Data General

Customer Documentation

# Programmer's Reference for the DG/UX™ System (Volume 3)

AViiON®
PRODUCT LINE

# Programmer's Reference for the DG/UX™ System (Volume 3)

093-701102-00

*For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.*

Ordering No. 093-701102

Revision 00, June 1991

Programmer's Reference for the DG/UX System (Volume 3)

093-701102-00

| Revision History: | Effective with: |
|---|---|
| Original Release – June 1991 | DG/UX 5.4 |

The chapters in Volume 3 were previously part of Volume 2 (093-701056).

# Preface

This is Volume 3 of the *Programmer's Reference for the DG/UX™ System*. The *Programmer's Reference* describes the programming features of the DG/UX system. It contains individual manual pages that describe commands, system calls, subroutines, file formats, and other useful topics, such as the ASCII table shown on **ascii**(5).

This manual is part of a five-volume reference set. The other manuals are the *System Manager's Reference for the DG/UX System* and the *User's Reference for the DG/UX System*. These manuals contain in printed (typeset) form the online entries released with the DG/UX System in **/usr/catman** for access by the **man** command.

The *Programmer's Reference* provides neither a general overview of the DG/UX system nor details of the implementation of the system. For more details about some of the most often used programming tools, see *Programmer's Guide: ANSI C and Programming Support Tools*, *Programmer's Guide: System Services and Application Packaging Tools*, and the Data General supplements to these two manuals. Other related manuals are listed under "Related Manuals" at the end of this manual.

## Man Pages

For historical reasons, each entry is called a "manual page" or "man page," though an entry may occupy more than one physical page and may contain more than one entry. If the man page contains more than one entry, it is alphabetized under its "primary" name; for example, the **utmp** manual page describes the **utmp** and **wtmp** files.

Manual pages are assigned to classes ranging from 0 through 8 for easy cross-reference. The class number appears in parentheses following the name; for example, in **accept**(1M) the "1" indicates that **accept** is a command, and the "M" indicates that the man page is in the *System Manager's Reference*.

A command followed by a (1) or (1G) usually means that it is described in the *User's Reference*. (Class 1 commands appropriate for use by programmers are located in the *Programmer's Reference*.) A man page name with a (1M), (4M), (7), or (8) following it means that the entry is in the *System Manager's Reference*. Names with (2) or (3*x*), (4), (5) [except **editread**(5)], or (6F) are in the *Programmer's Reference*. Occasionally, DG/UX man pages refer to other products' man pages, which are not part of the DG/UX documentation; these are so noted.

# Manual Organization

Volume 1 contains two chapters:

**Chapter 1: Commands (1)**
This chapter describes commands that support C and other programming languages.

**Chapter 2: System Calls (2)** This chapter describes the access to services provided by the DG/UX kernel, including the C language interface and a description of returned error codes.

Volume 2 contains one chapter:

**Chapter 3: Subroutines and Libraries (3)** This chapter describes the available subroutines and subroutine libraries. Their binary versions reside in various system libraries in the directories **/lib** and **/usr/lib**. See **intro**(3) for descriptions of these libraries and the files in which they are stored. Although these man pages are alphabetized together, each has a letter associated with the number 3 indicating the pertinent library:

| | |
|---|---|
| 3C | C Programming Language Libraries |
| 3E | ELF Library Routines |
| 3G | General Library Routines |
| 3M | Mathematical Library Routines |
| 3N | Networking Support Utilities |
| 3S | Standard I/O Library Routines |
| 3X | Specialized Libraries |

Volume 3 contains three chapters and one appendix:

**Chapter 4: File Formats (4)** This chapter documents the structure of particular kinds of files; for example, the format of the output of the link editor is given in **a.out**(4). Excluded are files used by only one command (for example, the assembler's intermediate files). In general, the C language structures corresponding to these formats can be found in the directories **/usr/include** and **/usr/include/sys**.

**Chapter 5: Miscellaneous Features (5)** This chapter contains a variety of facilities. Included are descriptions of character sets, macro packages, and other things.

**Chapter 6: Communications Protocols (6)** This chapter contains a description of the **unix_ipc** communications facility.

**Appendix A: Contents and Permuted Index Man Pages**
These manual pages contain information extracted from the DG/UX man pages in all five reference volumes.

## Man Page Format

Each man page has at least some of the following sections:

| | |
|---|---|
| NAME | gives the primary name (and secondary names, as the case may be) and briefly states its purpose. |
| SYNOPSIS | summarizes the usage of the program being described. |
| DESCRIPTION | discusses how to use these commands. |
| EXAMPLES | gives examples of usage, where appropriate. |
| FILES | contains the file names that are referenced by the program. |
| EXIT CODES | discusses values set when the command terminates. The value set is available in the shell environment variable "?" (see **sh**(1)). |
| DIAGNOSTICS | discusses the error messages that may be produced. Messages that are intended to be self-explanatory are not listed. |
| SEE ALSO | offers pointers to related information. |
| NOTES | gives information that may be helpful under the particular circumstances described. |

Some man pages may contain other heads such as ENVIRONMENT and CAVEATS.

## Man Page Notation Conventions

This manual uses certain symbols and styles of type to indicate different meanings in man pages. Those symbol and typeface conventions are defined in the following list. You should familiarize yourself with these conventions before reading the manual.

The description of convention meanings uses the terms "command line," "format line," and "syntax line." A command line is an example of a command string that you should type verbatim; it is preceded by a system prompt. A format line shows how to structure a command; it shows the variables that must be supplied and the available options. A syntax line is a fragment of program code that shows how to use a particular routine; some syntax lines contain variables.

| Convention | Meaning |
|---|---|
| **boldface** | This font is used for section heads and subsection heads. It is also used to distinguish input from output in examples where the two are intermixed. |
| `constant width/ monospace` | In command formats and code syntax: This typeface indicates text (including punctuation) that you type verbatim from your keyboard.<br>In text: This typeface is used for examples, code samples, pathnames, and the names of commands, files, directories, and manual pages.<br>In all contexts: The following characters, which have special meanings explained below, do not have special meaning but simply represent themselves when they appear in constant-width font: `< > [ ] { } \|`. In constant-width font they are are I/O redirection operators, brackets, braces, and the pipe symbol. |
| *italic* | In format lines: This font represents variables for which you supply values; for example, the names of your directories and files, your username and password, and possible arguments to commands. |
| [*optional*] | In format lines: Regular-font brackets surround an optional argument. Don't type the brackets; they only set off what is optional. These brackets should not be confused with constant-width brackets. |
| *choice1*\|*choice2* | In format lines: The vertical bar indicates a choice between *choice1* and *choice2*. |
| ... | In format lines and syntax lines: You can repeat the preceding argument as many times as desired. |
| { } | In format lines: These regular-font braces surround either two or more choices or syntax elements that are repeatable as a group. |
| < > | In command lines and other examples: Angle brackets distinguish a command sequence or a keystroke (such as <**Ctrl-D**>, <**Esc**>, and <**3dw**>) from surrounding text. Note that these angle brackets are in regular type and that you do not type them; there are, however, constant-width versions of these symbols that you do type. |
| `$, %, #` | In command lines and other examples: These symbols represent the system command prompt symbols used for the Bourne and Korn shells, the C shell, and the superuser, respectively. Note that your system might use different symbols for the command prompts. |

## Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

### Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative. A list of related documents appears at the end of this manual with the TIPS order form.

For a complete list of AViiON® and DG/UX™ manuals, see the *Guide to AViiON® and DG/UX™ System Documentation* (069-701085). The on-line version of this manual found in **/usr/release/doc_guide** contains the most current list.

### Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service options. If you are within the United States or Canada, contact the Data General Service Center by calling 1-800-DG-HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

## Joining Our Users Group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive FOCUS monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1-800-877-4787 or 1-512-345-5316.

End of Preface

# Contents

# Chapter 4
# File Formats

This chapter contains in printed form all the online manual entries for file formats. The entries are in alphabetical order except for **intro**(4), which is first.

For other file format manual pages (4M), see the *System Manager's Reference for the DG/UX System*.

## NAME

`intro` – introduction to file formats

## DESCRIPTION

This section outlines the formats of various files. The C structure declarations for the file formats are given where applicable. Usually, the header files containing these structure declarations can be found in the directories `/usr/include` or `/usr/include/sys`. For inclusion in C language programs, however, the syntax `#include` <*filename.h*> or `#include` <`sys/`*filename.h*> should be used.

## SEE ALSO

`intro`(4M).

## NAME

a.out – assembler and link editor output

## SYNOPSIS

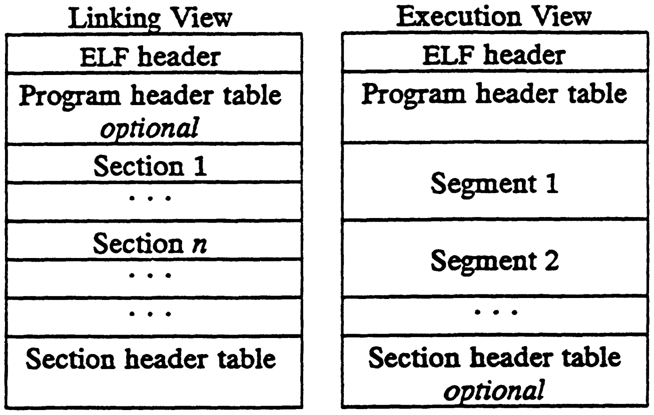
```
#include <elf.h>          /* for ELF executables*/

#include <a.out.h>/* for COFF executables */
```

## DESCRIPTION

The filename a.out is the default output filename from the link editor ld(1). The link editor will make a.out executable if there were no errors in linking. The output file of the assembler, as(1), also follows the common object file format of the a.out file although the default filename is different.

### ELF (Executable and Linking Format) Files

Programs that manipulate ELF files may use the library that elf(3E) describes. An overview of the file format follows. For more complete information, see the references given below.



An ELF header resides at the beginning and holds a "road map" describing the file's organization. Sections hold the bulk of object file information for the linking view: instructions, data, symbol table, relocation information, and so on. Segments hold the object file information for the program execution view. As shown, a segment may contain one or more sections.

A program header table, if present, tells the system how to create a process image. Files used to build a process image (execute a program) must have a program header table; relocatable files do not need one. A section header table contains information describing the file's sections. Every section has an entry in the table; each entry gives information such as the section name, the section size, etc. Files used during linking must have a section header table; other object files may or may not have one.

Although the figure shows the program header table immediately after the ELF header, and the section header table following the sections, actual files may differ. Moreover, sections and segments have no specified order. Only the ELF header has a fixed position in the file.

When an a.out file is loaded into memory for execution, three logical segments are set up: the text segment, the data segment (initialized data followed by uninitialized, the latter actually being initialized to all 0's), and a stack. The text segment is not writable by the program; if other processes are executing the same a.out file, the processes will share a single text segment.

The data segment starts at the next maximal page boundary past the last text address. (If the system supports more than one page size, the "maximal page" is the largest supported size.) When the process image is created, the part of the file holding the end of text and the beginning of data may appear twice. The duplicated chunk of text that appears at the beginning of data is never executed; it is duplicated so that the operating system may bring in pieces of the file in multiples of the actual page size without having to realign the beginning of the data section to a page boundary. Therefore, the first data address is the sum of the next maximal page boundary past the end of text plus the remainder of the last text address divided by the maximal page size. If the last text address is a multiple of the maximal page size, no duplication is necessary. The stack is automatically extended as required. The data segment is extended as requested by the `brk`(2) system call.

## COFF (Common Object File Format) Files

A common object file consists of a file header, a UNIX system header (if the file is link editor output), a table of section headers, relocation information, (optional) line numbers, a symbol table, and a string table. The order is given below:

File header.
UNIX system header.
Section 1 header.
...
Section n header.
Section 1 data.
...
Section n data.
Section 1 relocation.
...
Section n relocation.
Section 1 line numbers.
...
Section n line numbers.
Symbol table.
String table.

The last three parts of an object file (line numbers, symbol table and string table) may be missing if the program was linked with the `-s` option of `ld`(1) or if they were removed by `strip`(1). Also note that the relocation information will be absent after linking unless the `-r` option of `ld`(1) was used. The string table exists only if the symbol table contains symbols with names longer than eight characters.

The sizes of each section (contained in the header, discussed below) are in bytes.

When an `a.out` file is loaded into memory for execution, three logical segments are set up: the text segment, the data segment (initialized data followed by uninitialized, the latter actually being initialized to all 0's), and a stack. On the M88K computer the text segment typically starts at location 0x00010000 plus the byte offset in the a.out file of the text section data.

The first 16 bits of `a.out` files is the magic number. For non-executable `a.out` files and executables linked in the m88kbcs SDE, the magic number is 0555.For executables linked in the dgux SDE, the magic number is 0541. See `sde`(1). The optional header of an `a.out` file produced by `ld`(1) also has a magic number whose value is 0413. The headers (file header, optional header, and section headers) appear at the beginning of `a.out` files and determine the address of the text segment when it is loaded into memory. The first text address will equal 0x00010000 plus the size of the headers, and will vary depending upon the number of section headers in the `a.out` file. In an `a.out` file with three sections (.text, .data, and .bss), the first text address is at 0x000100B8 on the M88K computer.The text segment is not writable by the program; if other processes are executing the same `a.out` file, the processes will share a single text segment.

On the M88K computer the stack begins at location 0xF000000 and grows toward lower addresses. The stack is automatically extended as required. The data segment is extended only as requested by the `brk`(2) system call.

For relocatable files the value of a word in the text or data portions that is not a reference to an undefined external symbol is exactly the value that will appear in memory when the file is executed. If a word in text or data involves a reference to an undefined external symbol, there will be a relocation entry for the word, the storage class of the symbol-table entry for the symbol will be marked as an "external symbol", and the value and section number of the symbol-table entry will be undefined. When the file is processed by the link editor and the external symbol becomes defined, the value of the symbol will be added to the word in the file.

The format of the `filehdr` header is

```
struct filehdr
{
    unsigned short   f_magic;    /* magic number */
    unsigned short   f_nscns;    /* number of sections */
    long             f_timdat;   /* time and date stamp */
    long             f_symptr;   /* file ptr to symtab */
    long             f_nsyms;    /* # symtab entries */
    unsigned short   f_opthdr;   /* sizeof(opt hdr) */
    unsigned short   f_flags;    /* flags */
};
```

The format of the optional header is

```
typedef struct aouthdr
{
    short   magic;        /* magic number */
    short   vstamp;       /* version stamp */
    long    tsize;        /* text size in bytes, padded */
    long    dsize;        /* initialized data (.data) */
    long    bsize;        /* uninitialized data (.bss) */
    long    entry;        /* entry point */
    long    text_start;   /* base of text used for this file */
    long    data_start;   /* base of data used for this file */
} AOUTHDR;
```

The format of the section header is

```
struct scnhdr
{
    char            s_name[8];  /* section name */
    long            s_paddr;    /* physical address */
    long            s_vaddr;    /* virtual address */
    long            s_size;     /* section size */
    long            s_scnptr;   /* file ptr to raw data */
    long            s_relptr;   /* file ptr to relocation */
    long            s_lnnoptr;  /* file ptr to line numbers */
    unsigned long   s_nreloc;   /* # reloc entries */
    unsigned long   s_nlnno;    /* # line number entries */
    long            s_flags;    /* flags */
};
```

Object files have one relocation entry for each relocatable reference in the text or data. If relocation information is present, it will be in the following format:

```
struct reloc
{
    long            r_vaddr;    /* (virtual) address of reference
    long            r_symndx;   /* index into symbol table */
    unsigned short  r_type;     /* relocation type */
    unsigned short  r_offset;   /* high 16 bits of expression */
};
```

The start of the relocation information is *s_relptr* from the section header. If there is no relocation information, *s_relptr* is 0.

The format of each symbol in the symbol table is

```
#define  SYMNMLEN  8
#define  FILNMLEN  14
#define  DIMNUM    4

struct syment
{
    union                           /* all ways to get a symbol name
    {
        char        _n_name[SYMNMLEN]; /* name of symbol */
        struct
        {
            long    _n_zeroes;      /* == 0L if in string table */
            long    _n_offset;      /* location in string table */
        } _n_n;
        char        *_n_nptr[2];    /* allows overlaying */
    } _n;
    long            n_value;        /* value of symbol */
    short           n_scnum;        /* section number */
    unsigned short  n_type;         /* type and derived type */
    char            n_sclass;       /* storage class */
    char            n_numaux;       /* number of aux entries */
    char            n_pad1;         /* pad to 4 byte multiple */
    char            n_pad2;         /* pad to 4 byte multiple */
};
```

```
#define  n_name    _n._n_name
#define  n_zeroes  _n._n_n._n_zeroes
#define  n_offset  _n._n_n._n_offset
#define  n_nptr    _n._n_nptr[1]
```

Some symbols require more information than a single entry; they are followed by *auxiliary entries* that are the same size as a symbol entry. The format follows:

```
union auxent {
     struct {
          long  x_tagndx;
          union {
                struct {
                       unsigned longx_lnno;
                       unsigned longx_size;
                } x_lnsz;
                long   x_fsize;
          } x_misc;
          union {
                struct {
                       long  x_lnnoptr;
                       long  x_endndx;
                }  x_fcn;
                struct {
                       unsigned shortx_dimen[4];
                } x_ary;
                         struct {
                                 unsigned long   x_dimen1[2];
                    } x_ary1;



          } x_fcnary;
          unsigned short  x_tvndx;
          char x_pad1;
          char x_pad2;
     } x_sym;

        struct {
           unsigned long x_dimen2[5];
        } x_ary2;

        union {
          char  x_fname[FILNMLEN];
                struct {
                     long _x_zeroes;      /* 0 if name is in string table*/
                     long _x_offset;      /* offset into string table */
                } _x_x;
                char      *_x_xptr[2];    /* allows for overlaying */
        } x_file;
     } x_file;

     struct {
          long      x_scnlen;
```

```
                unsigned short  x_nreloc;
                unsigned short  x_nlinno;
           } x_scn;

           struct {
                long          x_tvfill;
                unsigned short   x_tvlen;
                unsigned short   x_tvran[2];
           } x_tv;
      };
```

Indexes of symbol table entries begin at *zero*. The start of the symbol table is *f_symptr* (from the file header) bytes from the beginning of the file. If the symbol table is stripped, *f_symptr* is 0. The string table (if one exists) begins at *f_symptr* + (*f_nsyms* * SYMESZ) bytes from the beginning of the file.

**SEE ALSO**

as(1), att_dump(1), cc(1), ld(1), ld-coff(1), brk(2), elf(3E), filehdr(4), ldfcn(4), linenum(4), reloc(4), syms(4).

The "Object Files" chapter in the *Programmer's Guide: ANSI C and Programming Support Tools*.

## NAME

acct – per-process accounting file format

## SYNOPSIS

```
#include <sys/acct.h>
```

## DESCRIPTION

Files produced as a result of calling acct(2) have records in the form defined by <sys/acct.h>, whose contents are:

```
typedef  ushort comp_t;  /* "floating point" */
                         /* 13-bit fraction, 3-bit exponent  */


struct  acct
{
   char     ac_flag;     /* Accounting flag */
   char     ac_stat;     /* Exit status */
   ushort   ac_uid;      /* Accounting user ID */
   ushort   ac_gid;      /* Accounting group ID */
   dev_t    ac_tty;      /* control typewriter */
   time_t   ac_btime;    /* Beginning time */
   comp_t   ac_utime;    /* acctng user time in clock ticks */
   comp_t   ac_stime;    /* acctng system time in clock ticks */
   comp_t   ac_etime;    /* acctng elapsed time in clock ticks */
   comp_t   ac_mem;      /* memory usage in kbytes */
   comp_t   ac_io;       /* chars trnsfrd by read/write */
   comp_t   ac_rw;       /* number of block reads/writes */
   char     ac_comm[8];  /* command name */
};
```

Also defined are the following symbolic names:

```
AFORK /* has executed fork, but no exec */ ASU   /* used super-
user privileges */ ACCTF /* record type: 00 = acct */
```

In *ac_flag*, the AFORK flag is turned on by each fork(2) and turned off by an exec(2). The *ac_comm* field is inherited from the parent process and is reset by any exec. Each time the system charges the process with a clock tick, it also adds to *ac_mem* the current process size, computed as follows:

(data size) + (text size) / (number of in-core processes using text)

The value of *ac_mem / (ac_stime + ac_utime)* can be viewed as an approximation to the mean process size, as modified by text-sharing.

The structure `tacct.h`, which resides with the source files of the accounting commands, represents the total accounting format used by the various accounting commands:

```
/*
 *  total accounting (for acct period), also for day
 */

struct tacct {
   uid_t          ta_uid;        /* userid */
   char           ta_name[8];    /* login name */
   float          ta_cpu[2];     /* cum. cpu time, p/np (mins) */
   float          ta_kcore[2];   /* cum kcore-minutes, p/np */
   float          ta_con[2];     /* cum. connect time, p/np, mins */
   float          ta_du;         /* cum. disk usage */
   long           ta_pc;         /* count of processes */
   unsigned short ta_sc;         /* count of login sessions */
   unsigned short ta_dc;         /* count of disk samples */
   unsigned short ta_fee;        /* fee for special services */
};
```

## SEE ALSO

acct(2), exec(2), fork(2).
acct(1M) in the *System Manager's Reference for the DG/UX System.*
acctcom(1) in the *User's Reference for the DG/UX System.*

## NOTES

The *ac_mem* value for a short-lived command gives little information about the actual size of the command because *ac_mem* may be incremented while a different command (like the shell) is being executed by the process.

## NAME

ar – DG/UX common archive file format

## DESCRIPTION

The archive command ar is used to combine several files into one. Archives are used mainly as libraries to be searched by the link editor ld.

Each archive begins with the archive magic string.

```
#define  ARMAG   "!<arch>\n"   /* magic string */
#define  SARMAG  8             /* length of magic string */
```

Following the archive magic string are the archive file members. Each file member is preceded by a file member header which is of the following format:

```
#define  ARFMAG     "`\n"  /* header trailer string */

struct  ar_hdr            /* file member header */
{
    char    ar_name[16]; /* '/' terminated file member name */
    char    ar_date[12]; /* file member date */
    char    ar_uid[6];   /* file member user identification */
    char    ar_gid[6];   /* file member group identification */
    char    ar_mode[8];  /* file member mode (octal) */
    char    ar_size[10]; /* file member size */
    char    ar_fmag[2];  /* header trailer string */
};
```

All information in the file member headers is in printable ASCII. The numeric information contained in the headers is stored as decimal numbers (except for *ar_mode* which is in octal). Thus, if the archive contains printable files, the archive itself is printable.

If the file member name fits, the *ar_name* field contains the name directly, and is terminated by a slash (/) and padded with blanks on the right. If the member's name does not fit, *ar_name* contains a slash (/) followed by a decimal representation of the name's offset in the archive string table described below.

The *ar_date* field is the modification date of the file at the time of its insertion into the archive. Common format archives can be moved from system to system as long as the portable archive command ar is used.

Each archive file member begins on an even byte boundary; a newline is inserted between files if necessary. Nevertheless, the size given reflects the actual size of the file exclusive of padding.

Notice there is no provision for empty areas in an archive file.

Each archive that contains object files [see a.out(4)] includes an archive symbol table. This symbol table is used by the link editor ld to determine which archive members must be loaded during the link edit process. The archive symbol table (if it exists) is always the first file in the archive (but is never listed) and is automatically created and/or updated by ar.

The archive symbol table has a zero length name (i.e., ar_name[0] is '/'), ar_name[1]==' ', etc.). All "words" in this symbol table have four bytes, using the machine-independent encoding shown below. (All machines use the encoding described here for the symbol table, even if the machine's "natural" byte order is different.)

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0x01020304 | 01 | 02 | 03 | 04 |

The contents of this "file" are as follows:

1. The number of symbols. Length: 4 bytes.
2. The array of offsets into the archive file. Length: 4 bytes * "the number of symbols".
3. The name string table. Length: *ar_size* – 4 bytes * ("the number of symbols" + 1).

As an example, the following symbol table defines 4 symbols. The archive member at file offset 114 defines `name` and `object`. The archive member at file offset 426 defines `function` and a second version of `name`.

| Offset | +0 | +1 | +2 | +3 | |
|---|---|---|---|---|---|
| 0 | 4 | | | | 4 offset entries |
| 4 | 114 | | | | `name` |
| 8 | 114 | | | | `object` |
| 12 | 426 | | | | `function` |
| 16 | 426 | | | | `name` |
| 20 | n | a | m | e | |
| 24 | \0 | o | b | j | |
| 28 | e | c | t | \0 | |
| 32 | f | u | n | c | |
| 36 | t | i | o | n | |
| 40 | \0 | n | a | m | |
| 44 | e | \0 | | | |

The number of symbols and the array of offsets are managed with `sgetl` and `sputl`. The string table contains exactly as many null terminated strings as there are elements in the offsets array. Each offset from the array is associated with the corresponding name from the string table (in order). The names in the string table are all the defined global symbols found in the common object files in the archive. Each offset is the location of the archive header for the associated symbol.

If some archive member's name is more than 15 bytes long, a special archive member contains a table of file names, each followed by a slash and a new-line. This string table member, if present, will precede all "normal" archive members. The special archive symbol table is not a "normal" member, and must be first if it exists. The *ar_name* entry of the string table's member header holds a zero length name `ar_name[0]=='/'`, followed by one trailing slash (`ar_name[1]=='/'`), followed by blanks (`ar_name[2]==' '`, etc.). Offsets into the string table begin at zero. Example *ar_name* values for short and long file names appear below.

| Offset | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 | +8 | +9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | f | i | l | e | _ | n | a | m | e | _ |
| 10 | s | a | m | p | l | e | / | \n | l | o |
| 20 | n | g | e | r | f | i | l | e | n | a |
| 30 | m | e | x | a | m | p | l | e | / | \n |

| Member Name | *ar_name* | Note |
|---|---|---|
| `short-name` | `short-name/` | Not in string table |
| `file_name_sample` | `/0` | Offset 0 in string table |
| `longerfilenamexample` | `/18` | Offset 18 in string table |

## SEE ALSO

`ar`(1), `ld`(1), `strip`(1), `sputl`(3X), `a.out`(4).

## NOTES

`strip` will remove all archive symbol entries from the header. The archive symbol entries must be restored via the `-ts` options of the `ar` command before the archive can be used with the link editor `ld`.

## NAME

`checklist` – list of file systems processed by fsck and ncheck

## DESCRIPTION

`Checklist` may reside in directory `/etc` and contain a list of special file names. Each special file name is contained on a separate line and corresponds to a file system. Each file system will then be automatically processed by the `fsck`(1M) and `ncheck`(1M) commands. You have to create the `checklist` file yourself; the system does not create it for you.

If you have your special files in `fstab`, you do not need to create a `checklist` file to get `fsck` to process them.

## SEE ALSO

`fsck`(1M) and `ncheck`(1M) in the *System Manager's Reference for the DG/UX System.*
`fstab`(4).

## NAME

compver – compatible versions file

## DESCRIPTION

compver is an ASCII file used to specify previous versions of the associated package which are upward compatible. It is created by a package developer.

Each line of the file specifies a previous version of the associated package with which the current version is backward compatible.

Since some packages may require installation of a specific version of another software package, compatibility information is extremely crucial. Consider, for example, a package called "A" which requires version "1.0" of application "B" as a prerequisite for installation. If the customer installing "A" has a newer version of "B" (version 1.3), the compver file for "B" must indicate that "1.3" is compatible with version "1.0" in order for the customer to install package "A".

## NOTES

The comparison of the version string disregards white space and tabs. It is performed on a word-by-word basis. Thus "Version        1.3" and "Version 1.3" would be considered the same.

## EXAMPLE

A sample compver file is shown below.

```
Version 1.3
Version 1.0
```

## SEE ALSO

pkginfo(4).

## NAME

`copyright` – copyright information file

## DESCRIPTION

`copyright` is an ASCII file used to provide a copyright notice for a package. The text may be in any format. The full file contents (including comment lines) is displayed on the terminal at the time of package installation.

## SEE ALSO

`pkginfo`(4).

## NAME

core – format of core image file

## DESCRIPTION

The system writes out a core image of a terminated process when any of several errors occur. See signal(2) for the list of reasons; the most common are memory violations, illegal instructions, and user-generated quit signals. The core image is called core and is written in the process's working directory (if possible; normal access controls apply). A process with an effective user id different from the real user id will not produce a core image.

The first section of the core image is a copy of the system's per-user data for the process, including the registers as they were at the time of the fault. The remainder represents the actual contents of the user's core area when the core image was written. The text segment is not dumped.

The format of the information in the first section is described by the *user* structure of the system, defined in /usr/include/sys/user.h.

## SEE ALSO

sdb(1), dbx(1), setuid(2), signal(2).
crash(1M) in the *System Manager's Reference for the DG/UX System*.

## NAME

cpio – format of cpio archive

## DESCRIPTION

The header structure, when the -c option of cpio(1) is not used, is:

```
struct {
        short   h_magic,
                h_dev;
        ushort  h_ino,
                h_mode,
                h_uid,
                h_gid;
        short   h_nlink,
                h_rdev,
                h_mtime[2],
                h_namesize,
                h_filesize[2];
        char    h_name[h_namesize rounded to word];
} Hdr;
```

When the -c option is used, the header information is described by:

```
    sscanf(Chdr,"%6o%6o%6o%6o%6o%6o%6o%6o%11lo%6o%11lo%s",
            &Hdr.h_magic, &Hdr.h_dev, &Hdr.h_ino, &Hdr.h_mode,
            &Hdr.h_uid, &Hdr.h_gid, &Hdr.h_nlink, &Hdr.h_rdev,
            &Longtime, &Hdr.h_namesize,&Longfile,Hdr.h_name);
```

*Longtime* and *Longfile* are equivalent to *Hdr.h_mtime* and *Hdr.h_filesize*, respectively. The contents of each file are recorded in an element of the array of varying length structures, *archive*, with other items describing the file. Every instance of *h_magic* contains the constant 070707 (octal). The items *h_dev* through *h_mtime* have meanings explained in stat(2). The length of the null-terminated path name *h_name*, including the null byte, is given by *h_namesize*.

The last record of the *archive* always contains the name TRAILER!!!. Special files, directories, and the trailer are recorded with *h_filesize* equal to zero.

## SEE ALSO

stat(2).
cpio(1), find(1) in the *User's Reference for the DG/UX System*.

## NAME

d_passwd – log-in programs and passwords for dial-up devices

## SYNOPSIS

/etc/d_passwd

## DESCRIPTION

This file contains an entry for programs (such as shells) that login(1) can invoke for users logging into the system via dial-up devices. Each entry includes the pathname of the shell program for which a dialup password is required and the encrypted password that the user must provide in order to invoke the program. You have to create a d_passwd file yourself; the system does not create one for you.

A dial-up device is any device that has an entry in the /etc/dialups file. See dialups(4). You have to create a dialups file yourself; the system does not create one for you.

When a user logs into a dial-up device, login searches the d_passwd file to see if it contains an entry for the shell program specified in the user's passwd entry. If such an entry is found, login requires that the user provide a second ("dial-up") password in addition to their personal password. The program name in the user's passwd entry and the program name in the d_passwd file must match exactly. E.g., /bin/csh and /usr/bin/csh will not be matched even though they reference the same file.

The program /usr/bin/sh is treated as a special case. If d_passwd contains an entry for /usr/bin/sh, the password for that entry will be used as the default dial-up password for all users whose passwd shell program doesn't match any of the other d_passwd entries. In the case where no matching entry is found for a user and no /usr/bin/sh entry exists, the user is not prompted for a dial-up password.

Here is a sample d_passwd entry:

```
/bin/csh:xxxxxx:
```

where xxxxxx is the encrypted password.

## SEE ALSO

login(1), dialups(4).

## NAME

depend – software dependencies files

## DESCRIPTION

depend is an ASCII file used to specify information concerning software dependencies for a particular package. The file is created by a software developer.

Each entry in the depend file describes a single software package. The instance of the package is described after the entry line by giving the package architecture and/or version. The format of each entry and subsequent instance definition is:

```
type pkg name
        (arch)version
        (arch)version
        ...
```

The fields are:

*type* Defines the dependency type. Must be one of the following characters:

- P Indicates a prerequisite for installation, for example, the referenced package or versions must be installed.
- I Implies that the existence of the indicated package or version is incompatible.
- R Indicates a reverse dependency. Instead of defining the package's own dependencies, this designates that another package depends on this one. This type should be used only when an old package does not have a depend file but it relies on the newer package nonetheless. Therefore, the present package should not be removed if the designated old package is still on the system since, if it is removed, the old package will no longer work.

*pkg* Indicates the package abbreviation.

*name* Specifies the full package name.

*(arch)version* Specifies a particular instance of the software. A version name cannot begin with a left parenthesis. The instance specifications, both *arch* and *version*, are completely optional but must each begin on a new line that begins with white space. A null version set equates to any version of the indicated package.

## EXAMPLE

Here is a sample depend file:

```
I msvr 3B2 Messaging Server
P ctc Cartridge Tape Utilities
P dfm Directory and File Management Utilities
P ed Editing Utilities
P ipc Inter-Process Communication Utilities
P lp Line Printer Spooling Utilities
P shell Shell Programming Utilities
P sys System Header Files
          Release 3.0
P sysadm System Administration Utilities
P term Terminal Filters Utilities
```

```
P terminfo Terminal Information Utilities
P usrenv User Environment Utilities
P uucp Basic Networking Utilities
P x25 X.25 Network Interface
            Issue 1 Version 1
            Issue 1 Version 2
P windowing AT&T Windowing Utilities
            (3B2)Version 1
R cms 3B2 Call Management System
```

## SEE ALSO

pkginfo(4).

## NAME

dialups – devices requiring a dial-up password.

## SYNOPSIS

/etc/dialups

## DESCRIPTION

This file contains the pathnames of devices that require an additional password, called a dial-up password, from users who attempt to log into it. An example entry might be /dev/tty16. For such devices, the login(1) command prompts the user for the dial-up password after the user has provided a valid log-in name and personal password.

Dial-up passwords must appear in the /etc/d_passwd file along with the programs (such as a shell) that login will execute after a succesful log-in at the given device.

You have to create the dialups and d_passwd files yourself; the system does not create them for you.

## SEE ALSO

login(1), d_passwd(4).

## NAME

`dirent` – file system independent directory entry

## SYNOPSIS

```
#include <sys/dirent.h>
#include <sys/types.h>
```

## DESCRIPTION

Different file system types may have different directory entries. The `dirent` structure defines a file system independent directory entry, which contains information common to directory entries in different file system types. A set of these structures is returned by the `getdents`(2) system call.

The `dirent` structure is defined below.

```
struct dirent {
              long            d_ino;
              off_t           d_off;
              unsigned short  d_reclen;
              char            d_name[1];
      };
```

The *d_ino* is a number which is unique for each file in the file system. The field *d_off* is the offset of that entry in the file system directory. The field *d_name* is the beginning of the character array giving the name of the directory entry. This name is null terminated and may have at most MAXNAMLEN characters. This results in file system independent directory entries being variable length entities. The value of *d_reclen* is the record length of this entry. This length is defined to be the number of bytes between the current entry and the next one, so that it will always result in the next entry being on a long boundary.

## FILES

`/usr/include/sys/dirent.h`

## SEE ALSO

`getdents`(2).

## NAME

`dumptab` – tape table file for `dump2`

## DESCRIPTION

`/etc/dumptab` is an ASCII file containing an entry describing media characteristics for each medium made available to `dump2`.

This table file contains lines in one of three formats:

a. comment lines (must start with a "#")

b. lines specifying the capacity of the medium:

*medium-name buffer-size* <capacity>

c. lines giving the density, tape length, and IRG for the medium:

*medium-name buffer-size density tape-length* <IRG>

Fields are separated by white space. The fields are desribed below:

`medium-name`
: descriptive label for the medium.

`buffer-size`
: size (in 1024-byte blocks) of the buffers written to the medium.

`capacity`
: formatted capacity of the medium (in bytes). The `capacity` can also be specified as a number followed by a upper or lowercase `b`, `k`, `m`, or `g` to indicate bytes, kilobytes, megabytes, or gigabytes, respectively.

`density` density at which data is written to the device (in bpi).

`tape-length`
: length of the tape (in feet).

`IRG` inter-record gap size used by the device (in tenths per inch).

## SEE ALSO

`dump2`(1M).

## NAME

filehdr – file header for common object files

## SYNOPSIS

```
#include <filehdr.h>
```

## DESCRIPTION

Every common object file begins with a 20-byte header. The following C `struct` declaration is used:

```
struct  filehdr  {
   unsigned short  f_magic ;   /* magic number */
   unsigned short  f_nscns ;   /* number of sections */
   long            f_timdat ;  /* time & date stamp */
   long            f_symptr ;  /* file ptr to symtab */
   long            f_nsyms ;   /* # symtab entries */
   unsigned short  f_opthdr ;  /* sizeof(opt hdr) */
   unsigned short  f_flags ;   /* flags */
} ;
```

*F_symptr* is the byte offset into the file at which the symbol table can be found. Its value can be used as the offset in `fseek`(3S) to position an I/O stream to the symbol table. The UNIX system optional header is 28-bytes. The magic number for the M88000 is:

```
#define MC88MAGIC 0540
```

The value in *f_timdat* is obtained from the `time`(2) system call. Flag bits currently defined are:

```
#define F_RELFLG  0000001 /* relocation entries stripped */
#define F_EXEC    0000002 /* file is executable */
#define F_LNNO    0000004 /* line numbers stripped */
#define F_LSYMS   0000010 /* local symbols stripped */
#define F_AR32W   0001000 /* non-DEC host */
#define F_BM32B   0020000 /* file contains WE 32100 code */
#define F_BM32MAU 0040000 /* file reqs MAU to execute */
```

## SEE ALSO

`time`(2), `fseek`(3S), `a.out`(4).

## NAME

fs – file system format

## SYNOPSIS

```
#include <ufs/disk_format.h>
```

## DESCRIPTION

There is a at most one filesystem for each logical disk. The basic components of a the file system are the File Manager Information Areas (FMIA's), Disk Allocation Regions (DAR's), and a table of entries containing information about each DAR called the DAR Information Area.

### The FMIA

Two copies of the FMIA are maintained to reduce its vulnerability to corruption. The copies are placed in the first and last blocks of the file system. The FMIA in the first block (the Primary FMIA) is contained in the first DAR, but the FMIA contained in the last block of the logical disk (the Secondary FMIA) is not contained in the last DAR.

The following is the definition of a FMIA. This contains the per-filesystem information. When a filesystem is mounted, this structure is used to generate memory databases for the newly mounted entry.

```
typedef struct
    {
    df_self_id_type           self_id;
    df_fsid_type              fsid;
    uint32e_type              minor_device_number;
    uint32e_type              dar_size;
    uint32e_type              file_nodes_per_dar;
    boolean16e_type           fsck_required;
    uint16e_type              revision;
    byte8e_type               fname[DF_FS_LABEL_SIZE];
    byte8e_type               fpack[DF_FS_LABEL_SIZE];
    uint8e_type               default_des_exponent;
    uint8e_type               default_ies_exponent;
    uint8e_type               default_dir_des_exponent;
    uint8e_type               default_dir_ies_exponent;
    uint32e_type              first_anniversary;
    uint32e_type              second_anniversary;
    uint32e_type              fs_size;
    uint32e_type              space_used;
    uint32e_type              number_of_used_file_nodes;
    uint32e_type              first_log_lda;
    uint32e_type              second_log_lda;
    uint32e_type              log_size;
    boolean_field_type        shrink_operation_in_progress;
    boolean_field_type        grow_operation_in_progress;
    skip_type                 reserved:14;
    byte8e_type               pad_to_block[DF_PADDING_PER_FMIA_BLOCK];
    }   df_fmia_block_type  ;
```

*self_id* is the self-identification information. The block kind is DF_FMIA_BLOCK. The block number is:

```
#define DF_PRIMARY_FMIA_ADDRESS   0
```

The file node number is:

```
#define DF_NODE_NUMBER_FOR_NON_FILES    012345670123
```

The following fields are assumed to be correct by `fsck`(1M).

*fsid* is the filesystem identifier unique among mounted file systems on a single host. It is kept on disk so that it will stay the same if possible from mount to mount. If it doesn't, NFS accesses using filehandles based on a previous mount will fail.

*minor_device_number* is the assigned extended minor device number. It is kept on disk so that it will stay the same if possible from mount to mount. If the value in this field on disk is not in the valid range for extended minor device numbers, it is file manager's responsibility to correct the problem at mount time.

*dar_size* is the size of a DAR in blocks. The minimum value for this field is:

```
#define DF_MIN_DAR_SIZE     4032
```

and the maximum value is:

```
#define DF_MAX_DAR_SIZE(fs_size)
```

`mkfs`(1M) defines the default for this field; for efficiency, it should be a multiple of:

```
#define DF_BITS_PER_BITMAP_BLOCK 4032
```

whenever possible; 4 to 12 MB (two to six bitmap blocks' worth) per DAR seems a reasonable default DAR size given current disk sizes. As disks grow by orders of magnitude in size, DAR sizes should likely grow linearly with the square root of the disk sizes.

*file_nodes_per_dar* is the number of file nodes for each DAR. This value must be a multiple of:

```
#define DF_FILE_NODE_MULTIPLE_REQUIREMENT     64
```

The minimum value for this field is

```
#define DF_MIN_FILE_NODES_PER_DAR 64
```

and the maximum value is:

```
#define DF_MAX_FILE_NODES_PER_DAR(dar_size)
```

`mkfs`(1M) defines this field's default, which is to have about one file node for each four user data blocks, similar to 4.2 BSD.

*fsck_required* indicates that `fsck`(1M) needs to be run. If this field is not zero (FALSE), the filesystem needs to be checked before it can be mounted.

*revision* is the revision number of the FMIA. Used to determine the type of filesystem that the FMIA resides on.

`fsck(1M)` will attempt to correct the following fields if they are invalid:

*fname* is used by `statfs(2)`, `fstatfs(2)`, `labelit(1M)`, `volcopy(1M)`, `frec(1M)`, Initialized to zeros, when used it is considered an ASCII string not necessarily terminated by a NULL byte.

*fpack* is used by `statfs(2)`, `fstatfs(2)`, `labelit(1M)`, `volcopy(1M)`, `frec(1M)`, Initialized to zeros, when used it is considered an ASCII string not necessarily terminated by a NULL byte.

The following exponent fields pertain to the size of elements used to access user data blocks. Data elements are equal sized sets of contiguous blocks of a file. These data elements are either pointed to directly from the file node or indirectly through an index structure. Index elements are arrays of block numbers. The index structure is hierarchical; an index block number may point to another index element or, if the bottom is reached, point to a data element. The direct or indexed access of data elements depends on the size of the file and the block being accessed; blocks at the beginning of the file can be accessed through the direct access to provide faster access for smaller files since they are generally more common. The following fields control the sizes of these elements, allowing the user to choose values more suitable for the types of files that will typically fill the file system. For more information about data access from the inode, see `inode(4)`.

*default_des_exponent* specifies the default data element size for non-directory files. The default data element size in blocks is 2 raised to the default_des_exponent power. The default value for this field is:

```
#define DF_DEFAULT_DEFAULT_DES_EXPONENT 4
```

The maximum value is:

```
#define DF_MAX_DES_EXPONENT 31
```

although it is also limited to the base 2 logarithm of the largest power of two that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

*default_ies_exponent* specifies the default index element size for non-directory files. The default index element size in blocks is 2 raised to the default_ies_exponent power. The default value for this field is:

```
#define DF_DEFAULT_DEFAULT_IES_EXPONENT 0
```

The maximum value is:

```
#define DF_MAX_IES_EXPONENT 15
```

although it is also limited to the base 2 logarithm of the largest power of two that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

*default_dir_des_exponent* specifies the default data element size for directories and CPDs. The default data element size in blocks is 2 raised to the default_dir_des_exponent power. The default value for this field is:

```
#define DF_DEFAULT_DEFAULT_DES_EXPONENT 4
```

The maximum value is:

```
#define DF_MAX_DES_EXPONENT 31
```

although it is also limited to the base 2 logarithm of the largest power of two that is less than or equal to

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

*default_dir_ies_exponent* specifies the default index element size for directories and CPDs. The default index element size in blocks is 2 raised to the default_dir_ies_exponent power. The default value for this field is:

```
#define DF_DEFAULT_DEFAULT_IES_EXPONENT 0
```

The maximum value is:

```
#define DF_MAX_IES_EXPONENT 15
```

although it is also limited to the base 2 logarithm of the largest power of two that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar).
```

*fs_size* is the number of blocks in the filesystem. `fsck`(1M) will check this against the disk size as reported by the device driver.

*space_used* is the total (user and system) space used on this filesystem, including any space wasted at the end due to an incomplete DAR.

*number_of_used_file_nodes* is the number of file nodes used in the file system, not including the wasted file nodes with node numbers 0 and 1.

*first_anniversary* is the first anniversary of each file in blocks. When a file first consumes this much space, the filesystem should change the DAR from which it gets space for the file. The minimum value of this field is 2 raised to the default_des_exponent power; the default value is:

```
#define DF_DEFAULT_FIRST_ANNIVERSARY(dar_size)
```

*second_anniversary* the second anniversary of each file in blocks. A file should change the DAR from which the filesystem gets space each time its space utilization crosses a multiple of the second anniversary. The second anniversary must be greater than or equal to the first anniversary. The default value of this field is:

```
#define DF_DEFAULT_SECOND_ANNIVERSARY(dar_size)
```

*first_log_lda* and *second_log_lda* give the logical disk address of the two halves of the fast recovery log. They will be zero if the file system was not mounted for fast recovery when the filesystem was last mounted or if /f4fsck/fP has been run over the file system.

*log_size* is the size in 512-byte blocks of each half of the fast recovery log.

*shrink_operation_in_progress* is set if the filesystem is in the process of being shrunk.

*grow_operation_in_progress* is set if the filesystem is in the process of being grown.

## The Disk Allocation Region (DAR)

The DAR is similar to the BSD cylinder group; however, the DAR is not necessarily associated with a physical disk cylinder as it is in BSD. The purpose of the DAR is to spread files throughout the filesystem while maintaining a locality between inodes and the data blocks associated with them.

The DAR consists of three parts: a bitmap, a file node tat :, and the data blocks allocated to files as they are needed.

The bitmap records the space allocation in the DAR. A bit in the bitmap represents a block in the DAR (this includes the blocks allocated for the bitmap and the file node table). If the bitmap value is 1, it is used; otherwise, it is free. The size of the bitmap is a function of the size of the DAR and is provided (in blocks) by:

```
#define DF_DAR_BITMAP_SIZE(dar_size)
```

The file node table contains entries for each file in the DAR. A file node entry (called an inode) contains information about the file. The first block of the table is after the bitmap. The number of file nodes in the DAR is a field in the FMIA. The number of blocks allocated to the table (in blocks) is:

```
#define DF_DAR_FILE_NODE_TABLE_SIZE(file_nodes_per_dar)
```

The ile node table element (the inode) is discussed in inode(4).

The data blocks take up the remaining blocks of the DAR.

With the exception of the blocks of the DAR Information Area and the Secondary FMIA, all blocks in the file system are contained in DAR's. The number of DAR's in a file system is a function of the size of the file system, the size of each DAR, and the file nodes contained in each DAR. This is provided by:

```
#define DF_NUMBER_OF_DARS(fs_size, dar_size, nodes_per_dar)
```

The last DAR of the file system may be the smaller than the other DAR's. If the space before the DAR Information Area and the Secondary FMIA is large enough to contain the DAR's bitmap and file node table, then the DAR will be created; otherwise, the space between the end of the last DAR and the beginning of the DAR Information Area is wasted. Since the bitmap in the last DAR is the same size as the other DAR's, if the last DAR is smaller the bitmap will have bits indicating the allocation of data blocks that do not exist (in fact it i legal for no data blocks to exist in the last DAR). In this case, the non-existent blocks are marked as allocated. The following macros provide values associated with the space before the DAR Information Area:

```
#define DF_LAST_DAR_SIZE(fs_size, dar_size, nodes_per_dar)
```

```
#define DF_FS_WASTED_SPACE(fs_size, dar_size, nodes_per_dar)
```

### The DAR Information Area

At the end of the file system, a table of entries exist for each DAR in the file system. It is located such that its last block of entries is before the last block of the file system containing the Secondary FMIA. This location is provided by:

```
#define DF_DARE_TABLE_ADDRESS(fs_size,dar_size,file_nodes_per_dar)
```

A definition for a DAR entry is:

```
typedef struct
    {
    uint32e_type                file_nodes_used;
    uint32e_type                space_used;
    uint32e_type                directories_used;
    df_file_node_number_type    free_file_node_number;
    byte8e_type                 reserved[DF_RESERVED_BYTES_PER_DAR];
    } df_dar_entry_type;
```

*file_nodes_used* Number of file_nodes in use from the DAR the entry represents.

*space_used* is the number of data blocks in use from the DAR. This explicitly excludes DAR Information Area blocks, the block containing the Secondary FMIA, and blocks marked as allocated in the last DAR but do not exist. This field includes the following system blocks: the Primary FMIA for the first DAR only, the DAR's bitmap blocks and the DAR's file node blocks.

*directories_used* is the number of directories in the DAR.

*free_file_node_number* is the file node number of next free file node in the DAR. This functions as the head of the DAR's free file node list.

## SEE ALSO

fstatfs(2), mount(2), statfs(2), inode(4). frec(1M), fsck(1M), labelit(1M), mkfs(1M), volcopy(1M) in the *System Manager's Reference for the DG/UX System*.

## NAME

`fspec` – format specification in text files

## DESCRIPTION

You many want to maintain text files on the DG/UX system with tabs that are not set at every eighth column. You must usually convert such files to a standard format, frequently by replacing all tabs with the appropriate number of spaces, before they can be processed by DG/UX system commands. A format specification in the first line of a text file specifies how tabs are to be expanded in the rest of the file.

A format specification consists of a sequence of parameters separated by blanks and surrounded by the brackets `<:` and `:>`. Each parameter consists of a keyletter, possibly followed immediately by a value. The following parameters are recognized:

t*tabs* The `t` parameter specifies the tab settings for the file. The value of *tabs* must be one of the following:

1. A list of column numbers separated by commas, indicating tabs set at the specified columns;
2. A – followed immediately by an integer *n*, indicating tabs at intervals of *n* columns;
3. A – followed by the name of a canned tab specification.

Standard tabs are specified by `t-8`, or equivalently, `t1,9,17,25`,etc. The canned tabs are defined by the `tabs`(1) command.

s*size* The `s` parameter specifies a maximum line size. The value of *size* must be an integer. Size is checked after tabs have been expanded, but before the margin is prepended.

m*margin* The `m` parameter specifies a number of spaces to be prepended to each line. The value of *margin* must be an integer.

`d` The `d` parameter takes no value. It indicates that the line containing the format specification is to be deleted from the converted file.

`e` The `e` parameter takes no value. It indicates that the current format is to prevail only until another format specification is encountered in the file.

Default values, which are assumed for parameters not supplied, are `t-8` and `m0`. If the `s` parameter is not specified, no size checking is performed. If the first line of a file does not contain a format specification, the above defaults are assumed for the entire file. The following is an example of a line containing a format specification:

```
* <:t5,10,15 s72:> *
```

For programming language source files, if you can disguise a format specification as a comment, you don't need to code the `d` parameter.

## SEE ALSO

ed(1), `newform`(1), `tabs`(1) in the *User's Reference for the DG/UX System.*

## NAME

`fstab` – static information about file systems

## SYNOPSIS

```
#include <mntent.h>
```

## DESCRIPTION

The file `/etc/fstab` describes the file systems and swapping areas used by the local machine. The system administrator can modify it with a text editor or by invoking the `sysadm`(1M) system administration utility. It is read by commands that mount, dump, restore, and check the consistency of file systems, as well as by the system in providing swap space. The file consists of a number of lines like this:

```
fsname dir type opts freq passno
```

for example:

```
/dev/dsk/usr  /usr  dg/ux  rw  1  1
```

would indicate a mount for a local file system, and

```
titan:/usr/titan  /usr/titan  nfs  rw,hard  0 0
```

would indicate an NFS file system mount.

A High Sierra CDROM would be indicated using the following line:

```
/dev/pdsk/4 /cdrom cdrom ro 0 0
```

A DOS floppy would be indicated using the following line:

```
/dev/pdsk/3 /pdd/floppy dos rw 0 0
```

A swap area could be indicated using the following line:

```
/dev/dsk/swap1 swap1_area swap sw 0 0
```

The `fstab` format was changed in order to support NFS file systems as well as local file systems. The old-style `fstab` entries are supported, but not recommended.

The entries from this file are accessed using the routines in `getmntent`(3C), which returns a structure of the following form:

```
struct mntent {
        char  *mnt_fsname;  /* file system name */
        char  *mnt_dir;     /* file system path prefix */
        char  *mnt_type;    /* dg/ux, nfs, swap, cdrom, or ignore */
        char  *mnt_opts;    /* rw, ro, hard, soft, bg, fg */
        int   mnt_freq;     /* highest dump level */
        int   mnt_passno;   /* pass number on parallel fsck */
};
```

Fields are separated by white space; a #, as the first non-white character, indicates a comment. The *mnt_type* field determines how the *mnt_fsname* and *mnt_opts* fields will be interpreted. The following is a list of the file system types currently supported, and the way each of them interprets these fields:

| *Type* | *Field* | *Interpretation* |
|---|---|---|
| dg/ux | mnt_fsname | Must be a block special device unless this is a ramdisk, in which case, it is a symbolic link to the mounted memory file system. |
| | mnt_opts | Valid options are ro, rw, bg, and fg. If this has the ramdisk option, other options include use_wired_memory, max_file_space and max_file_count. |
| cdrom | mnt_fsname | Must be a block special device. |
| | mnt_opts | Valid options are ro, bg, fg. |
| dos | mnt_fsname | Must be a block special device. |
| | mnt_opts | Common options are ro, rw, bg, fg. |
| nfs | mnt_fsname | The hostname of the server and the pathname on the server of the directory to be served. A colon separates the pathname and hostname. |
| | mnt_opts | Valid options are ro, rw, hard, soft, bg, fg. |
| swap | mnt_fsname | Must be a block special device swap section. |
| | mnt_opts | Ignored. |

If the *mnt_type* is specified as *ignore*, the entry is ignored. This is useful to show disks not currently used.

Entries identified as *swap* are made available as swap space by the *swapon(1M)* command at the end of the system reboot procedure.

When the *mnt_fsname* field is interpreted as a block special device, programs that require the corresponding character special device must construct the name by changing *dsk* to *rdsk* in the pathname.

If the *mnt_opts* field is a comma-separated list of options that includes `rw` or `ro`, the file system is mounted read-write or read-only. If this includes `hard` or `soft`, the NFS file system is mounted `hard` or `soft`. If the list includes `bg` or `fg`, and failed attempt to mount will cause `mount` to retry in the background or in the foreground. For more details on these options, see `mount`(1M).

The field *mnt_freq* indicates how often each file system should be dumped by the *dump2*(1M) command (and triggers that command's `w` option, which determines what file systems should be dumped). Most systems set the *mnt_freq* field to 1, indicating that file systems are dumped each day. Some programs, like *sysadm*, may use a different set of entries here.

The final field *mnt_passno* is used by the consistency checking program `fsck`(1M) to allow overlapped checking of file systems during a reboot. All file systems with a *mnt_passno* of 1 are checked first simultaneously, then all file systems with *mnt_passno* of 2 are checked, and so on. A value of 0 indicates that the file system will not be checked. The <*mnt_passno*> of the root file system should be 0, as the root cannot be checked since it is already mounted.

Programs read the `/etc/fstab` file but never write to it. It is the duty of the system administrator to maintain this file. The order of records in `/etc/fstab` is important because *fsck* and *mount* process the file sequentially; file systems must appear after file systems they are mounted within. For example, if you have an entry for `/usr/spool`, it must appear after the entry for `/usr`.

## FILES

/etc/fstab

## SEE ALSO

`dump2`(1M), `fsck`(1M), `mount`(1M), `swapon`(1M), `sysadm`(1M), `getfsent`(3X), `getmntent`(3C).

## NAME

group – group file

## SYNOPSIS

/etc/group

## DESCRIPTION

Group contains for each group the following information:

- group name
- encrypted password
- numerical group id
- a comma-separated list of all users allowed in the group

This is an ASCII file. The fields are separated by colons; each group is separated from the next by a newline. If the password field is null, no password is demanded.

This file resides in the /etc directory. Because of the encrypted passwords, it can and does have general read permission and can be used, for example, to map numerical group IDs to names.

A group file can have a line beginning with a plus sign (+), which means to incorporate entries from the Yellow Pages (YP).

NOTE: You must be using the DG/UX Open Network Computing/Network File System (ONC/NFS) to use this feature.

There are two styles of + entries: By itself, + means to insert the entire contents of the YP group file at that point; +*name* means to insert the entry (if any) for *name* from the YP at that point. If a + entry has a non-null password or group member field, the contents of that field will override what is contained in the YP. The numerical group ID field cannot be overridden.

Entries beginning with a minus (-) are also allowed, and have the format -*name*, which means to consider *name* to not be in the group file, regardless of subsequent entries to the contrary. Minus entries can be used to exclude specific groups that are present in the YP group database.

Grpck can be used to verify entries in the group file. See pwck(1M) in the *System Manager's Reference for the DG/UX System*.

## EXAMPLE

```
+myproject:::bill, steve
+:
```

If these entries appear at the end of a group file, then the group will have members *bill* and *steve* and the password and group ID of the YP entry for the group *myproject*. All the groups listed in the Yellow Pages will be pulled in and placed after the entry for *myproject*.

## FILES

/etc/group

## SEE ALSO

setgroups(2), crypt(3C), passwd(4), groups(1), newgrp(1), passwd(1), pwck(1M).

## NOTES

The passwd(1) command won't change group passwords.

Normally, group-ids less than 100 are reserved for system-level use (DG/UX software).

## NAME

`hfm` – high sierra file manager

## DESCRIPTION

The DG/UX kernel provides configurable support for High Sierra and ISO 9660 formatted Compact Discs (CDs). The high sierra file manager lets the system administrator mount a CD into the UNIX file system hierarchy. A mounted CD will appear as a readonly UNIX file system. The mode of all files from the CD will be readonly and executable for user, group and other.

Filenames in High Sierra or ISO 9660 format are uppercase, but for convenience, they are translated to lowercase by the high sierra file manager. All input filenames are similarly translated to uppercase. High Sierra and ISO 9660 mounted file systems can be NFS exported in the same way as any normal DG/UX file system. The mount point must be added to `/etc/exports` and the `exportfs`(8) command must be executed after the file system is mounted. This will be automatic if the mount of the CD is in your `/etc/fstab` file. Since most current CDs available in high sierra or ISO 9660 format are for PC's, the high sierra file manager will be most useful when used with a DOS emulator.

In order to use the high sierra file manager, you must configure the hfm() pseudo device into your kernel.

```
sd(insc(),*)
st(insc(),*)
inen()
loop()
pmt()
prf()
meter()
hfm()          # this is the line that must be added.
```

Once the kernel is built and running, you may use the `mount`(1M) command to add the high sierra or ISO 9660 file system to the UNIX file system hierarchy.

```
        mount -t cdrom /dev/pdsk/4 /pdd/cdrom
```

The special device mentioned in the mount command is the block special representation of the CD device in `/dev/pdsk`. The type "cdrom" must be used with mount to route the mount request to the correct file manager.

You may add a line to the /etc/fstab file to have the mount occur when the system is brought up to init level 3.

```
/dev/pdsk/4 /pdd/cdrom  cdrom ro x 0
```

The `umount`(1M) command may be used to unmount the CD from the file system hierarchy

```
        umount /pdd/cdrom
```

To export the file system on the CD, in lieu of adding it to `/etc/exports`:

```
        exportfs -iv /pdd/cdrom
```

When the `mount`(1M) command is issued, the CD device will lock the CD platter into the unit until a successful `umount`(1M) is issued.

The high sierra file manager does not support the path table or the extended attribute record from files on the CD, as these are unnecessary to the UNIX file system implementation.

## SEE ALSO

`config`(1M), `mount`(1M), `umount`(1M), `fstab`(4), `exportfs`(8).

## NAME

holidays – accounting information used to distinguish prime and non-prime days

## SYNOPSIS

/usr/lib/acct/holidays

## DESCRIPTION

The holidays file distinguishes between *prime* and *non-prime* time for the accounting system. It divides weekdays into two pieces, and it divides the year into prime and non-prime days. Weekends are always non-prime. Additional company holidays can be specified as non-prime.

Comment lines are denoted by an asterisk in column one.

The first non-comment line contains three fields, separated by white space. The first field is the four-digit current year. The second field is the start of prime time, specified as four digits in the form *hhmm* (for hour and minute). The third field is the start of non-prime time, specified in the same way. The hours must be between 0 and 23, inclusive, and the minutes must be between 0 and 59, inclusive.

Subsequent lines define up to 20 non-prime days. The first field is the day of year, where January 1 has the value 1. The second field is the calendar date. The third field is the holiday name.

## EXAMPLE

```
* Prime/Nonprime Table for UNIX Accounting System
*
* Curr  Prime   Non-Prime
* Year  Start   Start
*
  1989  0830    1700
*
* Day of        Calendar        Company
* Year          Date            Holiday
*
    2           Jan 2           New Year's Day Observed
  149           May 29          Memorial Day
  184           Jul 3           Day Before Independence Day
  185           Jul 4           Independence Day
  247           Sep 4           Labor Day
  327           Nov 23          Thanksgiving
  328           Nov 24          Day After Thanksgiving
  359           Dec 25          Christmas Day
```

## SEE ALSO

acctcon(1M), acctprc(1M).

`mnemonic`
A one-character abbreviation for the menu's *name*.

`name` A one or two word name for the menu.

`title` A string, such as "Main Menu" which is used as the title for the menu.

`visible` A boolean indication of whether this menu will be displayed. If the value is `${NO}`, the menu will not be shown by `idi(1)`.

**operation Class**

Instances of the *operation* class are the basic actions which can be performed by the user. *Operations* may contain queries which must be answered before performing the action. *Operations* are added to *menus* with the `add` statement.

The following attributes are allowed for the *operation* class:

| operation Attribute Set | | |
|---|---|---|
| Name | Type | Default |
| access-groups | *name-list* | "" |
| access-names | *name-list* | "*" |
| action | *command* | "" |
| confirm | *value* | "" |
| description | *value* | "No description" |
| entry-action | *command* | "" |
| exit-action | *command* | "" |
| help | *value* | "No help for this operation." |
| mnemonic | *value* | "" |
| name | *value* | "Unnamed" |
| repeat | *value* | "" |
| visible | *boolean* | "${YES}" |

The attributes have the following meanings:

`access-groups`
A whitespace-separated list of group names which are allowed access to this operation. A star ("*") means that all groups are allowed access.

`access-names`
A whitespace-separated list of user names which are allowed access to this operation. A star ("*") means that all users are allowed access.

`action` A shell command line to execute when this operation is selected (after any queries for the operation are answered and confirmed). This command is not executed if the operation is canceled.

`confirm` A string to use as a confirmation prompt which must be answered before the operation is executed. If the value of this attribute is the empty string, no confirmation is performed.

`description`
A one-line description of the operation.

`entry-action`
A shell command line to execute as soon as the operation is selected, before any screens or queries are presented. If the value of the `repeat` attribute is not empty, the `entry-action` is performed once for each iteration of the operation.

`exit-action`
: A shell command line to execute after all processing of the operation has completed. This command is executed after the `action` command, and is executed even if the operation is canceled. If the value of the `repeat` attribute is not empty, the `exit-action` is performed after all iterations of the operation.

`help` A message to display if the user requests help on the operation.

`mnemonic`
: A one-character abbreviation for the operation's *name*.

`name` A one or two word name for the operation.

`repeat` A string to present before repeating the operation. If the value of this attribute is the empty string, the operation is performed only once. Otherwise, the string is presented, and the user is given the opportunity to repeat or cancel the operation.

`visible` A boolean indication of whether the operation will be made available. If the value is `${NO}`, the operation will appear in the parent menu but will not be available.

## text Class

Instances of the *text* class are simple text holders. *Text* objects may be added to *querygroups* with the `add` statement.

The following attributes are allowed for the *text* class:

| text Attribute Set | | |
|---|---|---|
| **Name** | **Type** | **Default** |
| value | *value* | "" |
| visible | *boolean* | "${YES}" |

The attributes have the following meanings:

`value` A text string to display.

`visible` A boolean indication of whether the text will be displayed.

## screen Class

Instances of the *screen* class are holders for *querygroups*. All of the *querygroups* of a certain *screen* are guaranteed to be evaluated at the same time and before the *querygroups* of any later *screens*. The interface driver may also display *screens* as separate windows. *Screens* may be added to *operations* with the `add` statement.

The following attributes are allowed for the *screen* class:

| screen Attribute Set | | |
|---|---|---|
| **Name** | **Type** | **Default** |
| entry-action | *command* | "" |
| exit-action | *command* | "" |
| title | *value* | "Untitled" |
| visible | *boolean* | "${YES}" |

The attributes have the following meanings:

`entry-action`
A shell command line to execute when entering the screen.

`exit-action`
A shell command line to execute when leaving the screen. This is executed after all queries for the screen are validated, and is executed even if the user terminates the screen.

`title` A string such as "Add a User" which is used as a title for the screen.

`visible` A boolean indication of whether the screen (and any querygroups below it) will be displayed. This attribute is evaluated after an operation is chosen, at the same time as all other screens for the operation, and before the `visible` attributes of the querygroups are evaluated.

**querygroup Class**

Instances of the *querygroup* class are used to group similar queries. The interface driver may use *querygroup* information to display related queries in a more attractive manner. *Querygroup*s may be added to *screen*s with the add statement.

The following attributes are allowed for the *querygroup* class:

| querygroup Attribute Set | | |
|---|---|---|
| **Name** | **Type** | **Default** |
| orientation | *direction* | "${HORIZONTAL}" |
| title | *value* | "" |
| visible | *boolean* | "${YES}" |

The attributes have the following meanings:

`orientation`
The preferred layout of queries within the querygroup. The value may be either `$VERTICAL` or `$HORIZONTAL`. The default is `$VERTICAL`. This attribute may be ignored by the display driver.

`title` A string describing the queries within the querygroup. This attribute may be ignored by the display driver.

`visible` A boolean indication of whether the querygroup (and any queries below it) will be displayed. This attribute is evaluated after a screen is entered, and is evaluated at the same time as the `visible` attributes of all other querygroups for the screen.

**Queries**

The following attributes are allowed for all query types: *textquery*, *boolquery*, *selectquery*, and *rangequery*:

| Query Attribute Set | | |
|---|---|---|
| **Name** | **Type** | **Default** |
| confirm | *value* | "" |
| confirm-value | *value* | "" |
| default | *value* | "" |
| help | *value* | "No help available." |
| preserve | *boolean* | "${NO}" |
| prompt | *value* | "" |
| variable | *value* | "" |

The attributes have the following meanings:

`confirm` The string to use as a confirmation prompt which must be answered by the user before execution continues. Confirmation is performed if the value entered for the query matches the *confirm-value*.

`confirm-value`
An `ed(1)`-style regular expression. If the value entered for a query matches *confirm-value*, confirmation of the value is sought (using the *confirm* string as the prompt).

`default` The default value of the *variable*.

`help` The text string to display if the user requests help on the query.

`preserve`
An indication of whether the value of *variable* should be saved in a global variable. If the value of this attribute is `${YES}`, the *variable*'s value (after being validated and confirmed) is saved in a global `idl` variable named *variable*. If the value of this attribute is `${NO}`, the *variable* is destroyed when the operation is complete.

`prompt` The text string to be displayed when the query is presented.

`variable`
The name of an `idl` variable that is set by the query. *variable*s may be referenced in other attribute strings by using the `$`*variable* notation.

## textquery Class

Instances of the *textquery* class describe how to retrieve an arbitrary text entry from the user. *Textqueries* may be added to *querygroup*s or to *screen*s with the `add` statement.

The following attributes are allowed for the *textquery* class:

| **textquery Attribute Set** | | |
|---|---|---|
| **Name** | **Type** | **Default** |
| confirm | *value* | "" |
| confirm-value | *value* | "" |
| default | *value* | "" |
| help | *value* | "No help available." |
| max-columns | *number* | "40" |
| max-lines | *number* | "1" |
| preserve | *boolean* | "${NO}" |
| prompt | *value* | "Enter text" |
| semantics | *command* | "" |
| semantics-message | *value* | "" |
| show-columns | *number* | "" |
| show-lines | *number* | "" |
| syntax | *command* | "" |
| syntax-message | *value* | "" |
| variable | *value* | "Text" |

The `confirm`, `confirm-value`, `default`, `help`, `preserve`, `prompt`, and `variable` attributes are generic Query Attributes. The other attributes have the following meanings:

`max-columns`
: The maximum number of columns of text accepted for the query.

`max-lines`
: The maximum number of lines of text accepted for the query.

`semantics`
: A command string to execute on the administered host to determine if the value entered for the query is semantically correct. The command must return zero if the value is correct, and return non-zero if the string is not correct. The command may be a builtin command.

`semantics-message`
: The custom error message to display if the semantics check fails. If the value of this attribute is empty, the error message is generated by `idi` from the prompt and the entered value.

`show-columns`
: The maximum number of columns to display at one time. The default value for this attribute is the value of *max-columns*. This attribute may be ignored by the display driver.

`show-lines`
: The maximum number of lines to display at one time. The default value for this attribute is the value of *max-lines*. This attribute may be ignored by the display driver.

`syntax`
: A command string to execute on the administering host to determine if the value entered for the query is syntactically correct. The command must return zero if the value is correct, and return non-zero if the string is not correct. The command may be a builtin command.

`syntax-message`
: The custom error message to display if the syntax check fails. If the value

of this attribute is empty, the error message is generated by `idi` from the prompt and the entered value.

**boolquery Class**

Instances of the *boolquery* class describe how to retrieve a positive or negative response from the user. *Boolqueries* may be added to *querygroups* with the `add` statement.

The following attributes are allowed for the *boolquery* class:

| **boolquery Attribute Set** | | |
|---|---|---|
| **Name** | **Type** | **Default** |
| confirm | *value* | "" |
| confirm-value | *value* | "" |
| default | *boolean* | "${YES}" |
| help | *value* | "No help available." |
| preserve | *boolean* | "${NO}" |
| prompt | *value* | "Enter yes or no" |
| variable | *value* | "Bool" |

The `confirm`, `confirm-value`, `default`, `help`, `preserve`, `prompt`, and `variable` attributes are generic Query Attributes.

**selectquery Class**

Instances of the *selectquery* class describe how to retrieve one or more choices from a list of choices. *Selectqueries* may be added to *querygroups* with the add statement.

The following attributes are allowed for the *selectquery* class:

| **selectquery Attribute Set** | | |
|---|---|---|
| **Name** | **Type** | **Default** |
| abort-message | *value* | "No possible values." |
| assign-values | *value-list* | "" |
| confirm | *value* | "" |
| confirm-value | *value* | "" |
| default | *value* | "" |
| exclusive | *boolean* | "${YES}" |
| help | *value* | "No help available." |
| number | *boolean* | "${YES}" |
| packed | *boolean* | "${YES}" |
| possible-values | *value-list* | "" |
| preserve | *boolean* | "${NO}" |
| prompt | *value* | "Enter selection" |
| variable | *value* | "Selection" |

The `confirm`, `confirm-value`, `default`, `help`, `preserve`, `prompt`, and `variable` attributes are generic Query Attributes. The other attributes have the following meanings:

`abort-message`

The message to display if an operation must be aborted because the value of *possible-values* for this query is empty.

`assign-values`
: A newline-separated list of values which may be assigned to the *variable* when the user selects one of the *possible-values*. This value of this attribute may be a backquoted string which is executed to dynamically produce the list described.

`exclusive`
: If the value of this attribute is `${YES}`, only one of the *possible-value*s for the query may be selected. If the value of this attribute is `${NO}`, more than one of the values may be selected.

`number`
: If the value of this attribute is `${YES}`, the *possible-values* of the query may be automatically numbered by the interface driver. If the value of this attribute is `${NO}`, the *possible-values* will not be numbered. This attribute should be set to `${NO}` when the *possible-values* are numbers so that there is no confusion between the *possible-values* and the automatically-generated numbers.

`packed`
: If the value of this attribute is `${YES}`, the interface driver may conserve screen space when presenting the query. If the value is `${NO}`, screen space may not be conserved.

`possible-values`
: A newline-separated list of choices for the query. The value of this attribute may be a backquoted string which is executed to produce the list of values.

## rangequery Class

Instances of the *rangequery* class describe how to retrieve a number within a given range from the user. *Rangequeries* may be added to *querygroup*s with the `add` statement.

The following attributes are allowed for the *rangequery* class:

| rangequery Attribute Set | | |
|---|---|---|
| **Name** | **Type** | **Default** |
| confirm | *value* | "" |
| confirm-value | *value* | "" |
| default | *value* | "0" |
| help | *value* | "No help available" |
| preserve | *boolean* | "${NO}" |
| prompt | *value* | "Enter value" |
| range | *number-list* | "0 1" |
| semantics | *command* | "" |
| semantics-message | *value* | "" |
| syntax | *command* | "" |
| syntax-message | *value* | "" |
| variable | *value* | "Range" |

The `confirm`, `confirm-value`, `default`, `help`, `preserve`, `prompt`, and `variable` attributes are generic Query Attributes. The other attributes have the following meanings:

`range` A whitespace-separated list of two numbers which are the minimum and maximum values for the query. The value of this attribute may be a backquoted string which is executed to produce the list of numbers.

`semantics`
A command string to execute on the administered host to determine if the value entered for the query is semantically correct. The command must return zero if the value is correct, and return non-zero if the string is not correct. The command may be a builtin command.

`semantics-message`
The custom error message to display if the semantics check fails. If the value of this attribute is empty, the error message is generated by `idi` from the prompt and the entered value.

`syntax` A command string to execute on the administering host to determine if the value entered for the query is syntactically correct. The command must return zero if the value is correct, and return non-zero if the string is not correct. The command may be a builtin command.

`syntax-message`
The custom error message to display if the syntax check fails. If the value of this attribute is empty, the error message is generated by `idi` from the prompt and the entered value.

## set Statement

The `set` statement causes the `idl` variable named *name* to take on the value *value*. The *value* is available globally for the duration of program.

## add Statement

The `add` statement causes the database object named *name1* to be added as a subobject of the database object named *name2*.

The following rules apply:

a. Both *names* must be defined previously.

b. Any number of *menus* or *operations* may be added to a *menu*.

c. Any number of *screens* may be added to an *operation*.

d. Any number of *querygroups* may be added to a *screen*.

e. Any number of queries ( *textquery*, *boolquery*, *selectquery*, or *rangequery* ) may be added to a *querygroup*.

f. An number of *texts* may be added to a *querygroup*.

g. At most one *textquery* may be added to a *selectquery*.

## export Statement

The `export` statement exports the `idl` variable named *name* (along with the variable's value) into the environment of all sub-shells. This is a function similar to the `export` command of the shell (`sh`(1)).

## Compiler Directives

The following compiler directives can be used to alter the behavior of the compiler or interpreter.

`%dir` *name*
: Interpret subsequent `%include` lines relative to *name*. Such a line overrides any previous `%dir` directive.

`%include` *name*
: Read the contents of the file *name* as if the contents were present in the current file.

`%print` [ `object` ]
: If *object* is given, print debugging information about *object*. Otherwise, print information about all objects.

## Variable Substitution

The `action`, `assign-values`, `confirm`, `default`, `help`, `possible-values`, `preserve`, `prompt`, `range`, `semantics`, and `syntax` attributes are processed so that `idl` variables may be used inside of the values for these attributes.

Variable expansion may be indicated by any of these forms:

`$`*var* or `${`*var*`}`
: If *var* is set, substitute the value of *var*. Otherwise, substitute an empty string.

`$#`*var* or `${#`*var*`}`
: Substitute the number of words found in the value of *var*. Words are separated by whitespace.

`${`*var*`:-`*val*`}`
: If *var* is set and non-null, substitute the value of *var*. Otherwise, substitute *val*.

`${`*var*`:+`*val*`}`
: If *var* is set and non-null, substitute *val*. Otherwise, substitute an empty string.

`${`*var*`:?`*val1*`:`*val2*`}`
: If *var* is set and non-null, substitute *val1*. Otherwise, substitute *val2*.

`${`*var*`:<`*prefix*`}`
: If *var* is set and non-null, substitute its value previxed by *prefix*.

Otherwise, substitute an empty string.

`${`*var*`:=`*text1*`:`*value1*`;`*text2*`:`*value2*`;`*textn*`:`*valuen*`}`
: Compare the value of *var* with each of the *text*s, and substitute the *value* associated with the matching *text*. As many text and value pairs as are required may be included. An empty *text* may be specified to indicate a default case. If *var* matches none of the *text*s, substitute an empty string.

If the colon (`:`) is omitted from the above expressions, `idi` only checks whether *var* is set or not.

In all cases, *var* must be a sequence of alphanumeric characters and underscores, optionally followed by an index specification of the form

*name*`[`*index*`]`

where the *index* is used to select only some of the words or lines from the value of *name*. If the *index* begins with =, the *index*-th line is substituted; otherwise, the *index*-th word is substituted. Words are separated by one or more whitespace characters. The *index* is subjected to variable substitution and may consist of a single number or two numbers separated by a -. The first word or line of a variable's value is numbered `1`. If the first number of a range is omitted, it defaults to `1`. If the last member of a range is omitted, it defaults to `$#name`. The index `*` selects all words or lines.

If a *val* or *prefix* contains any of colon (`:`), semi-colon (`;`), or right brace (`}`), the character must be preceeded by a backslash (\) to escape its special meaning.

Any variables found within double quotes (") are expanded. All characters between back quotes (‘) are expanded and passed to the shell (`sh`(1)) for execution, and the result of the shell execution is inserted in place of the back-quoted string. A backslash (\) preceding either $ or ‘ causes the character to lose its special meaning.

The *value* or *text* part of any of the above expressions may contain other variable references.

## Pre-defined Variables

The following variables are used internally by `idi`(1) and should not be changed. These variables should be used in place of the strings they represent (for example, always use "${YES}" instead of "yes").

`YES` This is defined to be the affirmative string, `yes`.

`NO` This is defined to be the negative string, `no`.

`HORIZONTAL`
: This is defined to be `horizontal`. This may be used as the value for the `orientation` attribute of querygroups.

`VERTICAL`
: This is defined to be `vertical`. This may be used as the value for the `orientation` attribute of querygroups.

`NO_DEFAULT`
: This is defined to be `[ No default ]`. This may be used as the value for the `default` attribute of selectqueries. When this is used, the interface driver will leave the `default` for the selectquery empty if possible.

`SKILL_LEVELS`
: This is defined to be the list of possible skill levels: `Novice`

`Intermediate Expert`. Note that this variable's value varies based on the current locale.

The following global variables are set by `idi` at run-time:

`InterfaceName`
: The name of the chosen interface. This will be either `ascii` or `motif`. This is the only means for changing the behavior of the program based on the chosen interface.

`OperationName`
: The value of the `name` attribute of the current operation. This may be used to generalize query prompts:

```
prompt = "Host Name to ${OperationName}"
```

`SkillLevel`
: The chosen level of expertise. This will be one of the values from the ${SKILL_LEVELS} variable. This variable may be set in an `idl` file to control the behavior of the interface driver.

## Builtin Commands

Several builtin commands are provided for use in values for the `action`, `semantics`, and `syntax` attributes. The builtin commands are the following:

`:Confirm` *confirmation-string*
: Present the *confirmation-string* to the user using the appropriate interface driver. Return zero if the string is confirmed; return non-zero if it is not confirmed.

`:DoOp` *operation-name* [ *confirmation-string* ]
: Perform the *operation-name* operation. If the *confirmation-string* is used, ask for confirmation before the operation is performed. If the confirmation fails, exit with status 0; otherwise, exit with the exit status of the operation.

`:Echo` *message*
: Echo the *message* to the display.

`:Error` *message*
: Display the error *message* in a way appropriate for the interface driver.

`:Help` *help-message*
: Present a *help-message* to the user.

`:Log` *message*
: Append the *message* to the log file. The *message* is written regardless of the verbosity level chosen by the user.

`:Match` *regexp string*
: Return zero if the *string* matches the given `egrep(1)`-style regular expression, *regexp*; otherwise, return non-zero. This command is useful in the `syntax` attribute of queries.

`:Numeric` *lower-bound upper-bound value*
: Return zero if the integer *value* given is within the range specified by *lower-bound* and *upper-bound*. This command is useful in the `syntax` attribute of queries.

`:Quit` *exit-code*
: Terminate the program with *exit-code* as the status code.

`:Restart`
: Restart the interface driver. This takes into account new or changed description files.

`:Run` *command*
: Execute an interactive *command* on the host system. The standard input, output, and error file descriptors are set appropriately.

`:Set` *variable value*
: Set the global *variable* to *value*. The *variable* is then available for use by other queries. The *variable* is created if it does not exist, or modified if it does exist.

`:Show` Dump the values of all variables to stdout. This is useful for debugging.

`:Unimp` *message*
: Display a *message* indicating that some feature is unimplemented. *message* should describe the feature not implemented.

`:Unset` *variable*
: Remove the global *variable* and its value. This command should only be used for *variable*s which are set using the `:Set` builtin command.

`:Warning` *message*
: Display the warning *message* in a way appropriate for the interface driver.

## EXAMPLES

Below is a sample `idl` file which creates a single menu with several operations which could be used to manage the /etc/ethers database file.

```
      ##################################################################
      #
      #  Some patterns used here
      #
      ##################################################################

      set STD_HOST_NAME_PATTERN   = "^[a-zA-Z][-.a-zA-Z0-9]*\$"

      set STD_HOST_NAME_HELP =
      "Enter an Internet host name.  A host name may contain the characters:
                  a-z  A-Z  0-9  .  -
      It should begin with a letter (a-z or A-Z) and be no more
      than 32 characters in length.  It should not contain a . or -
      as the last character."

set STD_ETHER_ADDRESS_PATTERN                    =
"^[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]

      set STD_ETHER_ADDRESS_HELP =
      "Enter an Ethernet address.  An Ethernet address has the form:
                  aa:bb:cc:dd:ee:ff
      where a, b, c, d, e, f are two-digit hexadecimal numbers 00 and ff.
      The numbers are separated by colons.  You must enter all 17 characters."
```

```
set dg_EthersFile = "/etc/ethers"

##################################################################
#
#  Main menu
#
##################################################################

menu main
    name = "Main"
    title = "Main Menu"
    description = "Top level menu"
    help =
"This is the first level menu.  It contains a sub-menu for
manipulating the ethers database."
end

##################################################################
#
#  Ether menu
#
##################################################################

menu dg_Ether
    name = "Ether"
    mnemonic = E
    title = "Ethers Menu"
    description = "Manipulate the ethers databases"
    help =
"This menu provides access to the ethers databases.  There are
operations for adding, deleting, modifying, and listing entries
from the database."
end

##################################################################
#
#  Operations
#
##################################################################

operation dg_EtherAdd
    name = Add
    mnemonic = A
    action = "admether -o add -a ${NetAddress}
    description = "Add an entry to the ethers database"
    help =
"The Add operation takes a host name and an Ethernet address and adds
an entry to the ethers database."
    exit-action = ":Unset DefaultString"
end

operation dg_EtherDelete
    name = Delete
    mnemonic = D
    action = "admether -odelete
    description = "Delete entry from the ethers database"
    confirm = "Delete ${HostName} from the ethers database?"
    help =
"The Delete operation takes one or more host names and
deletes the corresponding entry or entries from the
ethers database."
end

operation dg_EtherModify
    name = Modify
    mnemonic = M
    action =
"admether -o modify -n ${NewHostName} -a ${NetAddress}
    description = "Modify an entry in the ethers database"
    help =
"The Modify operation takes a host name and allows the user to modify
the corresponding entry in the ethers file.
The user may modify the host name and the Ethernet address."
    exit-action = ":Unset DefaultString"
end

operation dg_EtherList
    name = List
    mnemonic = L
    action = "admether -o list"
    description = "List entries from the ethers database"
    help =
"The List operation displays the contents of the ethers database
for one or more hosts."
end

##################################################################
#
#  Screens, querygroups, and queries
#
##################################################################

screen dg_AddEtherScreen
    title = "Add an Ethers Entry"
    entry-action = ":Set DefaultString 00:00:00:00:00:00 NewName"
end

#
#  This querygroup and its queries are used for entering a
#  new ether entry.  The defaults are stored in the DefaultString
#  variable, and should be set by the screen.
#

querygroup dg_NewEtherEntryQG
end
```

```
      textquery dg_HostNameText
          prompt = "Host Name"
          variable = HostName
          syntax = ":Match ${STD_HOST_NAME_PATTERN} ${HostName}"
          help = "${STD_HOST_NAME_HELP}

      This is the name of the host as it should appear in the
      ethers database."
          #
          #  Do different checks based on whether we're adding or
          #  listing.
          #
semantics = "${OperationName=Add:test -z '`grep ${HostName} ${dg_EthersFile}`';\
:test -n '`grep ${HostName} ${dg_EthersFile}`'}"
          default = "${DefaultString[2]}"
      end

      textquery dg_EthernetText
          prompt = "Ethernet address"
          variable = NetAddress
          syntax = ":Match ${STD_ETHER_ADDRESS_PATTERN} ${NetAddress}"
          help = "${STD_ETHER_ADDRESS_HELP}

      This is the Ethernet address of the host as it should appear
      in the ethers database."
          default = "${DefaultString[1]}"
      end

      #
      #  This screen, querygroup, and query are shared between Delete
      #  and List, because both operations need to choose one or more
      #  existing host names.
      #

      screen dg_HostNameListScreen
          title = "${OperationName} Ethers Entry(ies)"
      end

      querygroup dg_HostNameListQG
      end

      selectquery dg_HostName
          prompt = "Host Name(s)"
          possible-values = "all
      `admether -o list -q | cut -f2 -d' '`"
          exclusive = "$NO"
          variable = HostName
          default = "${NO_DEFAULT}"
          help = "
      This is the name of the host(s) to ${OperationName}."
      end

      #
```

```
#  This screen and its queries are used for getting a single
#  existing entry which will be modified.
#

screen dg_ModifyEtherScreen1
    title = "Modify an Ethers Entry"
end

querygroup dg_ModifyEtherQG1
end

screen dg_ModifyEtherScreen2
    title = "Modify an Ethers Entry"
    entry-action = ":Set DefaultString `admether -o list -q ${HostName}`"
end

selectquery dg_OldHostName
    prompt = "Old Host Name"
    possible-values = "`admether -o list -q | cut -f2 -d' '`"
    exclusive = "$YES"
    variable = HostName
    help = "
This is the name of the host whose database entry is to
be modified."
end

add dg_Ether to main
add dg_EtherAdd to dg_Ether
    add dg_AddEtherScreen to dg_EtherAdd
                add dg_NewEtherEntryQG to dg_AddEtherScreen
                    add dg_HostNameText to dg_NewEtherEntryQG
                    add dg_EthernetText to dg_NewEtherEntryQG

add dg_EtherDelete to dg_Ether
    add dg_HostNameListScreen to dg_EtherDelete
                add dg_HostNameListQG to dg_HostNameListScreen
                    add dg_HostName to dg_HostNameListQG

add dg_EtherModify to dg_Ether
    add dg_ModifyEtherScreen1 to dg_EtherModify
                add dg_ModifyEtherQG1 to dg_ModifyEtherScreen1
                    add dg_OldHostName to dg_ModifyEtherQG1

    add dg_ModifyEtherScreen2 to dg_EtherModify
                add dg_NewEtherEntryQG to dg_ModifyEtherScreen2

add dg_EtherList to dg_Ether
    add dg_HostNameListScreen to dg_EtherList
```

**SEE ALSO**

ed(1), egrep(1), idi(1), idc(1), sh(1).

## NAME

`inittab` – script for `init`

## DESCRIPTION

The file `/etc/inittab` controls process dispatching by `init`. The processes most typically dispatched by `init` are servers.

The `inittab` file is composed of entries that are position dependent and have the following format:

*id* : *rstate* : *action* : *process*

Each entry is delimited by a newline, however, a backslash (\) preceding a newline indicates a continuation of the entry. Up to 512 characters per entry are permitted. Comments may be inserted in the *process* field using the convention for comments described in `sh`(1). There are no limits (other than maximum entry size) imposed on the number of entries in the `inittab` file. The entry fields are:

*id* This is one or two characters used to uniquely identify an entry.

*rstate* This defines the run level in which this entry is to be processed. Run-levels effectively correspond to a configuration of processes in the system. That is, each process spawned by `init` is assigned a run level or run levels in which it is allowed to exist. The run levels are represented by a number ranging from `0` through `6`. As an example, if the system is in run level `1`, only those entries having a `1` in the *rstate* field are processed. When `init` is requested to change run levels, all processes that do not have an entry in the *rstate* field for the target run level are sent the warning signal `SIGTERM` and allowed a 5-second grace period before being forcibly terminated by the kill signal `SIGKILL`. The *rstate* field can define multiple run levels for a process by selecting more than one run level in any combination from `0` through `6`. If no run level is specified, then the process is assumed to be valid at all run levels `0` through `6`. There are three other values, `a`, `b` and `c`, which can appear in the *rstate* field, even though they are not true run levels. Entries which have these characters in the *rstate* field are processed only when an `init` or `telinit` process requests them to be run (regardless of the current run level of the system). See `init`(1M). They differ from run levels in that `init` can never enter run level `a`, `b` or `c`. Also, a request for the execution of any of these processes does not change the current run level. Furthermore, a process started by an `a`, `b` or `c` command is not killed when `init` changes levels. They are killed only if their line in `inittab` is marked `off` in the *action* field, their line is deleted entirely from `inittab`, or `init` goes into single-user state.

*action* Key words in this field tell `init` how to treat the process specified in the *process* field. The actions recognized by `init` are as follows:

`respawn` If the process does not exist, then start the process; do not wait for its termination (continue scanning the `inittab` file), and when the process dies, restart the process. If the process currently exists, do nothing and continue scanning the `inittab` file.

`wait` When `init` enters the run level that matches the entry's *rstate*, start the process and wait for its termination. All subsequent reads of the `inittab` file while `init` is in the same run level cause `init` to ignore this entry.

| | |
|---|---|
| `once` | When `init` enters a run level that matches the entry's *rstate*, start the process, do not wait for its termination. When it dies, do not restart the process. If `init` enters a new run level and the process is still running from a previous run level change, the program is not restarted. |
| `boot` | The entry is to be processed only at `init`'s boot-time read of the `inittab` file. `init` is to start the process, not wait for its termination; and when it dies, not restart the process. In order for this instruction to be meaningful, the *rstate* should be the default or it must match `init`'s run level at boot time. This action is useful for an initialization function following a hardware reboot of the system. |
| `bootwait` | The entry is to be processed the first time `init` goes from single-user to multi-user state after the system is booted. (If `initdefault` is set to `2`, the process runs right after the boot.) `init` starts the process, waits for its termination and, when it dies, does not restart the process. |
| `powerfail` | Execute the process associated with this entry only when `init` receives a power fail signal, `SIGPWR` [see `signal`(2)]. |
| `powerwait` | Execute the process associated with this entry only when `init` receives a power fail signal, `SIGPWR`, and wait until it terminates before continuing any processing of `inittab`. |
| `off` | If the process associated with this entry is currently running, send the warning signal `SIGTERM` and wait 5 seconds before forcibly terminating the process with the kill signal `SIGKILL`. If the process is nonexistent, ignore the entry. |
| `ondemand` | This instruction is really a synonym for the `respawn` action. It is functionally identical to `respawn` but is given a different keyword in order to divorce its association with run levels. This instruction is used only with the `a`, `b` or `c` values described in the *rstate* field. |
| `initdefault` | An entry with this action is scanned only when `init` is initially invoked. `init` uses this entry, if it exists, to determine which run level to enter initially. It does this by taking the highest run level specified in the *rstate* field and using that as its initial state. If the *rstate* field is empty, this is interpreted as `0123456` and `init` therefore enters run level `6`. This will cause the system to loop, that is, it will go to firmware and reboot continuously. Additionally, if `init` does not find an `initdefault` entry in `inittab`, it requests an initial run level from the user at reboot time. |
| `sysinit` | Entries of this type are executed before `init` tries to access the console (i.e., before the `Console Login:` prompt). It is expected that this entry will be only used to initialize devices on which `init` might try to ask the run level question. These entries are executed and waited for before continuing. |

*process* This is a command to be executed. The entire `process` field is prefixed with `exec` and passed to a forked `sh` as `sh -c 'exec` *command*`'`. For this reason, any legal `sh` syntax can appear in the *process* field.

**SEE ALSO**

`init`(1M), `ttymon`(1M), `exec`(2), `open`(2), `signal`(2)

`sh`(1), `who`(1) in the *User's Reference Manual*

## NAME

`inode` – file node structure

## SYNOPSIS

```
#include <ufs/disk_format.h>
```

## DESCRIPTION

The inode table for a file system is distributed across the disk: a table exists in each disk allocation region (DAR). For more information about the file system layout, refer to `fs`(4).

The file node's purpose is to provide access to data blocks associated with the file. The data blocks are allocated in chunks of contiguous physical blocks called data elements. In the case that the file is less than the data element size, the file is fragmented. In this case, the file has only one data element and its size is determined by the fragment exponent field. If the file grows, the fragmented data element is copied to a full sized element, and the allocation to the file will always be in data element sized chunks, causing the actual size of the file to be less than or equal to the blocks allocated to it.

Data elements are accessed directly or indirectly depending on the size of the file. The file node has an array of direct data elements, pointing to the first block of the data element. If the size of the file is greater than the number of direct data element pointers, then indirect access is used.

Indirect data element access is provided through indexing. An index structure consists of index blocks containing pointers to data elements. Depending on the depth of the index structure, index entries point to data elements or other index blocks. There are three index structures rooted in the file node; each of the three differs in the levels of indexing. If the file node represents a directory, only the first index level is used.

In the case of the first index structure, the pointer in the file node points to the first block containing the index entries (an index may span blocks); the entries at this level point to data elements. The second index structure points to the first block containing index entries. Each index entry at this level points to the first block of an index containing the same number of entries as the previous level. These index entries contain pointers to data elements. The third index structure is similar to the previous two but has another level of indexing before the index containing the data element pointers.

This expansion of index levels produces a tree, where the leaves of the tree are data elements. The number at each level multiplies itself by the number of index entries.

To access a data block, it must be determined if it is accessible directly or through indexing. If direct access is possible, the data element needs to be determined along with the particular block within the data element. If the block is deep enough in the file to require indexing, the level of indexing must be determined by finding what range of blocks each index covers. After the index structure is determined, the path of entries through the index structure is required.

The inode table in the DAR is made up of entries of the following structure:

```
typedef struct
    {
    boolean_field_type           is_allocated                 : 1;
    boolean_field_type           is_fragmented                : 1;
    field_type                   fragment_size_exponent       : 3;
    field_type                   des_exponent                 : 5;
    field_type                   ies_exponent                 : 4;
    field_type                   pad_to_double_word           : 9;
    field_type                   partial_block_byte_count     : 9;
    uint32e_type                 whole_block_count;
    uint32e_type                 generation_number;
    uint32e_type                 dar_index;
    df_file_node_number_type     space_parent;
    uint32e_type                 maximum_space_usage;
    uint32e_type                 current_space_usage;
    uint32e_type                 maximum_file_node_usage;
    uint32e_type                 current_file_node_usage;
    df_file_mode_type            mode;
    uint16e_type                 user_id;
    uint16e_type                 group_id;
    int16e_type                    link_count;
    df_time_type                 time_last_accessed;
    df_time_type                 time_last_modified;
    df_time_type                 time_attributes_last_changed;
    union
        {
        struct
            {
            uint32e_type           data[DF_DIRECT_ELEMENT_COUNT];
            union
                {
                struct
                  {
                  uint32e_type index_array[DF_MAX_DIR_INDEX_LEVEL];
                  df_din_type  din;
                  } directory;
                struct
                  {
                  uint32_type   index_array[DF_MAX_INDEX_LEVEL];
                  } regular;
                } index;
            } element_addresses;
        struct
            {
            uint16e_type   major_device_number;
            uint16e_type   minor_device_number;
            byte8e_type    pad_to_union_size[48];
            } represented_device;
        } contents;
        byte8e_type reserved[DF_RESERVED_BYTES_PER_FILE_NODE];
    } df_file_node_type;
```

*is_allocated* indicates whether this is a free file node or not. If FALSE it is a free file node; if TRUE, then this is a valid file node.

*is_fragmented* is TRUE when the first (and only) element of the file is reduced in size from the data element size to the fragment size specified by fragment_size_exponent; otherwise, all data elements (if any) are the full data element size and fragment_size_exponent is invalid.

*fragment_size_exponent* specifies, when valid, the size of the fragmented data element which contains the file's data. The size in blocks of the fragment is 2 raised to the fragment_size_exponent power. It must be large enough to fit the total size of the file in the fragment. Because all fragments must fit into a single file system buffer, the maximum fragment size is:

```
#define DF_MAX_FRAGMENT_SIZE      16
```

blocks, although the fragment_size_exponent field is large enough to support fragment sizes up to 128 (2 ^ 7) blocks.

*des_exponent* specifies the data element size. The data element size in blocks is 2 raised to the des_exponent power. The maximum data element size is therefore 2 ^ 31 blocks. The maximum value for this field is:

```
#define DF_MAX_DES_EXPONENT 31
```

although it is also limited to the base 2 logarithm of the largest power of 2 that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

*ies_exponent* specifies the index element size. The index element size in blocks is 2 raised to the ies_exponent power. The maximum index element size is therefore 2 ^ 15 blocks. The maximum value for this field is:

```
#define DF_MAX_IES_EXPONENT 15
```

although it is also limited to the base 2 logarithm of the largest power of 2 that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

*partial_block_byte_count* is the count of the number of bytes to the end of file following the last whole block. All possible values, i.e., 0 to 511, are legal.

*whole_block_count* is the number of 512 byte blocks logically in the file before EOF. The file size as reported by `stat`(2) is:

```
((whole_block_count * 512) + partial_block_byte_count).
```

*generation_number* is incremented each time an inode is freed and is kept valid on free nodes so that subsequent uses of the same file node number are guaranteed to have different UFID values.

*dar_index* is the current allocation hint (index of a DAR to use for data and file node allocation). DAR indexes are zero based.

*space_parent* is the parent file node number. In the file node for the root of the filesystem, the value of space_parent is:

```
   #define DF_ROOT_FILE_NODE_NUMBER 2
```

therefore, the filesystem root is its own space parent.

*maximum_space_usage* is the maximum usage limit in blocks for the file plus all its space descendants. It must be set to UINT32_MAX for non-CPD directories and other non-directory files, as well as for CPD's which have no allocation limit. On the root of each filesystem, this limit is not applied to the superuser.

*current_space_usage* is the current usage in blocks for the file plus all its space descendants, if any. If not a CPD, then it is the number of blocks actually used to store the file's contents on disk, including both index and data elements. For a CPD, it is that plus the current_space_allocation fields of all files which name this CPD as their space parent.

*maximum_file_node_usage* is the maximum file node usage limit for the file plus all its space descendants. Must be UINT32_MAX for non-CPD directories and other non-directory files, as well as for CPDs with no file node allocation limit. On the root of each filesystem, this limit is not applied to the superuser. On all other CPD's it is applied equally to all users.

*current_file_node_usage* is the current file node usage count for the file plus all its space descendants. It must be 1 for non-CPD directories and other non-directory files. For a CPD, it is 1 plus the current_file_node_usage fields of all files which name this CPD as their space parent.

*mode* is the file's mode. See `stat`(2).

*user_id* is user id of the file.

*group_id* is the group id of the file.

*link_count* is the number of links (directory entries) to the file. Must be greater than zero.

*time_last_accessed* is the time the file's contents were last accessed (i.e., read or executed).

*time_last_modified* is the time the file's contents were last modified (i.e., written or truncated).

*time_attributes_last_changed* is the time one of the file's attributes (mode, user_id, group_id, link_count, child_count, etc.) was last changed.

*contents* is a union containing represented_device for block-special or character-special files, and containing element_addresses for all other file types.

*represented_device* is the device numbers of the device represented by a character or block special file. The padding bytes (pad_to_union_size) must be set to zero.

*element_addresses* are the disk addresses of the data elements and index elements of the file. The "data" field contains the addresses of the first:

```
#define DF_DIRECT_ELEMENT_COUNT       10
```

data elements in the file. The "index" field contains the addresses of the first index element of each level for regular files. For directory files, we only have 1 level of indexing, with the other two index fields being used to store the directory manager information.

Since all the file nodes in a DAR are not necessarily allocated, a list of free file nodes must be maintained. The head of the list is contained in each DAR entry. The DAR entry contains the file node number of a file node in the DAR, that file node should be unallocated and the following structure contains the fields for a free file node:

```
typedef struct
    {
    boolean_field_type        is_allocated : 1;
    df_file_node_number_type next_free_file_node_number;
    uint32e_type              generation_number;
    byte8e_type  pad_to_file_node_size[DF_FREE_FILE_NODE_PADDING];
    } df_free_file_node_type;
```

*is_allocated* is TRUE when this is a valid file_node. If FALSE, then this is a free file_node.

*generation_number* is kept valid on free nodes so that subsequent uses of the same file node number are guaranteed to have different UFID values.

*next_free_file_node_number* is the file node number of ne.: free file_node on the DAR free file_node list.

## SEE ALSO

stat(2), dg_stat(2), fs(4); fsck(1M), mkfs(1M) in the *System Manager's Reference for the DG/UX System*.

## NAME

`issue` – issue identification file

## DESCRIPTION

The file `/etc/issue` contains the *issue* or project identification to be printed as part of the login prompt. This is an ASCII file containing any text you choose and is read by program `getty` and then written to any terminal spawned or respawned from the `inittab`(4) file.

## FILES

`/etc/issue`

## SEE ALSO

`gettydefs`(4)
`login`(1) in the *User's Reference for the DG/UX System*.

## NAME

ldfcn – COFF executable file access routines

## SYNOPSIS

```
#include <stdio.h>
#include <sys/types.h>
#include <filehdr.h>
#include <ldfcn.h>
```

## DESCRIPTION

The executable file access routines are a collection of functions for reading a COFF executable file that is in DG/UX executable file format. Although the calling program must know the detailed structure of the parts of the executable file that it processes, the routines effectively insulate the calling program from knowledge of the overall structure of the executable file.

The interface between the calling program and the executable file access routines is based on `LDFILE` defined as `struct ldfile`, declared in the header file `ldfcn.h`. This structure provides uniform access to simple executable files and to executable files that are members of an archive file.

The function `ldopen`(3X) allocates and initializes the `LDFILE` structure and returns a pointer to the structure to the calling program. The fields of the `LDFILE` structure may be accessed individually through macros defined in `ldfcn.h` and contain the following information:

| | |
|---|---|
| `LDFILE` | `*ldptr;` |
| `TYPE(ldptr)` | The file magic number, used to distinguish between archive members and simple executable files. |
| `IOPTR(ldptr)` | The file pointer returned by `fopen`(3S) and used by the standard input/output functions. |
| `OFFSET(ldptr)` | The file address of the beginning of the executable file; the offset is non-zero if the executable file is a member of an archive file. |
| `HEADER(ldptr)` | The file header structure of the executable file. |

The executable file access functions may be divided into four categories:

(1) Functions that open or close an executable file

`ldopen`(3X) and `ldaopen`(3X) open an executable file
`ldclose`(3X) and `ldaclose`(3X) close an executable file

(2) Functions that read header or symbol table information.

`ldahread`(3X) reads the archive header of a member of an archive file
`ldfhread`(3X) reads the file header of an executable file
`ldshread`(3X) reads a section header of an executable file
`ldsyshread`(3X) reads the system header of an executable file
`ldtbread`(3X) reads a symbol table entry of an executable file
`ldgetname`(3X) retrieves a symbol name from a symbol table entry.

(3) Functions that position an executable file at (seek to) the start of a particular section.

`Ldohseek`(3X) seeks to the system header of an executable file
`ldsseek`(3X) seeks to a section of an executable file
`ldtbseek`(3X) seeks to the symbol table of an executable file

(4) The function `ldtbindex`(3X) returns the index of a particular executable file symbol table entry.

These functions are described in detail on their respective manual pages.

All the functions except `ldaopen`(3X), `ldgetname`(3X), `ldopen`(3X), and `ldtbindex`(3X) return either `SUCCESS` or `FAILURE`, both constants defined in `ldfcn.h`. `Ldaopen`(3X) and `ldopen`(3X) both return pointers to an `LDFILE` structure.

Additional access to an executable file is provided through a set of macros defined in `ldfcn.h`. These macros parallel the standard input/output file reading and manipulating functions, translating a reference of the `LDFILE` structure into a reference to its file descriptor field.

The following macros are provided:

```
GETC(ldptr)
FGETC(ldptr)
GETW(ldptr)
UNGETC(c, ldptr)
FGETS(s, n, ldptr)
FREAD(ptr, sizeof (*ptr), nitems, ldptr)
FSEEK(ldptr, offset, ptrname)
FTELL(ldptr)
REWIND(ldptr)
FEOF(ldptr)
FERROR(ldptr)
FILENO(ldptr)
SETBUF(ldptr, buf)
```

See the manual entries for the corresponding standard input/output library functions for details on these macros.

The program must be loaded with the executable file access routine library `libld.a`.

## SEE ALSO

`fseek`(3S), `ldahread`(3X), `ldclose`(3X), `ldfhread`(3X), `ldgetname`(3X), `ldohseek`(3X), `ldopen`(3X), `ldshread`(3X), `ldsseek`(3X), `ldtbindex`(3X), `ldtbread`(3X), `ldtbseek`(3X), `intro`(5).

## NOTES

The executable file format is used only for executable files (load modules), not for object files.

`limits` – header file for implementation-specific constants

## SYNOPSIS

`#include <limits.h>`

## DESCRIPTION

The header file `limits.h` is a list of minimal magnitude limitations imposed by a specific implementation of the operating system.

```
ARG_MAX     5120                    /* max length of arguments to exec */
CHAR_BIT    8                       /* max # of bits in a "char" */
CHAR_MAX    255                     /* max value of a "char" */
```

```
CHAR_MIN    0                         /* min value of a "char" */
CHILD_MAX   25                        /* max # of processes per user id */
EDMC??
CLK_TCK     _sysconf(3)               /* clock ticks per second */
DBL_DIG     15                        /* digits of precision of a "double" */
DBL_MAX     1.79769313486223179E+308/* max decimal value of a "double"*/
DBL_MIN     2.2250738585071991E-308 /* min decimal value of a "double"*/
FCHR_MAX    2147483647                /* max size of a file in bytes */
FLT_DIG     6                         /* digits of precision of a "float" */
FLT_MAX     3.40282347E+38F           /* max decimal value of a "float" */
FLT_MIN     1.17549435E-38F           /* min decimal value of a "float" */
HUGE_VAL    7.237005145973118E-75     /* error value returned by Math lib */
INT_MAX     2147483647                /* max value of an "int" */
INT_MIN     (-2147483647-1)           /* min value of an "int" */
LINK_MAX    1000                      /* max # of links to a single file */
LOGNAME_MAX8                          /* max # of characters in a login name */
LONG_BIT    32                        /* # of bits in a "long" */
LONG_MAX    2147483647                /* max value of a "long int" */
LONG_MIN    (-2147483647-1)           /* min value of a "long int" */
MAX_CANON   255                       /* max bytes in a line for canonical
                                      processing */
MAX_INPUT   512                       /* max size of a char input buffer */
MB_LEN_MAX 5                          /* max # of bytes in a multibyte
                                      character */
NAME_MAX    14                        /* max # of characters in a file name */
NGROUPS_MAX16                         /* max # of groups for a user */
NL_ARGMAX   9                         /* max value of "digit" in calls to the
                                      NLS printf() and scanf() */
NL_LANGMAX 14                         /* max # of bytes in a LANG name */
NL_MSGMAX   32767                     /* max message number */
NL_NMAX     1                         /* max # of bytes in N-to-1 mapping
                                      characters */
NL_SETMAX   255                       /* max set number */
NL_TEXTMAX 255                        /* max # of bytes in a message string */
NZERO       20                        /* default process priority */
OPEN_MAX    64                        /* max # of files a process can have
                                      open */
PASS_MAX    8                         /* max # of characters in a password */
PATH_MAX    1023                      /* max # of characters in a path name */
PID_MAX     30000                     /* max value for a process ID */
PIPE_BUF    8192                      /* max # bytes atomic in write to a pipe */
PIPE_MAX    8192                      /* max # bytes written to a pipe
                                      in a write */
SCHAR_MAX   127                       /* max value of a "signed char" */
SCHAR_MIN   (-128)                    /* min value of a "signed char" */
SHRT_MAX    32767                     /* max value of a "short int" */
SHRT_MIN    (-32768)                  /* min value of a "short int" */
STD_BLK     512                       /* # bytes in a physical I/O block */
SYS_NMLN    256                       /* 4.0 size of utsname elements */
```

```
                                           /* also defined in sys/utsname.h */
SYSPID_MAX 1                               /* max pid of system processes */
TMP_MAX    17576                           /* max # of unique names generated
                                           by tmpnam */
UCHAR_MAX  255                             /* max value of an "unsigned char" */
UID_MAX    60000                           /* max value for a user or group ID */
UINT_MAX   4294967295                      /* max value of an "unsigned int" */
ULONG_MAX  4294967295                      /* max value of an "unsigned long int" */
USHRT_MAX  65535                           /* max value of an "unsigned short int" */
USI_MAX    4294967295                      /* max decimal value of an "unsigned" */
WORD_BIT   32                              /* # of bits in a "word" or "int" */
```

The following POSIX definitions are the most restrictive values to be used by a POSIX conformant application. Conforming implementations shall provide values at least this large.

```
_POSIX_ARG_MAX          4096     /* max length of arguments to exec */
_POSIX_CHILD_MAX        6        /* max # of processes per user ID */
_POSIX_LINK_MAX         8        /* max # of links to a single file */
_POSIX_MAX_CANON        255      /* max # of bytes in a line of input */
_POSIX_MAX_INPUT        255      /* max # of bytes in terminal
                                 input queue */
_POSIX_NAME_MAX         14       /* # of bytes in a filename */
_POSIX_NGROUPS_MAX       0       /* max # of groups in a process */
_POSIX_OPEN_MAX         16       /* max # of files a process can have open */
_POSIX_PATH_MAX         255      /* max # of characters in a pathname */
_POSIX_PIPE_BUF         512      /* max # of bytes atomic in write
                                 to a pipe */
```

## SEE ALSO

passwd(4).

## NAME

linenum – line number entries in a common object file

## SYNOPSIS

```
#include  <linenum.h>
```

## DESCRIPTION

When invoked with the -g option, the cc command generates an entry in the object file for each C source line on which a breakpoint is possible. debuggers such as sdb(1) and dbx(1) can then reference line numbers in the source. The structure of the line number entries appears below.

```
struct  lineno
{
        union
        {
                long     l_symndx ;
                long     l_paddr ;
        }                l_addr ;
        union
        {
                struct
                {
                         unsigned short _l_lnno;
                         unsigned short _l_pad;
                }        _l_l;
                long     _l_lnno;
        }                _l;
};
```

Numbering starts with 1 for each function. The initial line number entry for a function has *l_lnno* equal to zero, and the symbol table index of the function's entry is in *l_symndx*. Otherwise, *l_lnno* is non-zero, and *l_paddr* is the physical address of the code for the referenced line. Thus the overall structure is the following:

| *l_addr* | *l_lnno* |
|---|---|
| function symtab index | 0 |
| physical address | line |
| physical address | line |
| ... | |
| function symtab index | 0 |
| physical address | line |
| physical address | line |
| ... | |

## SEE ALSO

cc(1), sdb(1), dbx(1), a.out(4).

## NAME

`master` – format of a master file

## DESCRIPTION

Information about configurable kernel components is contained in a set of *master files* that are kept in the *master file directory* (by default, `/usr/etc/master.d`). This information is used by the `config`(1M) program to configure a kernel image. There are four types of configurable kernel components: device drivers, socket protocols, STREAMS modules, and tunable parameters.

Each layered kernel product available on the system has its own master file in the master file directory. For example, the TCP/IP product includes the master file `/usr/etc/master.d/tcpip`. The base DG/UX System itself uses `/usr/etc/master.d/dgux` as its master file. If you create your own device drivers or other configurable kernel compenents, you will need to create a new master file to supply information about the new components. Remember that every file found in the master file directory is examined when `config`(1M) is run, so backup or duplicate copies of master files should not be stored there, since they will cause errors when components are defined in more than one place. If you are not adding a new configurable component, you will probably only use the master files as reference when setting up your *system file* (see `system`(4)).

A *master file* can contain entries describing device drivers, socket protocols, STREAMS modules, tunable parameters, and aliases. Different types of information are grouped into their own sections with their own entry format. Each section is prefaced by a line containing a section name, whose first character is the dollar sign ($). A master file may have any number (including zero) of each type of section, and they may appear in any order. Six different types of sections are supported:

| | |
|---|---|
| `$device` | Describes drivers for hardware devices and pseudo-devices. |
| `$protocol` | Describes protocols that can be supported by the `socket`(2) system call. |
| `$stream` | Describes STREAMS modules. |
| `$keyword` | Describes user-tunable system parameters. |
| `$alias` | Defines aliases for the keywords defined in any of the above types of sections. These aliases can them be used in a system file in place of the master file keywords. |
| `$local_alias` | Defines constants for use only within the master file. |

Each entry in a section consists of a single line broken into a number of fields separated by blanks and/or tabs. Comments are preceded by a pound sign (#) and can begin at any position on a line. Blank lines and comments are ignored.

### Device Entries

Entries in a `$device` section have three fields:

Field 1: Device name as specified in the system file. The kernel uses this name as a prefix to names for device driver routines in `conf.c`.

Field 2: Restriction flags on this device. Flags are:

| | |
|---|---|
| o | Only one device of this type is allowed. |
| r | This device is required and will be automatically be configured into any kernels configured against this master file. |

| | |
|---|---|
| s | This device is a DG/UX-style STREAMS device. |
| S | This device is a System V-style STREAMS device. |
| N | This STREAMS device uses the new (System V.4) style open/close interface. |
| z | This device may be configured either explicitly or implicitly as part of a nested declaration of another device. For example, "st(insc(),4)" declares the device "insc()" implicitly. |
| n | No restrictions. |

Field 3: STREAMS Concurrency Set. The concurrency set name specifies the STREAMS set to which a given STREAMS module or STREAMS device driver belongs. STREAMS concurrency only occurs within each set: modules or drivers belonging to the same set are guaranteed never to run concurrently. A set may contain drivers, modules, or both. Two exceptional cases allow for more concurrency: the pseudo-set named `module` means that each instance of such a STREAMS device or module will have its own private set; and the pseudo-set named `stream` means that locking is granular to the individual STREAMS themselves. All other set name values specify a named set. The concurrency set name has no meaning for non-STREAMS devices, which by convention are assigned to the set named `default`.

### Protocol Entries

Entries in a `$protocol` section have six fields:

Field 1: Name to be used in the system file to reference this protocol.

Field 2: The protocol's protocol number as defined in the `/etc/protocols` file.

Field 3: The protocol's domain number as defined in the `<sys/socket.h>` header file.

Field 4: The protocol's type as defined in the `<sys/socket.h>` header file.

Field 5: The *infix name*. The kernel will use this name to generate names for the protocol's control routines. You may use any name you want and then match this name with the names of your protocol control routines.

Field 6: Restriction flags on this protocol. Flags are:

| | |
|---|---|
| r | This protocol is required and will be automatically be configured into any kernels configured against this master file. |
| d | This protocol will be the default protocol used for `socket`(2) calls of the listed Domain and Type. |
| u | This protocol is a UNIX domain protocol. |
| n | No restrictions. |

### STREAMS Module Entries

Entries in a `$stream` section have four fields:

Field 1: Name of the stream control module as given in the system file.

Field 2: The *infix name*. The kernel will use this name to generate names for the stream's control module routines. You may use any name you want and then match this name with the names of your stream control module routines.

Field 3: Restriction flags on this module. Flags are:

| | |
|---|---|
| N | This STREAMS module uses the new (System V.4) style open/close interface. |
| n | No restrictions. |

Field 4: STREAMS Concurrency Set. The concurrency set name specifies the STREAMS set to which a given STREAMS module or STREAMS device driver belongs. STREAMS concurrency only occurs within each set: modules or drivers belonging to the same set are guaranteed never to run concurrently. A set may contain drivers, modules, or both. Two exceptional cases allow for more concurrency: the pseudo-set named `module` means that each instance of such a STREAMS device or module will have its own private set; and the pseudo-set named `stream` means that locking is granular to the individual STREAMS themselves. All other set name values specify a named set.

### Tunable Parameter Entries

Entries in a `$keyword` section have four fields, the last of which is optional:

Field 1: Name of kernel variable to be set.

Field 2: The default value that the variable will have, unless it is overridden in the system file.

Field 3: The kernel variable's data type. This must not be a type that requires use of any header file besides `/usr/src/uts/aviion/ext/c_generics.h`.

Field 4: The implied value for a variable that is listed in the system file without a value. This is useful for things like function pointers, whose value is represented by a string that would otherwise be inconvenient to type.

### Alias Entries

Entries in an `$alias` section have two fields:

Field 1: Alias name.

Field 2: Name of master file entry being referenced.

### Local Alias Entries

Entries in a `$local_alias` section have two fields:

Field 1: Alias name.

Field 2: The value which this alias name will have. This can be either a numeric or character string value.

## SEE ALSO

`system`(4).
`config`(1M), `sysdef`(1M) in the *System Manager's Reference for the DG/UX System* *Installing the DG/UX System*. *Customizing the DG/UX System*. *Managing the DG/UX System*.

## NAME

mfs – memory file system

## DESCRIPTION

The DG/UX kernel provides support for memory file systems. These are file systems that live entirely in memory without any backing store on disk. Files in memory file systems do not persist between system instantiations. Memory file systems are faster than normal file systems and are ideal for temporary files and for putting common executables in them to avoid any disk I/O on execution. A memory file system has the same semantics as a normal DG/UX file system. Memory file systems can be NFS-exported just like regular DG/UX file systems.

A memory file system can be instantiated via the `mount`(1M) command:

```
mount -o ramdisk /dev/ml /pdd/memory
```

The "ramdisk" option instructs the DG/UX file system to create a memory file system instead of trying to mount the device "/dev/ml" on the directory. The "/dev/ml" pseudo device must not exist at the time of the mount command. The pseudo device node will be created during the mount to reference the mounted on directory. Any naming convention can be used for this memory device with the exception that the name must reference a path in `/dev`. The example name "/pdd/memory" is the directory in the DG/UX file system hierarchy where the memory file system will be created. This may be any directory.

There are several options:

```
mount -o ramdisk,use_wired_memory /dev/ml /pdd/memory
```

"use_wired_memory" is a boolean option that will instruct the file manager to use wired memory to hold data for the memory file system instead of unwired memory (the default is to use unwired memory). This is useful if you have lots of expansion memory for the file system, since data in the file system will always reside in memory and never be swapped out. (But see the CAUTIONS section below.)

```
mount -o ramdisk,max_file_space=20000 /dev/ml /pdd/memory
```

"max_file_space=*n*" gives the number of blocks that can be allocated to the memory file system to hold data. No space is ever allocated up front, so using a high value will not lead to trouble. The amount of actual space that can be given to a memory file system is the minimum of the value assigned by this attribute and the total amount of the resource (wired or unwired memory) available on the system. If space is not available to allocate blocks to a memory file system, then the operation that requests space will return an ENOSPC result. The default amount of space allocated to a memory file system is 2048 blocks.

```
mount -o ramdisk,max_file_count=50000 /dev/ml /pdd/memory
```

"max_file_count=*n*" gives the number file nodes that can be allocated in the memory file system. This is counted separately from the "max_file_space" attribute. The default number is 16384.

Memory file systems can be unmounted via the `umount`(1M) command:

```
umount /pdd/memory
```

The `umount` will not work until all the files have been removed from the file system. This is to protect against unintended data loss.

There is no limit to the number of memory file systems that may be created on a given system. Memory limitations, both wired and unwired, will ultimately govern how large they may grow.

## SEE ALSO

`mount`(1M), `umount`(1M), `fstab`(4), `exportfs`(8).

## CAUTIONS

Do not over-commit the swap space available to the system. Because of the way DG/UX allocates memory, if you establish a large memory file system, start some very large application, then fill the memory file system, you might exhaust the swap space on the system. This will cause the system to thrash and to kill random processes in order to recover the swap space.

Do not mount a memory file system on `/tmp`, since the recovery mechanism of `ex`(1) and `vi`(1) depends on the persistence of temporary files in the `/tmp` directory.

Do not use the use_wired_memory option unless your system has enough expansion (physical) memory.

Use of the use_wired_memory option is also strongly discouraged on diskless workstations.

## NAME

mnttab – mounted file system table

## SYNOPSIS

```
#include <mntent.h>
```

## DESCRIPTION

mnttab resides in the directory /etc and consists of a list of currently mounted file systems. The file contains a number of lines like this:

```
fsname dir type opts freq passno
```

for example:

```
/dev/dsk/usr  /usr  dg/ux  rw  1  1
```

would indicate a mount for a local filesystem, and

```
titan:/usr/titan  /usr/titan  nfs  rw,hard  0 0
```

would indicate an NFS filesystem mount. The entries from this file are accessed using the routines in getmntent(3), which returns a structure of the following form:

```
struct  mntent  {
   char   *mnt_fsname;    /* filesystem name */
   char   *mnt_dir;       /* filesystem path prefix */
   char   *mnt_type;      /* dg/ux, nfs, swap, cdrom, or ignore */
   char   *mnt_opts;      /* rw, ro, hard, soft, fg, bg, memory */
   int    mnt_freq;       /* highest dump level */
   int    mnt_passno;     /* pass number on parallel fsck */
};
```

Fields are separated by white space; a #, as the first non-white character, indicates a comment. The *mnt_type* field determines how the *mnt_fsname* and *mnt_opts* fields will be interpreted. The following is a list of the filesystem types currently supported, and the way each of them interprets these fields:

| *Type* | *Field* | *Interpretation* |
|---|---|---|
| dg/ux | mnt_fsname | Must be a block special device. |
| | mnt_opts | Valid options are ro, rw, bg, and fg. If this has the ramdisk option, other options include use_wired_memory, max_file_space and max_file_count. |
| cdrom | mnt_fsname | Must be a block special device. |
| nfs | mnt_fsname | The hostname of the server and the pathname on the server of the directory to be served. A colon separates the pathname and hostname. |
| | mnt_opts | Valid options are ro, rw, hard, soft. |
| swap | mnt_fsname | Must be a block special device swap section. |
| | mnt_opts | Ignored. |

If the *mnt_type* is specified as `ignore` then the entry is ignored. This is useful to show disks not currently used.

Entries identified as swap are made available as swap space by the swapon(1M) command at the end of the system reboot procedure.

When the *mnt_fsname* field is interpreted as a block special device, programs that require the corresponding character special device must construct the name by changing `dsk` to `rdsk` in the pathname.

If the *mnt_opts* field is a comma-separated list of options that includes `ro` or `rw`, then the filesystem is mounted read-write or read-only. If this includes `hard` or `soft`, then the NFS filesystem is mounted `hard` or `soft`.

The field *mnt_freq* indicates how often each filesystem should be dumped by the dump(1M) command (and triggers that command's w option, which determines what filesystems should be dumped). Most systems set the *mnt_freq* field to 1, indicating that filesystems are dumped each day.

The final field *mnt_passno* is used by the consistency checking program `fsck`(1M) to allow overlapped checking of filesystems during a reboot. All filesystems with a *mnt_passno* of 1 are checked first simultaneously, then all filesystems with *mnt_passno* of 2 are checked, and so on. The <*mnt_passno*> of the root filesystem should be 0, as the root cannot be checked since it is already mounted.

The maximum number of entries in `mnttab` is based on the system parameter NMOUNT located in `/usr/src/uts/mv/cf/config.h`, which defines the number of allowable mounted special files.

## SEE ALSO

`mount`(1M), `setmnt`(1M) in the *System Manager's Reference for the DG/UX System.*

## NAME

`netconfig` – network configuration database

## SYNOPSIS

`#include <netconfig.h>`

## DESCRIPTION

The network configuration database, `/etc/netconfig`, is a system file used to store information about networks connected to the system and available for use. The `netconfig` database and the routines that access it [see `getnetconfig`(3N)] are part of the UNIX System V Network Selection component. The Network Selection component also includes the environment variable `NETPATH` and a group of routines that access the `netconfig` database using `NETPATH` components as links to the `netconfig` entries. `NETPATH` is described in `sh`(1); the `NETPATH` access routines are discussed in `getnetpath`(3N).

`netconfig` contains an entry for each network available on the system. Entries are separated by newlines. Fields are separated by whitespace and occur in the order in which they are described below. Whitespace can be embedded as "\\*blank*" or "\*tab*". Backslashes may be embedded as "\\\\". Each field corresponds to an element in the `struct netconfig` structure. `struct netconfig` and the identifiers described on this manual page are defined in `/usr/include/netconfig.h`.

*network ID*
: A string used to uniquely identify a network. *network ID* consists of non-null characters, and has a length of at least 1. No maximum length is specified. This namespace is locally significant and the local system administrator is the naming authority. All *network IDs* on a system must be unique.

*semantics*
: The *semantics* field is a string identifing the "semantics" of the network, i.e., the set of services it supports, by identifying the service interface it provides. The *semantics* field is mandatory. The following semantics are recognized.

| | |
|---|---|
| `tpi_clts` | Transport Provider Interface, connectionless |
| `tpi_cots` | Transport Provider Interface, connection oriented |
| `tpi_cots_ord` | Transport Provider Interface, connection oriented, supports orderly release. |

*flag*
: The *flag* field records certain two-valued ("true" and "false") attributes of networks. *flag* is a string composed of a combination of characters, each of which indicates the value of the corresponding attribute. If the character is present, the attribute is "true." If the character is absent, the attribute is "false." "–" indicates that none of the attributes is present. Only one character is currently recognized:

| | |
|---|---|
| `v` | Visible ("default") network. Used when the environment variable `NETPATH` is unset. |

*protocol family*
: The *protocol family* and *protocol name* fields are provided for protocol-specific applications.

The *protocol family* field contains a string that identifies a protocol family. The *protocol family* identifier follows the same rules as those for *network IDs*, that is, the string consists of non-null characters; it has a length of at least `1`; and there is no maximum length specified. A "–" in the *protocol family* field indicates that no protocol family identifier applies, that is, the network is experimental. The following are examples:

| | |
|---|---|
| `loopback` | Loopback (local to host). |
| `inet` | Internetwork: UDP, TCP, etc. |
| `implink` | ARPANET imp addresses |
| `pup` | PUP protocols: e.g. BSP |
| `chaos` | MIT CHAOS protocols |
| `ns` | XEROX NS protocols |
| `nbs` | NBS protocols |
| `ecma` | European Computer Manufacturers Association |
| `datakit` | DATAKIT protocols |
| `ccitt` | CCITT protocols, X.25, etc. |
| `sna` | IBM SNA |
| `decnet` | DECNET |
| `dli` | Direct data link interface |
| `lat` | LAT |
| `hylink` | NSC Hyperchannel |
| `appletalk` | Apple Talk |
| `nit` | Network Interface Tap |
| `ieee802` | IEEE 802.2; also ISO 8802 |
| `osi` | Umbrella for all families used by OSI (e.g., protosw lookup) |
| `x25` | CCITT X.25 in particular |
| `osinet` | AFI = 47, IDI = 4 |
| `gosip` | U.S. Government OSI |

*protocol name*

The *protocol name* field contains a string that identifies a protocol. The *protocol name* identifier follows the same rules as those for *network IDs*, that is, the string consists of non-NULL characters; it has a length of at least `1`; and there is no maximum length specified. The following protocol names are recognized. A "–" indicates that none of the names listed applies.

| | |
|---|---|
| `tcp` | Transmission Control Protocol |
| `udp` | User Datagram Protocol |
| `icmp` | Internet Control Message Protocol |

*network device*

The *network device* is the full pathname of the device used to connect to the transport provider. Typically, this device will be in the `/dev` directory. The *network device* must be specified.

*directory lookup libraries*

The *directory lookup libraries* support a "directory service" (a name-to-address mapping service) for the network. This service is implemented by the UNIX System V Name-to-Address Mapping feature. If a network is not provided with such a library, the *netdir* feature will not work. A "–" in this field indicates the absence of any lookup libraries, in which case name-to-address mapping for the network is non-functional. The directory lookup library field consists of a comma-separated list of full pathnames to dynamically linked libraries. Commas may be embedded as "`\,`"; backslashs as "`\\`".

Lines in `/etc/netconfig` that begin with a sharp sign (#) in column 1 are treated as comments.

The struct netconfig structure includes the following members corresponding to the fields in in the netconfig database entries:

| | |
|---|---|
| `char * nc_netid` | Network ID, including NULL terminator |
| `unsigned long nc_semantics` | Semantics |
| `unsigned long nc_flag` | Flags |
| `char * nc_protofmly` | Protocol family |
| `char * nc_proto` | Protocol name |
| `char * nc_device` | Full pathname of the network device |
| `unsigned long nc_nlookups` | Number of directory lookup libraries |
| `char ** nc_lookups` | Full pathnames of the directory lookup libraries themselves |
| `unsigned long nc_unused[9]` | Reserved for future expansion (not advertised to user level) |

The nc_semantics field takes the following values, corresponding to the semantics identified above:

```
NC_TPI_CLTS
NC_TPI_COTS
NC_TPI_COTS_ORD
```

The nc_flag field is a bitfield. The following bit, corresponding to the attribute identified above, is currently recognized. NC_NOFLAG indicates the absence of any attributes.

```
NC_VISIBLE
```

## FILES

/etc/netconfig
/usr/include/netconfig.h

## SEE ALSO

netdir_getbyname(3N), getnetconfig(3N), getnetpath(3N), netconfig(4)
*Network Programmer's Guide*
*System Administrator's Guide*

## NAME

passwd – password file

## SYNOPSIS

`/etc/passwd`

## DESCRIPTION

The `passwd` file contains for each user the following information:

| | |
|---|---|
| name | User's login name. Contains no uppercase characters and must not be greater than USR_NAME (see `limits`(4)) characters long. |
| password | encrypted password. |
| numerical user id | This is the user's id in the system and it must be unique. Otherwise, users with the same uid will be able to access each other's files. |
| numerical group id | This is the number of the group that the user belongs to. |
| user's real name | Some system administrators use this field to contain the user's office, extension, home phone, and so on. For historical reasons this field is called the GCOS field. |
| initial working directory | The directory that the user is positioned in when they log in — this is also known as the home directory. |
| shell | program to use as shell when the user logs in. |

The user's real name field may contain '&', meaning to insert the login name.

The password file is an ASCII file. Each field within each user's entry is separated from the next by a colon. Each user is separated from the next by a new-line. If the password field is null, no password is demanded; if the shell field is null, `/bin/sh` is used.

This file resides in directory `/etc`. Because of the encrypted passwords, it has general read permission. It can be used, for example to map numerical user IDs to names.

The encrypted password consists of 13 characters chosen from a 64-character alphabet ( . , / , 0-9 , A-Z , a-z ), except when the password is null. In that case, the encrypted password is also null. Password aging is affected for a particular user if the user's encrypted password in the password file is followed by a comma and a non-null string of characters from the above alphabet (such a string must first be introduced by the superuser).

The first character of the age denotes the maximum number of weeks for which a password is valid. If you try to login after your password has expired, you must supply a new one. The next character denotes the minimum period in weeks that must elapse before the password may be changed. The remaining characters define the week (counted from the beginning of 1970) when the password was last changed ( a null string is equivalent to zero). The first and second characters have numerical values in the range 0-63 that correspond to the 64-character alphabet shown above (i.e., / = 1 week; z = 63 weeks). If both characters are equal to zero (derived from the string "." or ".."), you must change your password the next time you login. The age will disappear from your entry in the password file. If the second character is greater than the first (signified, e.g., by the string "./"), then only the superuser will be able to change the password.

The passwd file can also have lines beginning with a plus (+), which means to incorporate entries from the Yellow Pages.

NOTE: You must be using the DG/UX Open Network Computing/Network File System (ONC/NFS) to use this feature. If you use DG/UX ONC/NFS, see passwd(5).

There are three styles of + entries: all by itself, + means to insert the entire contents of the Yellow Pages password file at that point; +*name* means to insert the entry (if any) for *name* from the Yellow Pages at that point; +@*name* means to insert the entries for all members of the network group *name* at that point. If a + entry has a non-null password, directory, user's real name, or shell field, they will override what is contained in the Yellow Pages. The numerical user ID and group ID fields cannot be overridden.

Entries beginning with a minus sign (-) are also allowed. They have two formats: -*name* and -@*name*. The meaning of these formats is the same as for +*name* and +@*name*, respectively, except that the action is reversed; all members matched are considered to be excluded from the password file, regardless of subsequent entries. Minus entries can be used to exclude specific entries from the Yellow Pages.

## EXAMPLE

Here is a sample /etc/passwd file:

```
root:q.mJzTnu8icF.:0:10:God:/:/bin/csh
tut:6k/7KCFRPNVXg:508:10:Bill Tuthill:/usr/tut:/bin/csh
+john:
-@documentation:no-login:
+:::Guest
john::605:20:John Smith:/usr/john:
```

In this example, there are specific entries for users root and tut, in case the Yellow Pages are not running. (See *Managing ONC/NFS and Its Facilities on the DG/UX System.*) The user john will have his password entry in the Yellow Pages incorporated without change; anyone in the netgroup documentation will have their password field disabled, and anyone else will be able to login with their usual password, shell, and home directory, but with a GCOS field of Guest.

The second entry for john in this example will not be used if the Yellow Pages are running; the first entry for a given user name will be used if multiple entries exist.

Appropriate precautions must be taken to lock the /etc/passwd file against simultaneous changes if it is to be edited with a text editor; vipw(1M) does the necessary locking. The password file can be scanned for inconsistencies using pwck(1M).

## FILES

/etc/passwd

## SEE ALSO

login(1), passwd(1), pwck(1M), useradd(1M), vipw(1M), crypt(3C), getpwent(3C), group(4), limits(4), passwd(5).

## NAME

`pkginfo` – package characteristics file

## DESCRIPTION

`pkginfo` is an ASCII file that describes the characteristics of the package along with information that helps control the flow of installation. It is created by the software package developer.

Each entry in the `pkginfo` file is a line that establishes the value of a parameter in the following form:

*PARAM="value"*

There is no required order in which the parameters must be specified within the file. Each parameter is described below. Only fields marked with an asterisk are mandatory.

| | |
|---|---|
| *PKG** | Abbreviation for the package being installed, generally three characters in length (for example, `dir` or `pkg`). All characters in the abbreviation must be alphanumeric and the first may not be numeric. The abbreviation is limited to a maximum length of nine characters. `install`, `new`, and `all` are reserved abbreviations. |
| *NAME** | Text that specifies the package name (maximum length of 256 ASCII characters). |
| *ARCH** | A comma-separated list of alphanumeric tokens that indicate the architecture (for example, `3B2`) associated with the package. The `pkgmk` tool may be used to create or modify this value when actually building the package. The maximum length of a token is 16 characters and it cannot include a comma. |
| *VERSION** | Text that specifies the current version associated with the software package. The maximum length is 256 ASCII characters and the first character cannot be a left parenthesis. The `pkgmk` tool may be used to create or modify this value when actually building the package. |
| *CATEGORY** | A comma-separated list of categories under which a package may be displayed. A package must at least belong to the system or application category. Categories are case-insensitive and may contain only alphanumerics. Each category is limited in length to 16 characters. |
| *DESC* | Text that describes the package (maximum length of 256 ASCII characters). |
| *VENDOR* | Used to identify the vendor that holds the software copyright (maximum length of 256 ASCII characters). |
| *HOTLINE* | Phone number and/or mailing address where further information may be received or bugs may be reported (maximum length of 256 ASCII characters). |
| *EMAIL* | An electronic address where further information is available or bugs may be reported (maximum length of 256 ASCII characters). |
| *VSTOCK* | The vendor stock number, if any, that identifies this product (maximum length of 256 ASCII characters). |
| *CLASSES* | A space-separated list of classes defined for a package. The order of the list determines the order in which the classes are installed. Classes listed first will be installed first (on a media by media basis). |

| | |
|---|---|
| | This parameter may be modified by the request script. |
| *ISTATES* | A list of allowable run states for package installation (for example, `"S s 1"`). |
| *RSTATES* | A list of allowable run states for package removal (for example, `"S s 1"`). |
| *BASEDIR* | The pathname to a default directory where "relocatable" files may be installed. If blank, the package is not relocatable and any files that have relative pathnames will not be installed. An administrator can override the default directory. |
| *ULIMIT* | If set, this parameter is passed as an argument to the `ulimit` command, which establishes the maximum size of a file during installation. |
| *ORDER* | A list of classes defining the order in which they should be put on the medium. Used by `pkgmk` in creating the package. Classes not defined in this field are placed on the medium using the standard ordering procedures. |
| *MAXINST* | The maximum number of package instances that should be allowed on a machine at the same time. By default, only one instance of a package is allowed. This parameter must be set in order to have multiple instances of a package. |
| *PSTAMP* | Production stamp used to mark the `pkgmap` file on the output volumes. Provides a means for distinguishing between production copies of a version if more than one is in use at a time. If PSTAMP is not defined, the default is used. The default consists of the UNIX system machine name followed by the string "*YYMMDDHHMM*" (year, month, date, hour, minutes). |
| *INTONLY* | Indicates that the package should only be installed interactively when set to any non-NULL value. |
| *PREDEPEND* | Used to maintain compatibility with pre-SVR4 package dependency checking. Pre-SVR4 dependency checks were based on whether or not the name file for the required package existed in the `/var/options` directory. This directory is not maintained for SVR4 packages since the `depend` file is used for checking dependencies. However, entries can be created in this directory to maintain compatibility. Setting the PREDEPEND parameter to `y` or `yes` creates a `/usr/option` entry for the package. (Packages that are new for SVR4 do not need to use this parameter.) |

## EXAMPLES

Here is a sample `pkginfo`:

```
PKG="oam"
NAME="OAM Installation Utilities"
VERSION="3"
VENDOR="AT&T"
HOTLINE="1-800-ATT-BUGS"
EMAIL="attunix!olsen"
VSTOCK="0122c3f5566"
CATEGORY="system.essential"
ISTATES="S 2"
RSTATES="S 2"
```

## SEE ALSO

`compver`(4), `copyright`(4), `depend`(4), `pkgmap`(4).

## NOTES

Developers may define their own installation parameters by adding a definition to this file. A developer-defined parameter must begin with a capital letter,

## NAME

pkgmap – package contents description file

## DESCRIPTION

pkgmap is an ASCII file that provides a complete listing of the package contents. It is automatically generated by pkgmk(1) using the information in the prototype file.

Each entry in pkgmap describes a single "deliverable object file." A deliverable object file includes shell scripts, executable objects, data files, directories, etc. The entry consists of several fields of information, each field separated by a space. The fields are described below and must appear in the order shown.

*part* An optional field designating the part number in which the object resides. A part is a collection of files, and is the atomic unit by which a package is processed. A developer can choose the criteria for grouping files into a part (e.g., based on class). If no value is defined in this field, part 1 is assumed.

*ftype* A one-character field that indicates the file type. Valid values are:

| | |
|---|---|
| f | a standard executable or data file |
| e | a file to be edited upon installation or removal |
| v | volatile file (one whose contents are expected to change) |
| d | directory |
| x | an exclusive directory |
| l | linked file |
| p | named pipe |
| c | character special device |
| b | block special device |
| i | installation script or information file |
| s | symbolic link |

*class* The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. It is not specified if the ftype is i (information file).

*pathname* The pathname where the object will reside on the target machine, such as /usr/bin/mail. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable.

For linked files (ftype is either l or s), pathname must be in the form of *path1=path2*, with *path1* specifying the destination of the link and *path2* specifying the source of the link.

*pathname* may contain variables which support relocation of the file. A $*parameter* may be embedded in the pathname structure. $BASEDIR can be used to identify the parent directories of the path hierarchy, making the entire package easily relocatable. Default values for *parameter* and BASEDIR must be supplied in the pkginfo file and may be overridden at installation.

*major* The major device number. The field is only specified for block or character special devices.

*minor* The minor device number. The field is only specified for block or character special devices.

*mode* The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files, packaging information files or non-installable files.

*owner* The owner of the file (for example, bin or root). The field is limited to 14 characters in length. A question mark (?) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what owner an installation script will be executed.

Can be a variable specification in the form of $[A-Z]. Will be resolved at installation time.

*group* The group to which the file belongs (for example, "bin" or "sys"). The field is limited to 14 characters in length. A question mark (?) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what group an installation script will be executed.

Can be a variable assignment in the form of `$[A-Z]`. Will be resolved at installation time.

*size* The actual size of the file in bytes. This field is not specified for named pipes, special devices, directories or linked files.

*cksum* The checksum of the file contents. This field is not specified for named pipes, special devices, directories or linked files.

*modtime* The time of last modification, as reported by the stat(2) function call. This field is not specified for named pipes, special devices, directories or linked files.

Each pkgmap must have one line that provides information about the number and maximum size (in 512-byte blocks) of parts that make up the package. This line is in the following format:

:*number_of_parts maximum_part_size*

Lines that begin with "#" are comment lines and are ignored.

When files are saved during installation before they are overwritten, they are normally just copied to a temporary pathname. However, for files whose mode includes execute permission (but which are not editable), the existing version is linked to a temporary pathname and the original file is removed. This allows processes which are executing during installation to be overwritten.

## EXAMPLES

The following is an example of a pkgmap file.

```
:2 500
1 i pkginfo 237 1179 541296672
1 b class1 /dev/diskette 17 134 0644 root other
1 c class1 /dev/rdiskette 17 134 0644 root other
1 d none bin 0755 root bin
1 f none bin/INSTALL 0755 root bin 11103 17954 541295535
1 f none bin/REMOVE 0755 root bin 3214 50237 541295541
1 l none bin/UNINSTALL=bin/REMOVE
1 f none bin/cmda 0755 root bin 3580 60325 541295567
1 f none bin/cmdb 0755 root bin 49107 51255 541438368
1 f class1 bin/cmdc 0755 root bin 45599 26048 541295599
```

```
1 f class1 bin/cmdd 0755 root bin 4648 8473 541461238
1 f none bin/cmde 0755 root bin 40501 1264 541295622
1 f class2 bin/cmdf 0755 root bin 2345 35889 541295574
1 f none bin/cmdg 0755 root bin 41185 47653 541461242
2 d class2 data 0755 root bin
2 p class1 data/apipe 0755 root other
2 d none log 0755 root bin
2 v none log/logfile 0755 root bin 41815 47563 541461333
2 d none save 0755 root bin
2 d none spool 0755 root bin
2 d none tmp 0755 root bin
```

## SEE ALSO

pkginfo(4).

## NOTES

The pkgmap file may contain only one entry per unique pathname.

## NAME

`profile` – setting up an environment at login time

## DESCRIPTION

If you are using the Bourne shell and your login directory contains a file named `.profile`, that file will be executed (via `exec .profile`) before your session begins; `.profiles` are handy for setting exported environment variables and terminal modes. If the file `/etc/profile` exists, it will be executed for every user before the `.profile`. The following example is typical (except for the comments):

```
#  Make some environment variables global
export MAIL PATH
#  Set file creation mask
umask 22
#  Tell me when new mail comes in
MAIL=/usr/mail/myname
#  Add my /bin directory to the shell search sequence
PATH=$PATH:$HOME/bin
```

## FILES

`$HOME/.profile`
`/etc/profile`

## SEE ALSO

`environ`(5), `term`(5).
`env`(1), `login`(1), `mail`(1), `sh`(1), `stty`(1), `su`(1) in the *User's Reference for the DG/UX System*.

## NAME

`prototype` – package information file

## DESCRIPTION

`prototype` is an ASCII file used to specify package information. Each entry in the file describes a single deliverable object. An object may be a data file, directory, source file, executable object, etc. This file is generated by the package developer.

Entries in a `prototype` file consist of several fields of information separated by white space. Comment lines begin with a "#" and are ignored. The fields are described below and must appear in the order shown.

*part* An optional field designating the part number in which the object resides. A part is a collection of files, and is the atomic unit by which a package is processed. A developer can choose criteria for groupig files into a part (e.g., based on class). If this field is not used, part 1 is assumed.

*ftype* A one-character field which indicates the file type. Valid values are:

| | |
|---|---|
| `f` | a standard executable or data file |
| `e` | a file to be edited upon installation or removal |
| `v` | volatile file (one whose contents are expected to change) |
| `d` | directory |
| `x` | an exclusive directory |
| `l` | linked file |
| `p` | named pipe |
| `c` | character special device |
| `b` | block special device |
| `i` | installation script or information file |
| `s` | symbolic link |

*class* The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. The field is not specified for installation scripts. (`admin` and all classes beginning with capital letters are reserved class names.)

*pathname* The pathname where the file will reside on the target machine, e.g., `/usr/bin/mail` or `bin/ras_proc`. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable. The form

*path1=path2*

may be used for two purposes: to define a link and to define local pathnames.

For linked files, *path1* indicates the destination of the link and *path2* indicates the source file. (This format is mandatory for linked files.)

For local pathnames, *path1* indicates the pathname an object should have on the machine where the entry is to be installed and *path2* indicates either a relative or fixed pathname to a file on the host machine which contains the actual contents.

A pathname may contain a variable specification, which will be resolved at the time of installation. This specification should have the form $[A-Z].

*major* The major device number. The field is only specified for block or character special devices.

*minor* The minor device number. The field is only specified for block or character special devices.

*mode* The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.

*owner* The owner of the file (for example, `bin` or `root`). The field is limited to 14 characters in length. A question mark (?) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.

Can be a variable specification in the form of `$[A-Z]`. Will be resolved at installation time.

*group* The group to which the file belongs (for example, `bin` or `sys`). The field is limited to 14 characters in length. A question mark (?) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.

Can be a variable specification in the form of $[A-Z]. Will be resolved at installation time.

An exclamation point (!) at the beginning of a line indicates that the line contains a command. These commands are used to incorporate files in other directories, to locate objects on a host machine, and to set permanent defaults. The following commands are available:

`search` Specifies a list of directories (separated by white space) to search for when looking for file contents on the host machine. The basename of the *path* field is appended to each directory in the ordered list until the file is located.

`include` Specifies a pathname which points to another prototype file to include. Note that `search` requests do not span `include` files.

`default` Specifies a list of attributes (mode, owner, and group) to be used by default if attribute information is not provided for prototype entries which require the information. The defaults do not apply to entries in `include` prototype files.

*param=value* Places the indicated parameter in the current environment.

The above commands may have variable substitutions embedded within them, as demonstrated in the two example `prototype` files below.

Before files are overwritten during installation, they are copied to a temporary pathname. The exception to this rule is files whose mode includes execute permission, unless the file is editable (i.e, *ftype* is `e`). For files which meet this exception, the existing version is linked to a temporary pathname, and the original file is removed. This allows processes which are executing during installation to be overwritten.

## EXAMPLES

Example 1:

```
!PROJDIR=/usr/proj
!BIN=$PROJDIR/bin
!CFG=$PROJDIR/cfg
!LIB=$PROJDIR/lib
!HDRS=$PROJDIR/hdrs
```

```
!search /usr/myname/usr/bin /usr/myname/src /usr/myname/hdrs
i pkginfo=/usr/myname/wrap/pkginfo
i depend=/usr/myname/wrap/depend
i version=/usr/myname/wrap/version
d none /usr/wrap 0755 root bin
d none /usr/wrap/usr/bin 0755 root bin
! search $BIN
f none /usr/wrap/bin/INSTALL 0755 root bin
f none /usr/wrap/bin/REMOVE 0755 root bin
f none /usr/wrap/bin/addpkg 0755 root bin
!default 755 root bin
f none /usr/wrap/bin/audit
f none /usr/wrap/bin/listpkg
f none /usr/wrap/bin/pkgmk
# the following file starts out zero length but grows
v none /usr/wrap/logfile=/dev/null 0644 root bin
# the following specifies a link (dest=src)
l none /usr/wrap/src/addpkg=/usr/wrap/bin/rmpkg
! search $SRC
!default 644 root other
f src /usr/wrap/src/INSTALL.sh
f src /usr/wrap/src/REMOVE.sh
f src /usr/wrap/src/addpkg.c
f src /usr/wrap/src/audit.c
f src /usr/wrap/src/listpkg.c
f src /usr/wrap/src/pkgmk.c
d none /usr/wrap/data 0755 root bin
d none /usr/wrap/save 0755 root bin
d none /usr/wrap/spool 0755 root bin
d none /usr/wrap/tmp 0755 root bin
d src /usr/wrap/src 0755 root bin
```

Example 2:

```
# this prototype is generated by 'pkgproto' to refer
# to all prototypes in my src directory
!PROJDIR=/usr/dew/projx
!include $PROJDIR/src/cmd/prototype
!include $PROJDIR/src/cmd/audmerg/protofile
!include $PROJDIR/src/lib/proto
```

## SEE ALSO

pkginfo(4), pkgmk(1).

## NOTES

Normally, if a file is defined in the prototype file but does not exist, that file is created at the time of package installation. However, if the file pathname includes a directory that does not exist, the file will not be created. For example, if the prototype file has the following entry:

```
f none /usr/dev/bin/command
```

and that file does not exist, it will be created if the directory /usr/dev/bin already exists or if the prototype also has an entry defining the directory:

```
d none /usr/dev/bin
```

## NAME

`rcsfile` – format of RCS file

## DESCRIPTION

An RCS file is an ASCII file. Its contents are described by the grammar below. The text is free format, that is, spaces, tabs and new lines have no significance except in strings. Strings are enclosed by '@'. For a string to contain a '@', the '@' must be doubled.

The meta-syntax uses the following conventions: '|' (bar) separates alternatives; '{' and '}' enclose optinal phrases; '{' and '}*' enclose phrases that may be repeated zero or more times; '{' and '}+' enclose phrases that must appear at least once and may be repeated; '<' and '>' enclose nonterminals.

```
rcstext    ::= admin {delta}* desc {deltatext}*

admin      ::= head     {num};
                 access   {id}*;
                 symbols  {id : num}*;
                 locks    {id : num}*;
                 comment  {string};

delta      ::= <num>
                 date     num;
                 author   id;
                 state    {id};
                 branches {num}*;
                 next     {num};

desc       ::= desc    <string>

deltatext  ::= <num>
                 log      <string>
                 text     <string>

num        ::= {digit{.}}+

digit      ::= 0 | 1 | ... | 9

id         ::= letter{idchar}*

letter     ::= A | B | ... | Z | a | b | ... | z

idchar     ::= Any printing ASCII character except space,
               tab, carriage return, new line, and special.

special    ::= ; | : | , | @

string     ::= @{any ASCII character, with '@' doubled}*@
```

Identifiers are case sensitive. Keywords are lowercase only. The sets of keywords and identifiers may overlap.

The *delta* nodes form a tree. All nodes whose numbers consist of a single pair (e.g., 2.3, 2.1, 1.3, etc.) are on the "trunk", and are linked through the "next" field in order of decreasing numbers. The "head" field in the <admin> node points to the head of that sequence (i.e., contains the highest pair).

All *delta* nodes whose numbers consist of 2n fields (where n > 2) (e.g., 3.1.1.1, 2.1.2.2, etc.) are linked as follows. All nodes whose first (2n)-1 number fields are identical are linked through the "next" field in order of increasing numbers. For each such sequence, the *delta* node whose number is identical to the first 2(n-1) number fields of the deltas on that sequence is called the branchpoint. The "branches" field of a node contains a list of the numbers of the first nodes of all sequences for which it is a branchpoint. This list is ordered in increasing numbers.

Example:

```
                             Head
                               |
                               |
                               v
                           ---------
     / \          / \      |       |      / \           / \
    /   \        /   \     |  2.1  |     /   \         /   \
   /     \      /     \    |       |    /     \       /     \
  /1.2.1.3\    /1.3.1.1\   |       |   /1.2.2.2\    /1.2.2.1.1.1\
  ---------    ---------   ---------   ---------    -------------
      ^            ^           |           ^              ^
      |            |           |           |              |
      |            |           v           |              |
     / \           |       ---------      / \             |
    /   \          |       \  1.3  /     /   \            |
   /     \         ---------\     /     /     \------------
  /1.2.1.1\                  \   /     /1.2.2.1\
  ---------                   \ /      ---------
      ^                        |           ^
      |                        |           |
      |                        v           |
      |                    ---------       |
      |                    \  1.2  /       |
      ----------------------\     /---------
                             \   /
                              \ /
                               |
                               |
                               v
                           ---------
                           \  1.1  /
                            \     /
                             \   /
                              \ /
```

**Fig. 1: A Revision Tree**

## SEE ALSO

ci(1), co(1), ident(1), rcs(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rlog(1), sccstorcs(1).

## NAME

reloc – relocation information for a common object file

## SYNOPSIS

```
#include <reloc.h>
```

## DESCRIPTION

Common object (COFF) files have one relocation entry for each relocatable reference in the text or data. If relocation information is present, it will be in the following format:

```
struct reloc
{
       long      r_vaddr ; /* (virtual) address of reference */
       long      r_symndx ;/* index into symbol table */
       ushort    r_type ;  /* relocation type */
       unsigned short r_offset;/* high 16 bits of expression*/
} ;

#defineR_ABS      0
#defineR_PCR16L   128
#defineR_PCR26L   129
#defineR_VRT16    130
#defineR_HVRT16   131
#defineR_LVRT16   132
#defineR_VRT32    133
```

As the link editor reads each input section and performs relocation, the relocation entries are read. They direct how references found within the input section are treated.

| | |
|---|---|
| R_ABS | The reference is absolute and no relocation is necessary. The entry will be ignored. |
| R_PCR16L | A "PC-relative" 16-bit reference to the symbol's virtual address. |
| R_PCR26L | A "PC-relative" 26-bit reference to the symbol's virtual address. |
| R_VRT16 | Direct 16-bit reference to the symbol's virtual address. |
| R_HVRT16 | Same as R_VRT16, except, only the high 16 bits are used in the relocation. |
| R_LVRT16 | Same as R_VRT16, except, only the low 16 bits are used in the relocation. |
| R_VRT32 | Direct 32-bit reference to the symbol's virtual address. |

Relocation entries are generated automatically by the assembler and automatically used by the link editor. Link editor options exist for both preserving and removing the relocation entries from object files.

## SEE ALSO

as(1), ld-coff(1), a.out(4), syms(4).

## NAME

`sccsfile` – format of SCCS file

## DESCRIPTION

An SCCS file is an ASCII file. It consists of six logical parts:

| | |
|---|---|
| *checksum* | |
| *delta table* | information about each delta |
| *user names* | login names and/or numerical group IDs of users who may add deltas |
| *flags* | definitions of internal keywords |
| *comments* | arbitrary descriptive information about the file |
| *body* | the actual text lines intermixed with control lines |

Throughout an SCCS file there are lines that begin with the ASCII SOH (start of heading) character (octal 001). We call this character *the control character*, and represent it graphically as @. Any line described below that does not begin with the control character is prevented from doing so.

Entries of the form DDDDD represent a five-digit string (a number between 00000 and 99999).

Each logical part of an SCCS file is described in detail below.

*Checksum*

The checksum is the first line of an SCCS file. The form of the line is:

```
@hDDDDD
```

The value of the checksum is the sum of all characters, except those of the first line. The @h provides a *magic number* of (octal) 064001.

*Delta table*

The delta table consists of a variable number of entries of the form:

```
@s DDDDD/DDDDD/DDDDD
@d type <SCCS ID> yr/mo/da hr:mi:se pgmr DDDDD DDDDD
@i DDDDD . . .
@x DDDDD . . .
@g DDDDD . . .
@m <MR number>
  .
  .
  .
@c comments . . .
  .
  .
  .
@e
```

The first line (@s) contains the number of lines inserted/deleted/unchanged. The second line (@d) contains the type of the delta (currently, normal: D, and removed: R); the SCCS ID of the delta; the date and time of creation of the delta; the login name corresponding to the real user ID at the time the delta was created; and the serial numbers of the delta and its predecessor

The @i, @x, and @g lines are optional; they contain the serial numbers of deltas included, excluded, and ignored, respectively.

The @m lines (optional) each contain one MR number associated with the delta; the @c lines contain comments associated with the delta.

The @e line ends the delta table entry.

*User names*

The list of login names and/or numerical group IDs of users who may add deltas to the file, separated by new-lines. The lines containing these login names and/or numerical group IDs are surrounded by the bracketing lines @u and @U. An empty list lets anyone to make a delta. Any line starting with a ! prohibits the succeeding group or user from making deltas.

*Flags*

Keywords used internally (see admin(1) for more information on their use). Each flag line takes the form:

@f *flag*<optional text>

The following flags are defined:

| | |
|---|---|
| @f t | <type of program> |
| @f v | <program name> |
| @f i | <keyword string> |
| @f b | |
| @f m | <module name> |
| @f f | <floor> |
| @f c | <ceiling> |
| @f d | <default-sid> |
| @f n | |
| @f j | |
| @f l | <lock-releases> |
| @f q | <user defined> |
| @f z | <reserved for use in interfaces> |

The t flag defines the replacement for the %Y% identification keyword. The v flag controls prompting for MR numbers as well as comments; if the optional text is present it defines an MR number validity checking program.

The i flag controls the warning/error aspect of the No id keywords message. When the i flag is not present, this message is only a warning; when the i flag is present, this message will cause a fatal error; the file will not be gotten, or the delta will not be made.

When the b flag is present the -b keyletter may be used on the *get* command to cause a branch in the delta tree.

The m flag defines the first choice for the replacement text of the %M% identification keyword. The f flag defines the the release below which no deltas may be added (also known as the floor release).

The c flag defines the the release above which no deltas may be added (also known as the ceiling release).

The d flag defines the default SID to be used when none is specified on a *get* command.

The n flag causes *delta* to insert a null delta (a delta that applies *no* changes) in those releases that are skipped when a delta is made in a *new* release (e.g., when delta 5.1 is made after delta 2.7, releases 3 and 4 are skipped). The absence of the n flag causes skipped releases to be completely empty.

The j flag causes *get* to allow concurrent edits of the same base SID.

The l flag defines a *list* of releases that are *locked* against editing (get(1) with the -e keyletter).

The q flag defines the replacement for the %Q% identification keyword.

The z flag is used in certain specialized interface programs.

*Comments*

Arbitrary text is surrounded by the bracketing lines @t and @T. The comments section typically will contain a description of the file's purpose.

*Body*

The body consists of text lines and control lines. Text lines do not begin with the control character, control lines do. There are three kinds of control lines:

| | |
|---|---|
| @I DDDDD | Insert |
| @D DDDDD | Delete |
| @E DDDDD | End |

The digit string is the serial number corresponding to the delta for the control line.

## SEE ALSO

admin(1), delta(1), get(1), prs(1) in the *User's Reference for the DG/UX System.*

## NAME

`scr_dump` – format of curses screen image file

## SYNOPSIS

`scr_dump(`*file*`)`

## DESCRIPTION

The `curses`(3X) function `scr_dump()` copies the contents of the screen into a file. The format of the screen image is as described below.

The name of the tty is 20 characters long and the modification time (the *mtime* of the tty that this is an image of) is of the type *time_t*. All other numbers and characters are stored as `chtype` (see `<curses.h>`). No newlines are stored between fields.

```
<magic number: octal 0433>
<name of tty>
<mod time of tty>
columns <lines>
<line length> <chars in line>   for each line on the screen
<line length> <chars in line>
   .
   .
   .
<labels?>                       1, if soft screen labels are present
<cursor row> <cursor column>
```

Only as many characters as are in a line will be listed. For example, if the *<line length>* is `0`, there will be no characters following *<line length>*. If *<labels?>* is TRUE, following it will be

```
<number of labels>
<label width>
<chars in label 1>
<chars in label 2>
   .
   .
   .
```

## SEE ALSO

`curses`(3X).

## NAME

sde-chooser – execute environment-sensitive tool

## SYNOPSIS

sde-chooser [-e *sde-target*] *path* [*tool-args*]

## DESCRIPTION

The action of a number of software development tools depends on the current software development environment [see sde(5)]. Such tools have different versions in each environment. Sde-chooser finds and executes the correct version of such a tool.

For example, when a command line such as "as foo.s" is executed, a small program named as in /usr/bin executes sde-chooser with the appropriate arguments. Sde-chooser in turn executes the correct version of as.

Sde-chooser is not normally invoked from a shell command line, but it can be with the following arguments:

| | |
|---|---|
| -e *sde-target* | Specifies a software development environment explicitly. If this option is not given, sde-chooser uses the current software development environment [see sde-target(1)]. |
| *path* | The path to the desired tool within an environment. Path is given as an absolute path but it is interpreted as being relative to /usr/sde/<*sde-target*>. For example, /usr/bin/as invokes /usr/sde/<*sde-target*>/usr/bin/as, where <*sde-target*> is a software development environment. |
| *tool-args* | All remaining arguments to sde-chooser are passed to the selected tool. |

## SEE ALSO

sde-target(1), sde(5), elink(5).

## NAME

sdetab – software development environment data base

## DESCRIPTION

The `sdetab` file contains information used by certain software development tools to customize SDE targets. The actual file used is `/usr/etc/sdetab`, which is an `elink` to the appropriate file (see `sde`(5) and `elink`(5)).

Each entry in the `sdetab` file consists of a key followed by one or more attributes separated by a colon, `:`. Blank lines and comments (from the pound sign, `#`, to the end of the line) are ignored. The backslash, `\`, may be used to quote characters.

Currently, `ld`(1) uses the key `fmagic` to determine the magic number of the executable it produces.

## FILES

`/usr/etc/sdetab`

## SEE ALSO

`sde-target`(1), `sde`(5), `elink`(5).

## NAME

space – disk space requirement file

## DESCRIPTION

space is an ASCII file that gives information about disk space requirements for the target environment. It defines space needed beyond that which is used by objects defined in the prototype file—for example, files which will be installed with the installf command. It should define the maximum amount of additional space which a package will require.

The generic format of a line in this file is:

*pathname blocks inodes*

Definitions for the fields are as follows:

| | |
|---|---|
| *pathname* | Specifies a directory name which may or may not be the mount point for a filesystem. Names that do not begin with a slash (/) indicate relocatable directories. |
| *blocks* | Defines the number of disk blocks required for installation of the files and directory entries contained in the pathname (using a 512-byte block size). |
| *inodes* | Defines the number of inodes required for installation of the files and directory entries contained in the pathname. |

## EXAMPLE

```
# extra space required by config data which is
# dynamically loaded onto the system
data  500   1
```

## SEE ALSO

installf(1M), prototype(4)

## NAME

strftime – language specific strings

## DESCRIPTION

There can exist one printable file per locale to specify its date and time formatting information. These files must be kept in the directory /usr/lib/locale/*<locale>*/LC_TIME. The contents of these files are:

1. abbreviated month names (in order)
2. month names (in order)
3. abbreviated weekday names (in order)
4. weekday names (in order)
5. default strings that specify formats for locale time (%X) and locale date (%x).
6. default format for cftime, if the argument for cftime is zero or null.
7. AM (ante meridian) string
8. PM (post meridian) string

Each string is on a line by itself. All wh:te space is significant. The order of the strings in the above list is the same order in which they must appear in the file.

## EXAMPLE

/usr/lib/locale/C/LC_TIME

```
Jan
Feb
 ...
January
February
 ...
Sun
Mon
 ...
Sunday
Monday
 ...
%H:%M:%S
%m/%d/%y
%a %b %d %T %Z %Y
AM
PM
```

## FILES

/usr/lib/locale/*<locale>*/LC_TIME

## SEE ALSO

ctime(3C), setlocale(3C), strftime(3C).

## NAME

syms – common object file symbol table format

## SYNOPSIS

```
#include  <syms.h>
```

## DESCRIPTION

Common object files contain information to support symbolic software testing [see sdb(1)]. Line number entries [see `linenum`(4)] and extensive symbolic information permit testing at the C *source* level. Every object file's symbol table is organized as shown below.

```
File name 1.
        Function 1.
                Local symbols for function 1.
        Function 2.
                Local symbols for function 2.
        ...
        Static externs for file 1.

File name 2.
        Function 1.
                Local symbols for function 1.
        Function 2.
                Local symbols for function 2.
        ...
        Static externs for file 2.
...

Defined global symbols.
Undefined global symbols.
```

The entry for a symbol is a fixed-length structure. The members of the structure hold the name (null padded), its value, and other information. The C structure is given below.

```
#define  SYMNMLEN   8
#define  FILNMLEN   14
#define  DIMNUM     4

struct  syment
{
    union                           /* all ways to get symbol name */
    {
        char        _n_name[SYMNMLEN]; /* symbol name */
        struct
        {
            long    _n_zeroes;     /* == 0L when in string table */
            long    _n_offset;     /* location of name in table */
        } _n_n;
        char        *_n_nptr[2];   /* allows overlaying */
    } _n;
    long            n_value;       /* value of symbol */
    short           n_scnum;       /* section number */
    unsigned short  n_type;        /* type and derived type */
    char            n_sclass;      /* storage class */
```

```
        char            n_numaux;      /* number of aux entries */
        char            n_pad1;        /* pad to 4 byte multiple */
        char            n_pad2;        /* pad to 4 byte multiple */
    };
    };
    };

    #define  n_name     _n._n_name
    #define  n_zeroes   _n._n_n._n_zeroes
    #define  n_offset   _n._n_n._n_offset
    #define  n_nptr     _n._n_nptr[1]
```

Meaningful values and their explanations can be found in syms.h;. anyone who needs to interpret the entries should seek more information there. Some symbols require more information than a single entry; they are followed by *auxiliary entries* that are the same size as a symbol entry. The format follows:

```
    union auxent
    {
         struct
         {
              long        x_tagndx;
              union
              {
                   struct
                   {
                        unsigned shortx_lnno;
                        unsigned shortx_size;
                   } x_lnsz;
                   long  x_fsize;
              } x_misc;
              union
              {
                   struct
                   {
                        long  x_lnnoptr;
                        long  x_endndx;
                   }    x_fcn;
                   struct
                   {
                        unsigned shortx_dimen[DIMNUM];
                   }    x_ary;
              }         x_fcnary;
              unsigned short  x_tvndx;
              char  pad1;
              char  pad2;
         }    x_sym;
         struct
         {
              char  x_fname[FILNMLEN];
         }    x_file;
            struct
            {
                    long    x_scnlen;
```

```
                        unsigned short  x_nreloc;
                        unsigned short  x_nlinno;
                }       x_scn;

            struct
            {
                long        x_tvfill;
                unsigned short    x_tvlen;
                unsigned short    x_tvran[2];
            }   x_tv;
        };
```
-in -2

Indexes of symbol table entries begin at *zero*.

## SEE ALSO
sdb(1), a.out(4), linenum(4).

## CAUTION
Symbols declared as type long are recorded in the symbol table as type int.

## NAME

`system` – format of a kernel description file

## DESCRIPTION

The *system file* contains information about the hardware and system-dependent parameters found on your system. This information is used in conjunction with one or more *master files* as input into the `config`(1M) program. The `config`(1M) program is used to generate a `conf.c` file, which is then compiled and linked with kernel libraries to form a kernel image. A more complete description of the system file is found in *Managing the DG/UX System*.

Each line in a the system file is a separate entry. An entry contains one or more fields, separated by one or more space and/or tab characters. Any line with a number sign (#) in column 1 is treated as a comment and is ignored. Blank lines are also ignored. Each non-comment entry represents a device, STREAMS module, protocol, or tunable sysem parameter. Entries of any type may appear in any order.

### Device Entries

An entry of the form:

*devname*(*parameters*)

or

*devname*@*devcode*(*parameters*)

specifies a device or pseudo-device to be configured into the kernel.

The device name *devname* must be listed in a `$device` section of one of the master files.

The *devcode* notation, if present, specifies that a non-default hardware device code will be used for that device. The device code must appear as a two-digit hexadecimal number.

The *parameters* string represents a specific unit or instantiation of the device; its interpretation is left to the specific device driver. If *parameters* is the null string, the driver's default parameter values will be used. Note that the *parameters* string may itself be a device specification, such as:

sd(insc(),*)

### Protocol Entries

Each single-word entry that matches an entry in a master file's `$protocol` section specifies a socket protocol to be configured into the kernel.

### STREAMS Module Entries

Each single-word entry that matches an entry in a master file's `$stream` section specifies a STREAMS module to be configured into the kernel.

### Tunable Parameter Entries

Each one or two-word entry whose first word matches an entry in a master file's `$keyword` section specifies a tunable system parameter for which a non-default value should be configured into the kernel. The first word of the entry names the parameter that is to be tuned; the second word specifies its value. The value field may be omitted if an implied value is specified in the master file. Note that the implied value may be different from the default value.

## SEE ALSO

`config`(1M), `sysdef`(1M), `master`(4).
*Installing the DG/UX System*, *Customizing the DG/UX System*, *Managing the DG/UX System*.

## NAME

terminfo – terminal and printer capability database

## DESCRIPTION

Terminfo is a compiled database of terminal and printer device capabilities. The capabilities of each type of device are described in a data file that has a name of the following form: /usr/lib/terminfo/?/*, where * stands for the device name and ? stands for the first character of the name. For example,

    /usr/lib/terminfo/d/d215

is the terminfo entry for Data General's DASHER D215 terminal and terminals that behave like it.

Terminfo data files are obtained by compiling source descriptions with the tic(1M) command. Terminfo source descriptions describe, in special code, how basic operations are performed on a terminal or printer. They also describe padding requirements, initialization sequences, and so on. The section entitled "Preparing a Terminfo Description" explains how to build a terminfo source description. Applications such as vi(1) and curses(3X) refer to the compiled terminfo database so that they can work with a variety of terminals without changes to the program code.

Entries in a terminfo source file consist of a number of comma-separated fields. The white space after each comma is ignored. The first line names the device, and the remaining lines describe its capabilities.

### Device Names

The first line of each device description in the terminfo source file gives the names by which terminfo knows the device. Each name is separated by bar ( | ) characters. The first name specifies the most common abbreviation for the device (this is the one to use for the environment variable TERM; see profile(4)). The last name should be a long name that fully identifies the device. All other names are synonyms for the device name. All names but the last should contain no blanks; the last, verbose name may contain blanks for readability.

Device names (except for the verbose entry) should be chosen using the following conventions. First, the particular vendor and model of the device should be specified in the root name, for example, att4425 for the AT&T 4425 terminal. Second, device modes or user preferences should be indicated by appending a hyphen and an indicator of the mode, for example, d410–w for the Data General DASHER D410 series in wide mode (more than 80 columns). See term(5) for examples and more information on choosing names and synonyms.

### Device Capabilities

Lines after the first line of a device description describe the device's capabilities. Terminfo device capabilities are of three general types: boolean capabilities indicate that the device has some particular feature, numeric capabilities specify a numeric value associated with a particular feature, for example, the size of a terminal screen, and string capabilities give a sequence which can be used to perform particular device operations.

In the table below, the variable is the name by which a C programmer (at the terminfo level) accesses the capability. The capname is the short name for this variable used in the text of the database. It is used by a person updating the database and by the tput(1) command when asking what the value of the capability is for a particular device. See Also refers to the numbered subsection in "Terminfo Terminal Capabilities" or the lettered subsection in "Terminfo Printer Capabilities" where the capability is described in detail.

Capability names have no fixed length limit, but an informal limit of 5 characters has been adopted to keep them short. Most of the time, names are chosen to be the same as or similar to the ANSI X3.64-1979 standard. Semantics are also intended to match those of the description.

All string capabilities listed below may have padding described, with the exception of those used for input. Input capabilities, listed under the strings section in the table below, have names beginning with `key_`. The following indicators may appear at the end of the description for a variable.

(G) indicates that the string needs to be instantiated by `tparm()` with arguments (parms) as given ($\#_i$ as described below). `Tparm()` will substitute the arguments into the string to create a customized version. (See `curses`(3X) for more information on `tparm()` and the strings it creates.)

(*) indicates that padding may be based on the number of lines affected.

($\#_i$) indicates the $i^{\text{th}}$ parameter.

| Variable | Cap-name | See Also | Description |
|---|---|---|---|
| `Boolean Capabilities:` | | | |
| auto_left_margin | bw | 1 | `cub1` wraps back from column 0 |
| auto_right_margin | am | 1,13 | Device has automatic margins |
| back_color_erase | bce | 12 | Screen erased with background color |
| can_change | ccc | 12 | Device can redefine existing color |
| ceol_standout_glitch | xhp | 14 | Standout not erased by overwriting (HP) |
| col_addr_glitch | xhpa | B | Only positive motion for hpa/mhpa |
| cpi_changes_res | cpix | A,G | Character pitch affects resolution |
| cr_cancels_micro_mode | crxm | B | Using cr disables micro mode |
| eat_newline_glitch | xenl | 14 | Newline ignored after 80 columns (Concept) |
| erase_overstrike | eo | 6 | Overstrikes are erased by blanks |
| generic_type | gn | 13 | Generic line type (e.g., dialup, switch) |
| hard_copy | hc | 1 | Hardcopy device |
| hard_cursor | chts | 6 | Cursor is hard to see |
| has_meta_key | km | 13 | Device can send meta-characters (e.g., key sets eighth bit) |
| has_print_wheel | daisy | E | Printer needs operator to change character sets |
| has_status_line | hs | 10 | Terminal has extra "status line" |
| hue_lightness_saturation | hls | 12 | Device uses only HLS color notation (Tektronix) |
| insert_null_glitch | in | 5 | Insert mode distinguishes nulls |
| lpi_changes_res | lpix | A,G | Line pitch affects resolution |
| memory_above | da | 4 | Display may be retained above screen |
| memory_below | db | 4 | Display may be retained below screen |
| move_insert_mode | mir | 5 | Safe to move in insert mode |
| move_standout_mode | msgr | 6 | Safe to move in standout modes |

| | | | |
|---|---|---|---|
| needs_xon_xoff | **nxon** | 14 | Padding won't work, XON/XOFF needed |
| no_esc_ctlc | **xsb** | 14 | Beehive (F1=<ESC>, F2=<Ctrl-C>) |
| non_rev_rmcup | **nrrmc** | 6 | smcup does not reverse rmcup |
| no_pad_char | **npc** | 13 | Pad character doesn't exist |
| over_strike | **os** | 1,6 | Device overstrikes (hardcopy device) |
| prtr_silent | **mc5i** | 13 | Printer won't echo on screen |
| row_addr_glitch | **xvpa** | B | Only positive motion for vpa/mvpa |
| semi_auto_right_margin | **sam** | B | Printing in last column causes cr |
| status_line_esc_ok | **eslok** | 10 | Escape sequences work on status line |
| dest_tabs_magic_smso | **xt** | 13 | Destructive tabs, magic smso character (t1061) |
| tilde_glitch | **hz** | 14 | Hazeltine; can't print tildes (~) |
| transparent_underline | **ul** | 6 | Underline character overstrikes |
| xon_xoff | **xon** | 1,13 | Device uses XON/XOFF handshaking |

Numeric Capabilities:

| | | | |
|---|---|---|---|
| buffer_capacity | **bufsz** | I | Bytes buffered before printing |
| columns | **cols** | 1 | Number of columns in a line |
| dot_vert_spacing | **spinv** | F | Vertical pin spacing (pins/inch) |
| dot_horz_spacing | **spinh** | F | Horizontal dot spacing (dots/inch) |
| init_tabs | **it** | 8 | Initial spacing of tab settings |
| label_height | **lh** | 7 | Number of rows in each soft label |
| label_width | **lw** | 7 | Number of columns in each soft label |
| lines | **lines** | 1 | Number of lines on screen or page |
| lines_of_memory | **lm** | 13 | Lines of memory; variable if 0 |
| magic_cookie_glitch | **xmc** | 6 | Number of blanks left by smso/rmso |
| max_colors | **colors** | 12 | Maximum number of colors on-screen |
| max_micro_address | **maddr** | B | Maximum limit on micro_..._address |
| max_micro_jump | **mjump** | B | Maximum limit on parm_..._micro |
| max_pairs | **pairs** | 12 | Maximum number of color-pairs |
| micro_col_size | **mcs** | A | Horizontal step size in micro mode |
| micro_line_size | **mls** | A | Vertical step size in micro mode |
| no_color_video | **ncv** | 12 | Video attributes unusable with color |
| number_of_pins | **npins** | F | Number of pins in print head |
| num_labels | **nlab** | 7 | Number of soft labels available (starting from 1) |
| output_res_char | **orc** | A | Horizontal resolution (steps/column) |
| output_res_line | **orl** | A | Vertical resolution (steps/line) |
| output_res_horz_inch | **orhi** | A | Horizontal resolution (steps/inch) |
| output_res_vert_inch | **orvi** | A | Vertical resolution (steps/inch) |
| padding_baud_rate | **pb** | 9 | Lowest baud rate requiring padding |
| print_rate | **cps** | I | Average speed (characters/second) |
| virtual_terminal | **vt** | 13 | UNIX system virtual terminal number |
| wide_char_size | **widcs** | A | Character size in double wide mode |
| width_status_line | **wsl** | 10 | Number of columns in status line |

String Capabilities:

| | | | |
|---|---|---|---|
| acs_chars | **acsc** | 11 | Graphic character set pairs aAbBcC (vt100+) |

| | | | |
|---|---|---|---|
| back_tab | cbt | 8 | Back tab |
| bell | bel | 1 | Audible signal (bell) |
| carriage_return | cr | 1,9 | Carriage return (*) |
| change_char_pitch | cpi | A,G | Set pitch to #1 characters/inch (G) |
| change_line_pitch | lpi | A,G | Set pitch to #1 lines/inch (G) |
| change_res_horz | chr | A | Set horizontal resolution to #1 (G) |
| change_res_vert | cvr | A | Set vertical resolution to #1 (G) |
| change_scroll_region | csr | 4 | Scrolling area lines #1 through #2 (vt100) (G) |
| char_padding | rmp | 5 | Like ip but when in replace mode |
| char_set_names | csnm | E | Name of character set #1 (G) |
| clear_all_tabs | tbc | 8 | Clear all tab stops |
| clear_margins | mgc | 8 | Clear left and right soft margins |
| clear_screen | clear | 1 | Clear screen and home cursor (*) |
| clr_bol | el1 | 3 | Clear to beginning of line |
| clr_eol | el | 3,14 | Clear to end of line |
| clr_eos | ed | 3 | Clear to end of display (*) |
| column_address | hpa | 2 | Horizontal position to column #1 (G) |
| command_character | cmdch | 13 | Prototype settable command character |
| cursor_address | cup | 2 | Move cursor to row #1, column #2 (G) |
| cursor_down | cud1 | 1 | Move cursor down one line |
| cursor_home | home | 2 | Home cursor (especially if no cup) |
| cursor_invisible | civis | 6 | Make cursor invisible |
| cursor_left | cub1 | 1 | Move cursor left one space |
| cursor_mem_address | mrcup | 2 | Like cup but memory relative (G) |
| cursor_normal | cnorm | 6 | Make cursor normal (undo civis/cvvis) |
| cursor_right | cuf1 | 1 | Move cursor right one space (non-destructive) |
| cursor_to_ll | ll | 2 | Move cursor to column 0 of last line |
| cursor_up | cuu1 | 2 | Move cursor up one line |
| cursor_visible | cvvis | 6 | Make cursor very visible |
| define_char | defc | E | Define character #1 with width #2 and descender #3 (G) |
| delete_character | dch1 | 5 | Delete character (*) |
| delete_line | dl1 | 4 | Delete line (*) |
| dis_status_line | dsl | 10 | Disable status line |
| down_half_line | hd | 13 | Move cursor down one half-line (forward 1/2 linefeed) |
| ena_acs | enacs | 6 | Initialize alternate character set |
| enter_alt_charset_mode | smacs | 6 | Enable alternate character set mode |
| enter_am_mode | smam | 13 | Enable automatic margins |
| enter_blink_mode | blink | 6 | Enable blinking mode |
| enter_bold_mode | bold | 6 | Enable bold (extra bright) mode |
| enter_ca_mode | smcup | 6 | String to send before using cup |
| enter_delete_mode | smdc | 5 | Begin delete mode |
| enter_dim_mode | dim | 6 | Enable half-bright mode |
| enter_doublewide_mode | swidm | D | Enable double wide printing |
| enter_draft_quality | sdrfq | G | Set draft quality printing |
| enter_insert_mode | smir | 5 | Begin insert mode |

| | | | |
|---|---|---|---|
| enter_italics_mode | **sitm** | D | Enable italics |
| enter_leftward_mode | **slm** | B | Enable leftward carriage motion |
| enter_micro_mode | **smicm** | B | Enable micro motion capabilities |
| enter_near_letter_quality | **snlq** | G | Set near-letter-quality printing |
| enter_normal_quality | **snrmq** | G | Set normal quality printing |
| enter_protected_mode | **prot** | 6 | Enable protected mode |
| enter_reverse_mode | **rev** | 6 | Enable reverse video mode |
| enter_secure_mode | **invis** | 6 | Enable blank mode (invisible text) |
| enter_shadow_mode | **sshm** | D | Enable shadow printing |
| enter_standout_mode | **smso** | 6 | Enable standout mode |
| enter_subscript_mode | **ssubm** | D | Enable subscript printing |
| enter_superscript_mode | **ssupm** | D | Enable superscript printing |
| enter_underline_mode | **smul** | 6 | Enable underscore mode |
| enter_upward_mode | **sum** | B | Enable upward carriage motion |
| enter_xon_mode | **smxon** | 13 | Enable XON/XOFF handshaking |
| erase_chars | **ech** | 5 | Erase #1 characters (G) |
| exit_alt_charset_mode | **rmacs** | 6 | Disable alternate character set mode |
| exit_am_mode | **rmam** | 13 | Disable automatic margins |
| exit_attribute_mode | **sgr0** | 6 | Disable all video attributes (G) |
| exit_ca_mode | **rmcup** | 6 | String to send when done with cup |
| exit_delete_mode | **rmdc** | 5 | End delete mode |
| exit_doublewide_mode | **rwidm** | D | Disable double wide printing |
| exit_insert_mode | **rmir** | 5 | End insert mode |
| exit_italics_mode | **ritm** | D | Disable italics |
| exit_leftward_mode | **rlm** | B | Enable rightward carriage motion (the normal state) |
| exit_micro_mode | **rmicm** | B | Disable micro motion capabilities |
| exit_shadow_mode | **rshm** | D | Disable shadow printing |
| exit_standout_mode | **rmso** | 6 | Disable standout mode |
| exit_subscript_mode | **rsubm** | D | Disable subscript printing |
| exit_superscript_mode | **rsupm** | D | Disable superscript printing |
| exit_underline_mode | **rmul** | 6 | Disable underscore mode |
| exit_upward_mode | **rum** | B | Enable downward carriage motion (the normal state) |
| exit_xon_mode | **rmxon** | 13 | Disable XON/XOFF handshaking |
| flash_screen | **flash** | 6 | Visible bell (must not move cursor) |
| form_feed | **ff** | 13 | Hardcopy device page eject (*) |
| from_status_line | **fsl** | 10 | Return from status line |
| init_1string | **is1** | 8 | Device initialization string 1 |
| init_2string | **is2** | 8 | Device initialization string 2 |
| init_3string | **is3** | 8 | Device initialization string 3 |
| init_file | **if** | 8 | Name of initialization data file |
| init_prog | **iprog** | 8 | Path name of initialization program |
| initialize_color | **initc** | 12 | Define color #1 as RGB #2-#4 (G) |
| initialize_pair | **initp** | 12 | Define color-pair #1 as RGB #2-#7 (G) |
| insert_character | **ich1** | 5 | Insert new blank character |
| insert_line | **il1** | 4 | Add new blank line (*) |
| insert_padding | **ip** | 5 | Padding after character inserted (*) |
| key_a1 | **ka1** | 7 | KEY_A1, Upper left of keypad |

| | | | |
|---|---|---|---|
| key_a3 | **ka3** | 7 | KEY_A3, Upper right of keypad |
| key_b2 | **kb2** | 7 | KEY_B2, Center of keypad |
| key_backspace | **kbs** | 7 | KEY_BACKSPACE, Sent by backspace key |
| key_beg | **kbeg** | 7 | KEY_BEG, Sent by beginning key (beg key) |
| key_btab | **kcbt** | 7 | KEY_BTAB, Sent by back-tab key |
| key_c1 | **kc1** | 7 | KEY_C1, Lower left of keypad |
| key_c3 | **kc3** | 7 | KEY_C3, Lower right of keypad |
| key_cancel | **kcan** | 7 | KEY_CANCEL, Sent by cancel key |
| key_catab | **ktbc** | 7 | KEY_CATAB, Sent by clear-all-tabs key |
| key_clear | **kclr** | 7 | KEY_CLEAR, Sent by clear-screen key (erase key) |
| key_close | **kclo** | 7 | KEY_CLOSE, Sent by close key |
| key_command | **kcmd** | 7 | KEY_COMMAND, Sent by command key (cmd key) |
| key_copy | **kcpy** | 7 | KEY_COPY, Sent by copy key |
| key_create | **kcrt** | 7 | KEY_CREATE, Sent by create key |
| key_ctab | **kctab** | 7 | KEY_CTAB, Sent by clear-tab key |
| key_dc | **kdch1** | 7 | KEY_DC, Sent by delete-character key |
| key_dl | **kdl1** | 7 | KEY_DL, Sent by delete-line key |
| key_down | **kcud1** | 7 | KEY_DOWN, Sent by cursor-down key (down-arrow key) |
| key_eic | **krmir** | 7 | KEY_EIC, Sent by end-insert-mode key |
| key_end | **kend** | 7 | KEY_END, Sent by end key |
| key_enter | **kent** | 7 | KEY_ENTER, Sent by enter/send key |
| key_eol | **kel** | 7 | KEY_EOL, Sent by clear-to-end-of-line key |
| key_eos | **ked** | 7 | KEY_EOS, Sent by clear-to-end-of-screen key |
| key_exit | **kext** | 7 | KEY_EXIT, Sent by exit key |
| key_f0 | **kf0** | 7 | KEY_F(0), Sent by function key F0 |
| key_f1 | **kf1** | 7 | KEY_F(1), Sent by function key F1 |
| key_f2 | **kf2** | 7 | KEY_F(2), Sent by function key F2 |
| key_f3 | **kf3** | 7 | KEY_F(3), Sent by function key F3 |
| key_f4 | **kf4** | 7 | KEY_F(4), Sent by function key F4 |
| key_f5 | **kf5** | 7 | KEY_F(5), Sent by function key F5 |
| key_f6 | **kf6** | 7 | KEY_F(6), Sent by function key F6 |
| key_f7 | **kf7** | 7 | KEY_F(7), Sent by function key F7 |
| key_f8 | **kf8** | 7 | KEY_F(8), Sent by function key F8 |
| key_f9 | **kf9** | 7 | KEY_F(9), Sent by function key F9 |
| key_f10 | **kf10** | 7 | KEY_F(10), Sent by function key F10 |
| key_f11 | **kf11** | 7 | KEY_F(11), Sent by function key F11 |
| key_f13 | **kf13** | 7 | KEY_F(12), Sent by function key F12 |
| key_f14 | **kf14** | 7 | KEY_F(13), Sent by function key F13 |
| key_f14 | **kf14** | 7 | KEY_F(14), Sent by function key F14 |
| key_f15 | **kf15** | 7 | KEY_F(15), Sent by function key F15 |
| key_f16 | **kf16** | 7 | KEY_F(16), Sent by function key F16 |
| key_f17 | **kf17** | 7 | KEY_F(17), Sent by function key F17 |
| key_f18 | **kf18** | 7 | KEY_F(18), Sent by function key F18 |

| | | | |
|---|---|---|---|
| key_f19 | kf19 | 7 | KEY_F(19), Sent by function key F19 |
| key_f20 | kf20 | 7 | KEY_F(20), Sent by function key F20 |
| key_f21 | kf21 | 7 | KEY_F(21), Sent by function key F21 |
| key_f22 | kf22 | 7 | KEY_F(22), Sent by function key F22 |
| key_f23 | kf23 | 7 | KEY_F(23), Sent by function key F23 |
| key_f24 | kf24 | 7 | KEY_F(24), Sent by function key F24 |
| key_f25 | kf25 | 7 | KEY_F(25), Sent by function key F25 |
| key_f26 | kf26 | 7 | KEY_F(26), Sent by function key F26 |
| key_f27 | kf27 | 7 | KEY_F(27), Sent by function key F27 |
| key_f28 | kf28 | 7 | KEY_F(28), Sent by function key F28 |
| key_f29 | kf29 | 7 | KEY_F(29), Sent by function key F29 |
| key_f30 | kf30 | 7 | KEY_F(30), Sent by function key F30 |
| key_f31 | kf31 | 7 | KEY_F(31), Sent by function key F31 |
| key_f32 | kf32 | 7 | KEY_F(32), Sent by function key F32 |
| key_f33 | kf33 | 7 | KEY_F(13), Sent by function key F33 |
| key_f34 | kf34 | 7 | KEY_F(34), Sent by function key F34 |
| key_f35 | kf35 | 7 | KEY_F(35), Sent by function key F35 |
| key_f36 | kf36 | 7 | KEY_F(36), Sent by function key F36 |
| key_f37 | kf37 | 7 | KEY_F(37), Sent by function key F37 |
| key_f38 | kf38 | 7 | KEY_F(38), Sent by function key F38 |
| key_f39 | kf39 | 7 | KEY_F(39), Sent by function key F39 |
| key_f40 | kf40 | 7 | KEY_F(40), Sent by function key F40 |
| key_f41 | kf41 | 7 | KEY_F(41), Sent by function key F41 |
| key_f42 | kf42 | 7 | KEY_F(42), Sent by function key F42 |
| key_f43 | kf43 | 7 | KEY_F(43), Sent by function key F43 |
| key_f44 | kf44 | 7 | KEY_F(44), Sent by function key F44 |
| key_f45 | kf45 | 7 | KEY_F(45), Sent by function key F45 |
| key_f46 | kf46 | 7 | KEY_F(46), Sent by function key F46 |
| key_f47 | kf47 | 7 | KEY_F(47), Sent by function key F47 |
| key_f48 | kf48 | 7 | KEY_F(48), Sent by function key F48 |
| key_f49 | kf49 | 7 | KEY_F(49), Sent by function key F49 |
| key_f50 | kf50 | 7 | KEY_F(50), Sent by function key F50 |
| key_f51 | kf51 | 7 | KEY_F(51), Sent by function key F51 |
| key_f52 | kf52 | 7 | KEY_F(52), Sent by function key F52 |
| key_f53 | kf53 | 7 | KEY_F(53), Sent by function key F53 |
| key_f54 | kf54 | 7 | KEY_F(54), Sent by function key F54 |
| key_f55 | kf55 | 7 | KEY_F(55), Sent by function key F55 |
| key_f56 | kf56 | 7 | KEY_F(56), Sent by function key F56 |
| key_f57 | kf57 | 7 | KEY_F(57), Sent by function key F57 |
| key_f58 | kf58 | 7 | KEY_F(58), Sent by function key F58 |
| key_f59 | kf59 | 7 | KEY_F(59), Sent by function key F59 |
| key_f60 | kf60 | 7 | KEY_F(60), Sent by function key F60 |
| key_f61 | kf61 | 7 | KEY_F(61), Sent by function key F61 |
| key_f62 | kf62 | 7 | KEY_F(62), Sent by function key F62 |
| key_f63 | kf63 | 7 | KEY_F(63), Sent by function key F63 |
| key_find | kfnd | 7 | KEY_FIND, Sent by find key |
| key_help | khlp | 7 | KEY_HELP, Sent by help key |
| key_home | khome | 7 | KEY_HOME, Sent by home key |
| key_ic | kich1 | 7 | KEY_IC, Sent by insert-character key |

| | | | |
|---|---|---|---|
| | | | (enter-insert-mode key) |
| key_il | kil1 | 7 | KEY_IL, Sent by insert-line key |
| key_left | kcub1 | 7 | KEY_LEFT, Sent by cursor-left key (left-arrow key) |
| key_ll | kll | 7 | KEY_LL, Sent by home-down key |
| key_mark | kmrk | 7 | KEY_MARK, Sent by mark key |
| key_message | kmsg | 7 | KEY_MESSAGE, Sent by message key |
| key_move | kmov | 7 | KEY_MOVE, Sent by move key |
| key_next | knxt | 7 | KEY_NEXT, Sent by next-object key |
| key_npage | knp | 7 | KEY_NPAGE, Sent by next-page key |
| key_open | kopn | 7 | KEY_OPEN, Sent by open key |
| key_options | kopt | 7 | KEY_OPTIONS, Sent by options key |
| key_ppage | kpp | 7 | KEY_PPAGE, Sent by previous-page key |
| key_previous | kprv | 7 | KEY_PREVIOUS, Sent by previous-object key |
| key_print | kprt | 7 | KEY_PRINT, Sent by print key (copy key) |
| key_redo | krdo | 7 | KEY_REDO, Sent by redo key |
| key_reference | kref | 7 | KEY_REFERENCE, Sent by reference key (ref key) |
| key_refresh | krfr | 7 | KEY_REFRESH, Sent by refresh key |
| key_replace | krpl | 7 | KEY_REPLACE, Sent by replace key |
| key_restart | krst | 7 | KEY_RESTART, Sent by restart key |
| key_resume | kres | 7 | KEY_RESUME, Sent by resume key |
| key_right | kcuf1 | 7 | KEY_RIGHT, Sent by cursor-right key (right-arrow key) |
| key_save | ksav | 7 | KEY_SAVE, Sent by save key |
| key_sbeg | kBEG | 7 | KEY_SBEG, Sent by shifted beginning key |
| key_scancel | kCAN | 7 | KEY_SCANCEL, Sent by shifted cancel key |
| key_scommand | kCMD | 7 | KEY_SCOMMAND, Sent by shifted command key (cmd key) |
| key_scopy | kCPY | 7 | KEY_SCOPY, Sent by shifted copy key |
| key_screate | kCRT | 7 | KEY_SCREATE, Sent by shifted create key |
| key_sdc | kDC | 7 | KEY_SDC, Sent by shifted delete-character key |
| key_sdl | kDL | 7 | KEY_SDL, Sent by shifted delete-line key |
| key_select | kslt | 7 | KEY_SELECT, Sent by select key |
| key_send | kEND | 7 | KEY_SEND, Sent by shifted end key |
| key_seol | kEOL | 7 | KEY_SEOL, Sent by shifted clear-to-end-of-line key |
| key_sexit | kEXT | 7 | KEY_SEXIT, Sent by shifted exit key |
| key_sf | kind | 7 | KEY_SF, Sent by scroll-forward key (scroll-down key) |
| key_sfind | kFND | 7 | KEY_SFIND, Sent by shifted find key |
| key_shelp | kHLP | 7 | KEY_SHELP, Sent by shifted help key |

| | | | |
|---|---|---|---|
| key_shome | **kHOM** | 7 | KEY_SHOME, Sent by shifted home key |
| key_sic | **kIC** | 7 | KEY_SIC, Sent by shifted input key |
| key_sleft | **kLFT** | 7 | KEY_SLEFT, Sent by shifted cursor-left key (left-arrow key) |
| key_smessage | **kMSG** | 7 | KEY_SMESSAGE, Sent by shifted message key |
| key_smove | **kMOV** | 7 | KEY_SMOVE, Sent by shifted move key |
| key_snext | **kNXT** | 7 | KEY_SNEXT, Sent by shifted next key |
| key_soptions | **kOPT** | 7 | KEY_SOPTIONS, Sent by shifted options key |
| key_sprevious | **kPRV** | 7 | KEY_SPREVIOUS, Sent by shifted previous-object key |
| key_sprint | **kPRT** | 7 | KEY_SPRINT, Sent by shifted print key |
| key_sr | **kri** | 7 | KEY_SR, Sent by scroll-backward key (scroll-up key) |
| key_sredo | **kRDO** | 7 | KEY_SREDO, Sent by shifted redo key |
| key_sreplace | **kRPL** | 7 | KEY_SREPLACE, Sent by shifted replace key |
| key_sright | **kRIT** | 7 | KEY_SRIGHT, Sent by shifted cursor-right key (right-arrow key) |
| key_srsume | **kRES** | 7 | KEY_SRSUME, Sent by shifted resume key |
| key_ssave | **kSAV** | 7 | KEY_SSAVE, Sent by shifted save key |
| key_ssuspend | **kSPD** | 7 | KEY_SSUSPEND, Sent by shifted suspend key |
| key_stab | **khts** | 7 | KEY_STAB, Sent by set-tab key |
| key_sundo | **kUND** | 7 | KEY_SUNDO, Sent by shifted undo key |
| key_suspend | **kspd** | 7 | KEY_SUSPEND, Sent by suspend key |
| key_undo | **kund** | 7 | KEY_UNDO, Sent by undo key |
| key_up | **kcuu1** | 7 | KEY_UP, Sent by cursor-up key (up-arrow key) |
| keypad_local | **rmkx** | 7 | Disable "keypad-transmit" mode |
| keypad_xmit | **smkx** | 7 | Enable "keypad-transmit" mode |
| lab_f0 | **lf0** | 7 | Label on function key F0 if not F0 |
| lab_f1 | **lf1** | 7 | Label on function key F1 if not F1 |
| lab_f2 | **lf2** | 7 | Label on function key F2 if not F2 |
| lab_f3 | **lf3** | 7 | Label on function key F3 if not F3 |
| lab_f4 | **lf4** | 7 | Label on function key F4 if not F4 |
| lab_f5 | **lf5** | 7 | Label on function key F5 if not F5 |
| lab_f6 | **lf6** | 7 | Label on function key F6 if not F6 |
| lab_f7 | **lf7** | 7 | Label on function key F7 if not F7 |
| lab_f8 | **lf8** | 7 | Label on function key F8 if not F8 |
| lab_f9 | **lf9** | 7 | Label on function key F9 if not F9 |
| lab_f10 | **lf10** | 7 | Label on function key F10 if not F10 |
| label_off | **rmln** | 7 | Disable soft labels |
| label_on | **smln** | 7 | Enable soft labels |
| meta_off | **rmm** | 13 | Disable "meta mode" |
| meta_on | **smm** | 13 | Enable "meta mode" (eight-bit I/O) |

| | | | |
|---|---|---|---|
| micro_column_address | mhpa | B | Like column_address for micro adjustment (G) |
| micro_down | mcud1 | B | Like cursor_down for micro adjustment |
| micro_left | mcub1 | B | Like cursor_left for micro adjustment |
| micro_right | mcuf1 | B | Like cursor_right for micro adjustment |
| micro_row_address | mvpa | B | Like row_address for micro adjustment (G) |
| micro_up | mcuu1 | B | Like cursor_up for micro adjustment |
| newline | nel | 1 | Newline (like CR followed by LF) |
| order_of_pins | porder | F | Matches data bits to print head pins |
| orig_colors | oc | 12 | Set all color(-pair)s to defaults |
| orig_pair | op | 12 | Set color-pair to the default (G) |
| pad_char | pad | 13 | Pad character (rather than null) |
| parm_dch | dch | 5 | Delete #1 characters (G*) |
| parm_delete_line | dl | 4 | Delete #1 lines (G*) |
| parm_down_cursor | cud | 1 | Move cursor down #1 lines (G*) |
| parm_down_micro | mcud | B | Like parm_down_cursor for micro adjustment (G) |
| parm_ich | ich | 4 | Insert #1 blank characters (G*) |
| parm_index | indn | 1 | Scroll forward #1 lines (G) |
| parm_insert_line | il | 4 | Add #1 new blank lines (G*) |
| parm_left_cursor | cub | 1 | Move cursor left #1 spaces (G) |
| parm_left_micro | mcub | B | Like parm_left_cursor for micro adjustment (G) |
| parm_right_cursor | cuf | 1 | Move cursor right #1 spaces (G*) |
| parm_right_micro | mcuf | B | Like parm_right_cursor for micro adjustment (G) |
| parm_rindex | rin | 1 | Scroll backward #1 lines (G) |
| parm_up_cursor | cuu | 1 | Move cursor up #1 lines (G*) |
| parm_up_micro | mcuu | B | Like parm_up_cursor for micro adjustment (G) |
| pkey_key | pfkey | 7 | Program PFkey #1 to type #2 (G) |
| pkey_local | pfloc | 7 | Program PFkey #1 to execute #2 (G) |
| pkey_xmit | pfx | 7 | Program PFkey #1 to transmit #2 (G) |
| plab_norm | pln | 7 | Program soft label #1 to show #2 (G) |
| print_screen | mc0 | 13 | Print contents of screen |
| prtr_non | mc5p | 13 | Enable printer for #1 bytes |
| prtr_off | mc4 | 13 | Disable printer |
| prtr_on | mc5 | 13 | Enable printer |
| repeat_char | rep | 13 | Repeat character #1 #2 times (G*) |
| req_for_input | rfi | 13 | Send next input character (for ptys) |
| reset_1string | rs1 | 8 | Device full reset string 1 |
| reset_2string | rs2 | 8 | Device full reset string 2 |
| reset_3string | rs3 | 8 | Device full reset string 3 |
| reset_file | rf | 8 | Name of file containing reset string |
| restore_cursor | rc | 4,10 | Move cursor to position of last sc |

| | | | |
|---|---|---|---|
| row_address | vpa | 2 | Vertical position to row #1 (G) |
| save_cursor | sc | 4,10 | Save cursor position for next rc |
| scroll_forward | ind | 1 | Scroll text up one line |
| scroll_reverse | ri | 1 | Scroll text down one line |
| select_char_set | scs | E | Select character set #1 (G) |
| set_attributes | sgr | 6 | Define video attributes #1-#9 (G) |
| set_background | setb | 12 | Set active background color to #1 (G) |
| set_bottom_margin | smgb | C | Set bottom margin at current line |
| set_bottom_margin_parm | smgbp | C | Set bottom margin at line #1 or #2 lines from bottom (G) |
| set_color_pair | scp | 12 | Set current color-pair to #1 (G) |
| set_foreground | setf | 12 | Set active foreground color to #1 (G) |
| set_left_margin | smgl | 8 | Set soft left margin |
| set_left_margin_parm | smglp | C | Set left margin at column #1 (right margin at #2) (G) |
| set_right_margin | smgr | 8 | Set soft right margin |
| set_right_margin_parm | smgrp | C | Set right margin at column #1 (G) |
| set_tab | hts | 8 | Set tab in all rows, current column |
| set_top_margin | smgt | C | Set top margin at current line |
| set_top_margin_parm | smgtp | C | Set top margin at line #1 (bottom margin at line #2) (G) |
| set_window | wind | 4 | Set current window to lines #1-#2, columns #3-#4 (G) |
| start_bit_image | sbim | F | Start printing bit image graphics, #1 dots wide (G) |
| start_char_set_def | scsd | E | Start defining character set #1, containing #2 characters (G) |
| stop_bit_image | rbim | F | End printing bit image graphics |
| stop_char_set_def | rcsd | E | End defining character set #1 (G) |
| subscript_characters | subcs | D | "Subscript-able" characters |
| superscript_characters | supcs | D | "Superscript-able" characters |
| tab | ht | 8 | Tab to next hardware tab stop |
| these_cause_cr | docr | B | Any of these characters causes cr |
| to_status_line | tsl | 10 | Go to status line, column #1 (G) |
| underline_char | uc | 6 | Underscore character and move past |
| up_half_line | hu | 13 | Move up one half-line (reverse 1/2 linefeed) |
| xoff_character | xoffc | 13 | XOFF character |
| xon_character | xonc | 13 | XON character |
| zero_motion | zerom | B | No motion for subsequent character |

## PREPARING A TERMINFO DESCRIPTION

At a mininum for a terminal, a `terminfo` source file should specify capabilities to do the following:

- Clear the screen
- Specify screen size
- Specify how to scroll the screen
- Specify how to move the cursor to any point on the screen
- Display whatever graphic embellishments are available (e.g., reverse video)
- Specify whether the cursor wraps around when it reaches the end of a line
- Specify a scrolling region, if possible
- Insert and delete lines and characters, if available

- Save and restore the cursor position, if possible
- Describe special keys, if any
- Specify how to handle special cases of terminal behavior, if any

The most effective way to prepare a new device description is by imitating the description of a similar device in `terminfo` and building up the new description gradually, testing whether `vi`(1) works with the compiled description. That is, first create a `terminfo` source file that includes what you have determined to be the minimum set of capabilities needed for the new device. Next, compile the source with the `tic`(1M) command. Use `vi`(1) and determine whether the device displays what it is supposed to display. Make alterations or add more advanced capabilities to the source file as appropriate, recompile the source, and repeat the test. Repeat this cycle until the description is complete and correct.

You can obtain the source description for a given device by using the `-I` option of `infocmp`(1M). You may copy and edit this description to accurately describe the device that you wish to enter into the `terminfo` database. Most reference manuals for terminals and printers list the codes that make the device perform specific operations. Use these codes to describe capabilities of the new device.

To test a new device description, set the environment variable `TERMINFO` to the pathname of a directory containing the compiled description. Programs will then search that directory for terminal information instead of `/usr/lib/terminfo`. To get the padding for insert-line correct on a terminal (if the manufacturer did not document it) a severe test is to comment out `xon`, edit a large file at 9600 baud with `vi`(1), delete 16 or so lines from the middle of the screen, then hit the `u` key several times quickly. If the display is corrupted, more padding is usually needed. An analagous test can be used for insert-character.

Be aware that a very unusual device may expose deficiencies in the ability of `terminfo` to describe it or the ability of programs such as `vi`(1) to work with that device.

## Similar Devices

If there are two very similar devices, one can be defined as being just like the other with certain exceptions. The string capability `use` can be given with the name of the similar device. The capabilities given before `use` override those in the device type included by `use`.

More than one `use` capability may be specified. Statements that contain `use` exhibit left-to-right precedence. That is, the earliest `use` statement has priority when more than one statement defines the same capability.

A capability can be canceled by placing @ to the left of the capability definition. For example:

```
att4424-2|Teletype 4424 in display function group ii,
    rev@, sgr@, smul@, use=att4424,
```

defines an AT&T 4424 terminal that does not have the `rev`, `sgr`, and `smul` capabilities, and hence cannot do highlighting. This is useful for different modes of a device, or for different user preferences.

## Parameterized Strings

Cursor addressing and other strings requiring parameters for the device are described by a parameterized string capability, with `printf`(3S)-like escapes (`%x`) in it. The parameter mechanism uses a stack and special `%` codes to manipulate it in the manner of a Reverse Polish Notation (postfix) calculator.

Typically a sequence pushes one of the parameters onto the stack and then prints it in some format. When a sequence pushes a value, the value is placed onto the top of the `terminfo` stack, leaving the source unchanged. The complement to a "push" is the "pop", which removes the topmost value from the `terminfo` stack, storing it elsewhere or using it in the current calculation.

**Stack and Variable Manipulation**

Parameterized strings can access arguments passed to `tparm()`. The arguments are referenced positionally, by number from `1` to `9`. `Terminfo` also provides 52 variables that parameterized strings can use. The variables are referenced by letter from `a` to `z` and from `A` to `Z`. The lowercase variable names represent automatic variables that do not retain their values between parameterized strings. The uppercase variable names represent static variables that do retain their values.

| | |
|---|---|
| %p[1-9] | Push the indicated parameter. |
| %'*c*' | Push the character constant '*c*'. |
| %{*n*} | Push the one or two digit decimal number constant *n*. |
| %P[a-zA-Z] | Pop the stack into the indicated variable. |
| %g[a-zA-Z] | Push the current contents of the indicated variable. |

**Printing Operations**

The following escapes print a value in a specified format.

%% Print the '%' character.

%c Pop the stack and print the value without interpretation, that is, as a single character.

%[[:]*flags*][*width*[.*precision*]][doxXs]

Pop the stack and print the value as a formatted string, converting to decimal (`d`), octal (`o`), lowercase hexadecimal (`x`), uppercase hexadecimal (`X`), or character (`s`) data as indicated. For information on the *flags*, *width*, and *precision* fields, and more information on the conversions, consult `printf`(3S). (The *flags* supported are `-`, `+`, `#`, and the space character.)

NOTE: The `-` flag must be preceded by a colon (`:`) to differentiate the flag from the `%-` escape described below.

**Arithmetic Operations**

The following escapes pop one or two operands off the stack, perform some arithmetic operation, and then push the result onto the stack. Binary operations are in postfix form and expect the first operand to be on the top of the stack.

NOTE: Whether arithmetic is signed or unsigned is unspecified.

| | |
|---|---|
| %+ | Push the sum of the two topmost values on the stack. |
| %- | Push the difference of the two topmost values on the stack. |
| %* | Push the product of the two topmost values on the stack. |
| %/ | Push the quotient of the two topmost values on the stack. |
| %m | Push the modulus of the two topmost values on the stack. |
| %& | Push the bitwise AND of the two topmost values on the stack. |
| %\| | Push the bitwise OR of the two topmost values on the stack. |
| %^ | Push the bitwise exclusive OR of the two topmost values on the stack. |
| %~ | Bitwise complement the topmost value on the stack. |

**Logical Operations**

The following escapes are like arithmetic operations except that they return boolean values. They pop one or two operands off the stack, perform some logical operation, and then push the result onto the stack. Possible results are 0 for `FALSE`, or 1 for `TRUE`.

NOTE: For logical operands, any nonzero value is considered `TRUE`.

| | |
|---|---|
| %= | Push `TRUE` if the two topmost operands are numerically equal. |
| %> | Push `TRUE` if the topmost operand is greater than the second operand. |
| %< | Push `TRUE` if the topmost operand is less than the second operand. |
| %A | Push `TRUE` if the two topmost operands are both logically `TRUE` (AND). |
| %O | Push `TRUE` if either of the two topmost operands are logically `TRUE` (OR). |
| %! | Logically invert the topmost operand (NOT). |

**Miscellaneous Operations**

| | |
|---|---|
| %l | Pop the stack, then push the length of the string indicated by that value. This escape is similar to `strlen`(3C). |
| %i | Add one to the first two parameters passed to `tparm()`, or to the single parameter if just one was passed. This is useful for ANSI terminals, which number cursor positions starting from one instead of zero. |

%?*expr*%t*then*%;
%?*expr*%t*then*%e*else*%;

"If-Then" and "If-Then-Else" (conditional) statements. *Expr*, *then*, and *else* are all parameterized substrings. In operation, `terminfo` evaluates *expr* and then pops the stack. If the popped value is logically `TRUE`, *then* is evaluated. Otherwise, if *else* was provided, *else* is evaluated. (*expr* typically calculates some logical expression, and *then* and *else* typically print corresponding strings.)

"If-Then-ElseIf" conditionals can be written as a string of "If-Then-Else" statements ala Algol 68, that is:

`%?` *c1* `%t` *b1* `%e` *c2* `%t` *b2* ... `%e` *cN* `%t` *bN* `%e` *E* `%;`

where *c[1-N]* are conditionals like *expr*, *b[1-N]* are bodies like *then*, and *E* is a body like *else*.

## A Sample Entry

The following entry, which describes the Concept-100 terminal, is among the more complex entries in the `terminfo` file as of this writing. It is provided here to illustrate the form and content of a `terminfo` entry, and to provide a point of reference for the text that follows.

```
concept100|c100|concept|c104|c100-4p|concept 100,
    am, db, eo, in, mir, ul, xenl,
    cols#80, lines#24, pb#9600, vt#8,
    bel=^G, blank=\EH, blink=\EC, clear=^L$<2*>, cnorm=\Ew, cr=^M$9,
    cub1=^H, cud1=^J, cuf1=\E=, cup=\Ea%p1%' '%+%c%p2%' '%+%c,
    cuu1=\E;, cvvis=\EW, dch1=\E^A$<16*>, dim=\EE, dl1=\E^B$<3*>,
    ed=\E^C$<16*>, el=\E^U$16, flash=\Ek$<20>\EK, ht=\t$8, il1=\E^R$<3*>,
    .ind=^J$9, ind=^J, ip=$<16*>,
    is2=\EU\Ef\E7\E5\E8\El\ENH\EK\E\0\Eo&\0\Eo\47\E, kbs=^h, kcub1=\E>,
    kcud1=\E<, kcuf1=\E=, kcuu1=\E;, kf1=\E5, kf2=\E6, kf3=\E7, khome=\E?,
    prot=\EI, rep=\Er%p1%c%p2%' '%+%c$<.2*>, rev=\ED,
    rmcup=\Ev\s\s\s\s$<6>\Ep\r\n, rmir=\E\0, rmkx=\Ex, rmso=\Ed\Ee,
    rmul=\Eg, rmul=\Eg, sgr0=\EN\0, smcup=\EU\Ev\s\s8p\Ep\r, smir=\E^P,
    smkx=\EX, smso=\EE\ED, smul=\EG,
```

Entries may continue onto multiple lines by placing white space at the beginning of each line except the first. Lines beginning with "#" are interpreted as comments.

### How to Describe Device Capabilities

In the example, the boolean capabilities appear in the second line. The numeric capabilities appear in the line that follows the booleans. The remainder of the entry consists of string capabilities.

The fact that a device has "automatic margins" (that is, an automatic return and linefeed when the end of a line is reached) is indicated by the boolean capability `am`. Thus, the device description simply gives `am`. Numeric capabilities are followed by the character '#' and then the value assigned. Thus `cols`, which indicates the number of columns the device has, specifies the value `80` for the Concept 100 as `cols#80`. The value may be specified in decimal, octal, or hexadecimal using normal C conventions. Finally, string-valued capabilities, such as `bel` (sound an audible alarm) are specified by the two- to five-character capability name, or capname for short, an '=', and then a string ending at the next following comma. The concept 100 responds to `<Ctrl-G>` by sounding its bell, so the description specifies `bel=^G`.

A delay in milliseconds may appear anywhere in a string capability, bracketed by `$<..>`, as in `el=\EK$<3>`. Padding characters are supplied by `tputs()` (see `curses`(3X)) to provide this delay. The delay can be either a number (for example, `20`); or a number followed by an '*' (for example, `3*`), a '/' (for example, `5/`), or both (for example, `10*/`). A '*' indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. (In the case of insert character, the factor is still the number of lines affected. This is always 1 unless the terminal has `in` defined and the software uses it.) When an '*' is specified, it is sometimes useful to give a delay of the form `3.5` to specify a delay per unit to tenths of milliseconds. (Only one decimal place is allowed.) A '/' indicates that the padding is mandatory. Otherwise, if the device has `xon` defined, the padding information is advisory and is only used for cost estimates or when the device is in raw mode. Mandatory padding is transmitted regardless of the setting of `xon`.

A number of escape sequences are provided in the string valued capabilities for easy encoding of characters there. Both `\E` and `\e` map to an ESCAPE character, `^x` maps to a `<Ctrl-x>` for any appropriate *x*, and the sequences `\n`, `\l`, `\r`, `\t`, `\b`, `\f`, and `\s` give a newline, linefeed, return, tab, backspace, formfeed, and space, respectively. Other escapes include: `\^` for caret (^); `\\` for backslash (\); `\,` for comma (,); `\:` for colon (:); and `\0` for null. (`\0` actually produces `\200`, which does not terminate a string but behaves as a null character on most devices.) Finally, characters may be given as three octal digits after a backslash (e.g., `\123`).

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the first `ind` in the example above. Note that when capabilities are defined more than once, a prior definition overrides a later definition.

## TERMINFO TERMINAL CAPABILITIES

The following subsections describe `terminfo` terminal capabilities in detail. Subsections are numbered for cross-reference to the table that appears earlier in this man page.

## 1. Basic Capabilities

The number of columns on each line for the terminal is given by the `cols` numeric capability. If the terminal has a screen, then the number of lines on the screen is given by the `lines` capability. If the terminal cursor wraps around to the beginning of the next line when it reaches the right margin, then the `am` capability should be given. If the terminal can clear its screen, leaving the cursor in the home position, then this is given by the `clear` string capability. If the terminal overstrikes (rather than clearing a position when a character is overwritten) then it should have the `os` capability. If the terminal is a printing terminal, with no soft copy unit, give it both `hc` and `os`. (`os` applies to storage scope terminals, such as the Tektronix 4010 series, as well as hardcopy and APL terminals.) If there is a code to move the cursor to the left edge of the current row, give this as `cr`. (Normally this is carriage return, `^M`.) If there is a code to produce an audible signal (bell, beep, etc) give this as `bel`. If the terminal uses the XON-XOFF flow control protocol, like most terminals, specify the boolean capability `xon`.

If there is a code to move the cursor one position to the left (such as backspace) that capability should be given as `cub1`. Similarly, codes to move to the right, up, and down should be given as `cuf1`, `cuu1`, and `cud1`. These local cursor motions should not alter the text they pass over; for example, you would not normally use `cuf1=\s` because the space would erase the character moved over.

It is important to remember that the local cursor motions encoded in `terminfo` are undefined at the left and top edges of a screen terminal. Programs should never attempt to backspace around the left edge, unless `bw` is specified, and should never attempt to move the cursor up locally off the top.

To scroll text up, a program moves the cursor to the bottom left corner of the screen and sends the `ind` (index) string. To scroll text down, a program moves the cursor to the top left corner of the screen and sends the `ri` (reverse index) string. The strings `ind` and `ri` are undefined when the cursor is not on their respective corners of the screen.

Parameterized versions of the scrolling sequences are `indn` and `rin` which have the same semantics as `ind` and `ri` except that they take one parameter, and scroll that many lines. They are also undefined except at the appropriate corners of the screen.

The `am` capability tells whether the cursor sticks at the right edge of the screen when text is output, but this does not necessarily apply to a `cuf1` from the last column. The only local motion which is defined from the left edge is if `bw` is given, then a `cub1` from the left edge moves to the right edge of the previous row. If `bw` is not given, the effect is undefined. `bw` is useful for drawing a box around the edge of the screen, for example. If the terminal has switch selectable automatic margins, the `terminfo` file usually assumes that this is on; i.e., `am`. If the terminal has a command which moves to the first column of the next line, that command can be given as `nel` (newline). It does not matter if the command clears the remainder of the current line, so if the terminal has no `CR` and `LF` it may still be possible to craft a working `nel` out of one or both of them.

These capabilities suffice to describe hardcopy and screen terminals. Thus the model 33 teletype is described as follows:

```
33|tty33|tty|model 33 teletype,
    bel=^G, cols#72, cr=^M, cud1=^J, hc, ind=^J, os,
```

The Lear Siegler ADM-3 is described as follows:

```
adm3|lsi adm3,
    am, bel=^G, clear=^Z, cols#80, cr=^M, cub1=^H,
    cud1=^J, ind=^J, lines#24,
```

2. **Cursor Motions**

   If the terminal has a fast way to home the cursor (to the very upper left corner of the screen) then this can be given as `home`; similarly a fast way of getting to the lower left-hand corner can be given as `ll`; this may involve going up with `cuu1` from the home position, but a program should never do this itself (unless `ll` does) because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as addressing to (0,0): to the top left corner of the screen, not of memory. (Thus, the `\EH` sequence on Hewlett-Packard terminals cannot be used for `home` without losing some of the other features on the terminal.)

   If the terminal has a way to move the cursor to any selected position on the screen, specify this with the `cup` string capability, which takes two parameters: the row and column of the new cursor position. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory.) If the terminal has memory relative cursor addressing, that can be indicated by the string capability `mrcup`.

   If the terminal has row or column absolute cursor addressing, these can be given as single parameter capabilities `hpa` (horizontal position absolute) and `vpa` (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to `cup`. If there are parameterized local motions (e.g., move *n* spaces to the right) these can be given as `cud`, `cub`, `cuf`, and `cuu` with a single parameter indicating how many spaces to move. These are primarily useful if the terminal does not have `cup`, as with the Tektronix 4025.

3. **Area Clears**

   If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as `el`. If the terminal can clear from the beginning of the line to the current position inclusive, leaving the cursor where it is, this should be given as `el1`. If the terminal can clear from the current position to the end of the display, then this should be given as `ed`. `ed` is only defined from the first column of a line. (Thus, it can be simulated by a request to delete a large number of lines, if a true `ed` is not available.)

4. **Insert/delete line**

   If the terminal can open a new blank line before the line containing the cursor, this should be given as `il1`; this is done only from the first position of a line. The cursor must then appear on the newly blank line. If the terminal can delete the line which the cursor is on, then this should be given as `dl1`; this is done only from the first position on the line to be deleted. Versions of `il1` and `dl1` which take a single parameter and insert or delete that many lines can be given as `il` and `dl`.

   If the terminal has a destructive programmable scrolling region (like the VT100), the command to set the region can be described with the `csr` string capability, which takes two parameters: the top and bottom lines of the scrolling region. It is possible to get the effect of insert or delete line using this command – the `sc` and `rc` (save and restore cursor) string capabilities are also useful. The cursor position is, alas, undefined after using this command. It must be reset using other terminfo capabilities such as `cup`, `home`, or `rc`. Inserting lines at the top or bottom of the screen can also be done using `ri` or `ind` on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

To determine whether a terminal has destructive scrolling regions or non-destructive scrolling regions, create a scrolling region in the middle of the screen, place data on the bottom line of the scrolling region, move the cursor to the top line of the scrolling region, and do a reverse index (`ri`) followed by a delete line (`dl1`) or index (`ind`). If the data that was originally on the bottom line of the scrolling region was restored into the scrolling region by the `dl1` or `ind`, then the terminal has non-destructive scrolling regions. Otherwise, it has destructive scrolling regions. Do not specify `csr` if the terminal has non-destructive scrolling regions, unless `ind`, `ri`, `indn`, `rin`, `dl`, and `dl1` all simulate destructive scrolling.

If the terminal has the ability to define a window as part of memory, which all commands affect, it should be given as the parameterized string `wind`. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order.

If the terminal can retain display memory above, then the `da` boolean capability should be given; if display memory can be retained below, then `db` should be given. These indicate that deleting a line or scrolling a full screen may bring non-blank lines up from below or that scrolling back with `ri` may bring down non-blank lines.

5. **Insert/Delete Character**

There are two basic kinds of intelligent terminals with respect to insert/delete character operations which can be described using `terminfo`. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly (i.e., all characters to the right of the insertion or deletion shift as a unit). Other terminals, such as the Concept-100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks.

You can determine the kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type "`abc    def`" using local cursor motions (not spaces) between the `abc` and the `def`. Then position the cursor before the `abc` and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to "fall off" the end, then your terminal does not distinguish between blanks and untyped positions. If the `abc` shifts over to the `def` which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal, and thus you should define the boolean capability `in`, which stands for "insert null". While these are two logically separate attributes (one line versus multiline insert mode, and special treatment of untyped spaces), we have seen no terminals whose insert mode cannot be described with the single attribute.

`Terminfo` can describe both terminals which have an insert mode and terminals which send a simple sequence to open a blank position on the current line. Give as `smir` the sequence to get into insert mode. Give as `rmir` the sequence to leave insert mode. Now give as `ich1` any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode do not specify `ich1`; terminals which send a sequence to open a screen position should specify it here. (If your terminal has both, insert mode is usually preferable to `ich1`. Do not give both unless the terminal actually requires both to be used in combination.)

If post-insert padding is needed, give this as a number of milliseconds padding in `ip` (a string capability). Any other sequence that may need to be sent after an insert of a single character may also be given in `ip`. If your terminal needs both to be placed into an 'insert mode' and a special code to precede each inserted character, then both `smir/rmir` and `ich1` can be given, and both are used.

The `ich` capability, with one parameter, *n*, repeats the effects of `ich1` *n* times.

If padding is necessary between characters typed while not in insert mode, give this as a number of milliseconds padding in `rmp`.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (e.g., if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability `mir` to speed up inserting in this case. Omitting `mir` affects only speed. Some terminals (notably Datamedia's) must not have `mir` because of the way their insert mode works.

Finally, you can give `dch1` to delete a single character, `dch` with one parameter, *n*, to delete *n* characters, and `smdc` and `rmdc` to enter and exit delete mode (any mode the terminal needs to be placed in for `dch1` to work).

A command to erase *n* characters (equivalent to outputting *n* blanks without explicitly moving the cursor) can be given as `ech` with one parameter.

6. **Highlighting, Underlining, and Visible Bells**
If your terminal has one or more kinds of display attributes (graphic embellishments to text), these can be represented in a number of different ways. You should choose one display form as "standout mode" (see `curses`(3X)), representing a good, high contrast, easy-on-the-eyes format for highlighting error messages and other attention getters. (If you have a choice, reverse video plus half-bright is good, or reverse video alone; however, different users have different preferences on different terminals.) The sequences to enter and exit standout mode are given as `smso` and `rmso`, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces on the screen, as on the TVI 912 and the Teleray 1061, then `xmc` should be given to tell how many spaces are left.

Codes to begin underlining and end underlining can be given as `smul` and `rmul` respectively. If the terminal has a code to underline the current character and move the cursor one space to the right, such as the Micro-Term MIME, this can be given as `uc`.

Other capabilities to enter various highlighting modes include `blink` (blinking), `bold` (bold or extra-bright), `dim` (dim or half-bright), `invis` (blanking or invisible text), `prot` (protected), `rev` (reverse video), `sgr0` (turn off all attribute modes), `smacs` (enter alternate-character-set mode), and `rmacs` (exit alternate-character-set mode). Turning on any of these modes singly may or may not turn off other modes. If a command is necessary before alternate character set mode is entered, give the sequence in `enacs` (enable alternate-character-set mode).

If there is a sequence to set arbitrary combinations of modes, this should be given as `sgr` (set attributes), taking nine parameters. Each parameter is either zero or non-zero, as the corresponding attribute is on or off. The nine parameters are, in order: standout, underline, reverse, blink, dim, bold, invisible, protected, and alternate character set. Not all modes need be supported by `sgr`, only those for which corresponding separate attribute commands exist. (See the example at the end of this section.)

Terminals with the "magic cookie" glitch (`xmc`) deposit special "cookies" when they receive mode-setting sequences, rather than having extra attribute bits for each character. These "cookies" affect the display algorithm to provide video attributes, but also take up (blank) space on the screen.

Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when the cursor is moved to a new line or is addressed. Programs using standout mode should exit standout mode before moving the cursor or sending a newline, unless the `msgr` capability, asserting that it is safe to move in standout mode, is present.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), then this can be given as `flash`; it must not move the cursor. A good flash can be done by changing the screen into reverse video, padding for 200 ms, then returning the screen to normal video.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to make, for example, a non-blinking underline into an easier to find block or blinking underline) give this sequence as `cvvis`. The boolean `chts` should also be given. If there is a way to make the cursor completely invisible, give that as `civis`. The capability `cnorm` should be given which undoes the effects of either of these modes.

If the terminal needs to be in a special mode when running a program that uses `terminfo` capabilities, the codes to enter and exit this mode can be given as `smcup` and `rmcup`. This arises, for example, from terminals like the Concept-100 with more than one page of memory. If the terminal has only memory relative cursor addressing and not screen relative cursor addressing, a window the size of the screen must be fixed into the terminal for cursor addressing to work properly. This is also used for the Tektronix 4025, where `smcup` sets the command character to the one used by `terminfo`. If the `smcup` sequence does not restore the screen after an `rmcup` sequence is output (to the state prior to outputting `rmcup`), specify the boolean capability `nrrmc`.

If your terminal generates underlined characters by using the underline character (with no special codes needed) even though it does not otherwise overstrike characters, then you should give the capability `ul`. For terminals where a character overstriking another leaves both characters on the screen, give the capability `os`. If overstrikes are erasable with a blank, then this should be indicated by giving `eo`.

Here is an example of highlighting: assume that a terminal needs the following escape sequences to turn on various modes.

| tparm parameter | attribute | escape sequence |
|---|---|---|
| | none | \E[0m |
| p1 | standout | \E[0;4;7m |
| p2 | underline | \E[0;3m |
| p3 | reverse | \E[0;4m |
| p4 | blink | \E[0;5m |
| p5 | dim | \E[0;7m |
| p6 | bold | \E[0;3;4m |
| p7 | invis | \E[0;8m |
| p8 | protect | not available |
| p9 | altcharset | ^O (off) ^N(on) |

Note that each escape sequence requires a `0` to turn off other modes before turning on its own mode. Combinations of attributes are allowed by appending a digit that represents each attribute, separated by a semicolon. For instance, underline + blink needs the sequence `\E[0;3;5m`. Note that, as suggested above, *standout* is set up to be the combination of *reverse* and *dim*. Also, since this terminal has no *bold* mode, *bold* is set up as the combination of *reverse* and *underline*. The terminal doesn't have *protect* mode, either, but that cannot be simulated in any way, so *p8* is ignored. The *altcharset* mode is different in that it requires either `<Ctrl-O>` or `<Ctrl-N>` depending on whether it is to be turned off or on. If all modes were to be turned on, the sequence would be `\E[0;3;4;5;7;8m^N`.

Now look at the cases in which different sequences are output. For example, `;3` is output when either *p2* or *p6* is true, that is, if either *underline* or *bold* modes are turned on. Writing out the above sequences, along with their dependencies, gives the following:

| sequence | when to output | terminfo translation |
|---|---|---|
| \E[0 | always | \E[0 |
| ;3 | if p2 or p6 | %?%p2%p6%\|%t;3%; |
| ;4 | if p1 or p3 or p6 | %?%p1%p3%\|%p6%\|%t;4%; |
| ;5 | if p4 | %?%p4%t;5%; |
| ;7 | if p1 or p5 | %?%p1%p5%\|%t;7%; |
| ;8 | if p7 | %?%p7%t;8%; |
| m | always | m |
| ^N or ^O | if p9 ^N, else ^O | %?%p9%t^N%e^O%; |

Putting this all together into the `sgr` sequence gives:

```
sgr=\E[0%?%p2%p6%|%t;3%;%?%p1%p3%|%p6%|%t;4%;%?%p5%t;5%;
        %?%p1%p5%|%t;7%;%?%p7%t;8%;m%?%p9%t^N%e^O%;,
```

7. Keypad

If the terminal has a keypad that transmits codes when special keys are pressed, this information can be given. Note that it is not possible to handle terminals where the keypad only works in local mode (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, give these codes as `smkx` and `rmkx`. Otherwise the keypad is assumed to always transmit.

The codes sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as `kcub1`, `kcuf1`, `kcuu1`, `kcud1`, and `khome` respectively. If there are function keys such as F0, F1, ..., F63, the codes they send can be given as `kf0`, `kf1`, `...`, `kf63`. If the first 11 keys have labels other than the default F0 through F10, the labels can be given as `lf0`, `lf1`, `...`, `lf10`. The codes transmitted by certain other special keys can be given: `kll` (home down), `kbs` (backspace), `ktbc` (clear all tabs), `kctab` (clear the tab stop in this column), `kclr` (clear screen or erase), `kdch1` (delete character), `kdl1` (delete line), `krmir` (exit insert mode), `kel` (clear to end of line), `ked` (clear to end of screen), `kich1` (insert character or enter insert mode), `kil1` (insert line), `knp` (next page), `kpp` (previous page), `kind` (scroll forward/down), `kri` (scroll backward/up), `khts` (set a tab stop in this column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, the other five keys can be given as `ka1`, `ka3`, `kb2`, `kc1`, and `kc3`. These keys are useful when the effects of a 3 by 3 directional pad are needed. Further keys are defined above in the capabilities list.

Strings to program function keys can be given as `pfkey`, `pfloc`, and `pfx`. A string to program their soft screen labels can be given as `pln`. Each of these strings takes two parameters: the function key number to program (from 0 to 10) and the string to program it with. Function key numbers out of this range may program undefined keys in a terminal-dependent manner. The difference between the capabilities is that `pfkey` causes the given key to act as if the user had typed the given string; `pfloc` causes the string to be executed by the terminal in local mode; and `pfx` causes the string to be transmitted to the computer. The capabilities `nlab`, `lw`, and `lh` define how many soft labels there are and how wide and high they are. If there are commands to turn the labels on and off, give them as `smln` and `rmln`. `smln` is normally output after one or more `pln` sequences to make sure that the change becomes visible.

**8. Tabs and Initialization**

If the terminal has hardware tabs, the command to advance to the next tab stop can be given as `ht` (usually *Ctrl-I*). A "backtab" command which moves leftward to the previous tab stop can be given as `cbt`. By convention, if the terminal driver modes indicate that tabs are being expanded by the computer rather than being sent to the terminal, programs should not use `ht` or `cbt` even if they are present, since the user may not have the tab stops properly set.

If the terminal has hardware tabs which are initially set every *n* spaces when the terminal is powered up, the numeric parameter `it` should be given, showing the number of spaces *n* to which the tabs are set. This is normally used by `tput init` (see `tput`(1)) to determine whether to set the mode for hardware tab expansion and whether to set the tab stops.

If the terminal has tab stops that can be saved in nonvolatile memory, the `terminfo` description can assume that they are properly set. If there are commands to set and clear tab stops, they can be given as `tbc` (clear all tab stops) and `hts` (set a tab stop in the current column of every row).

Other capabilities include: `is1`, `is2`, and `is3`, initialization strings for the terminal; `iprog`, the path name of a program to run to initialize the terminal; and `if`, the name of a file containing long initialization strings. These strings are expected to set the terminal into modes consistent with the rest of the `terminfo` description. They must be sent to the terminal each time the user logs in and be output in the following order: run the program `iprog`; output `is1`; output `is2`; set the margins using `mgc`, `smgl`, and `smgr`; set the tabs using `tbc` and `hts`; print the file `if`; and finally output `is3`. This is usually done using the `init` option of `tput`(1); see `profile`(4).

Most initialization is done with `is2`. Special terminal modes can be set up without duplicating strings by putting the common sequences in `is2` and special cases in `is1` and `is3`. Sequences that do a harder reset from a totally unknown state can be given as `rs1`, `rs2`, `rf`, and `rs3`, analogous to `is1`, `is2`, `if`, and `is3`. (The method using files, `if` and `rf`, is used for a few terminals, from `/usr/lib/tabset/*`; however, the recommended method is to use the initialization and reset strings.) These strings are output by `tput reset`, which is used when the terminal gets into a wedged state. Commands are normally placed in `rs1`, `rs2`, `rs3`, and `rf` only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set a terminal into 80-column mode would normally be part of `is2`, but on some terminals it causes an annoying glitch on the screen and is not normally needed since the terminal is usually already in 80-column mode.

If a more complex sequence is needed to set the tabs than can be described by using `tbc` and `hts`, the sequence can be placed in `is2` or `if`.

If there are commands to set and clear margins, they can be given as `mgc` (clear all margins), `smgl` (set left margin), and `smgr` (set right margin).

9. **Delays**

   Certain capabilities control padding in the terminal driver (see `termio`(7) and `tty`(7)). These are primarily needed by hardcopy terminals, and are used by `tput init` to set terminal driver modes appropriately. Delays embedded in the capabilities `cr`, `ind`, `cub1`, `ff`, and `tab` can be used to set the appropriate delay bits in the terminal driver. If `pb` (padding baud rate) is given, these values can be ignored at baud rates below the value of `pb`.

10. **Status Lines**

    If the terminal has an extra "status line" that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, into which a program can cursor address normally (such as the Heathkit h19's 25th line, or the 24th line of a VT100 which is set to a 23-line scrolling region), the capability `hs` should be given. Special strings that go to a given column of the status line and return from the status line can be given as `tsl` and `fsl`. (`fsl` must leave the cursor position in the same place it was before `tsl`. If necessary, the `sc` and `rc` strings can be included in `tsl` and `fsl` to get this effect.) The capability `tsl` takes one parameter, which is the column number of the new cursor position in the status line.

    If escape sequences and other special commands, such as tab, work while in the status line, the flag `eslok` can be given. A string which turns off the status line (or otherwise erases its contents) should be given as `dsl`. If the terminal has commands to save and restore the position of the cursor, give them as `sc` and `rc`. The status line is normally assumed to be the same width as the rest of the screen, e.g., `cols`. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded) the width, in columns, can be indicated with the numeric parameter `wsl`.

11. **Line Graphics**

    If the terminal has a line drawing alternate character set, the mapping of glyph to character would be given in `acsc`. The definition of this string is based on the alternate character set used in the DEC VT100 terminal, extended slightly with some characters from the AT&T 4410v1 terminal.

| glyph name | vt100+ character |
|---|---|
| arrow pointing right | + |
| arrow pointing left | , |
| arrow pointing down | . |
| solid square block | 0 |
| lantern symbol | I |
| arrow pointing up | – |
| diamond | ‘ |
| checker board (stipple) | a |
| degree symbol | f |
| plus/minus | g |
| board of squares | h |
| lower right corner | j |
| upper right corner | k |

| | |
|---|---|
| upper left corner | l |
| lower left corner | m |
| plus | n |
| scan line 1 | o |
| horizontal line | q |
| scan line 9 | s |
| left tee (├) | t |
| right tee (┤) | u |
| bottom tee (┴) | v |
| top tee | w |
| vertical line | x |
| bullet | ~ |

The best way to describe a new terminal's line graphics set is to add a third column to the above table with the characters for the new terminal that produce the appropriate glyphs when the terminal is in the alternate character set mode. For example,

| glyph name | vt100+ char | new tty char |
|---|---|---|
| upper left corner | l | R |
| lower left corner | m | F |
| upper right corner | k | T |
| lower right corner | j | G |
| horizontal line | q | , |
| vertical line | x | . |

Now write down the characters left to right, as in `acsc=lRmFkTjGq\,x.`

12. **Color Manipulation**

Let us define two methods of color manipulation: the Tektronix method and the HP method. The Tektronix method uses a set of N predefined colors (usually 8) from which a program can select "current" foreground and background colors. Thus a terminal can support up to N colors mixed into N*N color-pairs to be displayed on the screen at the same time. When using an HP method the program cannot define the foreground independently of the background, or vice-versa. Instead, the program must define an entire color-pair at once. Up to M color-pairs, made from 2*M different colors, can be defined this way. Most existing color terminals belong to one of these two classes.

The numeric capabilities `colors` and `pairs` define the number of colors and color-pairs that can be displayed on the screen at the same time. If a terminal can change the definition of a color (for example, the Tektronix 4100 and 4200 series terminals), this should be specified with the boolean capability `ccc` (can change color). To change the definition of a color (Tektronix method), use the parameterized string capability `initc` (initialize color). It requires four parameters: color number (ranging from 0 to `colors-1`) and three RGB (red, green, and blue) values (ranging from 0 to 1000).

Tektronix 4100 series terminals use a type of color notation called HLS (Hue Lightness Saturation) instead of RGB color notation. For such terminals one must define a boolean capability `hls`. The last three parameters of the `initc` string would then be HLS values: H, ranging from 0 to 360; and L and S, ranging from 0 to 100.

To set the current foreground or background to a given color, use parameterized string capabilities `setf` (set foreground) and `setb` (set background). They each require one parameter: the number of the color. To initialize a color-pair (HP method), use initp (initialize pair). It requires seven parameters: the number of a color-pair (ranging from 0 to pairs−1), and six RGB values: three for the foreground followed by three for the background. (When initc or initp is used, RGB or HLS arguments should be in the order "red, green, blue" or "hue, lightness, saturation", respectively.) To make a color-pair current, use the parameterized string capability scp (set color-pair). It takes one parameter, the number of a color-pair.

If a terminal can change the definitions of colors, but uses a color notation different from RGB and HLS, a mapping to either RGB or HLS must be developed and encoded in the initc and initp capabilities.

Some terminals (for example, most color terminal emulators for PCs) erase areas of the screen using the current background color. In such cases, the boolean capability bce (background color erase) should be defined. The string capability op (original pair) contains a sequence for setting the foreground and background colors to what they were at the terminal start-up time. Similarly, oc (original colors) contains a sequence for setting all colors (for the Tektronix method) or color-pairs (for the HP method) to the values they had at the terminal start-up time.

Some video attributes on some color terminals should not be combined with colors. For instance, some color terminals substitute color for video attributes, so each attribute can be displayed in only one color. Information about these video attributes should be packed into the numeric capability ncv (no color video). There is a one-to-one correspondence between the nine least significant bits of this capability and the video attributes. The following table depicts this correspondence.

| Attribute | Bit Position | Decimal Value |
|---|---|---|
| A_STANDOUT | 0 | 1 |
| A_UNDERLINE | 1 | 2 |
| A_REVERSE | 2 | 4 |
| A_BLINK | 3 | 8 |
| A_DIM | 4 | 16 |
| A_BOLD | 5 | 32 |
| A_INVIS | 6 | 64 |
| A_PROTECT | 7 | 128 |
| A_ALTCHARSET | 8 | 256 |

When a particular video attribute should not be used with colors, the corresponding ncv bit should be set to 1; otherwise it should be set to zero. To determine the information to pack into the ncv capability, you must add together the decimal values corresponding to those attributes that cannot coexist with colors. For example, if the terminal uses colors to simulate reverse video (bit number 2 and decimal value 4) and bold (bit number 5 and decimal value 32), the resulting value for ncv will be 36 (4 + 32).

13. **Miscellaneous**

If the terminal requires any character other than a null (zero) as a pad, then this can be given as pad. Only the first character of the pad string is used. If the terminal does not have a pad character, specify npc.

If the terminal can move up or down half a line, this can be indicated with hu (half-line up) and hd (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as ff (usually ^L).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters) this can be indicated with the parameterized string `rep`. The first parameter is the character to be repeated and the second is the number of times to repeat it. Thus, `tparm(repeat_char, 'x', 10)` produces the same effect as `xxxxxxxxxx`.

If the terminal has a programmable command character, such as the Tektronix 4025, this can be indicated with `cmdch`. A prototype command character is chosen which is used in all capabilities. This character is given in the `cmdch` capability to identify it. The following convention is supported on some UNIX systems: If the environment variable `CC` exists, all occurrences of the prototype character are replaced with the character in `CC`.

Terminal descriptions that do not represent a specific kind of known terminal, such as `switch`, `dialup`, `patch`, and `network`, should include the `gn` (generic) capability so that programs can complain that they cannot know how to talk to the terminal. (This capability does not apply to `virtual` terminal descriptions for which the escape sequences are known.) If the terminal is one of those supported by the UNIX system virtual terminal protocol, the terminal number can be given as `vt`. A line-turn-around sequence to be transmitted before doing reads should be specified in `rfi`.

If the terminal uses XON/XOFF handshaking for flow control, define `xon`. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters are not transmitted. Sequences to turn on and off XON/XOFF handshaking may be given in `smxon` and `rmxon`. If the characters used for handshaking are not `<Ctrl-S>` and *Ctrl-Q*, they may be specified with `xonc` and `xoffc`.

If the terminal has a "meta key" which acts as a shift key, setting the eighth bit of any character transmitted, this can be specified with the boolean capability `km`. Otherwise, software assumes that the eighth bit is parity and it is usually cleared. If strings exist to turn this "meta mode" on and off, they can be specified as `smm` and `rmm`.

If the terminal has more lines of memory than can fit on the screen at once, the number of lines of memory can be indicated with `lm`. A value of zero for `lm` indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

If the terminal cursor can wrap around to the beginning of the next line when it reaches the right margin, this can be specified with the boolean capability `am`. If a string exists to enable this wrapping, specify it as `smam`. A string to make the cursor stick in the last column of a line is specified as `rmam`.

Media copy strings which control an auxiliary printer connected to the terminal can be given as `mc0`: print the contents of the screen, `mc4`: turn off the printer, and `mc5`: turn on the printer. When the printer is on, all text sent to the terminal is sent to the printer. A variation, `mc5p`, takes one parameter, and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. If the text is not displayed on the terminal screen when the printer is on, specify `mc5i` (silent printer). All text, including `mc4`, is transparently passed to the printer while an `mc5p` is in effect.

## 14. Special Cases

The working model used by `terminfo` fits most terminals reasonably well. However, some terminals do not completely match that model, requiring special support by `terminfo`. These are not meant to be construed as deficiencies in the terminals;

they are just differences between the working model and the actual hardware. They may be unusual devices or, for some reason, do not have all the features of the `terminfo` model implemented.

Terminals which cannot display tilde (~) characters, such as certain Hazeltine terminals, should indicate `hz`.

Terminals which ignore a linefeed immediately after an `am` wrap, such as the Concept-100, should indicate `xenl`. Those terminals whose cursor remains on the rightmost column until another character has been received, rather than wrapping immediately upon receiving the rightmost character, such as the VT100, should also indicate `xenl`.

If `el` is required to get rid of standout mode (instead of writing normal text on top of it), `xhp` should be given.

Those Teleray terminals whose tabs overwrite blanks should indicate `xt` (destructive tabs). This capability is also taken to mean that it is not possible to position the cursor on top of a "magic cookie"; therefore, to erase standout mode, it is instead necessary to use delete and insert line.

Those Beehive Superbee terminals which do not transmit the `<ESC>` or `<Ctrl-C>` characters should specify `xsb`, indicating that the F1 key is to be used for `<ESC>` and the F2 key for *Ctrl-C*.

Most terminals can use padding as an alternative to XON-XOFF flow control. Some terminals, though, require XON-XOFF flow control. For these, specify the boolean capability `nxon`.

## TERMINFO PRINTER CAPABILITIES

The `terminfo` database allows you to define capabilities of printers as well as terminals. To find out what capabilities are available for printers as well as for terminals, see the table in the "Device Capabilities" section. Most subsections below are lettered for cross-reference to that table.

### Rounding Values

Because parameterized string capabilities work only with integer values, we recommend that `terminfo` designers create strings that expect numeric values that have been rounded. Application designers should note this and should always round values to the nearest integer before using them with a parameterized string capability.

### Printer Resolution

A printer's resolution is defined to be the smallest spacing of characters it can achieve. In general printers have independent resolution horizontally and vertically. Thus the vertical resolution of a printer can be determined by measuring the smallest achievable distance between consecutive printing baselines, while the horizontal resolution can be determined by measuring the smallest achievable distance between the leftmost edges of consecutive printed, identical, characters. (The terms "smallest distance" and "smallest step" will be used later to refer to these smallest achievable distances.)

All printers are assumed to be capable of printing with a uniform horizontal and vertical resolution. The view of printing that `terminfo` currently presents is one of printing inside a uniform matrix: All characters are printed at fixed positions relative to each "cell" in the matrix; furthermore, each cell has the same size given by the smallest horizontal and vertical step sizes dictated by the resolution. (The cell size can be changed as will be seen later.)

Many printers are capable of "proportional printing," where the horizontal spacing depends on the size of the character last printed. `Terminfo` does not make use of this capability, although it does provide enough capability definitions to allow an application to simulate proportional printing.

A printer must not only be capable of printing characters as close together as the horizontal and vertical resolutions suggest, but also of "moving" to a position an integral multiple of the resolution from a previous position. Thus printed characters can be spaced apart a distance that is an integral multiple of the smallest distance, up to the length or width of a single page.

Some printers can have different resolutions depending on different "modes." In "normal mode," the existing `terminfo` capabilities are assumed to work on columns and lines, just like a video terminal. Thus the old `lines` capability would give the length of a page in lines, and the `cols` capability would give the width of a page in columns. In "micro mode," many `terminfo` capabilities work on increments of lines and columns. With some printers the micro mode may be concomitant with normal mode, so that all the capabilities work at the same time.

## A. Specifying Printer Resolution

The printing resolution of a printer is given in several ways. Each specifies the resolution as the number of smallest steps per distance:

Numeric Capabilities for Specifying Characteristic Number of Smallest Steps

| | |
|---|---|
| **orhi** | Steps per inch horizontally |
| **orvi** | Steps per inch vertically |
| **orc** | Steps per column |
| **orl** | Steps per line |

When printing in normal mode, each character printed causes movement to the next column, except in special cases described later; the distance moved is the same as the per-column resolution. Some printers cause an automatic movement to the next line when a character is printed in the rightmost position; the distance moved vertically is the same as the per-line resolution. When printing in micro mode, these distances can be different, and may be zero for some printers.

Numeric Capabilities for Specifying Automatic Motion after Printing

| | |
|---|---|
| *Normal Mode:* | |
| **orc** | Steps moved horizontally |
| **orl** | Steps moved vertically |
| | |
| *Micro Mode:* | |
| **mcs** | Steps moved horizontally |
| **mls** | Steps moved vertically |

Some printers are capable of printing wide characters. The distance moved when a wide character is printed in normal mode may be different from when a regular width character is printed. The distance moved when a wide character is printed in micro mode may also be different from when a regular character is printed in micro mode, but the differences are assumed to be related: If the distance moved for a regular character is the same whether in normal mode or micro mode (**mcs=orc**), then the distance moved for a wide character is also the same whether in normal mode or micro mode. This doesn't mean the normal character distance is necessarily the same as the wide character distance, just that the distances don't change with a change in normal to micro mode. However, if the distance moved for a regular character is different in micro mode from the distance moved in normal mode (**mcs<orc**), the micro mode distance is assumed to be the same for a wide character printed in micro mode, as the table below shows.

Numeric Capabilities for Specifying
Automatic Motion after Printing Wide Character

| | |
|---|---|
| *Normal Mode or Micro Mode* (mcs = orc): | |
| **widcs** | Steps moved horizontally |
| *Micro Mode* (mcs < orc): | |
| **mcs** | Steps moved horizontally |

There may be control sequences to change the number of columns per inch (the character pitch) and to change the number of lines per inch (the line pitch). If these are used, the resolution of the printer changes, but the type of change depends on the printer:

String and Boolean Capabilities for
Changing the Character/Line Pitches

| | |
|---|---|
| **cpi** | Change character pitch |
| **cpix** | If set, `cpi` changes `orhi`, otherwise changes `orc` |
| **lpi** | Change line pitch |
| **lpix** | If set, `lpi` changes `orvi`, otherwise changes `orl` |
| **chr** | Change steps per column |
| **cvr** | Change steps per line |

The `cpi` and `lpi` string capabilities each require a single parameter, the pitch in columns (or characters) and lines per inch, respectively. The `chr` and `cvr` string capabilities each require a single parameter, the number of steps per column and line, respectively.

Using any of the control sequences in these strings will imply a change in some of the values of `orc`, `orhi`, `orl`, and `orvi`. Also, the distance moved when a wide character is printed, `widcs`, changes in relation to `orc`. The distance moved when a character is printed in micro mode, `mcs`, changes similarly, with one exception: if the distance is 0 or 1, then no change is assumed (see items marked with † in the following table).

Programs that use `cpi`, `lpi`, `chr`, or `cvr` should recalculate the printer resolution (and should recalculate other values — see the topic "Effect of Changing Printing Resolution" in the section "Dot-Matrix Graphics").

Specification of Printer Resolution
Effects of Changing the Character/Line Pitches

| *Before* | *After* |
|---|---|
| *Using* `cpi` *with* `cpix` *clear:* | |
| `orhi '` | `orhi` |
| `orc '` | $\mathbf{orc}=\frac{\mathbf{orhi}}{V_{cpi}}$ |

*Using* `cpi` *with* `cpix` *set:*

| | |
|---|---|
| `orhi ′` | $\mathbf{orhi}=\mathbf{orc}\cdot V_{cpi}$ |
| `orc ′` | `orc` |

*Using* `lpi` *with* `lpix` *clear:*

| | |
|---|---|
| `orvi ′` | `orvi` |
| `orl ′` | $\mathbf{orl}=\frac{\mathbf{orvi}}{V_{lpi}}$ |

*Using* `lpi` *with* `lpix` *set:*

| | |
|---|---|
| `orvi ′` | $\mathbf{orvi}=\mathbf{orl}\cdot V_{lpi}$ |
| `orl ′` | `orl` |

*Using* `chr`*:*

| | |
|---|---|
| `orhi ′` | `orhi` |
| `orc ′` | $V_{chr}$ |

*Using* `cvr`*:*

| | |
|---|---|
| `orvi ′` | `orvi` |
| `orl ′` | $V_{cvr}$ |

*Using* `cpi` *or* `chr`*:*

| | |
|---|---|
| `widcs ′` | $\mathbf{widcs}=\mathbf{widcs}'\frac{\mathbf{orc}}{\mathbf{orc}'}$ |
| `mcs ′` † | $\mathbf{mcs}=\mathbf{mcs}'\frac{\mathbf{orc}}{\mathbf{orc}'}$ |

$V_{cpi}$, $V_{lpi}$, $V_{chr}$, and $V_{cvr}$ are the parameters required by `cpi`, `lpi`, `chr`, and `cvr`, respectively. The `′` mark indicates the old value.

**B. Capabilities that Cause Movement**

In the following descriptions, "movement" refers to the motion of the "current position." With video terminals this would be the cursor; with some printers this is the carriage position. Other printers have different equivalents. In general, the current position is where a character would be displayed if printed.

`Terminfo` has string capabilities for control sequences that cause movement a number of full columns or lines. It also has equivalent string capabilities for control sequences that cause movement a number of smallest steps.

String Capabilities for Specifying
Single and Multiple Motions

| | |
|---|---|
| **mcub1** | Move 1 step left |
| **mcuf1** | Move 1 step right |
| **mcuu1** | Move 1 step up |
| **mcud1** | Move 1 step down |
| | |
| **mcub** | Move *N* steps left |
| **mcuf** | Move *N* steps right |
| **mcuu** | Move *N* steps up |
| **mcud** | Move *N* steps down |
| | |
| **mhpa** | Move *N* steps from the left |
| **mvpa** | Move *N* steps from the top |

The latter six strings each require a single parameter, *N*.

Some printers limit the motion to less than the width or length of a page. Also, some printers don't accept absolute motion to the left of the current position. `Terminfo` has capabilities for specifying these limits.

Numeric and Boolean Capabilities for
Specifying Limits to Motion

| | |
|---|---|
| **mjump** | Limit on use of `mcub1`, `mcuf1`, `mcuu1`, and `mcud1` |
| **maddr** | Limit on use of `mhpa` and `mvpa` |
| | |
| **xhpa** | If set, `hpa` and `mhpa` cannot move left |
| **xvpa** | If set, `vpa` and `mvpa` cannot move up |

If a printer needs to be in a "micro mode" for the motion capabilities described above to work, there are string capabilities defined to enter and exit this mode. A boolean capability is available for those printers where using a carriage return causes an automatic return to normal mode.

String and Boolean Capabilities for
Entering and Exiting Micro Mode

| | |
|---|---|
| **smicm** | Enter micro mode |
| **rmicm** | Exit micro mode |
| | |
| **crxm** | If set, using `cr` exits micro mode |

The movement made when a character is printed in the rightmost position varies among printers. Some make no movement, some move to the beginning of the next line, others move to the beginning of the same line. `Terminfo` has boolean capabilities for describing all three cases.

Boolean Capabilities for Specifying
What Happens After Character
Printed in Rightmost Position

| | |
|---|---|
| **sam** | Automatic move to beginning of same line |

Some printers can be put in a mode where the normal direction of motion is reversed. This mode can be especially useful when there are no capabilities for leftward or upward motion, because those capabilities can be built from the motion reversal capability and the rightward or downward motion capabilities. It is best to leave it up to an application to build the leftward or upward capabilities, though, and not enter them in the `terminfo` database. This allows several reverse motions to be strung together without intervening wasted steps that leave and reenter reverse mode.

String Capabilities for
Entering and Exiting Reverse Modes

| | |
|---|---|
| **slm** | Reverse sense of horizontal motions |
| **rlm** | Restore sense of horizontal motions |
| **sum** | Reverse sense of vertical motions |
| **rum** | Restore sense of vertical motions |

*While sense of horizontal motions reversed:*

| | |
|---|---|
| **mcub1** | Move 1 step right |
| **mcuf1** | Move 1 step left |
| **mcub** | Move *N* steps right |
| **mcuf** | Move *N* steps left |
| **cub1** | Move 1 column right |
| **cuf1** | Move 1 column left |
| **cub** | Move *N* columns right |
| **cuf** | Move *N* columns left |

*While sense of vertical motions reversed:*

| | |
|---|---|
| **mcuu1** | Move 1 step down |
| **mcud1** | Move 1 step up |
| **mcuu** | Move *N* steps down |
| **mcud** | Move *N* steps up |
| **cuu1** | Move 1 line down |
| **cud1** | Move 1 line up |
| **cuu** | Move *N* lines down |
| **cud** | Move *N* lines up |

The reverse motion modes should not affect the `mvpa` and `mhpa` absolute motion capabilities. The reverse vertical motion mode should, however, also reverse the action of the line "wrapping" that occurs when a character is printed in the rightmost position. Thus printers that have the standard `terminfo` capability `am` defined should experience motion to the beginning of the previous line when a character is printed in the rightmost position under reverse vertical motion mode.

The action when any other motion capabilities are used in reverse motion modes is not defined; thus, programs must exit reverse motion modes before using other motion capabilities.

Two miscellaneous capabilities complete the list of new motion capabilities. One of these is needed for printers that move the current position to the beginning of a line when certain control characters, such as "linefeed" or "formfeed," are used. The other is used for the capability of suspending the motion that normally occurs after printing a character.

String Capabilities for Specifying
Miscellaneous Motion

| | |
|---|---|
| **docr** | List of control characters causing `cr` |
| **zerom** | Prevent auto motion after printing next single character |

## C. Margins

`Terminfo` provides two strings for setting margins on terminals: one for the left margin and one for the right. Printers, however, have two additional margins, for the top and bottom of each page. Furthermore, instead of using motion strings to move the current position to a margin and then fixing the margin there, some printers require the specification of where a margin should be regardless of the current position. Therefore `terminfo` offers six additional strings for defining margins with printers.

String Capabilities for Setting Margins

| | |
|---|---|
| **smgl** | Set left margin at current column |
| **smgr** | Set right margin at current column |
| **smgb** | Set bottom margin at current line |
| **smgt** | Set top margin at current line |
| | |
| **smgbp** | Set bottom margin at line *N* |
| **smglp** | Set left margin at column *N* |
| **smgrp** | Set right margin at column *N* |
| **smgtp** | Set top margin at line *N* |

The last four strings each require one or more parameters that give the position of the margin or margins to set. If both of `smglp` and `smgrp` are defined, each requires a single parameter, *N*, that gives the column number of the left and right margin, respectively. If both of `smgtp` and `smgbp` are defined, they are used to set the top and bottom margin, respectively: `smgtp` requires a single parameter, *N*, the line number of the top margin; however, `smgbp` requires two parameters, *N* and *M*, that each give the line number of the bottom margin, the first counting from the top of the page and the second counting from the bottom. This accommodates the two methods used by different manufacturers to specify the bottom margin. When coding a `terminfo` entry for a printer that has a settable bottom margin, only the first or second parameter should be used, depending on the printer. When writing an application that uses `smgbp` to set the bottom margin, both arguments must be given.

If only one of `smglp` and `smgrp` is defined, then it requires two parameters, the column numbers of the left and right margins, in that order. Likewise, if only one of `smgtp` and `smgbp` is set, then it requires two parameters that give the top and bottom margins, in that order, counting from the top of the page. Thus when coding a `terminfo` entry for a printer that requires setting both left and right or top and bottom margins simultaneously, only one of `smglp` and `smgrp`, or `smgtp` and `smgbp`, should be defined; the other capability of the pair should not be included in the entry. When writing an application that uses these string capabilities, each pair should first be checked to see if both members of the pair are defined or if only one is defined; the defined capabilities should then be instantiated accordingly.

In counting lines or columns, line zero is the top line and column zero is the leftmost column. A zero value for the second argument with `smgbp` means the bottom line of the page.

All margins can be cleared with `mgc`.

## D. Shadows, Italics, Wide Characters, Superscripts, Subscripts

Five new sets of string capabilities are used to describe the methods printers have of enhancing printed text.

String Capabilities for Specifying Enhanced Printing

| | |
|---|---|
| **sshm** | Enter shadow-printing mode |
| **rshm** | Exit shadow-printing mode |

| | |
|---|---|
| **sitm** | Enter italicizing mode |
| **ritm** | Exit italicizing mode |
| | |
| **swidm** | Enter wide character mode |
| **rwidm** | Exit wide character mode |
| | |
| **ssupm** | Enter superscript mode |
| **rsupm** | Exit superscript mode |
| **supcs** | List of characters available as superscripts |
| | |
| **ssubm** | Enter subscript mode |
| **rsubm** | Exit subscript mode |
| **subcs** | List of characters available as subscripts |

If a printer requires the `sshm` control sequence before every character to be shadow-printed, the `rshm` string should be left undefined. Thus programs that find a control sequence in `sshm` but none in `rshm` should use the `sshm` control sequence before every character to be shadow-printed; otherwise, the `sshm` control sequence should be used once before the set of characters to be shadow-printed, followed by `rshm`. The same is also true of each of the `sitm/ritm`, `swidm/rwidm`, `ssupm/rsupm`, and `ssubm/rsubm` pairs.

Note that `terminfo` also has a capability for printing emboldened text (**bold**). While shadow printing and emboldened printing are similar in that they "darken" the text, many printers produce these two types of print in slightly different ways. Generally, emboldened printing is done by overstriking the same character one or more times. Shadow printing likewise usually involves overstriking, but with a slight movement up and/or to the side so that the character is "fatter."

`Terminfo` requires that enhanced printing modes be independent, so that it would be possible, for instance, to shadow print italicized subscripts.

As mentioned earlier, the amount of motion automatically made after printing a wide character should be given in the numeric capability `widcs`.

If only a subset of the printable ASCII characters can be printed as superscripts or subscripts, they should be listed in the `supcs` or `subcs` strings, respectively. If the `ssupm` (or `ssubm`) string contains control sequences, but the corresponding `supcs` (or `subcs`) string is undefined, a program can assume that all printable ASCII characters are available as superscripts (or subscripts).

Automatic motion made after printing a superscript or subscript must be the same as for regular characters. Thus, for example, printing any of the following two-character sequences will result in equivalent motion: `Bi` $B_i$ $B^i$

Note that the existing `msgr` boolean capability describes whether motion control sequences can be used while in "standout mode." This capability has been extended to cover the enhanced printing modes added here. `msgr` should be set for those printers that accept any motion control sequences without affecting shadow, italicized, widened, superscript, or subscript printing. Conversely, if `msgr` is not set, a program should exit these modes before attempting any motion.

## E. Alternate Character Sets

In addition to allowing you to define line graphics (described in the "Line Graphics" section), `terminfo` lets you define alternate character sets. The following capabilities cover printers and terminals with multiple selectable or definable character sets.

String and Boolean Capabilities for Specifying Alternate Character Sets

| | |
|---|---|
| **scs** | Select character set *N* |
| **scsd** | Start definition of character set *N*, *M* characters |
| **defc** | Define character *A*, *B* dots wide, descender *D* |
| **rcsd** | End definition of character set *N* |
| **csnm** | List of character set names |
| **daisy** | If set, printer has manually changed print wheels |

The `scs`, `rcsd`, and `csnm` strings each require a single parameter, *N*, a number from 0 to 63 that identifies the character set. The `scsd` string also requires the parameter *N* and another, *M*, that gives the number of characters in the set. The `defc` string requires three parameters: *A* gives the ASCII code representation for the character, *B* gives the width of the character in dots, and *D* is zero or one depending on whether the character is a "descender" or not. The `defc` string is also followed by a string of "image data" bytes that describe how the character looks (see below).

Character set 0 is the default character set present after the printer has been initialized. Not every printer has 64 character sets, of course; using `scs` with an argument that doesn't select an available character set should cause a null result from `tparm()`.

If a character set has to be defined before it can be used, the `scsd` control sequence must be used before defining the character set, and `rcsd` must be used after. They should also cause a null result from `tparm()` when used with an argument *N* that doesn't apply. If a character set still has to be selected after being defined, the `scs` control sequence must follow the `rcsd` control sequence. By examining the results of using each of the `scs`, `scsd`, and `rcsd` strings with a character set number in a call to `tparm()`, a program can determine which of the three are needed.

Between use of the `scsd` and `rcsd` strings, the `defc` string should be used to define each character. To print any character on printers covered by `terminfo`, the ASCII code is sent to the printer. This is true for characters in an alternate set as well as "normal" characters. Thus the definition of a character includes the ASCII code that represents it. In addition, the width of the character in dots is given, along with an indication of whether the character should descend below the print line (such as the lower case letter `g` in most character sets). The width of the character in dots also indicates the number of image data bytes that will follow the `defc` string. These image data bytes indicate where in a dot-matrix pattern ink should be applied to "draw" the character; the number of these bytes and their form are defined below in the "Dot-Matrix Graphics" section.

It's easiest for the creator of `terminfo` entries to refer to each character set by number; however, these numbers will be meaningless to the application developer. The `csnm` string alleviates this problem by providing names for each number.

When used with a character set number in a call to `tparm()`, the `csnm` string will produce the equivalent name. These names should be used as a reference only. No naming convention is specified, although anyone who creates a `terminfo` entry for a printer should use names consistent with the names found in user documents for the printer. Application developers should allow a user to specify a character set by number (leaving it up to the user to examine the `csnm` string to determine the correct number), or by name, where the application examines the `csnm` string to determine the corresponding character set number.

The boolean `daisy` indicates printers that have manually changed print wheels or font cartridges. However, the capabilities described above are likely to be used only with dot-matrix printers.

## F. Dot-Matrix Graphics

Dot-matrix printers typically have the capability of reproducing "raster graphics" images. Three new numeric capabilities and three new string capabilities help a program draw raster graphics images independent of the type of dot-matrix printer or the number of pins or dots the printer can handle at one time.

Numeric and String Capabilities for Specifying Dot-Matrix Graphics

| | |
|---|---|
| **npins** | Number of pins, *N*, in print head |
| **spinv** | Spacing of pins vertically in pins per inch |
| **spinh** | Spacing of dots horizontally in dots per inch |
| | |
| **porder** | Matches software bits to print head pins |
| **sbim** | Start printing bit image graphics, *B* bits wide |
| **rbim** | End printing bit image graphics |

The `sbim` sring requires a single parameter, *B*, the width of the image in dots.

The model of dot-matrix or raster graphics that `terminfo` presents is similar to the technique used for most dot-matrix printers: Each pass of the printer's print head is assumed to produce a dot-matrix that is *N* dots high and *B* dots wide. This is typically a wide, squat, rectangle of dots. The height of this rectangle in dots will vary from one printer to the next; this is given in the `npins` numeric capability. The size of the rectangle in fractions of an inch will also vary; it can be deduced from the `spinv` and `spinh` numeric capabilities. With these three values an application can divide a complete raster graphics image into several horizontal strips, perhaps interpolating to account for different dot spacing vertically and horizontally.

The `sbim` and `rbim` strings start and end a dot-matrix image, respectively. The `sbim` string requires a single parameter that gives the width of the dot-matrix in dots. A sequence of "image data" bytes is sent to the printer after the `sbim` string and before the `rbim` string. The number of bytes is an integral multiple of the width of the dot-matrix; the multiple and the form of each byte are determined by the `porder` string as described below.

The `porder` string is a comma-separated list of pin numbers optionally followed by a numerical offset. The offset, if given, is separated from the list with a semicolon. The position of each pin number in the list corresponds to a bit in an eight-bit data byte. The pins are numbered consecutively from 1 to `npins`, with 1 being the top pin. Note that the term "pin" is used loosely here; "ink-jet" dot-matrix printers don't have pins, but can be considered to have an equivalent method of applying a single dot of ink to paper. The bit positions in `porder` are in groups of eight; the first position of each group is the most significant bit and the last position is the least significant bit. An application produces eight-bit bytes in the order of the groups in `porder`.

An application computes the "image data" bytes from its internal image, mapping vertical dot positions in each print head pass into eight-bit bytes, using a 1 bit where ink should be applied and 0 where no ink should be applied. This can be reversed (0 bit for ink, 1 bit for no ink) by giving a negative pin number in `porder`. If a position is skipped in `porder`, a 0 bit is assumed (indicating no ink can be applied for this position). If a position has a lower case 'x' instead of a pin number, a 1 bit is assumed (indicating ink is always applied for this position). For consistency, a lower case ‘o’ can be used to represent a 0 filled (no-ink) bit. There must be a multiple of 8 bit positions used or skipped in `porder`; if not, 0 bits are used to fill the last byte in the least significant bits. The offset, if given, is added to each data byte; the offset can be negative.

Some examples may help clarify the use of the `porder` string. The AT&T 470, AT&T 475 and C.Itoh 8510 printers provide eight pins for graphics. The pins are identified top to bottom by the 8 bits in a byte, from least significant to most. The `porder` strings for these printers would be `8,7,6,5,4,3,2,1`. The AT&T 478 and AT&T 479 printers also provide eight pins for graphics. However, the pins are identified in the reverse order. The `porder` strings for these printers would be `1,2,3,4,5,6,7,8`. The AT&T 5310, AT&T 5320, DEC LA100, and DEC LN03 printers provide six pins for graphics. The pins are identified top to bottom by the decimal values 1, 2, 4, 8, 16 and 32. These correspond to the low six bits in an 8-bit byte, although the decimal values are further offset by the value 63. The `porder` string for these printers would be `,,6,5,4,3,2,1;63`, or alternately `o,o,6,5,4,3,2,1;63`.

## G. Effect of Changing Printing Resolution

If the control sequences to change the character pitch or the line pitch are used, the pin or dot spacing may change:

String and Boolean Capabilities for
Changing the Character and Line Pitches

| | |
|---|---|
| **cpi** | Change character pitch |
| **cpix** | If set, `cpi` changes `spinh` |
| **lpi** | Change line pitch |
| **lpix** | If set, `lpi` changes `spinv` |

Programs that use `cpi` or `lpi` should recalculate the dot spacing:

Dot-Matrix Graphics
Effects of Changing the Character and Line Pitches

| *Before* | *After* |
|---|---|
| *Using* `cpi` *with* `cpix` *clear:* | |
| `spinh '` | **spinh** |
| *Using* `cpi` *with* `cpix` *set:* | |
| `spinh '` | **spinh=spinh** $' \cdot \frac{\textbf{orhi}}{\textbf{orhi}'}$ |
| *Using* `lpi` *with* `lpix` *clear:* | |
| `spinv '` | `spinv` |
| *Using* `lpi` *with* `lpix` *set:* | |
| `spinv '` | **spinv=spinv** $' \cdot \frac{\textbf{orhi}}{\textbf{orhi}'}$ |

| | |
|---|---|
| *Using* `chr`: | |
| `spinh '` | `spinh` |
| *Using* `cvr`: | |
| `spinv '` | `spinv` |

`orhi'` and `orhi` are the values of the horizontal resolution in steps per inch, before using `cpi` and after using `cpi`, respectively. Likewise, `orvi'` and `orvi` are the values of the vertical resolution in steps per inch, before using `lpi` and after using `lpi`, respectively. Thus, the changes in the dots per inch for dot-matrix graphics follow the changes in steps per inch for printer resolution.

**H. Print Quality**

Many dot-matrix printers can alter the dot spacing of printed text to produce "near-letter-quality" printing or "draft quality" printing. Usually it is important to be able to choose one or the other because the rate of printing generally falls off as the quality improves. There are three new string capabilities used to describe these print quality levels.

String Capabilities for Specifying Print Quality

| | |
|---|---|
| **snlq** | Set near-letter-quality printing |
| **snrmq** | Set normal quality printing |
| **sdrfq** | Set draft quality printing |

The capabilities are listed in decreasing levels of quality. If a printer doesn't have all three levels, one or two of the strings should be left undefined as appropriate.

**I. Printing Rate and Buffer Size**

Because there is no standard protocol that can be used to keep a program synchronized with a printer, and because modern printers can buffer data before printing it, a program generally cannot determine at any time what has been printed. However, two new numeric capabilities can help a program estimate what has been printed.

Numeric Capabilities for Specifying Print Rate and Buffer Size

| | |
|---|---|
| **cps** | Nominal print rate in characters per second |
| **bufsz** | Buffer capacity in characters |

`cps` is the nominal or average rate at which the printer prints characters; if this value is not given, the rate should be estimated at one-tenth the prevailing baud rate. `bufsz` is the maximum number of subsequent characters buffered before the guaranteed printing of an earlier character, assuming proper flow control has been used. If this value is not given it is assumed that the printer does not buffer characters, but prints them as they are received.

As an example, if a printer has a 1000-character buffer, then sending the letter a followed by 1000 additional characters is guaranteed to cause the letter a to print. If the same printer prints at the rate of 100 characters per second, then it should take 10 seconds to print all the characters in the buffer, less if the buffer is not full. By keeping track of the characters sent to a printer, and determining the print rate and buffer size, a program can synchronize itself with the printer.

Note that most printer manufacturers advertise the maximum print rate, not the nominal print rate. A good way to get a value for `cps` is to generate a few pages of text, count the number of printable characters, and then see how long it takes to print the text.

Applications that use these values should recognize the variability in print rate. Straight text, in short lines, with no embedded control sequences will probably print at close to the advertised print rate and probably faster than the rate in cps. Graphics data with a lot of control sequences, or very long lines of text, will print at well below the advertised rate and below the rate in cps. If the application is using cps to decide how long it should take a printer to print a block of text, the application should pad the estimate. If the application is using cps to decide how much text has already been printed, it should shrink the estimate. The application will thus err in favor of the user, who wants, above all, to see all the output in its correct place.

## TERMINFO/TERMCAP CORRESPONDENCE

The table below presents the correspondence between terminfo and termcap(5) codes. The first two columns correspond to the first two columns in the previously presented table of terminfo capabilities. The last column shows the Termcap Code, which is the two-letter code that corresponds to the termcap(5) capability. The table is sorted alphabetically by Capname.

| Variable | Cap-name | Termcap Code |
|---|---|---|
| acs_chars | **acsc** | ac |
| auto_right_margin | **am** | am |
| back_color_erase | **bce** | be |
| bell | **bel** | bl |
| enter_blink_mode | **blink** | mb |
| enter_bold_mode | **bold** | md |
| buffer_capacity | **bufsz** | Ya |
| auto_left_margin | **bw** | bw |
| back_tab | **cbt** | bt |
| can_change | **ccc** | cc |
| change_res_horz | **chr** | ZC |
| hard_cursor | **chts** | HC |
| cursor_invisible | **civis** | vi |
| clear_screen | **clear** | cl |
| command_character | **cmdch** | CC |
| cursor_normal | **cnorm** | ve |
| max_colors | **colors** | Co |
| columns | **cols** | co |
| change_char_pitch | **cpi** | ZA |
| cpi_changes_res | **cpix** | YF |
| print_rate | **cps** | Ym |
| carriage_return | **cr** | cr |
| cr_cancels_micro_mode | **crxm** | YB |
| char_set_names | **csnm** | Zy |
| change_scroll_region | **csr** | cs |
| parm_left_cursor | **cub** | LE |
| cursor_left | **cub1** | le |
| parm_down_cursor | **cud** | DO |
| cursor_down | **cud1** | do |
| parm_right_cursor | **cuf** | RI |

| | | |
|---|---|---|
| cursor_right | **cuf1** | nd |
| cursor_address | **cup** | cm |
| parm_up_cursor | **cuu** | UP |
| cursor_up | **cuu1** | up |
| change_res_vert | **cvr** | ZD |
| cursor_visible | **cvvis** | vs |
| memory_above | **da** | da |
| has_print_wheel | **daisy** | YC |
| memory_below | **db** | db |
| parm_dch | **dch** | DC |
| delete_character | **dch1** | dc |
| define_char | **defc** | ZE |
| enter_dim_mode | **dim** | mh |
| parm_delete_line | **dl** | DL |
| delete_line | **dl1** | dl |
| these_cause_cr | **docr** | Zw |
| dis_status_line | **dsl** | ds |
| erase_chars | **ech** | ec |
| clr_eos | **ed** | cd |
| clr_eol | **el** | ce |
| clr_bol | **el1** | cb |
| ena_acs | **enacs** | eA |
| erase_overstrike | **eo** | eo |
| status_line_esc_ok | **eslok** | es |
| form_feed | **ff** | ff |
| flash_screen | **flash** | vb |
| from_status_line | **fsl** | fs |
| generic_type | **gn** | gn |
| hard_copy | **hc** | hc |
| down_half_line | **hd** | hd |
| hue_lightness_saturation | **hls** | hl |
| cursor_home | **home** | ho |
| column_address | **hpa** | ch |
| has_status_line | **hs** | hs |
| tab | **ht** | ta |
| set_tab | **hts** | st |
| up_half_line | **hu** | hu |
| tilde_glitch | **hz** | hz |
| parm_ich | **ich** | IC |
| insert_character | **ich1** | ic |
| init_file | **if** | if |
| parm_insert_line | **il** | AL |
| insert_line | **il1** | al |
| insert_null_glitch | **in** | in |
| scroll_forward | **ind** | sf |
| parm_index | **indn** | SF |
| initialize_color | **initc** | Ic |
| initialize_pair | **initp** | Ip |
| enter_secure_mode | **invis** | mk |

| | | |
|---|---|---|
| insert_padding | ip | ip |
| init_prog | iprog | iP |
| init_1string | is1 | i1 |
| init_2string | is2 | is |
| init_3string | is3 | i3 |
| init_tabs | it | it |
| key_sbeg | kBEG | &9 |
| key_scancel | kCAN | &0 |
| key_scommand | kCMD | *1 |
| key_scopy | kCPY | *2 |
| key_screate | kCRT | *3 |
| key_sdc | kDC | *4 |
| key_sdl | kDL | *5 |
| key_send | kEND | *7 |
| key_seol | kEOL | *8 |
| key_sexit | kEXT | *9 |
| key_sfind | kFND | *0 |
| key_shelp | kHLP | #1 |
| key_shome | kHOM | #2 |
| key_sic | kIC | #3 |
| key_sleft | kLFT | #4 |
| key_smove | kMOV | %b |
| key_smessage | kMSG | %a |
| key_snext | kNXT | %c |
| key_soptions | kOPT | %d |
| key_sprint | kPRT | %f |
| key_sprevious | kPRV | %e |
| key_sredo | kRDO | %g |
| key_srsume | kRES | %j |
| key_sright | kRIT | %i |
| key_sreplace | kRPL | %h |
| key_ssave | kSAV | !1 |
| key_ssuspend | kSPD | !2 |
| key_sundo | kUND | !3 |
| key_a1 | ka1 | K1 |
| key_a3 | ka3 | K3 |
| key_b2 | kb2 | K2 |
| key_beg | kbeg | @1 |
| key_backspace | kbs | kb |
| key_c1 | kc1 | K4 |
| key_c3 | kc3 | K5 |
| key_cancel | kcan | @2 |
| key_btab | kcbt | kB |
| key_close | kclo | @3 |
| key_clear | kclr | kC |
| key_command | kcmd | @4 |
| key_copy | kcpy | @5 |
| key_create | kcrt | @6 |
| key_ctab | kctab | kt |

| | | |
|---|---|---|
| key_left | **kcub1** | kl |
| key_down | **kcud1** | kd |
| key_right | **kcuf1** | kr |
| key_up | **kcuu1** | ku |
| key_dc | **kdch1** | kD |
| key_dl | **kdl1** | kL |
| key_eos | **ked** | kS |
| key_eol | **kel** | kE |
| key_end | **kend** | @7 |
| key_enter | **kent** | @8 |
| key_exit | **kext** | @9 |
| key_f0 | **kf0** | k0 |
| key_f1 | **kf1** | k1 |
| key_f10 | **kf10** | k; |
| key_f11 | **kf11** | F1 |
| key_f12 | **kf12** | F2 |
| key_f13 | **kf13** | F3 |
| key_f14 | **kf14** | F4 |
| key_f15 | **kf15** | F5 |
| key_f16 | **kf16** | F6 |
| key_f17 | **kf17** | F7 |
| key_f18 | **kf18** | F8 |
| key_f19 | **kf19** | F9 |
| key_f2 | **kf2** | k2 |
| key_f20 | **kf20** | FA |
| key_f21 | **kf21** | FB |
| key_f22 | **kf22** | FC |
| key_f23 | **kf23** | FD |
| key_f24 | **kf24** | FE |
| key_f25 | **kf25** | FF |
| key_f26 | **kf26** | FG |
| key_f27 | **kf27** | FH |
| key_f28 | **kf28** | FI |
| key_f29 | **kf29** | FJ |
| key_f3 | **kf3** | k3 |
| key_f30 | **kf30** | FK |
| key_f31 | **kf31** | FL |
| key_f32 | **kf32** | FM |
| key_f33 | **kf33** | FN |
| key_f34 | **kf34** | FO |
| key_f35 | **kf35** | FP |
| key_f36 | **kf36** | FQ |
| key_f37 | **kf37** | FR |
| key_f38 | **kf38** | FS |
| key_f39 | **kf39** | FT |
| key_f4 | **kf4** | k4 |
| key_f40 | **kf40** | FU |
| key_f41 | **kf41** | FV |
| key_f42 | **kf42** | FW |

| | | |
|---|---|---|
| key_f43 | **kf43** | FX |
| key_f44 | **kf44** | FY |
| key_f45 | **kf45** | FZ |
| key_f46 | **kf46** | Fa |
| key_f47 | **kf47** | Fb |
| key_f48 | **kf48** | Fc |
| key_f49 | **kf49** | Fd |
| key_f5 | **kf5** | k5 |
| key_f50 | **kf50** | Fe |
| key_f51 | **kf51** | Ff |
| key_f52 | **kf52** | Fg |
| key_f53 | **kf53** | Fh |
| key_f54 | **kf54** | Fi |
| key_f55 | **kf55** | Fj |
| key_f56 | **kf56** | Fk |
| key_f57 | **kf57** | Fl |
| key_f58 | **kf58** | Fm |
| key_f59 | **kf59** | Fn |
| key_f6 | **kf6** | k6 |
| key_f60 | **kf60** | Fo |
| key_f61 | **kf61** | Fp |
| key_f62 | **kf62** | Fq |
| key_f63 | **kf63** | Fr |
| key_f7 | **kf7** | k7 |
| key_f8 | **kf8** | k8 |
| key_f9 | **kf9** | k9 |
| key_find | **kfnd** | @0 |
| key_help | **khlp** | %1 |
| key_home | **khome** | kh |
| key_stab | **khts** | kT |
| key_ic | **kich1** | kI |
| key_il | **kil1** | kA |
| key_sf | **kind** | kF |
| key_ll | **kll** | kH |
| has_meta_key | **km** | km |
| key_move | **kmov** | %4 |
| key_mark | **kmrk** | %2 |
| key_message | **kmsg** | %3 |
| key_npage | **knp** | kN |
| key_next | **knxt** | %5 |
| key_open | **kopn** | %6 |
| key_options | **kopt** | %7 |
| key_ppage | **kpp** | kP |
| key_print | **kprt** | %9 |
| key_previous | **kprv** | %8 |
| key_redo | **krdo** | %0 |
| key_reference | **kref** | &1 |
| key_resume | **kres** | &5 |
| key_refresh | **krfr** | &2 |

| | | |
|---|---|---|
| key_sr | kri | kR |
| key_eic | krmir | kM |
| key_replace | krpl | &3 |
| key_restart | krst | &4 |
| key_save | ksav | &6 |
| key_select | kslt | *6 |
| key_suspend | kspd | &7 |
| key_catab | ktbc | ka |
| key_undo | kund | &8 |
| lab_f0 | lf0 | l0 |
| lab_f1 | lf1 | l1 |
| lab_f10 | lf10 | la |
| lab_f2 | lf2 | l2 |
| lab_f3 | lf3 | l3 |
| lab_f4 | lf4 | l4 |
| lab_f5 | lf5 | l5 |
| lab_f6 | lf6 | l6 |
| lab_f7 | lf7 | l7 |
| lab_f8 | lf8 | l8 |
| lab_f9 | lf9 | l9 |
| label_height | lh | lh |
| lines | lines | li |
| cursor_to_ll | ll | ll |
| lines_of_memory | lm | lm |
| change_line_pitch | lpi | ZB |
| lpi_changes_res | lpix | YG |
| label_width | lw | lw |
| max_micro_address | maddr | Yd |
| print_screen | mc0 | ps |
| prtr_off | mc4 | pf |
| prtr_on | mc5 | po |
| prtr_silent | mc5i | 5i |
| prtr_non | mc5p | pO |
| micro_col_size | mcs | Yf |
| parm_left_micro | mcub | Zg |
| micro_left | mcub1 | Za |
| parm_down_micro | mcud | Zf |
| micro_down | mcud1 | ZZ |
| parm_right_micro | mcuf | Zh |
| micro_right | mcuf1 | Zb |
| parm_up_micro | mcuu | Zi |
| micro_up | mcuu1 | Zd |
| clear_margins | mgc | MC |
| micro_column_address | mhpa | ZY |
| move_insert_mode | mir | mi |
| max_micro_jump | mjump | Ye |
| micro_line_size | mls | Yg |
| cursor_mem_address | mrcup | CM |
| move_standout_mode | msgr | ms |

| | | |
|---|---|---|
| micro_row_address | **mvpa** | Zc |
| no_color_video | **ncv** | NC |
| newline | **nel** | nw |
| num_labels | **nlab** | Nl |
| no_pad_char | **npc** | NP |
| number_of_pins | **npins** | Yh |
| non_rev_rmcup | **nrrmc** | NR |
| needs_xon_xoff | **nxon** | nx |
| orig_colors | **oc** | oc |
| orig_pair | **op** | op |
| output_res_char | **orc** | Yi |
| output_res_horz_inch | **orhi** | Yk |
| output_res_line | **orl** | Yj |
| output_res_vert_inch | **orvi** | Yl |
| over_strike | **os** | os |
| pad_char | **pad** | pc |
| max_pairs | **pairs** | pa |
| padding_baud_rate | **pb** | pb |
| pkey_key | **pfkey** | pk |
| pkey_local | **pfloc** | pl |
| pkey_xmit | **pfx** | px |
| plab_norm | **pln** | pn |
| order_of_pins | **porder** | Ze |
| enter_protected_mode | **prot** | mp |
| stop_bit_image | **rbim** | Zs |
| restore_cursor | **rc** | rc |
| stop_char_set_def | **rcsd** | Zt |
| repeat_char | **rep** | rp |
| enter_reverse_mode | **rev** | mr |
| reset_file | **rf** | rf |
| req_for_input | **rfi** | RF |
| scroll_reverse | **ri** | sr |
| parm_rindex | **rin** | SR |
| exit_italics_mode | **ritm** | ZR |
| exit_leftward_mode | **rlm** | ZS |
| exit_alt_charset_mode | **rmacs** | ae |
| exit_am_mode | **rmam** | RA |
| exit_ca_mode | **rmcup** | te |
| exit_delete_mode | **rmdc** | ed |
| exit_micro_mode | **rmicm** | ZT |
| exit_insert_mode | **rmir** | ei |
| keypad_local | **rmkx** | ke |
| label_off | **rmln** | LF |
| meta_off | **rmm** | mo |
| char_padding | **rmp** | rP |
| exit_standout_mode | **rmso** | se |
| exit_underline_mode | **rmul** | ue |
| exit_xon_mode | **rmxon** | RX |
| reset_1string | **rs1** | r1 |

| | | |
|---|---|---|
| reset_2string | **rs2** | r2 |
| reset_3string | **rs3** | r3 |
| exit_shadow_mode | **rshm** | ZU |
| exit_subscript_mode | **rsubm** | ZV |
| exit_superscript_mode | **rsupm** | ZW |
| exit_upward_mode | **rum** | ZX |
| exit_doublewide_mode | **rwidm** | ZQ |
| semi_auto_right_margin | **sam** | YE |
| start_bit_image | **sbim** | Zq |
| save_cursor | **sc** | sc |
| set_color_pair | **scp** | sp |
| select_char_set | **scs** | Zj |
| start_char_set_def | **scsd** | Zr |
| enter_draft_quality | **sdrfq** | ZG |
| set_background | **setb** | Sb |
| set_foreground | **setf** | Sf |
| set_attributes | **sgr** | sa |
| exit_attribute_mode | **sgr0** | me |
| enter_italics_mode | **sitm** | ZH |
| enter_leftward_mode | **slm** | ZI |
| enter_alt_charset_mode | **smacs** | as |
| enter_am_mode | **smam** | SA |
| enter_ca_mode | **smcup** | ti |
| enter_delete_mode | **smdc** | dm |
| set_bottom_margin | **smgb** | Zk |
| set_bottom_margin_parm | **smgbp** | Zl |
| set_left_margin | **smgl** | ML |
| set_left_margin_parm | **smglp** | Zm |
| set_right_margin | **smgr** | MR |
| set_right_margin_parm | **smgrp** | Zn |
| set_top_margin | **smgt** | Zo |
| set_top_margin_parm | **smgtp** | Zp |
| enter_micro_mode | **smicm** | ZJ |
| enter_insert_mode | **smir** | im |
| keypad_xmit | **smkx** | ks |
| label_on | **smln** | LO |
| meta_on | **smm** | mm |
| enter_standout_mode | **smso** | so |
| enter_underline_mode | **smul** | us |
| enter_xon_mode | **smxon** | SX |
| enter_near_letter_quality | **snlq** | ZK |
| enter_normal_quality | **snrmq** | ZL |
| dot_horz_spacing | **spinh** | Yc |
| dot_vert_spacing | **spinv** | Yb |
| enter_shadow_mode | **sshm** | ZM |
| enter_subscript_mode | **ssubm** | ZN |
| enter_superscript_mode | **ssupm** | ZO |
| subscript_characters | **subcs** | Zu |
| enter_upward_mode | **sum** | ZP |

| | | |
|---|---|---|
| superscript_characters | **supcs** | Zv |
| enter_doublewide_mode | **swidm** | ZF |
| clear_all_tabs | **tbc** | ct |
| to_status_line | **tsl** | ts |
| underline_char | **uc** | uc |
| transparent_underline | **ul** | ul |
| row_address | **vpa** | cv |
| virtual_terminal | **vt** | vt |
| wide_char_size | **widcs** | Yn |
| set_window | **wind** | wi |
| width_status_line | **wsl** | ws |
| eat_newline_glitch | **xenl** | xn |
| ceol_standout_glitch | **xhp** | xs |
| col_addr_glitch | **xhpa** | YA |
| magic_cookie_glitch | **xmc** | sg |
| xoff_character | **xoffc** | XF |
| xon_xoff | **xon** | xo |
| xon_character | **xonc** | XN |
| no_esc_ctlc | **xsb** | xb |
| dest_tabs_magic_smso | **xt** | xt |
| row_addr_glitch | **xvpa** | YD |
| zero_motion | **zerom** | Zx |

## FILES

`/usr/lib/terminfo/?/*`
: compiled device description database

`/usr/src/lib/libcurses/terminfo/*.ti`
: source device descriptions

`/usr/lib/tabset/*`
: tab settings for some devices, in a format appropriate to be output to the device (escape sequences that set margins and tabs)

## SEE ALSO

`curses`(3X), `printf`(3S), `term`(5), `profile`(4), `termcap`(5).
`captoinfo`(1M), `infocmp`(1M), `tic`(1M), `termio`(7), `tty`(7) in the *System Manager's Reference for the DG/UX System*.
`tput`(1) in the *User's Reference for the DG/UX System*.

## CAUTIONS

As described in the "Tabs and Initialization" section above, a device's initialization strings, `is1`, `is2`, and `is3`, if defined, must be output before a `curses`(3X) program is run. An available mechanism for outputting such strings is `tput init` (see `tput`(1) and `profile`(4)).

If a null character (`\0`) is encountered in a string, the null and all characters after it are lost. Therefore it is not possible to code a null character (`\0`) in a string capability and send it to a device (either a terminal or a printer). The suggestion of sending `\0200` where `\0` (null) is needed can succeed only if the device ignores the eighth bit. For example, because all eight bits are used in the standard international ISO character set, devices that adhere to this standard will treat `\0200` differently from `\0`.

Tampering with entries in `/usr/lib/terminfo/?/*` (for example, changing or removing an entry) can affect programs such as `vi`(1) that expect the entry to be present and correct. In particular, removing the description for the dumb terminal causes unexpected problems.

## NAME

`timezone` – set default system time zone and locale

## SYNOPSIS

`/etc/TIMEZONE`, `/etc/TIMEZONE.csh`

## DESCRIPTION

The files `/etc/TIMEZONE` and `/etc/TIMEZONE.csh` set and export the following environment variables:

| | |
|---|---|
| TZ | time zone |
| NLSPATH | search path for message catalogs |
| LANG | local language |

These files are included into other shell scripts (for example, `/etc/profile` and `/etc/cshrc`) to establish this localization information. `/etc/TIMEZONE` is also read by `/etc/init` to initialize the timezone and locale information for the system startup procedures.

To change the values of these environment variables, you may edit these files directly, or use `admdate`(1M) and `admnls`(1M), which can be invoked from `sysadm`(1M).

If `/etc/TIMEZONE` is missing, it is created at system startup by copying the file `/etc/TIMEZONE.proto`. If `/etc/TIMEZONE.csh` is missing, it is created at system startup by copying the file `/etc/TIMEZONE.csh.proto`.

NLSPATH and LANG are described in `environ`(5) and `setlocale`(3). The default value of NLSPATH (in the `proto` files) is "/usr/lib/nls/msg/%L/%N". The default value of LANG is "C".

TZ can be either the name of a timezone database file found under the directory /usr/lib/locale/TZ, preceded by a colon (e.g. ":US/Eastern"), or else a string that describes the timezone rules. The syntax of such a rule string can be described as follows:

| | | |
|---|---|---|
| *TZ* | → | *zone* |
| | | \| *zone signed_time* |
| | | \| *zone signed_time zone* |
| | | \| *zone signed_time zone dst* |
| *zone* | → | *letter letter letter* |
| *signed_time* | → | *sign time* |
| | | \| *time* |
| *time* | → | *hour* |
| | | \| *hour : minute* |
| | | \| *hour : minute : second* |
| *dst* | → | *signed_time* |
| | | \| *signed_time , dst_date , dst_date* |
| | | \| *, dst_date , dst_date* |
| *dst_date* | → | *julian* |
| | | \| *julian / time* |
| *letter* | → | *a* \| *A* \| *b* \| *B* \| ... \| *z* \| *Z* |
| *hour* | → | *00* \| *01* \| ... \| *23* |
| *minute* | → | *00* \| *01* \| ... \| *59* |
| *second* | → | *00* \| *01* \| ... \| *59* |
| *julian* | → | *001* \| *002* \| ... \| *366* |
| *sign* | → | – \| + |

## EXAMPLES

The contents of the file /etc/TIMEZONE could be

```
# Time Zone
TZ=:US/Eastern
export TZ
# Message catalog search path
NLSPATH=/usr/lib/nls/msg/%L/%N
export NLSPATH
# Language
LANG=C
export C
```

A simple setting for TZ for New Jersey could be

```
TZ=EST5EDT
```

where EST is the abbreviation for the main time zone, 5 is the difference, in hours, between GMT (Greenwich Mean Time) and the main time zone, and EDT is the abbreviation for the alternate time zone.

The most complex representation of the same setting, for the year 1986, is

```
TZ="EST5:00:00EDT4:00:00,117/2:00:00,299/2:00:00"
```

where EST is the abbreviation for the main time zone, 5:00:00 is the difference, in hours, minutes, and seconds between GMT and the main time zone, EDT is the abbreviation for the alternate time zone, 4:00:00 is the difference, in hours, minutes, and seconds between GMT and the alternate time zone, 117 is the number of the day of the year (Julian day) when the alternate time zone will take effect, 2:00:00 is the number of hours, minutes, and seconds past midnight when the alternate time zone will take effect, 299 is the number of the day of the year when the alternate time zone will end, and 2:00:00 is the number of hours, minutes, and seconds past midnight when the alternate time zone will end.

A southern hemisphere setting such as the Cook Islands could be

```
TZ="KDT9:30KST10:00,64/5:00,303/20:00"
```

This setting means that KDT is the abbreviation for the main time zone, KST is the abbreviation for the alternate time zone, KST is 9 hours and 30 minutes later than GMT, KDT is 10 hours later than GMT, the starting date of KDT is the 64th day at 5 AM, and the ending date of KDT is the 303rd day at 8 PM.

Starting and ending times are relative to the alternate time zone. If the alternate time zone start and end dates and the time are not provided, the days for the United States that year will be used and the time will be 2 AM. If the start and end dates are provided but the time is not provided, the time will be midnight.

Note that in most installations, TZ is set to the correct value by default when the user logs on, via the local /etc/profile file (see profile(4)).

## NOTES

When the longer format is used, the TZ variable must be surrounded by double quotes as shown.

The system administrator must change the Julian start and end days annually if the longer form of the TZ variable is used.

Setting the time during the interval of change from the main time zone to the alternate time zone or vice versa can produce unpredictable results.

## SEE ALSO

zic(1M), ctime(3C), setlocale(3C), profile(4), environ(5).

## NAME

utmp, wtmp – utmp and wtmp entry formats

## SYNOPSIS

```
#include <sys/types.h>
#include <limits.h>
#include <utmp.h>
```

## DESCRIPTION

These files, which hold user and accounting information for such commands as who(1), write(1), and login(1), have the following structure as defined by <utmp.h>:

```
#define UTMP_FILE "/etc/utmp"
#define WTMP_FILE "/etc/wtmp"
#define ut_name   ut_user

struct utmp {
  char    ut_user[USR_NAME]; /* User login name */
  char    ut_id[4];          /* /etc/inittab id (usually line #) */
  char    ut_line[12];       /* device name (console, lnxx) */
  short   ut_pid;            /* process id */
  short   ut_type;           /* type of entry */
  struct  exit_status {
    short   e_termination;   /* Process termination status */
    short   e_exit;          /* Process exit status */
  } ut_exit;                 /* The exit status of a process
                              * marked as DEAD_PROCESS. */
  time_t  ut_time;           /* time entry was made */
  char    ut_host[16];       /* hostname, if remote */
};

/*  Definitions for ut_type  */
#define EMPTY          0
#define RUN_LVL        1
#define BOOT_TIME      2
#define OLD_TIME       3
#define NEW_TIME       4
#define INIT_PROCESS   5  /* Process spawned by "init" */
#define LOGIN_PROCESS  6  /* A "getty" process waiting for login */
#define USER_PROCESS   7  /* A user process */
#define DEAD_PROCESS   8
#define ACCOUNTING     9
#define UTMAXTYPE      ACCOUNTING /* Largest legal value of ut_type */

/*  Special strings or formats used in the "ut_line" field when  */
/*  accounting for something other than a process  */
/*  No string for the ut_line field can be more than 11 chars +  */
/*  a NULL in length  */

#define RUNLVL_MSG  "run-level %c"
#define BOOT_MSG    "system boot"
#define OTIME_MSG   "old time"
#define NTIME_MSG   "new time"
```

**FILES**

/usr/include/utmp.h
/etc/utmp
/etc/wtmp

**SEE ALSO**

login(1), who(1), write(1), getut(3C), limits.h(4).

End of Chapter

# Chapter 5 Miscellaneous Features

This chapter contains in printed form all the online manual entries for miscellaneous features. The entries are in alphabetical order except for **intro**(5), which is first.

**NAME**

intro – introduction to miscellany

**DESCRIPTION**

This section describes miscellaneous facilities, such as macro packages and character set tables.

## NAME

`ascii` – map of ASCII character set

## DESCRIPTION

`ascii` is a map of the ASCII character set, giving both octal and hexadecimal equivalents of each character, to be printed as needed. It contains:

```
|000 nul |001 soh |002 stx |003 etx |004 eot |005 enq |006 ack |007 bel |
|010 bs  |011 ht  |012 nl  |013 vt  |014 np  |015 cr  |016 so  |017 si  |
|020 dle |021 dc1 |022 dc2 |023 dc3 |024 dc4 |025 nak |026 syn |027 etb |
|030 can |031 em  |032 sub |033 esc |034 fs  |035 gs  |036 rs  |037 us  |
|040 sp  |041 !   |042 "   |043 #   |044 $   |045 %   |046 &   |047 '   |
|050 (   |051 )   |052 *   |053 +   |054 ,   |055 -   |056 .   |057 /   |
|060 0   |061 1   |062 2   |063 3   |064 4   |065 5   |066 6   |067 7   |
|070 8   |071 9   |072 :   |073 ;   |074 <   |075 =   |076 >   |077 ?   |
|100 @   |101 A   |102 B   |103 C   |104 D   |105 E   |106 F   |107 G   |
|110 H   |111 I   |112 J   |113 K   |114 L   |115 M   |116 N   |117 O   |
|120 P   |121 Q   |122 R   |123 S   |124 T   |125 U   |126 V   |127 W   |
|130 X   |131 Y   |132 Z   |133 [   |134 \   |135 ]   |136 ^   |137 _   |
|140 `   |141 a   |142 b   |143 c   |144 d   |145 e   |146 f   |147 g   |
|150 h   |151 i   |152 j   |153 k   |154 l   |155 m   |156 n   |157 o   |
|160 p   |161 q   |162 r   |163 s   |164 t   |165 u   |166 v   |167 w   |
|170 x   |171 y   |172 z   |173 {   |174 |   |175 }   |176 ~   |177 del |

| 00 nul | 01 soh | 02 stx | 03 etx | 04 eot | 05 enq | 06 ack | 07 bel |
| 08 bs  | 09 ht  | 0a nl  | 0b vt  | 0c np  | 0d cr  | 0e so  | 0f si  |
| 10 dle | 11 dc1 | 12 dc2 | 13 dc3 | 14 dc4 | 15 nak | 16 syn | 17 etb |
| 18 can | 19 em  | 1a sub | 1b esc | 1c fs  | 1d gs  | 1e rs  | 1f us  |
| 20 sp  | 21 !   | 22 "   | 23 #   | 24 $   | 25 %   | 26 &   | 27 '   |
| 28 (   | 29 )   | 2a *   | 2b +   | 2c ,   | 2d -   | 2e .   | 2f /   |
| 30 0   | 31 1   | 32 2   | 33 3   | 34 4   | 35 5   | 36 6   | 37 7   |
| 38 8   | 39 9   | 3a :   | 3b ;   | 3c <   | 3d =   | 3e >   | 3f ?   |
| 40 @   | 41 A   | 42 B   | 43 C   | 44 D   | 45 E   | 46 F   | 47 G   |
| 48 H   | 49 I   | 4a J   | 4b K   | 4c L   | 4d M   | 4e N   | 4f O   |
| 50 P   | 51 Q   | 52 R   | 53 S   | 54 T   | 55 U   | 56 V   | 57 W   |
| 58 X   | 59 Y   | 5a Z   | 5b [   | 5c \   | 5d ]   | 5e ^   | 5f _   |
| 60 `   | 61 a   | 62 b   | 63 c   | 64 d   | 65 e   | 66 f   | 67 g   |
| 68 h   | 69 i   | 6a j   | 6b k   | 6c l   | 6d m   | 6e n   | 6f o   |
| 70 p   | 71 q   | 72 r   | 73 s   | 74 t   | 75 u   | 76 v   | 77 w   |
| 78 x   | 79 y   | 7a z   | 7b {   | 7c |   | 7d }   | 7e ~   | 7f del |
```

## SEE ALSO

`terminfo`(4).

## NAME

dg_mknod – data returned by the dg_mknod system call

## SYNOPSIS

```
#include <sys/types.h>
```

## DESCRIPTION

The system call dg_mknod takes a parameter that is a pointer to the structure defined by this include file. This structure defines the node that is created.

```
struct  dg_mknod
{
    mode_t        extended_mode;
    dev_t         device_number;
    char *        symbolic_link_target;
    unsigned long       desired_data_element_blocks;
    unsigned long       data_element_blocks_limit;
    unsigned long       desired_index_element_blocks;
    unsigned long       index_element_blocks_limit;
};
```

The fields of this structure are defined as follows:

`extended_mode`
: The file type and access permissions of the file. The file type is available by AND-ing this field with DG_FILE_TYPE_MASK. The access bits are available by AND-ing this field with (~ DG_FILE_TYPE_MASK). The file type and access are encoded using the constants defined in stat.h and dg_stat.h

`device_number`
: The device specifier to be used if the file to be created is of type 'block-special' or 'character-special'. This field is ignored otherwise.

`symbolic_target_link`
: A null-terminated pathname which will be the target of the file to be created if that file is of type 'symbolic link'. This field is ignored otherwise.

`desired_data_element_blocks`
: The preferred size (in 512-byte blocks) of the data elements of the file to be created. If this size is 0, then the default data element size for the containing file system will be used.

`data_element_blocks_limit`
: The maximum size (in 512-byte blocks) of the data elements of the file to be created. Values in the range starting at the preferred size and working towards the limit are tried until a valid data element size is found.

`desired_index_element_blocks`
: The preferred size (in 512-byte blocks) of the index elements of the file to be created. If this size is 0, then the default data element size for the containing file system will be used.

`index_element_blocks_limit`
: The maximum size (in 512-byte blocks) of the index elements of the file to be created. Values in the range starting at the preferred size and working towards the limit are tried until a valid data element size is found.

## FILES

/usr/include/sys/dg_mknod.h
/usr/include/sys/types.h

## SEE ALSO

dg_mknod(2), dg_stat(5), types(5).

## NAME

dg_stat – data returned by dg_stat and dg_fstat system call

## SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/dg_stat.h>
```

## DESCRIPTION

The system calls dg_stat, and dg_fstat return data whose structure is defined by this include file.

```
struct  dg_stat
{
    dev_t         st_dev;
    ino_t         st_ino;
    mode_t        st_mode;
    nlink_t             st_nlink;
    uid_t         st_uid;
    gid_t         st_gid;
    dev_t         st_rdev;
    off_t         st_size;
    time_t        st_atime;
    unsigned long       st_ausec;
    time_t        st_mtime;
    unsigned long       st_musec;
    time_t        st_ctime;
    unsigned long       st_cusec;
    long             st_pad1[114];
    unsigned long       st_blocks;
    mode_t        extended_mode;
    unsigned long       data_element_blocks;
    unsigned long       index_element_blocks;
    unsigned long       max_cpd_blocks;
    unsigned long       max_cpd_file_nodes;
    unsigned long       cur_cpd_blocks;
    unsigned long       cur_cpd_file_nodes;
};
```

The fields of this structure are defined as follows:

st_dev
: An identifier of the flat file store containing the file. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_ino
: An identifier of the per-file database within the flat file store. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_mode
: The mode of the file, encoded using the constants defined in stat.h. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_nlink`
: The number of links to the file. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_uid`
: The user-id of the file. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_gid`
: The group-id of the file. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_rdev`
: The represented device, giving the major and minor device numbers of the device represented by a special file. This field is meaningful only if the file is of type 'block-special' or 'character-special'. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_size`
: The size of the file in bytes. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_atime`
: The last time the file was accessed. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_ausec`
: The extended-precision portion of st_atime, in microseconds. If such precision is not available, this field will be zero.

`st_mtime`
: The last time the file's contents were modified. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_musec`
: The extended-precision portion of st_mtime, in microseconds. If such precision is not available, this field will be zero.

`st_ctime`
: The last time the file's attributes were changed. The meaning of this field is the same as that of the field of the same name in the stat structure.

`st_cusec`
: The extended-precision portion of st_ctime, in microseconds. If such precision is not available, this field will be zero.

`st_pad`
: Reserved space.

`st_blocks`
: The actual number of blocks allocated for the file.

`extended_mode`
: The extended mode of the file, encoded using the constants defined below and in stat.h.

`data_element_blocks`
: The number of 512-byte blocks used in each of the file's data elements.

`index_element_blocks`
: The number of 512-byte blocks used in each of the file's index elements.

`max_cpd_blocks`
: The maximum number of 512-byte blocks that can be allocated by this file and all of its space descendants. This field has meaning only if the file is a control-point directory. Otherwise, it will be zero. A node is a space descendant of a CPD if it is found in the directory tree descending from the CPD and if no file system mount point boundaries are crossed.

`max_cpd_file_nodes`
: The maximum number of file nodes that can be allocated by this file and all of its space descendants. This field has meaning only if the file is a control-point directory. Otherwise, it will be zero.

`cur_cpd_blocks`
: The current number of 512-byte blocks that have been allocated by this file and all of its space descendants. This field has meaning only if the file is a control-point directory. Otherwise, it will be zero.

`cur_cpd_file_nodes`
: The current number of file nodes that have been allocated by this file and all of its space descendants. This field has meaning only if the file is a control-point directory. Otherwise, it will be zero.

#define DG_FILE_TYPE_MASK      ((unsigned_long) 0xFFFFF000)

The bitmask used to extract the file's type from the *extended_mode* field. The result of AND-ing the file's *extended_mode* with this mask will be one of the following: DG_IFCPD, S_IFDIR, S_IFCHR, S_IFBLK, S_IFREG, S_IFLNK, S_IFIFO, S_IFSOCK. Logically, this field is equivalent to the S_IFMT mask defined in stat.h, except that DG_FILE_TYPE_MASK allows for detection of DG/UX-only extended file types, such as DG_IFCPD (see below).

#define DG_IFCPD      ((unsigned long) 0x00010000)

Control-point directory file type.

#define DG_IFSTREAMS      ((unsigned long) 0x00020000)

Streams special file type.

## FILES

`/usr/include/sys/dg_stat.h`
`/usr/include/sys/types.h`

## SEE ALSO

`dg_stat(2)`, `dg_fstat(2)`, `stat(5)`, `types(5)`.

## NAME

elink – Environment variable sensitive file link

## DESCRIPTION

An elink is the mechanism used to encode environment variable-sensitive references into symbolic links. This non-standard use of symbolic links is used by a number of software development tools such as cc to find files that pertain to a development environment selected with sde-target(1).

The elink mechanism is incorporated into a number of software development tools to support the generation of programs and libraries that conform to different standards on the same machine. It is implemented by inserting code into the error paths of special versions of some system library routines.

An elink is a symbolic link whose value conforms to the following grammar:

```
<elink>      ::= "elink:" <sp> <pathname> <sp> <comment>
<pathname>   ::= <pathname> <evref> <pathname>
               | <pathchars>
               |
<evref>      ::= "$" <evname>
               | "${" <evname> "}"
               | "${" <evname> ":-" <default> "}"
<evname>     ::= <id>
<default>    ::= <id>
<pathchars>  ::= <id>
               | <pathchars> "/" <pathchars>
<comment>    ::= "#" <text>
               |
```

<sp> is zero or more tab or space characters.
<id> is a sequence of identifier characters.
<text> is zero or more of any character except null.

This grammar is ambiguous in a number of ways that are not significant. For example, you can't tell how <evref> terminates if it is not the "${}" form and it is followed by an <id>.

Within one of the specially modified tools, when an operation such as open(2) is performed, nothing is done unless an error would be reported. In that case, the pathname argument is checked to see if it or any component is a symbolic link. If one is found, then the contents of the link are checked to see if they conform to the above grammar. If so, the <pathname> component is extracted, environment variable substitution is performed, and the operation is tried again, substituting the newly created pathname for the value of the symbolic link in the original argument. The previous steps are repeated until the operation succeeds or the argument does not resolve to a valid symbolic link (and an error is reported).

Environment variable substitution is defined as the replacement of all <evref> components in the <pathname> with the appropriate environment variable value. If a given environment variable is not defined, then the <default> value is used if it is supplied; otherwise "" is used.

For example, consider the following symbolic link:

```
/usr/lib/libc.a ->
elink:/usr/sde/${TARGET_BINARY_INTERFACE:-m88kdgux}/
      usr/lib/libc.a # See sde-target(1)
```

Links begin with "`elink:`" to give a visual cue that something is different about this symbolic link. The comment allows the insertion of other informational pointers.

This link makes reference to one environment variable although more could have been used. If the environment variable `TARGET_BINARY_INTERFACE` is not defined when a tool such as `ld`(1) attempts to open `/usr/lib/libc.a` then the tool will use the path `/usr/sde/m88kdgux/usr/lib/libc.a`. If `TARGET_BINARY_INTERFACE` is some value such as `m88kbcs`, the the path used to find `libc.a` will include the value of the variable such as `/usr/sde/m88kbcs/usr/lib/libc.a`.

It should be noted that the elink mechanism is incorporated only in a small set of tools. Other tools that attempt to use a pathname that contains an elink will get an error indicating that the file does not exist.

## SEE ALSO

`sde-target`(1), `sde`(5).

## NAME

environ – user environment

## DESCRIPTION

When a process begins execution, exec routines make available an array of strings called the environment [see exec(2)]. By convention, these strings have the form *variable=value*, for example, PATH=/sbin:/usr/sbin. These environmental variables provide a way to make information about a program's environment available to programs. The following environmental variables can be used by applications and are expected to be set in the target run-time environment.

HOME
: The name of the user's login directory, set by login(1) from the password file (see passwd(4)).

LANG
: The string used to specify localization information that allows users to work with different national conventions. The setlocale(3C) function looks for the LANG environment variable when it is called with "" as the *locale* argument. LANG is used as the default locale if the corresponding environment variable for a particular category is unset.

For example, when setlocale() is invoked as

```
setlocale(LC_CTYPE, ""),
```

setlocale() will query the LC_CTYPE environment variable first to see if it is set and non-null. If LC_CTYPE is not set or null, then setlocale() will check the LANG environment variable to see if it is set and non-null. If both LANG and LC_CTYPE are unset or null, the default C locale will be used to set the LC_CTYPE category.

Most commands will invoke

```
setlocale(LC_ALL, "")
```

prior to any other processing. This allows the command to be used with different national conventions by setting the appropriate environment variables.

The system-wide default value for LANG can be changed with the sysadm(1M) command.

The following environment variables are supported to correspond with each category of setlocale(3C):

LC_COLLATE
: This category specifies the collation sequence being used. The information corresponding to this category is stored in a database created by the colltbl(1M) command. This environment variable affects strcoll(3C), strxfrm(3C) and the regular expression code (see regexpr(3C)).

LC_CTYPE
: This category specifies character classification, character conversion, and widths of multibyte characters. The information corresponding to this category is stored in a database created by the chrtbl(1M) command. The default C locale corresponds to the 7-bit ASCII character set. This environment variable is used by ctype(3C), mbchar(3C), and many commands; for example: cat(1), ed(1), ls(1), and vi(1).

LC_MESSAGES This category specifies the language of the AT&T-style message database being used. For example, an application may have one message database with French messages, and another database with German messages. Message databases are created by the mkmsgs(1M) command. This environment variable is used by exstr(1), gettxt(1), gettxt(3C), and srchtxt(1). The X/Open-style message facility does not use this variable.

LC_MONETARY This category specifies the monetary symbols and delimiters used for a particular locale. The information corresponding to this category is stored in a database created by the montbl(1M) command. This environment variable is used by localeconv(3C).

LC_NUMERIC This category specifies the decimal and thousands delimiters. The information corresponding to this category is stored in a database created by the chrtbl(1M) command. The default C locale corresponds to "." as the decimal delimiter and no thousands delimiter. This environment variable is used by localeconv(3C), printf(3C), and strtod(3C).

LC_TIME This category specifies date and time formats. The information corresponding to this category is stored in a database specified in strftime(4). The default C locale corresponds to U.S. date and time formats. This environment variable is used by many commands and functions; for example: at(1), calendar(1), date(1), strftime(3C), and getdate(3C).

MSGVERB Controls which standard format message components fmtmsg selects when messages are displayed to stderr [see fmtmsg(1) and fmtmsg(3C)].

SEV_LEVEL Define severity levels and associate and print strings with them in standard format error messages [see addseverity(3C), fmtmsg(1), and fmtmsg(3C)].

NETPATH A colon-separated list of network identifiers. A network identifier is a character string used by the Network Selection component of the system to provide application-specific default network search paths. A network identifier must consist of non-NULL characters and must have a length of at least 1. No maximum length is specified. Network identifiers are normally chosen by the system administrator. A network identifier is also the first field in any /etc/netconfig file entry. NETPATH thus provides a link into the /etc/netconfig file and the information about a network contained in that network's entry. /etc/netconfig is maintained by the system administrator. The library routines described in getnetpath(3N) access the NETPATH environment variable.

NLSPATH Contains a sequence of templates which the X/Open-style message facility uses when attempting to locate message catalogs (see catopen(3C)). The AT&T-style message facility does not use this variable. Each template consists of an optional prefix, one or more substitution fields, a filename and an optional suffix.

For example:

```
NLSPATH="/usr/lib/nls/msg/%N.cat"
```

defines that catopen() should look for all message catalogs in the directory /usr/lib/nls/msg, where the catalog name should be constructed from the *name* parameter passed to catopen(), %N, with the suffix .cat.

Substitution fields consist of a % symbol, followed by a single-letter keyword. The following keywords are currently defined:

| | |
|---|---|
| %N | The value of the *name* parameter passed to catopen(). |
| %L | The value of LANG. |
| %l | The language element from LANG. |
| %t | The territory element from LANG. |
| %c | The codeset element from LANG. |
| %% | A single % character. |

An empty string is substituted if the specified value is not currently defined. The separators "_" and "." are not included in %t and %c substitutions.

Templates defined in NLSPATH are separated by colons (:). A leading colon or two adjacent colons (::) is equivalent to specifying %N.

For example:

```
NLSPATH=":%N.cat:/usr/lib/nls/msg/%L/%N.cat"
```

indicates to catopen() that it should look for the requested message catalog in *name*, *name*.cat and /usr/lib/nls/msg/$LANG/*name*.cat.

The system-wide default value for NLSPATH can be changed with the sysadm(1M) command.

PATH The sequence of directory prefixes that sh(1), time(1), nice(1), nohup(1), etc., apply in searching for a file known by an incomplete path name. The prefixes are separated by colons (:). login(1) sets PATH=/usr/bin. (For more detail, see sh(1).)

TERM The kind of terminal for which output is to be prepared. This information is used by commands, such as mm(1) or vi(1), which may exploit special capabilities of that terminal.

CFTIME Historically, the default format string to be used by the date(1) command and the ascftime() and cftime() routines (see strftime(3c)). If CFTIME is not set or is null, the default format string specified in the /lib/cftime/LANGUAGE file (if it exists) is used in its place (see cftime(4)). The use of CFTIME has generally been subsumed by LANG and LC_TIME.

CHRCLASS Historically, a value that corresponds to a file in /lib/chrclass containing character classification and conversion information. This information was used by commands (such as cat(1), ed(1), and sort(1)) to classify characters as alphabetic, printable, upper case, and so on, and to convert characters to upper or lower case. The use of `CHRCLASS` has generally been subsumed by `LANGF1 and LC_CTYPE`. For more detail, see `ctype`(3C).

`LANGUAGE` Historically, a language for which a printable file by that name exists in `/lib/cftime`. This information was used by commands (such as `date`(1), `ls`(1), and `sort`(1)) to print date and time information in the language specified. The use of `LANGUAGE` has generally been subsumed by `LANG` and `LC_TIME`.

`TZ` Time zone information. The contents of the environment variable named `TZ` are used by the functions `ctime`(3C), `localtime()` (see `ctime`(3C)), `strftime`(3C) `ascftime()` (see `strftime`(3C)), `cftime()` (see `strftime`(3C)), and `mktime`(3C) to override the default timezone. The value of `TZ` has one of the two forms (spaces inserted for clarity):

*:characters*
or:

*std offset dst offset, rule*

If `TZ` is of the first format (i.e., if the first character is a colon), the string following the colon is the name of the timezone that will be loaded in from the `/usr/lib/locale/TZ` directory. For example, if `TZ` was set to `:US/Eastern`, it would load the `/usr/lib/local/TZ/US/Eastern` timezone definition file. The timezones under this directory are produced with the `zic`(1) command.

The expanded format (for all `TZ`s whose value does not have a colon as the first character) is as follows:

*stdoffset* [ *dst* [ *offset* ] , [ *start* [ */time* ] , *end* [ */time* ] ] ]

Where:

*std* and *dst*
Three or more bytes that are the designation for the standard (*std*) and daylight savings time (*dst*) timezones. Only *std* is required, if *dst* is missing, then daylight savings time does not apply in this locale. Upper- and lower-case letters are allowed. Any characters except a leading colon (:), digits, a comma (,), a minus (–), a plus (+), or an ASCII NUL are allowed.

*offset* Indicates the value one must add to the local time to arrive at Coordinated Universal Time. The offset has the form:

*hh* [ : *mm* [ : *ss* ] ]

The minutes (*mm*) and seconds (*ss*) are optional. The hour (*hh*) is required and may be a single digit. The *offset* following *std* is required. If no *offset* follows *dst*, daylight savings time is assumed to be one hour ahead of standard time. One or more digits may be used; the value is always interpreted as a decimal number. The hour must be between 0 and 24, and the minutes (and seconds) if present between 0 and 59. Out of range values may cause unpredictable behavior. If preceded by a "–", the timezone is east of the Prime Meridian; otherwise it is west (which may be indicated by an optional preceding "+" sign).

*rule* Indicates when to change to and back from summer time. The *rule* has the form:

*start*/*time*, *end*/*time*

Which indicates when to change to and back from daylight savings time, where *start*/*time* describes when the change from standard time to daylight savings time occurs, and *end*/*time* describes when the change back happens. Each *time* field describes when, in current local time, the change is made.

The formats of *start* and *end* are one of the following:

`J`*n* The Julian day $n$ ($1 \leq n \leq 365$). Leap days are not counted. That is, in all years, February 28 is day 59 and March 1 is day 60. It is impossible to refer to the occasional February 29.

*n* The zero-based Julian day ($0 \leq n \leq 365$). Leap days are counted, and it is possible to refer to February 29.

`M`*m.n.d* The $d^{th}$ day, ($0 \leq d \leq 6$) of week $n$ of month $m$ of the year ($1 \leq n \leq 5, 1 \leq m \leq 12$), where week 5 means "the last $d$-day in month $m$" which may occur in either the fourth or the fifth week). Week 1 is the first week in which the $d^{th}$ day occurs. Day zero is Sunday.

The *time* has the same format as *offset* except that no leading sign ("−" or "+") is allowed. The default, if *time* is not given is 02:00:00.

Further names may be placed in the environment by the `export` command and *name=value* arguments in `sh`(1), or by `exec`(2). It is unwise to conflict with certain shell variables that are frequently exported by `.profile` files: `MAIL`, `PS1`, `PS2`, `IFS` (see `profile`(4)).

Whenever `ascftime`(), `cftime`(), `ctime`(), `localtime`(), `mktime`(), or `strftime`() is called, the time zone names contained in the external variable `tzname`() shall be set as if the `tzset`() function had been called.

Applications are explicitly allowed to change `TZ` and have the changed `TZ` apply to themselves.

The system-wide default value for `TZ` can be changed with the `sysadm`(1M) command.

NOTE:

There is an unfortunate potential for confusion with time zones identified by an offset from GMT. The `TZ` value *GMT+5*, according to the rules presented here, is equivalent to *EST5* — 5 hours West of GTM. There is also a timezone definition file that can be used by setting `TZ` to *:GMT+5*, but this file defines the time zone 5 hours *East* of GMT. Existing practice requires that both these notations be supported.

## SEE ALSO

`chrtbl`(1M), `colltbl`(1M), `montbl`(1M), `netconfig`(4), `strftime`(4), `passwd`(4), `profile`(4) in the *System Manager's Reference*.

`exec`(2), `addseverity`(3C), `catopen`(3C), `ctime`(3C), `ctype`(3C), `fmtmsg`(3C), `getdate`(3C), `getenv`(3C), `gettxt`(3C), `localeconv`(3C), `mbchar`(3C), `mktime`(3C), `printf`(3C), `strcoll`(3C), `strftime`(3C), `strtod`(3C), `strxfrm`(3C), `strftime`(4), `time`(4), `timezone`(4).

`cat`(1), `date`(1), `ed`(1), `gencat`(1), `fmtmsg`(1), `ls`(1), `login`(1), `mkmsgs`(1), `nice`(1), `nohup`(1), `sh`(1), `sort`(1), `time`(1), `vi`(1), `zic`(1) in the *User's Reference*.

`getnetpath`(3N), in the *Programmer's Guide: Networking Interfaces*.

`mm`(1) on the Documenter's Tool Kit (DTK) tape and the `mm` chapter in *Using the Documenter's Tool Kit on* and *Documenter's Tool Kit Technical Summary for the DG/UX System*.

## COPYRIGHTS

## NAME

eucioctl – generic interface to EUC handling TTY drivers and modules

## SYNOPSIS

```
#include <sys/eucioctl.h>

ioctl(int fd, I_STR, struct strioctl *sb);
```

## DESCRIPTION

This interface is implemented in TTY drivers and pushable *STREAMS* modules that handle EUC codes. It is intended as a generic interface for EUC handling, to eliminate an explosion of "module specific" ioctl calls that would otherwise be necessary, and to provide uniformity in dealing with EUC codesets in the TTY subsystem.

Several calls are defined. The first two calls take an argument, which is expected to be a pointer to an eucioc structure, defined in the header file <sys/eucioctl.h>:

```
struct eucioc {
      unsigned char eucw[4];
      unsigned char scrw[4];
};
typedef struct eucioc   eucioc_t;
```

In all cases, these calls return non-zero on failure. Failure should be usually taken as an indication that the current driver, or line discipline module, does not support EUC in which case errno will be set to EINVAL. For the EUC_WSET and EUC_WGET calls errno will be set will be set to EPROTO if the struct eucioc argument is invalid.

EUC_WSET
: This call takes a pointer to an eucioc structure, and uses it to set the EUC line discipline's local definition for the codeset widths to be used for subsequent operations. Within the *STREAM*, the line discipline may optionally notify other modules of this setting via M_CTL messages.

EUC_WGET
: This call takes a pointer to an eucioc structure, and returns in it the EUC codeset widths currently in use by the EUC line discipline. It need be recognized *only* by line discipline modules.

The following calls take no arguments. They should only fail if the driver (at the bottom of the TTY STREAM) does not recognize EUC codes. Drivers that support EUC, whether the STREAM contains modules that respond to the calls or not, will *recognize* the calls and acknowledge them. These calls are normally only *interpreted* by modules that have modes other than ASCII, and/or do some form of I/O conversion that normally prevents a program from receiving non-EUC characters in its byte stream. All of these calls, when received by modules, are passed down the TTY STREAM, to be ultimately acknowledged by the TTY driver.

EUC_MSAVE
: This call has no effect on modules that are currently in ASCII mode. Otherwise (i.e., for modules *not* in ASCII mode), the following actions are taken by all modules that recognize this call: (1) the current "mode" status is saved, (2) the mode is changed to ASCII mode immediately.

EUC_MREST
: If a mode was saved via a previous EUC_MSAVE call, the saved mode is restored, and the "saved state" flag is cleared. If the mode was not previously saved, this call has no effect. (The exact semantics are somewhat dependent on the module, since some

| | |
|---|---|
| | modules may respond to specific user-requests to switch modes, even while a mode is being saved via `EUC_MSAVE`.) |
| `EUC_IXLOFF` | If a module is currently in a state where "input conversion" is being performed on the incoming byte stream, then input conversion is turned off, and the module's "mode" status is saved. If no input conversion is being performed, there is no effect on the module. The purpose of this call is to provide a way of insuring a "pure" byte stream to the program. The byte stream while input conversion is off is, of course, not guaranteed to be a stream of EUC characters. Turning off input conversion is roughly equivalent to the old concept of "raw" mode, if used in conjunction with ICANON off. It should normally not be used by applications. |
| `EUC_IXLON` | If a module previously saved its state and turned off input conversion, then input conversion is restored (i.e., turned back on); otherwise, there is no effect. |
| `EUC_OXLOFF` | In a manner similar to `EUC_IXLOFF`, any "output conversion" is turned off, and the current mode status saved. |
| `EUC_OXLON` | In a manner similar to `EUC_IXLON`, any saved "output conversion" status is restored (i.e., output conversion is turned back on if previously turned off via `EUC_OXLOFF`). |

### Limitations

Drivers and modules that support EUC should all respond appropriately to these calls, depending on their type. Line disciplines must respond to `EUC_WSET` and `EUC_WGET`, changing their current codeset sizes to match `EUC_WSET` requests. All TTY STREAMS modules that do any input or output conversion should recognize the other calls; modules that do no codeset conversion are not required to recognize the calls, but *must* pass them through. Drivers that support EUC TTY STREAMS must all acknowledge the ON/OFF calls, whether the drivers themselves are affected or not, since these calls are purposely *not* acknowledged by modules which receive them; they are intended to be made available for affecting all modules in *the whole STREAM*.

## FILES

`/usr/include/sys/eucioctl.h`

## SEE ALSO

eucset(1).

## NOTES

Adherence to this protocol for all EUC handling modules is strongly encouraged in order to increase portability and language-independence of applications. These calls are intended as a small set of primitives to help reduce an anticipated plethora of module- and language-dependent operations.

## NAME

fcntl – file control options

## SYNOPSIS

```
#include <fcntl.h>
```

## DESCRIPTION

The `fcntl`(2) function helps you control open files. This include file describes *commands* and *arguments* to `fcntl` and open(2).

```
/* Flag values accessible to open(2) and fcntl(2) */
/* (The first three can only be set by open) */

#define O_RDONLY   0
#define O_WRONLY   1
#define O_RDWR     2
#define O_NDELAY   04    /* Non-blocking I/O */
#define O_APPEND   010   /* append (writes guaranteed at the end) */

/* Flag values accessible only to open(2) */
#define O_CREAT 00400    /* open with file create (uses 3rd open arg)*/
#define O_TRUNC 01000    /* open with truncation */
#define O_EXCL  02000    /* exclusive open */

/* fcntl(2) commands */
#define F_DUPFD   0      /* Duplicate fildes */
#define F_GETFD   1      /* Get the `close-on-exec' flag */
#define F_SETFD   2      /* Set the `close-on-exec' flag */
#define F_GETFL   3      /* Get file flags */
#define F_SETFL   4      /* Set file flags */
#define F_GETLK   5      /* Get record lock status */
#define F_SETLK   6      /* Set record lock or fail */
#define F_SETLKW  7      /* Set record lock or pend */
#define F_CHKFL   8      /* Check flags for validity */
#define F_FREESP  11     /* Free up file space */
#define F_GETOWN  65536 /* Get owner of fildes */
#define F_SETOWN  65537 /* Set owner of fildes */
```

## SEE ALSO

fcntl(2), open(2).

## NAME

hier – DG/UX file system hierarchy

## DESCRIPTION

The following outline gives a quick tour through a representative directory hierarchy. The basis of the outline is the DG/UX operating system. It is not exhaustive.

```
/       root
/dgux   the kernel binary (DG/UX System itself)
/lost+found
        directory for connecting detached files for fsck(1M)
/dev/   devices (7)
        console
                system console,
        tty[0-9]*
                terminals, tty(7)
        ttyp[0-9]*
                pseudo terminals,
        dsk/*   logical disks,
        rdsk/*  raw logical disks,
        pdsk/*  physical disks
        rpdsk/*
                raw physical disks
        mt/*    magnetic tapes,
        rmt/*   raw magnetic tapes,
        lp      line printer, lp(7)
        null    the null device; i.e., the "bit bucket"
        kmem    logical kernel memory
        mem     physical memory
        error   the error device error(7)
        ...
/bin/   utility programs, cf /usr/bin/ (1)
        as      Data General macro assembler
        cc      C compiler executive, cf /usr/lib/ccomp, /lib/cpp
        csh     C shell
        sh      Bourne shell
        ...
/lib/   object libraries, etc., cf /usr/lib/
        libc.a  system calls, standard I/O, etc. (2,3,3S)
        ...
        cpp     C preprocessor
        ...
/etc/   essential data and maintenance utilities; sect (1M)
...
        passwd  password file, passwd(5)
        group   group file, group(5)
        init    the parent of all processes, init(1M)
        inittab the init configuration table inittab(5)
        rc.init shell program to enter init states (0, 1, ...)  init(1M), rc(1M)
        rc[S0123456].d
                links to init.d scripts for actions in init states 0, 1, ...  init(1M),
                rc(1M)
        init.d  scripts for rc.d directories init(1M), rc(1M)
        getty   initial part of login sequence getty(1M)
        gettydefs
                terminal modes for getty gettydefs(5)
        login   the login program (final part of login sequence) login(1M)
        motd    message of the day, login(1)
        profile global sh(1) startup script sh(1)
        login.csh
                global csh(1) startup script csh(1)
        stdprofile
                prototype local sh(1) startup script sh(1)
        stdlogin
                prototype local csh(1) startup script csh(1)
        fstab   file system configuration table fstab(5)
        mount   mount(1M)
        mnttab  mounted file table, mnttab(5)
        dump    dump program dump(1M)
        dumpdates
                dump history, dump(1M)
        restore restore program restore(1M)
        cron    the clock server, cron(1M)
        wtmp,   login history, utmp(5)
        ermes   file containing text of system error messages, referenced by
                perror(3C)
        hosts   host name to network address mapping file, hosts(5)
        networks
                network name to network number mapping file, networks(5)
        protocols
                protocol name to protocol number mapping file, protocols(5)
        services
                network services definition file, services(5)
        ...
/tmp/   temporary files, usually on a fast device, cf /usr/tmp/
        e*      used by ed(1)
        ctm*    used by cc(1)
        ...
/usr/   mounted file system, general-pupose directory
        adm/    administrative information
                acct/*  system accounting data files
                sulog   log of the invocations of the su(1) command
/usr/bin/
        utility programs, to keep /bin/ small
        tmp/    temporaries, to keep /tmp/ small
                stm*    used by sort(1)
        dgc/*   the C compiler proper and associated files
        f77/*   the FORTRAN-77 compiler proper and associated files
        mail/*  the directory where mail messages are stored
        news/*  the directory where news items are stored
        include/
                standard #include files
                a.out.h object file layout, a.out(5)
                stdio.h standard I/O, intro(3)
                math.h (3M)
                net/    network header files
                ...
                sys/    system-defined layouts
                ...
        lib/    object libraries, etc., to keep /lib/ small
                acct/*  account programs and shell scripts
                crontab
                        file specifying actions for cron(1M) to take
                atrun   scheduler for at(1)
                lint/   utility files for lint
                        lint[12] subprocesses for lint(1)
                        llib-lc  dummy declarations for /lib/libc.a, used by lint(1)
                        llib-lm  dummy declarations for /lib/libc.m
                        ...
                ...
                tmac/   macros for nroff(1)
                        tmac.an
                        tmac.m
                        ...
                uucp/   programs and data for uucp(1c)
                        L.sys    remote system names and numbers
                        uucico   the real copy program
                        ...
                units   conversion tables for units(1)
                eign    list of English words to be ignored by ptx(1)
/usr/catman/
        online manual pages for man(1)
        u_man/
                User's Reference for the DG/UX System
                man0/   general: contents, permuted index
                        contents.0.z
                        index.0.z
                man1/   user commands and application programs
                        acctcom.1.z
                        alpq.1.z
                        ...
                man5/   miscellaneous features
                        editread.5.z
                        ...
        p_man/
                Programmer's Reference for the DG/UX System
                man1/   commands
                        admin.1.z
                        ar.1.z
                        ...
                man2/   system calls
                        accept.2.z
                        access.2.z
                        ...
                man3/   runtime libraries
                        a641.3c.z
                        ...
                man4/  file formats
                       a.out.4.z
                       ...
                man5/  miscellaneous features
                       ascii.5.z
                       ...
                man6/  networking protocols
                       unix_ipc.6f.z
        a_man/
                System Manager's Reference for the DG/UX System
                man1/  system maintenance commands
                       accept.1m.z
                       acct.1m.z
                       ...
                man4/  file formats for system maintenance commands
                       dfm.4m.z
                       ...
                man7/  special files
                       alp.7.z
                       ...
                man8/  system maintenance procedures
                       crash.8.z
                       ...
        preserve/
                editor temporaries preserved here after crashes/hangups
        public/ binaries of user programs - write permission to everyone
        spool/  delayed execution files
                at/    used by at(1)
                uucp/  work files and staging area for uucp(1c)
                       LOGFILE
                              summary log
                       LOG.* log file for one transaction
        tmp/    temporary files
        wd      initial working directory of a user, typically wd is the user's login
                name
                .profile   set environment for sh(1), environ(7)
                .cshrc     startup file for csh(1)
                .editreadrc
                           startup file for Editread command-line editor
                .exrc      startup file for ex(1)
                .mailrc    startup file for mail(1)
                .netrc     startup file for various network programs
                calendar   user's datebook for calendar(1)
```

## SEE ALSO

find(1), grep(1), ls(1) in the *User's Reference for the DG/UX System*.

## CAUTION

The position of files is subject to change without notice.

## NAME

`langinfo` – language information constants

## SYNOPSIS

```
#include <langinfo.h>
```

## DESCRIPTION

This header file contains the constants used to identify items of langinfo data. The mode of *items* is given in `nl_types`.

| | |
|---|---|
| `DAY_1` | Locale's equivalent of 'sunday' |
| `DAY_2` | Locale's equivalent of 'monday' |
| `DAY_3` | Locale's equivalent of 'tuesday' |
| `DAY_4` | Locale's equivalent of 'wednesday' |
| `DAY_5` | Locale's equivalent of 'thursday' |
| `DAY_6` | Locale's equivalent of 'friday' |
| `DAY_7` | Locale's equivalent of 'saturday' |
| `ABDAY_1` | Locale's equivalent of 'sun' |
| `ABDAY_2` | Locale's equivalent of 'mon' |
| `ABDAY_3` | Locale's equivalent of 'tue' |
| `ABDAY_4` | Locale's equivalent of 'wed' |
| `ABDAY_5` | Locale's equivalent of 'thur' |
| `ABDAY_6` | Locale's equivalent of 'fri' |
| `ABDAY_7` | Locale's equivalent of 'sat' |
| `MON_1` | Locale's equivalent of 'january' |
| `MON_2` | Locale's equivalent of 'febuary' |
| `MON_3` | Locale's equivalent of 'march' |
| `MON_4` | Locale's equivalent of 'april' |
| `MON_5` | Locale's equivalent of 'may' |
| `MON_6` | Locale's equivalent of 'june' |
| `MON_7` | Locale's equivalent of 'july' |
| `MON_8` | Locale's equivalent of 'august' |
| `MON_9` | Locale's equivalent of 'september' |
| `MON_10` | Locale's equivalent of 'october' |
| `MON_11` | Locale's equivalent of 'november' |
| `MON_12` | Locale's equivalent of 'december' |
| `ABMON_1` | Locale's equivalent of 'jan' |
| `ABMON_2` | Locale's equivalent of 'feb' |
| `ABMON_3` | Locale's equivalent of 'mar' |
| `ABMON_4` | Locale's equivalent of 'apr' |
| `ABMON_5` | Locale's equivalent of 'may' |

| | |
|---|---|
| ABMON_6 | Locale's equivalent of 'jun' |
| ABMON_7 | Locale's equivalent of 'jul' |
| ABMON_8 | Locale's equivalent of 'aug' |
| ABMON_9 | Locale's equivalent of 'sep' |
| ABMON_10 | Locale's equivalent of 'oct' |
| ABMON_11 | Locale's equivalent of 'nov' |
| ABMON_12 | Locale's equivalent of 'dec' |
| RADIXCHAR | Locale's equivalent of '.' |
| THOUSEP | Locale's equivalent of ',' |
| YESSTR | Locale's equivalent of 'yes' |
| NOSTR | Locale's equivalent of 'no' |
| CRNCYSTR | Locale's currency symbol |
| D_T_FMT | Locale's default format for date and time |
| D_FMT | Locale's default format for the date |
| T_FMT | Locale's default format for the time |
| AM_STR | Locale's equivalent of 'AM' |
| PM_STR | Locale's equivalent of 'PM' |

This information is retrived by `nl_langinfo`.

The items `CRNCYSTR`, `RADIXCHAR` and `THOUSEP` are extracted from the fields `currency_symbol`, `decimal_point` and `thousands_sep` in the structure returned by `localeconv`.

The items `T_FMT`, `D_FMT`, `D_T_FMT`, `YESSTR` and `NOSTR` are retrived from a special message catalog named `Xopen_info` which should be generated for each locale supported and installed in the appropriate directory [see `gettxt`(3C) and `mkmsgs`(1M)]. This catalog should have the messages in the order `T_FMT`, `D_FMT`, `D_T_FMT`, `YESSTR` and `NOSTR`.

All other items are as returned by `strftime`.

## SEE ALSO

`chrtbl`(1M), `mkmsgs`(1M), `gettxt`(3C), `localeconv`(3C), `nl_langinfo`(3C), `strftime`(3C), `cftime`(4), `nl_types`(5).

## NAME

`legend` – Debugging information technology

## DESCRIPTION

Legend debugging information (or legends for short) is used by the `sdb`(1) and `dbx`(1) debuggers when debugging an ELF executable and always used by the `mxdb`(1) debugger. It is created during compilation typically by `as`(1) which calls the `ctl`(1) translator.

Traditional UNIX compilation systems control debugging information by the use of a `-g` option. If the `-g` option is present on the compiler command line (e.g. "cc -g") then debugging information is generated. Legend technology provides a number of options that can't be coded into a single yes or no option but many existing applications have makefiles and shell scripts that users don't want to modify. The legend options, therefore, are controlled by an environment variable called LEGENDS.

## OPTIONS

The following values can be placed in the LEGENDS environment variable, separated by blanks, to control the generation of legends.

`-external`
: Store the legend data in a separate file. If the target file is named "prog.o", then the legend will be stored in a file named "prog.lg". This reduces the size of object files, libraries and executables, significantly saving link time as well as disk space.

`-no-external`
: Store legend data in the object file. This is the default.

`-compress`
: Legends come in two forms that allow you to make a speed/space trade-off. If present, this option requests that legends be generated in a compressed form. You can mix compressed and uncompressed legends into the same application.

`-no-compress`
: Don't compress the legend. This is the default.

`-keep-std`
: This option only makes sense when creating a COFF object file. If present, it directs the legend translator to preserve the COFF information in addition to generating a legend. This allows the use of COFF debuggers in addition to `mxdb`(1) on resulting executables. By default the COFF information is deleted.

`-no-keep-std`
: Don't preserve COFF information. This is the default.

`-v` Print the version of ctl to stderr.

`-warn` Print warning messages. They are suppressed by default.

## SEE ALSO

`ctl`(1), `cc`(1), `gcc`(1), `ghcc`(1), `ghf77`(1), `ghpc`(1), `as`(1), `mxdb`(1), `sdb`(1), `dbx`(1)

## NAME

`math` – math functions and constants

## SYNOPSIS

`#include <math.h>`

## DESCRIPTION

This file contains declarations of all the functions in the Math Library (described in Section 3M), as well as various functions in the C Library (Section 3C) that return floating-point values.

It defines the structure and constants used by the `matherr`(3M) error-handling mechanisms, including the following constant used as a error-return value:

| | |
|---|---|
| `HUGE` | The maximum value of a single-precision floating-point number. |

The following mathematical constants are defined for user convenience:

| | |
|---|---|
| `M_E` | The base of natural logarithms (*e*). |
| `M_LOG2E` | The base-2 logarithm of *e*. |
| `M_LOG10E` | The base-10 logarithm of *e*. |
| `M_LN2` | The natural logarithm of 2. |
| `M_LN10` | The natural logarithm of 10. |
| `M_PI` | $\pi$, the ratio of the circumference of a circle to its diameter. |
| `M_PI_2` | $\pi/2$. |
| `M_PI_4` | $\pi/4$. |
| `M_1_PI` | $1/\pi$. |
| `M_2_PI` | $2/\pi$. |
| `M_2_SQRTPI` | $2/\sqrt{\pi}$. |
| `M_SQRT2` | The positive square root of 2. |
| `M_SQRT1_2` | The positive square root of 1/2. |

The following mathematical constants are also defined in this header file:

| | |
|---|---|
| `MAXFLOAT` | The maximum value of a non-infinite single-precision floating point number. |
| `HUGE_VAL` | positive infinity. |

For the definitions of various machine-dependent constants, see `values`(5).

## SEE ALSO

`intro`(3), `matherr`(3M), `values`(5).

## NAME

`misalign` – handle misaligned memory access faults

## DESCRIPTION

The Motorola M88000 microprocessor family, on which the Data General AViiON computers are based, requires that data be aligned in memory to their lengths. If the address of a datum is not an integral multiple of the datum's length, a reference to the datum will cause a misaligned access fault. For example, if a program attempts to fetch a 16-bit value from an odd address, a misaligned access fault occurs. A misaligned access fault results in the delivery of a SIGBUS signal to the application. If the application has not defined a SIGBUS signal handler, the application terminates with a "Bus error" message.

A program can use the facilities defined herein to repair misaligned access faults that it incurs. These facilities can be useful in porting applications that were written for computers that don't impose alignment restrictions as strict as those of the M88000 family. The facilities are offered in three forms, for generality and convenience:

- functions to repair misaligned access faults with which you can construct your own SIGBUS signal handler
- predefined SIGBUS signal handlers that are built from the repair functions mentioned above
- a link-time mechanism to have one of the predefined SIGBUS signal handlers installed automatically when your program runs

To use these facilities in any of the three forms you must specify the misalignment handling library, `libmisalign.a`, to the linker. To do this you can simply include `-lmisalign` on the `cc` or `ld` command line. If you use the `ld` command, be sure to specify the misalignment handling library before specifying `libc`, as with `-lc`.

If your program does not care to handle SIGBUS signals other than those representing misaligned access faults, you can simply specify `-u misalign.auto-install` to the linker before specifying the misalignment handling library. With such a specification, a SIGBUS handler that catches SIGBUS signals and repairs misaligned access faults will be installed automatically when your program runs. You do not need to modify your original program to use misalignment handling in this way.

If your program does not care to handle SIGBUS signals other than those representing misaligned access faults but does want to establish signal handlers explicitly, you can use the predefined signal handlers `misalignment_sigbus_handler_ocs1` and `misalignment_sigbus_handler_abi1`. These signal handlers catch SIGBUS signals and repair misaligned access faults in the same way; they differ only in the target environments for which they are appropriate. If you establish the signal handler in a COFF environment (such as `m88kbcs`, `m88kocs`, or `m88kdguxcoff`), use `misalignment_sigbus_handler_ocs1`. If you establish the signal handler in an ELF environment (such as `m88kdguxelf`), use `misalignment_sigbus_handler_abi1`.

If a predefined signal handler catches a SIGBUS signal that does not represent a misaligned access fault, or if it cannot repair a misaligned access fault for any reason, it aborts the program by sending a SIGBUS signal to its own process using the `kill()` function. This same failure response occurs when `-u misalign.auto-install` is used, because one of the predefined handlers is installed automatically in that case.

If the failure treatment of the predefined handlers is inappropriate for your program, or if you want to handle SIGBUS signals other than those representing misaligned access faults, you can use the functions `repair_misalignment_ocs1` and

repair_misalignment_abi1. These functions attempt to repair misaligned access faults and indicate their success or failure. You can call one of these functions from your program's SIGBUS signal handler, then take other appropriate action in the case of failure. The two functions act the same; they differ only in their argument lists and the target environments for which they are appropriate.
repair_misalignment_ocs1 takes one argument, the second argument received by a signal handler that was established in a COFF environment.
repair_misalignment_abi1 takes two arguments, the second and third arguments received by a signal handler that was established in an ELF environment by a call to sigaction(2) with the SA_SIGINFO flag set.

The repair functions return an integer whose value indicates whether the repair was successful. If the return value is negative, the repair failed; otherwise, it succeeded. Furthermore, if the return value is zero, the site of the misaligned access fault was patched so that future faults will not occur; if the return value is positive, patching was not possible.

The remainder of this description applies to repair of misaligned access faults by any of the three forms described above (automatic installation of predefined handler, explicit installation of predefined handler, or direct use of repair function). The common facilities are referred to collectively as "misalignment handling."

Misalignment handling can not only emulate the faulting memory access but also patch the faulting instruction so that future faults will not occur. Patching can greatly speed up an application that suffers misaligned access faults. Note, however, that patching renders your program's text area less sharable. Pages that contain faulting instructions that are patched become private to your process.

If a faulting instruction appears to be in a delay slot (that is, the instruction appears to follow a flow control instruction with delayed branching selected), it is assumed that the instruction is indeed in a delay slot, and instructions are generated to patch the flow control instruction as well as the faulting instruction. Patching an instruction in a delay slot requires more instructions. If the resulting performance of your program is inadequate due to a large number of misaligned access faults, you may wish to instruct the compiler not to perform delay slot optimization. For gcc, use the -fno-delayed-branch option. For cc, use the -W0,-fno-delayed-branch option. For Green Hills compilers, use the -X307 option.

Three M88000 instructions can incur misaligned access faults: ld, st, and xmem. Misalignment handling handles all three instructions, but cannot maintain atomicity in most cases because the access must be done in pieces. The loss of atomicity is generally not important except for xmem, which is not typically generated by compilers.

You can control the behavior of misalignment handling by including an options file among the object files presented to the linker. The file misalign-options.c is provided as a prototype from which you can create your own version. The following table shows what behaviors the options file controls and what the defaults are when no options file is present. See the commentary in the prototype options file for complete information.

| **Behavior** | **Default** |
|---|---|
| Whether to patch | yes |
| Whether to patch in delay slots | yes |
| What registers to treat as scratch | r26 through r29 |
| How much bss area to preallocate | none |
| How to abort on failure | send SIGBUS signal to self |

## EXAMPLE

The following cc command compiles a program for debugging with mxdb(1) and links it with misalignment handling.

```
cc -g -mlegend -o example example.c -u misalign.auto-install -lmisalig
```

Mxdb can be used to determine where misaligned accesses occur. The following shell script produces a backtrace of the stack on each misaligned access. It then continues the program which allows misalignment handling to fix the access.

```
mxdb example <<EOF

  ,, Do a walkback on each SIGBUS.

  signal, catch bus, \
          action  { \
                  new-line; \
                  write MISALIGNED ACCESS; \
                  walkback, arg, locals; \
                  continue \
                  }

  continue        ,, Start the program.
  bye             ,, Quit when it is done.

EOF
```

The backslashes shown above are necessary.

If you use the above approach with patching enabled (the default), you should note two things. First, warnings of the following form may result but can be ignored:

```
Warning: instruction 00000000 not yet supported, ignored
```

Second, misaligned access faults can occur in the patch code sequences themselves. You need not worry about these faults, because in these cases the original faulting instruction is "repatched."

## SEE ALSO

sde(5), sigaction(2), kill(2), mxdb(1),
*Using the Multi-Extensible Debugger (Mxdb for DG/UX and 386/ix Systems),*
*88open Binary Compatibility Standard,*
*88open Object Compatibility Standard,*
*MC88100 RISC Microprocessor User's Manual.*

## NAME
`nl_types` – native language data types

## SYNOPSIS
`#include <nl_types.h>`

## DESCRIPTION
This header file contains the following definitions that relate to the X/open-sytle message facility:

| | |
|---|---|
| `nl_catd` | used by the message catalog functions `catopen`, `catgets` and `catclose` to identify a catalogue |
| `nl_item` | used by `nl_langinfo` to identify items of langinfo data. Values for objects of type `nl_item` are defined in `langinfo.h`. |
| `NL_SETD` | used by `gencat` when no `$set` directive is specified in a message text source file. This constant can be used in subsequent calls to `catgets` as the value of the set identifier parameter. |
| `NL_MGSMAX` | maximum number of messages per set |
| `NL_SETMAX` | maximum number of sets per catalogue. |
| `NL_TEXTMAX` | maximum size of a message in bytes. " 41" counts as one byte; a multibyte character counts as more than one byte. |
| `DEF_NLSPATH` | the default search path for locating catalogues. |

## SEE ALSO
`gencat`(1M), `catgets`(3C), `catopen`(3C), `nl_langinfo`(3C), `langinfo`(5).
`mkmsgs`(1), `gettxt`(3C) — AT&T-style message facilty.

## NAME

`printcap` – printer capability data base

## SYNOPSIS

`/etc/printcap`

## DESCRIPTION

`Printcap` is a simplified version of the `termcap`(5) data base used to describe line printers. The spooling system accesses the `printcap` file every time it is used, allowing dynamic addition and deletion of printers. Each entry in the data base is used to describe one printer. This data base may not be substituted for, as is possible for `termcap`, because it may allow accounting to be bypassed.

The default printer is normally `lp`, though the environment variable PRINTER may be used to override this. Each spooling utility supports an option, `-P`*printer*, to allow explicit naming of a destination printer.

### Capabilities

Refer to `termcap`(5) for a description of the file layout.

| Name | Type | Default | Description |
|---|---|---|---|
| af | str | NULL | name of accounting file |
| br | num | none | if lp is a tty, set baud rate (ioctl call) |
| cf | str | NULL | cifplot data filter |
| df | str | NULL | tex data filter (DVI format) |
| fc | num | 0 | if lp is a tty, clear flag bits (sgtty.h) |
| ff | str | "\f" | string to send for a form feed |
| fo | bool | false | print a form feed when device is opened |
| fs | num | 0 | like "fc" but set bits |
| gf | str | NULL | graph data filter (plot (3X) format) |
| hl | bool | false | print the burst header page last |
| ic | bool | false | driver supports nonstandard ioctl to indent printout |
| if | str | NULL | name of text filter which does accounting |
| lf | str | "/dev/console" | error logging file name |
| lo | str | "lock" | name of lock file |
| lp | str | "/dev/lp" | device name to open for output |
| mx | num | 1000 | maximum file size (in BUFSIZ blocks), 0 = unlimited |
| nd | str | NULL | next directory for list of queues (unimplemented) |
| nf | str | NULL | ditroff data filter (device independent troff) |
| of | str | NULL | name of output filtering program |
| pc | num | 200 | price per foot or page in hundredths of cents |
| pl | num | 66 | page length (in lines) |
| pw | num | 132 | page width (in characters) |
| px | num | 0 | page width in pixels (horizontal) |
| py | num | 0 | page length in pixels (vertical) |
| rf | str | NULL | filter for printing FORTRAN style text files |
| rg | str | NULL | restricted group; only group members can access |
| rm | str | NULL | machine name for remote printer |
| rp | str | "lp" | remote printer name argument |
| rs | bool | false | restrict remote users to those with local accounts |
| rw | bool | false | open the printer device for reading and writing |
| sb | bool | false | short banner (one line only) |
| sc | bool | false | suppress multiple copies |
| sd | str | "/usr/spool/lpd" | spool directory |
| sf | bool | false | suppress form feeds |

| | | | |
|---|---|---|---|
| sh | bool | false | suppress printing of burst page header |
| st | str | "status" | status file name |
| tf | str | NULL | troff data filter (cat phototypesetter) |
| tr | str | NULL | trailer string to print when queue empties |
| vf | str | NULL | raster image filter |
| xc | num | 0 | if lp is a tty, clear local mode bits [tty(4)] |
| xs | num | 0 | like "xc" but set bits |

If the local line printer driver supports indentation, the server must understand how to invoke it.

### Filters

The `lpd`(1M) server creates a pipeline of *filters* to process files for various printer types. The filters selected depend on the flags passed to `lpr`(1). The pipeline set up is:

| | | |
|---|---|---|
| -p | pr \| if | regular text + *pr*(1) |
| none | if | regular text |

The `if` filter is invoked with arguments:

*if* [ `-c` ] -wwidth `-l`length `-i`indent `-n` login `-h` host acct-file

The `-c` flag is passed only if the `-l` flag (pass control characters literally) is specified to *lpr*. *Width* and *length* specify the page width and length (from `pw` and `pl` respectively) in characters. The `-n` and `-h` parameters specify the login name and host name of the owner of the job respectively. *Acct-file* is passed from the `af` *printcap* entry.

If no `if` is specified, `of` is used instead, with the distinction that `of` is opened only once, while `if` is opened for every individual job. Thus, `if` is better suited to performing accounting. The `of` is only given the *width* and *length* flags.

All other filters are called as:

*filter* -x*width* -y*length* `-n` *login* `-h` *host acct-file*

where *width* and *length* are represented in pixels, specified by the `px` and `py` entries respectively.

All filters take *stdin* as the file, *stdout* as the printer, may log either to *stderr* or using `syslog`(3), and must not ignore SIGINT.

### Logging

Error messages generated by the line printer programs themselves (that is, the *lp** programs) are logged by `syslog`(3) using the *LPR* facility. Messages printed on *stderr* of one of the filters are sent to the corresponding `lf` file. The filters may, of course, use *syslog* themselves.

Error messages sent to the console have a carriage return and a line feed appended to them, rather than just a line feed.

## SEE ALSO

`lpc`(1M), `lpd`(1M), `lpq`(1), `lpr`(1), `lprm`(1), `termcap`(5).

## NAME

prof – profile within a function

## SYNOPSIS

```
#define MARK
#include <prof.h>

void MARK (name);
```

## DESCRIPTION

MARK introduces a mark called *name* that is treated the same as a function entry point. Execution of the mark adds to a counter for that mark, and program-counter time spent is accounted to the immediately preceding mark or to the function if there are no preceding marks within the active function.

*name* may be any combination of letters, numbers, or underscores. Each *name* in a single compilation must be unique, but may be the same as any ordinary program symbol.

For marks to be effective, the symbol MARK must be defined before the header file prof.h is included, either by a preprocessor directive as in the synopsis, or by a command line argument:

```
cc -p -DMARK foo.c
```

If MARK is not defined, the MARK(*name*) statements may be left in the source files containing them and are ignored. prof -g must be used to get information on all labels.

## EXAMPLE

In this example, marks can be used to determine how much time is spent in each loop. Unless this example is compiled with MARK defined on the command line, the marks are ignored.

```
#include <prof.h>
foo( )
{
        int i, j;
        . . .
        MARK(loop1);
        for (i = 0; i < 2000; i++) {
                . . .
        }
        MARK(loop2);
        for (j = 0; j < 2000; j++) {
                . . .
        }
}
```

## SEE ALSO

prof(1), profil(2), monitor(3C).

## NAME

`regexp`: `compile`, `step`, `advance` – regular expression compile and match routines

## SYNOPSIS

```
#define INIT declarations
#define GETC(void) getc code
#define PEEKC(void) peekc code
#define UNGETC(void) ungetc code
#define RETURN(ptr) return code
#define ERROR(val) error code

#include <regexp.h>

char *compile(char *instring, char *expbuf, char *endbuf, int eof);

int step(char *string, char *expbuf);

int advance(char *string, char *expbuf);

extern char *loc1, *loc2, *locs;
```

## DESCRIPTION

These functions are general purpose regular expression matching routines to be used in programs that perform regular expression matching. These functions are defined by the `<regexp.h>` header file.

The functions `step` and `advance` do pattern matching given a character string and a compiled regular expression as input.

The function `compile` takes as input a regular expression as defined below and produces a compiled expression that can be used with `step` or `advance`.

A regular expression specifies a set of character strings. A member of this set of strings is said to be matched by the regular expression. Some characters have special meaning when used in a regular expression; other characters stand for themselves.

The regular expressions available for use with the regexp functions are constructed as follows:

| *Expression* | *Meaning* |
|---|---|
| *c* | the character *c* where *c* is not a special character. |
| \*c* | the character *c* where *c* is any character, except a digit in the range `1-9`. |
| ^ | the beginning of the line being compared. |
| $ | the end of the line being compared. |
| . | any character in the input. |
| [*s*] | any character in the set *s*, where *s* is a sequence of characters and/or a range of characters, *e.g.*, [*c*–*c*]. |
| [^*s*] | any character not in the set *s*, where *s* is defined as above. |
| *r** | zero or more successive occurrences of the regular expression *r*. The longest leftmost match is chosen. |
| *rx* | the occurrence of regular expression *r* followed by the occurrence of regular expression *x*. (Concatenation) |
| *r*\\{*m*,*n*\\} | any number of *m* through *n* successive occurrences of the regular expression *r*. The regular expression *r*\\{*m*\\} matches exactly *m* occurrences; |

*r*\{*m*,\} matches at least *m* occurrences.

\(*r*\) the regular expression *r*. When *n* (where *n* is a number greater than zero) appears in a constructed regular expression, it stands for the regular expression *x* where *x* is the $n^{th}$ regular expression enclosed in \( and \) that appeared earlier in the constructed regular expression. For example, \(*r*\)*x*\(*y*\)*z*\2 is the concatenation of regular expressions *rxyzy*.

Characters that have special meaning except when they appear within square brackets ([]) or are preceded by \ are: ., *, [, \. Other special characters, such as $ have special meaning in more restricted contexts.

The character ^ at the beginning of an expression permits a successful match only immediately after a newline, and the character $ at the end of an expression requires a trailing newline.

Two characters have special meaning only when used within square brackets. The character - denotes a range, [*c*-*c*], unless it is just after the open bracket or before the closing bracket, [-*c*] or [*c*-] in which case it has no special meaning. When used within brackets, the character ^ has the meaning *complement of* if it immediately follows the open bracket (example: [^*c*]); elsewhere between brackets (example: [*c*^]) it stands for the ordinary character ^.

The special meaning of the \ operator can be escaped only by preceding it with another \, *e.g.* \\.

Programs must have the following five macros declared before the `#include <regexp.h>` statement. These macros are used by the `compile` routine. The macros GETC, PEEKC, and UNGETC operate on the regular expression given as input to `compile`. *NOTE:* If any of the macros below consist of more than 1 statement, then they should be surrounded with curly braces ({, }) or unexpected results will occur.

GETC This macro returns the value of the next character (byte) in the regular expression pattern. Successive calls to GETC should return successive characters of the regular expression.

PEEKC This macro returns the next character (byte) in the regular expression. Immediately successive calls to PEEKC should return the same character, which should also be the next character returned by GETC.

UNGETC This macro causes the argument c to be returned by the next call to GETC and PEEKC. No more than one character of pushback is ever needed and this character is guaranteed to be the last character read by GETC. The return value of the macro UNGETC(c) is always ignored.

RETURN(*ptr*) This macro is used on normal exit of the `compile` routine. The value of the argument *ptr* is a pointer to the character after the last character of the compiled regular expression. This is useful to programs which have memory allocation to manage.

ERROR(*val*) This macro is the abnormal return from the `compile` routine. The argument *val* is an error number [see ERRORS below for meanings]. This call should never return.

The syntax of the `compile` routine is as follows:

```
    compile(instring, expbuf, endbuf, eof)
```

The first parameter, *instring*, is never used explicitly by the `compile` routine but is useful for programs that pass down different pointers to input characters. It is sometimes used in the `INIT` declaration (see below). Programs which call functions to input characters or have characters in an external array can pass down a value of `(char *)0` for this parameter.

The next parameter, *expbuf*, is a character pointer. It points to the place where the compiled regular expression will be placed.

The parameter *endbuf* is one more than the highest address where the compiled regular expression may be placed. If the compiled expression cannot fit in `(endbuf-expbuf)` bytes, a call to `ERROR(50)` is made.

The parameter *eof* is the character which marks the end of the regular expression. This character is usually a `/`.

Each program that includes the `<regexp.h>` header file must have a `#define` statement for `INIT`. It is used for dependent declarations and initializations. Most often it is used to set a register variable to point to the beginning of the regular expression so that this register variable can be used in the declarations for `GETC`, `PEEKC`, and `UNGETC`. Otherwise it can be used to declare external variables that might be used by `GETC`, `PEEKC` and `UNGETC`. [See EXAMPLE below.]

The first parameter to the `step` and `advance` functions is a pointer to a string of characters to be checked for a match. This string should be null terminated.

The second parameter, *expbuf*, is the compiled regular expression which was obtained by a call to the function `compile`.

The function `step` returns non-zero if some substring of *string* matches the regular expression in *expbuf* and zero if there is no match. If there is a match, two external character pointers are set as a side effect to the call to `step`. The variable `loc1` points to the first character that matched the regular expression; the variable `loc2` points to the character after the last character that matches the regular expression. Thus if the regular expression matches the entire input string, `loc1` will point to the first character of *string* and `loc2` will point to the null at the end of *string*.

The function `advance` returns non-zero if the initial substring of *string* matches the regular expression in *expbuf*. If there is a match, an external character pointer, `loc2`, is set as a side effect. The variable `loc2` points to the next character in *string* after the last character that matched.

When `advance` encounters a `*` or `\{ \}` sequence in the regular expression, it will advance its pointer to the string to be matched as far as possible and will recursively call itself trying to match the rest of the string to the rest of the regular expression. As long as there is no match, `advance` will back up along the string until it finds a match or reaches the point in the string that initially matched the `*` or `\{ \}`. It is sometimes desirable to stop this backing up before the initial point in the string is reached. If the external character pointer `locs` is equal to the point in the string at sometime during the backing up process, `advance` will break out of the loop that backs up and will return zero.

The external variables `circf`, `sed`, and `nbra` are reserved.

## DIAGNOSTICS

The function `compile` uses the macro `RETURN` on success and the macro `ERROR` on failure (see above). The functions `step` and `advance` return non-zero on a successful match and zero if there is no match. Errors are:

| | |
|---|---|
| 11 | range endpoint too large. |
| 16 | bad number. |
| 25 | \ *digit* out of range. |
| 36 | illegal or missing delimiter. |
| 41 | no remembered search string. |
| 42 | `\( \)` imbalance. |
| 43 | too many `\(`. |
| 44 | more than 2 numbers given in `\{ \}`. |
| 45 | `}` expected after `\`. |
| 46 | first number exceeds second in `\{ \}`. |
| 49 | `[ ]` imbalance. |
| 50 | regular expression overflow. |

## EXAMPLE

The following is an example of how the regular expression macros and calls might be defined by an application program:

```
#define INIT         register char *sp = instring;
#define GETC        (*sp++)
#define PEEKC       (*sp)
#define UNGETC(c)    (--sp)
#define RETURN(*c)    return;
#define ERROR(c)     regerr

#include <regexp.h>

  . . .
     (void) compile(*argv, expbuf, &expbuf[ESIZE],'\0');
  . . .
     if (step(linebuf, expbuf))
                       succeed;
```

## SEE ALSO

regcmp(1), regcmp(3X).

## NAME

sde – software development environment

## DESCRIPTION

A *software development environment* (SDE) is a set of tools, libraries and system definitions that are specifically designed to work together to build an application that has certain qualities.

The environments provided in the DG/UX 5.4 release are:

| | |
|---|---|
| `m88kdguxELF` | Used to create ELF objects and executables that make use of full DG/UX 5.4 release features. |
| `m88kocs` | Used for creating COFF objects and executables that can be linked and run on other vendors' 88open OCS- (and BCS-) conforming platforms. |
| `m88kbcs` | Differs from the `m88kocs` because it allows the use of certain features (such as Berkeley signals) and optimizations (such as the macro implementation of `getc`) that are prohibited from the OCS environment. (This is unchanged from the DG/UX 4.3x release.) |
| `m88kdguxcoff` | Used to create COFF objects and executables that make use of DG/UX 4.3x level features. This option is interesting to software developers who have COFF-dependent tools, such as third-party debuggers, that they want to use on the DG/UX 5.4 release. (This is the same as `m88kdgux` on 4.3x.) |
| `m88kdgux` | The default for all past and future revisions. It refers to the largest feature set supported by the DG/UX system. In the DG/UX 5.4 release this is equal to `m88kdguxELF`. |

The following table shows the domain of certain standards across the different environments. "Yes" means the environment conforms to that standard.

| | BCS | OCS | POSIX | SVID/2 | SVID/3 | XPG/3 | ANSI C |
|---|---|---|---|---|---|---|---|
| `m88kdguxelf` | No | No | Yes | No | Yes | Yes | Yes |
| `m88kocs` | Yes | Yes | Yes | Yes | No | No | Yes |
| `m88kbcs` | Yes | No | Yes | Yes | No | No | Yes |
| `m88kdguxcoff` | No | No | Yes | Yes | No | No | Yes |

Support for multiple development environments is handled by the `sde-target`(1) mechanism. It allows you to specify the development environment that is appropriate for your needs, while other users (or you in another context) may be using a different development environment at the same time. You select your environment by setting the environment variable `TARGET_BINARY_INTERFACE` to one of the environment names listed above. The command `sde-target`(1) provides a convenient way to set that variable. (Note that the variable name has changed from `SDE_TARGET` in the DG/UX 4.3x release. The name was changed because additional variables that control the "sde target" in ways other than the binary interface are likely to be introduced in the future. The `sde-target` command will not change, but it might set multiple variables in the future.)

The environment variable set by `sde-target`(1) is used in two contexts. When you invoke a software development tool such as `/bin/cc` or `/bin/ld`, you are actually calling a small program that calls `sde-chooser`(1), which checks the environment variable and invokes the appropriate target-specific tool. Secondly, tools that read libraries, such as `ld`(1), use the `elink`(5) mechanism, which uses the environment variable to find the appropriate system libraries.

The commands, libraries, and other files that support a specific environment are placed in the directory `/usr/sde/<s>`, where <*s*> is the value of the environment variable `TARGET_BINARY_INTERFACE`. If `TARGET_BINARY_INTERFACE` is not set, the default (`m88kdgux`) is used.

Different environments need different header information at compile time. The DG/UX system has one set of include files that are customized by the use of conditional preprocessing under the control of target-specific macro names. The C compiler commands `cc`(1), `gcc`(1), and `ghcc`(1) predefine the following macro names according to the value of `TARGET_BINARY_INTERFACE`. (If you use another C compiler, you will need to do this manually with a `-D` option.)

| sde target | Target Macro Name |
|---|---|
| m88kdguxelf | _DGUX_TARGET |
| m88kocs | _M88KOCS_TARGET |
| m88kbcs | _M88KBCS_TARGET |
| m88kdguxcoff | _DGUXCOFF_TARGET |

The above mechanism using `sde-chooser` and `elinks` was chosen over a more "traditional" method of using the PATH environment variable to find the right tools because many sources that people maintain, such as `make` files and shell scripts, contain fully specified path names. Such references would ignore the path specification and perhaps invoke the wrong tool or read the wrong library.

## SEE ALSO

`sde-target`(1), `sde-chooser`(1), `sdetab`(4), `elink`(5).

## NAME

siginfo – signal generation information

## SYNOPSIS

```
#include <siginfo.h>
```

## DESCRIPTION

If a process is catching a signal, it may request information that tells why the system generated that signal [see sigaction(2)]. If a process is monitoring its children, it may receive information that tells why a child changed state [see waitid(2)]. In either case, the system returns the information in a structure of type siginfo_t, which includes the following information:

```
int si_signo/* signal number */
int si_errno/* error number */
int si_code /* signal code */
```

si_signo contains the system-generated signal number. (For the waitid(2) function, si_signo is always SIGCHLD.)

If si_errno is non-zero, it contains an error number associated with this signal, as defined in errno.h.

si_code contains a code identifying the cause of the signal. If the value of si_code is less than or equal to 0, then the signal was generated by a user process [see kill(2) and sigsend(2)] and the siginfo structure contains the following additional information:

```
pid_t si_pid/* sending process ID */
uid_t si_uid/* sending user ID */
```

Otherwise, si_code contains a signal-specific reason why the signal was generated, as follows:

| Signal | Code | Reason |
|---|---|---|
| SIGILL | ILL_ILLOPC | illegal opcode |
| | ILL_PRVOPC | privileged opcode |
| | ILL_PRVREG | privileged register |
| SIGFPE | FPE_INTDIV | integer divide by zero |
| | FPE_INTOVF | integer overflow |
| | FPE_FLTDIV | floating point divide by zero |
| | FPE_FLTOVF | floating point overflow |
| | FPE_FLTUND | floating point underflow |
| | FPE_FLTRES | floating point inexact result |
| | FPE_FLTINV | invalid floating point operation |
| | FPE_FLTSUB | subscript out of range |
| SIGSEGV | SEGV_MAPERR | address not mapped to object |
| | SEGV_ACCERR | invalid permissions for mapped object |
| SIGBUS | BUS_ADRALN | invalid address alignment |
| SIGTRAP | TRAP_BRKPT | process breakpoint |
| | TRAP_TRACE | process trace trap |
| SIGCHLD | CLD_EXITED | child has exited |
| | CLD_KILLED | child was killed |

| | | |
|---|---|---|
| | CLD_DUMPED | child terminated abnormally |
| | CLD_TRAPPED | traced child has trapped |
| | CLD_STOPPED | child has stopped |
| | CLD_CONTINUED | stopped child had continued |
| SIGPOLL | POLL_IN | data input available |
| | POLL_OUT | output buffers available |
| | POLL_MSG | input message available |
| | POLL_ERR | I/O error |
| | POLL_PRI | high priority input available |
| | POLL_HUP | device disconnected |

In addition, the following signal-dependent information is available for kernel-generated signals:

| Signal | Field | Value |
|---|---|---|
| SIGILL | caddr_t si_addr | address of faulting instruction SIGFPE |
| SIGSEGV | caddr_t si_addr | address of faulting memory reference SIGBUS |
| SIGCHLD | pid_t si_pid | child process ID |
| | int si_status | exit value or signal |
| SIGPOLL | long si_band | band event for POLL_IN, POLL_OUT, or POLL_MSG |

## SEE ALSO

sigaction(2), waitid(2), signal(5).

## NOTES

For SIGCHLD signals, if si_code is equal to CLD_EXITED, then si_status is equal to the exit value of the process; otherwise, it is equal to the signal that caused the process to change state.

## NAME

signal – base signals

## SYNOPSIS

```
#include <signal.h>
```

## DESCRIPTION

A signal is an asynchronous notification of an event. A signal is said to be generated for (or sent to) a process when the event associated with that signal first occurs. Examples of such events include hardware faults, timer expiration and terminal activity, as well as the invocation of the `kill` or `sigsend` system calls. In some circumstances, the same event generates signals for multiple processes. A process may request a detailed notification of the source of the signal and the reason why it was generated [see `siginfo`(5)].

Each process may specify a system action to be taken in response to each signal sent to it, called the signal's disposition. The set of system signal actions for a process is initialized from that of its parent. Once an action is installed for a specific signal, it usually remains installed until another disposition is explicitly requested by a call to either `sigaction`, `signal`, or `sigset`, or until the process execs [see `sigaction`(2) and `signal`(2)]. When a process execs, all signals whose dispositions have been set to catch the signal will be set to `SIG_DFL`. Alternatively, a process may request that the system automatically reset the disposition of a signal to `SIG_DFL` after it has been caught [see `sigaction`(2) and `signal`(2)].

A signal is said to be delivered to a process when the appropriate action for the process and signal is taken. During the time between the generation of a signal and its delivery, the signal is said to be pending [see `sigpending`(2)]. Ordinarily, this interval cannot be detected by an application. However, a signal can be blocked from delivery to a process [see `signal`(2) and `sigprocmask`(2)]. If the action associated with a blocked signal is anything other than to ignore the signal, and if that signal is generated for the process, the signal remains pending until either it is unblocked or the signal's disposition requests that the signal be ignored. If the signal disposition of a blocked signal requests that the signal be ignored, and if that signal is generated for the process, the signal is discarded immediately upon generation.

Each process has a signal mask that defines the set of signals currently blocked from delivery to it [see `sigprocmask`(2)]. The signal mask for a process is initialized from that of its parent.

The determination of which action is taken in response to a signal is made at the time the signal is delivered, allowing for any changes since the time of generation. This determination is independent of the means by which the signal was originally generated.

For a list of the signals supported by DG/UX, see <signal.h>.

`kill`(2), `pause`(2), `sigaction`(2), `sigset`(2), `sigaltstack`(2), `signal`(2), `sigprocmask`(2), `sigsend`(2), `sigsuspend`(2), `wait`(2), `sigsetops`(3C), `siginfo`(5), `ucontext`(5).

## NAME

stat – data returned by stat system call

## SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
```

## DESCRIPTION

The system calls **stat**, **fstat**, **lstat**, and **dg_mstat** return data whose structure is defined by this include file. The encoding of the field *st_mode* is also defined in this file.

```
/*
 * Structure of the result of stat
 */

struct  stat
{
     dev_t           st_dev;
     ino_t           st_ino;
     mode_t          st_mode;
     nlink_t         st_nlink;
     uid_t           st_uid;
     gid_t           st_gid;
     dev_t           st_rdev;
     off_t           st_size;
     time_t          st_atime;
     unsigned long   st_ausec;
     time_t          st_mtime;
     unsigned long   st_musec;
     time_t          st_ctime;
     unsigned long   st_cusec;
     timestruc_t     st_atim;
     timestruc_t     st_mtim;
     timestruc_t     st_ctim;
     long            st_blksize;
     long            st_blocks;
     char            st_fstype[16];
     char            st_pad5[408];
};

#define S_IFMT    0170000  /* type of file */
#define S_IFDIR   0040000  /* directory */
#define S_IFCHR   0020000  /* character special */
#define S_IFBLK   0060000  /* block special */
#define S_IFREG   0100000  /* regular */
#define S_IFLINK  0120000  /* symbolic link */
#define S_IFIFO   0010000  /* fifo */
#define S_IFSOCK  0140000  /* socket special file */
#define S_ISUID   04000    /* set user id on execution */
#define S_ISGID   02000    /* set group id on execution */
#define S_ISVTX   01000    /* save swapped text even after use */
#define S_IREAD   00400    /* read permission, owner */
#define S_IWRITE  00200    /* write permission, owner */
#define S_IEXEC   00100    /* execute/search permission, owner */
```

```
    #define S_ENFMT   02000   /* record locking enforcement flag */
    #define S_IRWXU   00700   /* read, write, execute search
                                 permission, owner */
    #define S_IRUSR   00400   /* read permission, owner */
    #define S_IWUSR   00200   /* write permission, owner */
    #define S_IXUSR   00100   /* execute/search permission, owner */
    #define S_IRWXG   00070   /* read, write, execute/search
                                 permission, group */
    #define S_IRGRP   00040   /* read permission, group */
    #define S_IWGRP   00020   /* write permission, group */
    #define S_IXGRP   00010   /* execute/search permission, group */
    #define S_IRWXO   00007   /* read, write, execute/search
                                 permission, other */
    #define S_IROTH   00004   /* read permission, other */
    #define S_IWOTH   00002   /* write permission, other */
    #define S_IXOTH   00001   /* execute/search permission, other */
```

## FILES

/usr/include/sys/stat.h
/usr/include/sys/types.h

## SEE ALSO

stat(2), types(5).

## NAME

`statfs` – data returned by the statfs system call

## DESCRIPTION

The system call `statfs` takes a parameter that is a pointer to the structure defined by this include file. This structure returns file system device statistics.

```
struct  statfs
{
    short       f_fstyp;
    long        f_bsize;
    long        f_frsize;
    long        f_blocks;
    long        f_bfree;
    long        f_bavail;
    long        f_files;
    long        f_ffree;
    char        f_fname [6];
    char        f_fpack [6];
    long        f_favail;
    long        fs_blocks;
    long        fs_bfree;
    long        fs_bavail;
    long        fs_files;
    long        fs_ffree;
    long        fs_favail;
};
```

The fields of this structure are defined as follows:

| | |
|---|---|
| `f_fstyp` | The type of the file system. |
| `f_bsize` | The file system block size, in bytes. |
| `f_frsize` | The file system fragment size, in bytes. |
| `f_blocks` | The maximum number of blocks that may exist in the control-point directory containing the pathname passed to *statfs*, taking into account the block limits of all CPDs on the path. If the pathname is a CPD, its own block limit is also taken into account. If the pathname is the root of a file system, this field is the maximum that applies to superusers, so it is the same as *fs_blocks*. If the pathname is not a file system root, the maximum applies to both superusers and non-superusers. |
| `f_bfree` | The number of free blocks in the control-point directory containing the pathname passed to *statfs*, taking into account the block limits of all CPDs on the path. If the pathname is a CPD, its own block limit is also taken into account. If the pathname is the root of a file system, this field is the number of blocks that can still be allocated by superusers, so it is the same as *fs_bfree*. If the pathname is not a file system root, the free count applies to both superusers and non-superusers. |
| `f_bavail` | This field is the same as *f_bfree* unless the pathname is the root of a file system. In that case it gives the number of blocks that can still be allocated by non-superusers. |
| `f_files` | The total number of files that may exist in the control-point directory containing the pathname passed to *statfs*, i.e. the number allocated plus |

| | |
|---|---|
| | the number that still may be created, taking into account the file limits of all CPDs on the path. If the pathname is a CPD, its own file limit is also taken into account. If the pathname is the root of a file system, this field is the maximum that applies to superusers, so it is the same as *fs_files*. If the pathname is not a file system root, the maximum applies to both superusers and non-superusers. |
| `f_ffree` | The number of files that still may be created in the control-point directory containing the pathname passed to *statfs*, taking into account the files limits of all CPDs on the path. If the pathname is a CPD, its own file limit is also taken into account. If the pathname is the root of a file system, this field is the number of files that can still be created by superusers, so it is the same as *fs_ffree*. If the pathname is not a file system root, the file count applies to both superusers and non-superusers. |
| `f_fname` | The file system name. This field will be null unless a label has been added to the file system with *labelit*. |
| `f_fpack` | The file system pack name. This field will be null unless a label has been added to the file system with *labelit*. |
| `f_favail` | This field is the same as *f_ffree*. |
| `fs_blocks` | The file system size, in blocks. |
| `fs_bfree` | The total number of free blocks on the file system. |
| `fs_bavail` | The number of free blocks on the file system available to non-superusers. |
| `fs_files` | The total number of files that may exist on the file system, i.e. the number allocated plus the number that still may be created. |
| `fs_ffree` | The number of files that still may be created on the file system. |
| `fs_favail` | The number of files that still may be created on the file system by non-superusers. |

## FILES

`/usr/include/sys/statfs.h`

## SEE ALSO

`statfs(2)`.

## NAME

stdarg – handle variable argument list

## SYNOPSIS

```
#include <stdarg.h>

va_list pvar;

void va_start(va_list pvar, parmN);

type va_arg(va_list pvar, type);

void va_end(va_list pvar);
```

## DESCRIPTION

This set of macros allows portable procedures that accept variable numbers of arguments of variable types to be written. Routines that have variable argument lists [such as `printf`] but do not use *stdarg* are inherently non-portable, as different machines use different argument-passing conventions.

`va_list` is a type defined for the variable used to traverse the list.

The `va_start()` macro is invoked before any access to the unnamed arguments and initializes `pvar` for subsequent use by `va_arg()` and `va_end()`. The parameter *parmN* is the identifier of the rightmost parameter in the variable parameter list in the function definition (the one just before the `, ...`). If this parameter is declared with the `register` storage class or with a function or array type, or with a type that is not compatible with the type that results after application of the default argument promotions, the behavior is undefined.

The parameter *parmN* is required under strict ANSI C compilation. In other compilation modes, *parmN* need not be supplied and the second parameter to the `va_start()` macro can be left empty [e.g., `va_start(pvar, );`]. This allows for routines that contain no parameters before the `...` in the variable parameter list.

The `va_arg()` macro expands to an expression that has the type and value of the next argument in the call. The parameter `pvar` should have been previously initialized by `va_start()`. Each invocation of `va_arg()` modifies `pvar` so that the values of successive arguments are returned in turn. The parameter *type* is the type name of the next argument to be returned. The type name must be specified in such a way so that the type of a pointer to an object that has the specified type can be obtained simply by postfixing a `*` to *type*. If there is no actual next argument, or if *type* is not compatible with the type of the actual next argument (as promoted according to the default argument promotions), the behavior is undefined.

The `va_end()` macro is used to clean up.

Multiple traversals, each bracketed by `va_start` and `va_end`, are possible.

## EXAMPLE

This example gathers into an array a list of arguments that are pointers to strings (but not more than `MAXARGS` arguments) with function `f1`, then passes the array as a single argument to function `f2`. The number of pointers is specified by the first argument to `f1`.

```
#include <stdarg.h>
#define MAXARGS   31

void f1(int n_ptrs, ...)
{
      va_list ap;
```

```
                    char *array[MAXARGS];
                    int ptr_no = 0;

                    if (n_ptrs > MAXARGS)
                         n_ptrs = MAXARGS;
                    va_start(ap, n_ptrs);
                    while (ptr_no < n_ptrs)
                         array[ptr_no++] = va_arg(ap, char*);
                    va_end(ap);
                    f2(n_ptrs, array);
              }
```

Each call to `f1` shall have visible the definition of the function or a declaration such as

```
      void f1(int, ...)
```

## SEE ALSO

`vprintf`(3S), `varargs`(5).

## NOTES

It is up to the calling routine to specify in some manner how many arguments there are, since it is not always possible to determine the number of arguments from the stack frame. For example, `execl` is passed a zero pointer to signal the end of the list. `printf` can tell how many arguments there are by the format. It is non-portable to specify a second argument of `char`, `short`, or `float` to `va_arg`, because arguments seen by the called function are not `char`, `short`, or `float`. C converts `char` and `short` arguments to `int` and converts `float` arguments to `double` before passing them to a function.

## NAME

`syslog.conf` – configuration file for syslogd system log server

## SYNOPSIS

`/etc/syslog.conf`

## DESCRIPTION

The file `/etc/syslog.conf` contains information used by the system log server (daemon), `syslogd`(1M), to forward a system message to appropriate log files and/or users.

A configuration entry is composed of two TAB-separated fields:

*selector* *action*

The *selector* field contains a semicolon-separated list of priority specifications of the form:

*facility.level*[*;facility.level*]

where *facility* is a system facility, or comma-separated list of facilities, and *level* is an indication of the severity of the condition being logged. Recognized values for *facility* include:

| | |
|---|---|
| `user` | Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file. |
| `kern` | Messages generated by the kernel. |
| `mail` | Reserved for the `mail` system. |
| `daemon` | System servers, such as `ftpd`(1M). |
| `auth` | Reserved for the `auth` system; it does not currently use the `syslog` mechanism. |
| `lpr` | Messages generated by the `lpr/lpd` line printer spooling system. |
| `news` | Reserved for the USENET network news system. |
| `uucp` | Reserved for the UUCP system; it does not currently use the `syslog` mechanism. |
| `cron` | Reserved for the `cron` system; it does not currently use the `syslog` mechanism. |
| `local0-7` | Reserved for local use. |
| `mark` | For timestamp messages produced internally by `syslogd`. |
| `*` | An asterisk indicates all facilities except for the `mark` facility. |

Recognized values for *level* are (in descending order of severity):

| | |
|---|---|
| `emerg` | For panic conditions that would normally be broadcast to all users. |
| `alert` | For conditions that should be corrected immediately, such as a corrupted system database. |
| `crit` | For warnings about critical conditions, such as hard device errors. |
| `err` | For other errors. |
| `warning` | For warning messages. |
| `notice` | For conditions that are not error conditions, but may require special handling. |

| | |
|---|---|
| `info` | Informational messages. |
| `debug` | For messages that are normally used only when debugging a program. |
| `none` | Do not send messages from the indicated *facility* to the selected file. For example, a *selector* of `*.debug;mail.none` will send all messages *except* mail messages to the selected file. |

The *action* field indicates where to forward the message. Values for this field can have one of four forms:

- A filename, beginning with a leading slash, which indicates that messages specified by the *selector* are to be written to the specified file. The file will be opened in append mode.
- The name of a remote host, prefixed with an @, as with: @*server*, which indicates that messages specified by the *selector* are to be forwarded to the `syslogd` on the named host.
- A comma-separated list of usernames, which indicates that messages specified by the *selector* are to be written to the named users if they are logged in.
- An asterisk, which indicates that messages specified by the *selector* are to be written to all logged-in users.

Blank lines are ignored. Lines for which the first nonwhite character is a '#' are treated as comments.

## EXAMPLE

With the following configuration file:

```
*.notice;mail.info      /usr/adm/notice
*.crit                  /usr/adm/critical
kern,mark.debug         /dev/console
kern.err                @server
*.emerg                 *
*.alert                 root,operator
*.alert;auth.warning    /usr/adm/auth
```

`syslogd` will log all mail system messages except `debug` messages and all `notice` (or higher) messages into a file named `/usr/adm/notice`. It logs all critical messages into `/usr/adm/critical`, and all kernel messages and 20-minute marks onto the system console.

Kernel messages of `err` (error) severity or higher are forwarded to the machine named *server*. Emergency messages are forwarded to all users. The users root and operator are informed of any `alert` messages. All messages from the authorization system of `warning` level or higher are logged in the file `/usr/adm/auth`.

## SEE ALSO

`logger(1)`, `syslogd(1M)`, `syslog(3C)`.

## NAME

tar – tape archive file format

## DESCRIPTION

tar (the tape archive command) dumps several files into one, in a medium suitable for transportation.

A "tar tape" or file is a series of blocks. Each block is of size TBLOCK. A file on the tape is represented by a header block which describes the file, followed by zero or more blocks which give the contents of the file. At the end of the tape are two blocks filled with binary zeros, as an end-of-file indicator.

The blocks are grouped for physical I/O operations. Each group of *n* blocks (where *n* is set by the b keyletter on the tar(1) command line — default is 20 blocks) is written with a single system call; on nine-track tapes, the result of this write is a single tape record. The last group is always written at the full size, so blocks after the two zero blocks contain random data. On reading, the specified or default group size is used for the first read, but if that read returns less than a full tape block, the reduced block size is used for further reads.

The header block looks like:

```
#define TBLOCK    512
#define NAMSIZ    100

union hblock {
      char dummy[TBLOCK];
      struct header {
            char name[NAMSIZ];
            char mode[8];
            char uid[8];
            char gid[8];
            char size[12];
            char mtime[12];
            char chksum[8];
            char linkflag;
            char linkname[NAMSIZ];
      } dbuf;
};
```

*Name* is a null-terminated string. The other fields are zero-filled octal numbers in ASCII. Each field (of width w) contains w-2 digits, a space, and a null, except *size* and *mtime*, which do not contain the trailing null and *chksum* which has a null followed by a space. *Name* is the name of the file, as specified on the *tar* command line. Files dumped because they were in a directory which was named in the command line have the directory name as prefix and */filename* as suffix. *Mode* is the file mode, with the top bit masked off. *Uid* and *gid* are the user and group numbers which own the file. *Size* is the size of the file in bytes. Links and symbolic links are dumped with this field specified as zero. *Mtime* is the modification time of the file at the time it was dumped. *Chksum* is an octal ASCII value which represents the sum of all the bytes in the header block. When calculating the checksum, the *chksum* field is treated as if it were all blanks. *Linkflag* is NULL if the file is "normal" or a special file, ASCII '1' if it is an hard link, and ASCII '2' if it is a symbolic link. The name linked-to, if any, is in *linkname,* with a trailing null. Unused fields of the header are binary zeros (and are included in the checksum).

The first time a given i-node number is dumped, it is dumped as a regular file. The second and subsequent times, it is dumped as a link instead. Upon retrieval, if a link entry is retrieved, but not the file it was linked to, an error message is printed and the tape must be manually re-scanned to retrieve the linked-to file.

The encoding of the header is designed to be portable across machines.

## SEE ALSO

`tar`(1).

## NOTE

Names or linknames longer than NAMSIZ produce error reports and cannot be dumped.

## NAME

termcap – terminal capability data base

## DESCRIPTION

Termcap is a data base of terminal descriptions used by the termcap(3X) library. All terminals are described in a file called /etc/termcap. Termcap entries describe, in special code, how basic operations are performed on a terminal. They also describe padding requirements, initialization sequences, and so on. The section entitled "Preparing a Termcap Description" that appears later explains how to build a termcap source description.

Entries in Termcap consist of a number of ':'-separated fields. The first line names the terminal, and the remaining lines describe its capabilities.

### Terminal Names

The first line of for each terminal description gives the names that are known for the terminal, separated by vertical bar (|) characters. The first name is always two characters long for compatibility with older systems which store the terminal type in a 16-bit word in a system-wide data base. The second name is the most common abbreviation for the terminal, the last name should be a long name fully identifying the terminal, and all others are understood as synonyms for the terminal name. All names but the first and last should be in lower case and contain no blanks; the last name may well contain upper case letters and blanks for readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions. First, the vendor and model of the terminal should be specified in the root name, for example, hp2621. This name should not contain hyphens. Terminal modes or user preferences should be indicated by appending a hyphen and an indicator of the mode. Therefore, a vt100 in 132-column mode would be vt100-w. The following suffixes should be used where possible:

| Suffix | Meaning | Example |
|---|---|---|
| -w | Wide mode (more than 80 columns) | vt100-w |
| -am | With automatic margins (usually default) | vt100-am |
| -nam | Without automatic margins | vt100-nam |
| -*n* | Number of lines on the screen | aaa-60 |
| -na | No arrow keys (leave them in local mode) | concept100-na |
| -*n*p | Number of pages of memory | concept100-4p |
| -rv | Reverse video | concept100-rv |

### Terminal Capabilties

Lines after the first line of a terminal description describe the terminal's capabilities. Capabilities in termcap are of three general types: Boolean capabilities, which indicate a terminal's particular features; numeric capabilities, which give the size of the display or other attributes; and string capabilities, which give character sequences that can be used to perform particular terminal operations.

The table below lists termcap capabilities alphabetically by name. The second field of the table indicates capability type. The characters in the Notes field in the table have the following meanings (more than one may apply to a capability):

| | |
|---|---|
| N | indicates numeric parameter(s) |
| P | indicates that padding may be specified |
| * | indicates that padding may be based on the number of lines affected |
| o | indicates that the capability is obsolete |

"Obsolete" capabilities have no `terminfo`(4) equivalents; either they were considered useless, or they have been subsumed by other capabilities. New software should not rely on them at all. The last field in the table gives a short description of the terminal capability.

| Name | Type | Notes | Description |
|---|---|---|---|
| ae | str | (P) | End alternate character set mode |
| AL | str | (NP*) | Add *n* new blank lines |
| al | str | (P*) | Add one new blank line |
| am | bool | | Terminal has automatic margins |
| as | str | (P) | Start alternate character set mode |
| bc | str | (o) | Backspace if not ^H |
| bl | str | (P) | Audible signal (bell) |
| bs | bool | (o) | Terminal can backspace with ^H |
| bt | str | (P) | Back tab |
| bw | bool | | `le` (backspace) wraps from column 0 to last column |
| CC | str | | Terminal settable command character in prototype |
| cd | str | (P*) | Clear to end of display |
| ce | str | (P) | Clear to end of line |
| ch | str | (NP) | Set cursor column (horizontal position) |
| cl | str | (P*) | Clear screen and home cursor |
| CM | str | (NP) | Memory-relative cursor addressing (motion) |
| cm | str | (NP) | Screen-relative cursor addressing (motion) |
| co | num | | Number of columns in a line |
| cr | str | (P) | Carriage return |
| cs | str | (NP) | Change scrolling region (VT100) |
| ct | str | (P) | Clear all tab stops |
| cv | str | (NP) | Set cursor row (vertical position) |
| da | bool | | Display may be retained above screen |
| dB | num | (o) | Milliseconds of `bs` delay needed (default 0) |
| db | bool | | Display may be retained below screen |
| DC | str | (NP*) | Delete *n* characters |
| dC | num | (o) | Milliseconds of `cr` delay needed (default 0) |
| dc | str | (P*) | Delete one character |
| dF | num | (o) | Milliseconds of `ff` delay needed (default 0) |
| DL | str | (NP*) | Delete *n* lines |
| dl | str | (P*) | Delete one line |
| dm | str | | Enter delete mode |
| dN | num | (o) | Milliseconds of `nl` delay needed (default 0) |
| DO | str | (NP*) | Move cursor down *n* lines |
| do | str | | Move cursor down one line |
| ds | str | | Disable status line |
| dT | num | (o) | Milliseconds of horizontal tab delay needed (default 0) |
| dV | num | (o) | Milliseconds of vertical tab delay needed (default 0) |
| ec | str | (NP) | Erase *n* characters |
| ed | str | | End delete mode |
| ei | str | | End insert mode |
| eo | bool | | Terminal can erase overstrikes with a blank |
| EP | bool | (o) | Terminal uses even parity |
| es | bool | | Escape sequences can be used on status line |
| ff | str | (P*) | Hardcopy terminal page eject |
| fs | str | | Return from status line |
| gn | bool | | Generic line type (*e.g.* dialup, switch) |

| | | | |
|---|---|---|---|
| `hc` | bool | | Hardcopy terminal |
| `HD` | bool | (o) | Half-duplex |
| `hd` | str | | Move a half-line down (forward 1/2 linefeed) |
| `ho` | str | (P) | Home cursor |
| `hs` | bool | | Terminal has extra "status line" |
| `hu` | str | | Move a half-line up (reverse 1/2 linefeed) |
| `hz` | bool | | Terminal cannot print tildes (Hazeltine) |
| `IC` | str | (NP*) | Insert *n* blank characters |
| `ic` | str | (P*) | Insert one blank character |
| `if` | str | | Name of file containing initialization string |
| `im` | str | | Enter insert mode |
| `in` | bool | | Insert mode distinguishes nulls |
| `ip` | str | (P*) | Insert padding after character inserted |
| `is` | str | | Terminal initialization string |
| `it` | num | | Tabs are initially every *n* positions |
| `K1` | str | | Sent by keypad upper left key |
| `K2` | str | | Sent by keypad upper right key |
| `K3` | str | | Sent by keypad center key |
| `K4` | str | | Sent by keypad lower left key |
| `K5` | str | | Sent by keypad lower right key |
| `k0-k9` | str | | Sent by function keys 0-9 |
| `kA` | str | | Sent by insert-line key |
| `ka` | str | | Sent by clear-all-tabs key |
| `kb` | str | | Sent by backspace key |
| `kC` | str | | Sent by clear-screen or erase key |
| `kD` | str | | Sent by delete-character key |
| `kd` | str | | Sent by down-arrow key |
| `kE` | str | | Sent by clear-to-end-of-line key |
| `ke` | str | | Out of "keypad transmit" mode |
| `kF` | str | | Sent by scroll-forward/down key |
| `kH` | str | | Sent by home-down key |
| `kh` | str | | Sent by home key |
| `kI` | str | | Sent by insert-character or enter-insert-mode key |
| `kL` | str | | Sent by delete-line key |
| `kl` | str | | Sent by left-arrow key |
| `kM` | str | | Sent by insert key while in insert mode |
| `km` | bool | | Terminal has a "meta" key (sets eighth bit) |
| `kN` | str | | Sent by next-page key |
| `kn` | num | (o) | Number of function (`k0-k9`) keys (default 0) |
| `ko` | str | (o) | Termcap entries for other non-function keys |
| `kP` | str | | Sent by previous-page key |
| `kR` | str | | Sent by scroll-backward/up key |
| `kr` | str | | Sent by right-arrow key |
| `kS` | str | | Sent by clear-to-end-of-screen key |
| `ks` | str | | Put terminal in "keypad transmit" mode |
| `kT` | str | | Sent by set-tab key |
| `kt` | str | | Sent by clear-tab key |
| `ku` | str | | Sent by up-arrow key |
| `l0-l9` | str | | Labels on function keys if not "`fn`" |
| `LC` | bool | (o) | Terminal is lowercase only |
| `LE` | str | (NP) | Move cursor left *n* positions |
| `le` | str | (P) | Move cursor left one position |
| `li` | num | | Number of lines on screen or page |

| | | | |
|---|---|---|---|
| ll | str | | Move cursor to last line, first column |
| lm | num | | Lines of memory if > li (0 means varies) |
| ma | str | (o) | Arrow key map |
| mb | str | | Turn on blinking attribute |
| md | str | | Turn on bold (extra bright) attribute |
| me | str | | Turn off all attributes |
| mh | str | | Turn on half-bright (dim) attribute |
| mi | bool | | Safe to move while in insert mode |
| mk | str | | Turn on blank attribute (characters invisible) |
| ml | str | (o) | Turn on memory lock above cursor |
| mm | str | | Turn on "meta mode" (transmit eighth bit) |
| mo | str | | Turn off "meta mode" |
| mp | str | | Turn on protected attribute |
| mr | str | | Turn on reverse-video attibute |
| ms | bool | | Safe to move in standout modes |
| mu | str | (o) | Memory unlock (turn off memory lock) |
| nc | bool | (o) | No correctly-working cr (Datamedia 2500, Hazeltine 2000) |
| nd | str | | Move cursor right one (non-destructive) space |
| NL | bool | (o) | \n is newline, not line feed |
| nl | str | (o) | Newline character if not \n |
| ns | bool | (o) | Terminal is a CRT but doesn't scroll |
| nw | str | (P) | Newline (behaves like cr followed by do) |
| OP | bool | (o) | Terminal uses odd parity |
| os | bool | | Terminal overstrikes |
| pb | num | | Lowest baud rate where delays are required |
| pc | str | | Pad character (default NUL) |
| pf | str | | Turn off printer |
| pO | str | (N) | Turn on printer for *n* bytes |
| po | str | | Turn on printer |
| ps | str | | Print contents of screen |
| pt | bool | (o) | Has hardware tabs (may need to be set with is) |
| rc | str | (P) | Restore cursor to position of last sc |
| rf | str | | Name of file containing reset string |
| RI | str | (NP) | Move cursor right *n* positions |
| rp | str | (NP*) | Repeat character *c n* times |
| rs | str | | Reset terminal completely to sane modes |
| sa | str | (NP) | Define video attributes |
| sc | str | (P) | Save cursor position |
| se | str | | End standout mode |
| SF | str | (NP*) | Scroll forward (up) *n* lines |
| sf | str | (P) | Scroll forward (up) one line |
| sg | num | | Number of garbage chars left by so or se (default 0) |
| so | str | | Begin standout mode |
| SR | str | (NP*) | Scroll backward (down) *n* lines |
| sr | str | (P) | Scroll backward (down) one line |
| st | str | | Set a tab in all rows, current column |
| ta | str | (P) | Tab to next hardware tab stop |
| tc | str | | Entry of similar terminal – must be last entry |
| te | str | | String to end programs that use termcap |
| ti | str | | String to begin programs that use termcap |
| ts | str | (N) | Go to status line, column *n* |
| UC | bool | (o) | Terminal is uppercase only |
| uc | str | | Underscore one character and move past it |

| | | | |
|---|---|---|---|
| ue | str | | End underscore mode |
| ug | num | | Number of garbage chars left by us or ue (default 0) |
| ul | bool | | Underline character overstrikes |
| UP | str | (NP*) | Move cursor up *n* lines |
| up | str | | Move cursor up one line |
| us | str | | Start underscore mode |
| vb | str | | Visible bell (must not move cursor) |
| ve | str | | Make cursor appear normal (undo vs/vi) |
| vi | str | | Make cursor invisible |
| vs | str | | Make cursor very visible |
| vt | num | | Virtual terminal number (not supported on all systems) |
| wi | str | (N) | Set current window |
| ws | num | | Number of columns in status line |
| xb | bool | | Beehive (f1=ESC, f2=^C). |
| xn | bool | | Newline ignored after column 80 (Concept) |
| xo | bool | | Terminal uses XOFF/XON (DC3/DC1) handshaking |
| xr | bool | (o) | Return acts like ce cr nl (Delta Data) |
| xs | bool | | Standout not erased by overwriting (Hewlett-Packard) |
| xt | bool | | Destructive tabs, magic so char (Teleray 1061) |
| xx | bool | (o) | Tektronix 4025 insert-line |

## PREPARING A TERMCAP DESCRIPTION

The most effective way to prepare a terminal description is by imitating the description of a similar terminal in termcap and building up your description gradually, using partial descriptions to check that they are correct.

To easily test a new terminal description, set the environment variable TERMCAP to the absolute pathname of a file containing the description you are working on and programs will look there rather than in /etc/termcap. TERMCAP can also be set to the termcap entry itself to avoid reading the file when starting up a program.

Be aware that a very unusual terminal may expose deficiencies in the ability of the termcap conventions to describe it.

### Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability tc can be given with the name of the similar terminal. This capability must be specified last, and the combined length of the entries must not exceed 1024 characters. The capabilities given before tc override those in the terminal type included by tc. A capability can be canceled by placing *xx*@ to the left of the tc invocation, where *xx* is the capability. For example, the entry

```
hn||2621-nl:ks@:ke@:tc=2621:
```

defines a "2621–nl" that does not have the ks or ke capabilities, and hence does not turn on the function key labels when in visual mode. This is useful for different modes of a terminal, or for different user preferences.

### Parameterized Strings

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with printf(3S)-like escapes %x in it, while other characters are passed through unchanged. The % encodings have the following meanings:

| | |
|---|---|
| %% | output % |
| %d | output value as in printf(%d) |

| | |
|---|---|
| `%2` | output value as in `printf(%2d)` |
| `%3` | output value as in `printf(%3d)` |
| `%.` | output value as in `printf(%c)` |
| `%+`*x* | add *x* to value, then do `%.` |
| `%>`*xy* | if value > *x* then add *y*, no output |
| `%r` | reverse order of two parameters, no output |
| `%i` | increment by one, no output |
| `%n` | exclusive-or all parameters with 0140 (Datamedia 2500), no output |
| `%B` | BCD (16*(value/10)) + (value%10), no output |
| `%D` | Reverse coding (value − 2*(value%16)), no output (Delta Data) |

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent `\E&a12c03Y` padded for 6 milliseconds. Note that the order of the row and column coordinates is reversed here and that the row and column are sent as two-digit integers. Thus its `cm` capability is `cm=6\E&%r%2c%2Y`.

The Microterm ACT-IV needs the current row and column sent simply encoded in binary preceded by a `^T`, `cm=^T%.%.`. Note that terminals that use `%.` need to be able to backspace the cursor (`le`) and to move the cursor up one line on the screen (`up`). This is necessary because it is not always safe to transmit `\n`, `^D`, `^H`, and `\r`, as the system may change or discard them. (Programs using `termcap` must set terminal modes so that tabs are not expanded, so `\t` is safe to send. This turns out to be essential for the Ann Arbor 4080.)

A final example is the Lear Siegler ADM−3a, which offsets row and column by a blank character, thus `cm=\E=%+ %+ `.

## A Sample Entry

The following entry, which describes the Concept−100, is among the more complex entries in the `termcap` file as of this writing. It is provided here to illustrate the form and content of a `termcap` entry, and to provide a point of reference for the text that follows.

```
ca|concept100|c100|concept|c104|concept100-4p|HDS Concept-100:\
  :al=3*\E^R:am:bl=^G:cd=16*\E^C:ce=16\E^U:cl=2*^L:cm=\Ea%+ %+ :\
  :co#80:.cr=9^M:db:dc=16\E^A:dl=3*\E^B:do=^J:ei=\E\200:eo:im=\E^P:in:\
  :ip=16*:is=\EU\Ef\E7\E5\E8\El\ENH\EK\E\200\Eo&\200\Eo\47\E:k1=\E5:\
  :k2=\E6:k3=\E7:kb=^h:kd=\E<:ke=\Ex:kh=\E?:kl=\E>:kr=\E=:ks=\EX:\
  :ku=\E;:le=^H:li#24:mb=\EC:me=\EN\200:mh=\EE:mi:mk=\EH:mp=\EI:\
  :mr=\ED:nd=\E=:pb#9600:rp=0.2*\Er%.%+ :se=\Ed\Ee:sf=^J:so=\EE\ED:\
  :.ta=8\t:te=\Ev    \200\200\200\200\200\200\Ep\r\n:\
  :ti=\EU\Ev  8p\Ep\r:ue=\Eg:ul:up=\E;:us=\EG:\
  :vb=\Ek\200\200\200\200\200\200\200\200\200\200\200\200\200\200\EK:\
  :ve=\Ew:vs=\EW:vt#8:xn:\
  :bs:cr=^M:dC#9:dT#8:nl=^J:ta=^I:pt:
```

Entries may continue onto multiple lines by giving a backslash (\) as the last character of a line, and empty fields may be included for readability (here between the last field on a line and the first field on the next). Comments may be included on lines beginning with pound sign (#).

## How to Describe Terminal Capabilities in a Termcap Entry

All capabilities have two-letter codes. The fact that the Concept has automatic margins (that is, an automatic return and linefeed when the end of a line is reached) is indicated by the Boolean capability `am`. Hence the description of the Concept includes `am` on the second line.

Numeric capabilities are followed by a pound sign (#) and then the value. On the third line of the example above, co, which indicates the number of columns in the display, gives the value "80" for the Concept.

Finally, string-valued capabilities, such as ce (the sequence to clear-to-end-of-line), are given by the two-letter code, an equals sign (=), then a string ending at the next following colon (:). A delay in milliseconds may appear after the = in such a capability, and causes padding characters to be supplied by `tputs`(3X) to provide this delay after the remainder of the string is sent. The delay can be either a number, for example, `20`, or a number followed by an asterisk (*), for example, `3*`. An * indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-line padding required. (In the case of insert-character, the factor is still the number of lines affected; this is always 1 unless the terminal has `in` and the software uses it.) When an * is specified, it is sometimes useful to give a delay containing a decimal point, for example `3.5` to specify a delay per line to tenths of milliseconds. (Only one decimal place is allowed.)

A number of escape sequences are provided in the string-valued capabilities for easy encoding of control characters there. `\E` maps to an ESC character, `^X` maps to a control-*X* for any appropriate *X*, and the sequences **\n** , **\r** , **\t** , **\b** , and `\f` map to linefeed, return, tab, backspace, and formfeed, respectively. Finally, characters may be given as three octal digits after a `\`, and the characters `^` and `\` may be given as `\^` and `\\`. If it is necessary to place a `:` in a capability it must be escaped in octal as `\072`. If it is necessary to place a NUL character in a string capability it must be encoded as `\200`. (The routines that deal with `termcap` use C strings and strip the eighth bit of the output very late, so that a `\200` comes out as a `\000` would.)

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the first `cr` and `ta` in the preceeding example.

## TERMCAP TERMINAL CAPABILITIES

The following subsections describe `termcap` capabilities in detail.

### Basic Capabilities

The number of columns on each line of the display is given by the co numeric capability. If the display is a CRT, then the number of lines on the screen is given by the `li` capability. If the cursor wraps around to the beginning of the next line when it reaches the right margin, then it should have the am capability. If the terminal can clear its screen, the code to do this is given by the `cl` string capability. If the terminal overstrikes (rather than clearing the position when a character is overwritten), it should have the os capability. If the terminal is a printing terminal, with no soft copy unit, give it both `hc` and `os`. (os applies to storage scope terminals, such as the Tektronix 4010 series, as well as to hard copy and APL terminals.) If there is a code to move the cursor to the left edge of the current row, give this as `cr`. (Normally this will be carriage-return, `^M`.) If there is a code to produce an audible signal (bell, beep, for example), give this as `bl`.

If there is a code (such as backspace) to move the cursor one position to the left, that capability should be given as `le`. Similarly, codes to move to the right, up, and down should be given as nd, up, and do, respectively. These local cursor motions should not alter the text they pass over; for example, you would not normally give "nd= " unless the terminal has the os capability, because the space would erase the character moved over.

A very important point here is that the local cursor motions encoded in `termcap` have undefined behavior at the left and top edges of a display. Programs should never attempt to backspace around the left edge, unless bw is given, and never attempt to move the cursor up off the top line using local cursor motions.

In order to scroll text up, a program moves the cursor to the bottom left corner of the screen and sends the `sf` (index) string. To scroll text down, a program moves the cursor to the top left corner of the screen and sends the `sr` (reverse index) string. The strings `sf` and `sr` have undefined behavior when the cursor is not on their respective corners of the screen. Parameterized versions of the scrolling sequences are SF and SR, which have the same semantics as `sf` and `sr` except that they take one parameter and scroll that many lines. They also have undefined behavior except at the appropriate corners of the screen.

The am capability tells whether the cursor sticks at the right edge of the screen when text is output there, but this does not necessarily apply to nd from the last column. Leftward local motion is defined from the left edge only when bw is given; then an `le` from the left edge will move to the right edge of the previous row. This is useful for drawing a box around the edge of the screen, for example. If the terminal has switch-selectable automatic margins, the `termcap` description usually assumes that this feature is on, that is, am. If the terminal has a command that moves to the first column of the next line, that command can be given as nw (newline). It is permissible for this to clear the remainder of the current line, so if the terminal has no correctly-working CR and LF it may still be possible to craft a working nw out of one or both of them.

These capabilities suffice to describe hardcopy and "glass-tty" terminals. Thus the Teletype model 33 is described as

```
T3|tty33|33|tty|Teletype model 33:\
      :bl=^G:co#72:cr=^M:do=^J:hc:os:
```

and the Lear Siegler ADM-3 is described as

```
13|adm3|3|LSI ADM-3:\
      :am:bl=^G:cl=^Z:co#80:cr=^M:do=^J:le=^H:li#24:sf=^J:
```

## Cursor Motions

If the terminal has a fast way to home the cursor (to the very upper left corner of the screen), this can be given as ho. Similarly, a fast way of getting to the lower left-hand corner can be given as `ll`; this may involve going up with up from the home position, but a program should never do this itself (unless `ll` does), because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as cursor address (0,0): to the top left corner of the screen, not of memory. (Therefore, the "\EH" (memory home) sequence on Hewlett-Packard terminals cannot be used for ho.)

To address the cursor (move it to an absolute position), the cm capability is given. cm takes two parameters: the row and column to move the cursor to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory. If the terminal has memory-relative cursor addressing, that can be indicated by an analogous CM boolean capability.)

Row or column absolute cursor addressing can be given as single parameter capabilities ch (horizontal position absolute) and cv (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to cm. If there are parameterized local motions (for example, move *n* positions to the right) these can be given as `DO`, `LE`, `RI`, and `UP` with a single parameter indicating how many positions to move. These are primarily useful if the terminal does not have `cm`, as with the Tektronix 4025.

**Area Clears**

If the terminal can clear from the current cursor position to the end of the line, leaving the cursor where it is, this should be given as `ce`. If the terminal can clear from the current cursor position to the end of the display, this should be given as `cd`. Programs must output `cd` only from the first column of a line. (Therefore, it can be simulated by a request to delete a large number of lines, if a true `cd` is not available.)

**Insert/Delete Line**

If the terminal can open a new blank line before the line containing the cursor, this should be given as `al`; programs must output this only from the first position of a line. The cursor must then appear at the left of the newly blank line. If the terminal can delete the line that the cursor is on, this should be given as `dl`; programs must output this only from the first position on the line to be deleted. Versions of `al` and `dl` which take a single parameter and insert or delete that many lines can be given as `AL` and `DL`. If the terminal has a settable scrolling region (like the VT100), the command to set this can be described with the `cs` capability, which takes two parameters: the top and bottom lines of the scrolling region. The cursor position is undefined after using this command. The program must reset the cursor position using other `termcap` capabilities such as `cm` or `rc`. It is possible to get the effect of insert or delete line using this command — the `sc` and `rc` (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using `sr` or `sf` on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

If the terminal has the ability to define a window as part of memory which all commands affect, it should be given as the parameterized string `wi`. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order.

If the terminal can retain display memory above the screen, then the `da` capability should be given; if display memory can be retained below, then `db` should be given. These indicate that deleting a line or scrolling may bring non-blank lines up from below, or that scrolling back with `sr` may bring down non-blank lines.

**Insert/Delete Character**

There are two basic kinds of intelligent terminals with respect to insert/delete character that can be described using `termcap`. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept-100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated or expanded to two untyped blanks. You can determine the kind of terminal you have by clearing the screen, and then typing text separated by cursor motions. Type `abc`      `def` using local cursor motions (not spaces) between the `abc` and the `def`. Then position the cursor before the `abc` and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the `abc` shifts over to the `def` which then move together around the end of the current line and onto the next as you insert, then you have the second type of terminal and should give the capability `in`, which stands for "insert null". While these are two logically separate attributes (one line versus multi-line insert mode, and special treatment of untyped spaces), we have seen no terminals whose insert mode cannot be described with the single attribute.

`Termcap` can describe both terminals that have an insert mode and terminals that have a sequence to open a blank position on the current line. Give as `im` the sequence to get into insert mode. Give as `ei` the sequence to leave insert mode. Now give as `ic` any sequence that needs to be sent just before each character to be inserted. Most terminals with a true insert mode will not require `ic`; it is mainly intended for terminals that use a sequence to open a screen position. (If your terminal has both, insert mode is usually preferable to `ic`. Do not give both unless the terminal actually requires both to be used in combination.) If post-insert padding is needed, give this as a number of milliseconds in `ip` (a string capability). Any other sequence that may need to be sent after insertion of a single character can also be given in `ip`. The `IC` capability, with one parameter *n*, will repeat the effects of `ic` *n* times.

It is occasionally necessary to move the cursor around while in insert mode to delete characters on the same line (for example, if there is a tab after the insertion position). If your terminal allows motion while in insert mode, you can give the Boolean capability `mi` to speed up inserting in this case. Omitting `mi` will affect only speed. Some terminals (notably Datamedia) must not have `mi` because of the way their insert mode works.

Finally, you can specify `dc` to delete a single character, `DC` with one parameter *n* to delete *n* characters, and delete mode by giving `dm` and `ed` to enter and exit delete mode (which is any mode the terminal needs to be placed into for `dc` to work).

**Highlighting, Underlining, and Visible Bells**

If your terminal has one or more kinds of display attributes, these can be represented in a number of different ways. You should choose one display form as standout mode, representing a good, high-contrast, easy-on-the-eyes format for highlighting error messages and other attention getters. (If you have a choice, reverse video plus half-bright is good, or reverse video alone.) The sequences to enter and exit standout mode are given as `so` and `se`, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces or garbage characters on the screen, as the TVI 912 and Teleray 1061 do, then the numeric capability `sg` should be given to tell how many characters are left.

Codes to begin and end underlining can be given as `us` and `ue`, respectively. If changing the underlining mode leaves blank spaces or garbage characters on the screen, specify `ug`, analagous to `sg`. If the terminal has a code to underline the current character and move the cursor one position to the right, such as the Microterm Mime, this can be given as `uc`.

Other capabilities to enter various highlighting modes include `mb` (blinking), `md` (bold or extra bright), `mh` (dim or half-bright), `mk` (blanking or invisible text), `mp` (protected), `mr` (reverse video), `me` (turn off all attribute modes), `as` (enter alternate character set mode), and `ae` (exit alternate character set mode). Turning on any of these modes singly may or may not turn off other modes.

If there is a sequence to set arbitrary combinations of attributes, this should be given as `sa` (set attributes), taking 9 parameters. Each parameter is either 0 or 1, as the corresponding attribute is on or off. The 9 parameters are, in order: standout, underline, reverse, blink, dim, bold, blank, protect, and alternate character set. Not all modes need be supported by `sa`, only those for which corresponding attribute commands exist.

Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when the cursor is moved to a new line or is addressed. Programs should exit standout mode on such terminals before moving the cursor or sending a newline. On terminals where this is not a problem, the Boolean capability ms should be given to indicate that this overhead is unnecessary.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), this can be given as vb; it must not move the cursor.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to change, for example, a non-blinking underline into an easier-to-find block or blinking underline), give this sequence as vs. If there is a way to make the cursor completely invisible, give that as vi. The capability ve, which undoes the effects of both vs and ve should also be given.

If your terminal correctly displays underlined characters (with no special codes needed) even though it does not overstrike, then you should give the Boolean capability ul. If overstrikes are erasable with a blank, this should be indicated by giving the Boolean capability eo.

**Keypad**

If the terminal has a keypad that transmits codes when the keys are pressed, termcap can represent. Note that it is not possible to handle terminals where the keypad only works in local mode (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, give these sequences as ks and ke. Otherwise the keypad is assumed to always transmit. The codes sent by the left-arrow, right-arrow, up-arrow, down-arrow, and home keys can be given as kl, kr, ku, kd, and kh, respectively. If there are function keys such as f0, f1, ..., f9, the codes they send can be given as k0, k1,. . . , k9. If these keys have labels other than the default f0 through f9, the labels can be given as l0, l1,. . . , l9. The codes transmitted by certain other special keys can be given: kH (home down), kb (backspace), ka (clear all tabs), kt (clear the tab stop in the current column), kC (clear screen or erase), kD (delete character), kL (delete line), kM (exit insert mode), kE (clear to end of line), kS (clear to end of screen), kI (insert character or enter insert mode), kA (insert line), kN (next page), kP (previous page), kF (scroll forward/down), kR (scroll backward/up), and kT (set a tab stop in the current column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, then the other five keys can be given as K1, K2, K3, K4, and K5. These keys are useful when the effects of a 3 by 3 directional pad are needed. The obsolete ko capability formerly used to describe "other" function keys has been completely supplanted by the above capabilities.

The ma entry is also used to indicate arrow keys that send single-character codes. This field is obsolete and redundant with kl, kr, ku, kd, and kh. It consists of groups of two characters. In each group, the first character is what an arrow key sends, and the second character is the corresponding cursor movement from vi(1). These commands are h for kl, j for kd, k for ku, l for kr, and H for kh. For example, the Mime would have ma=^Hh^Kj^Zk^Xl indicating arrow keys left (^H), down (^K), up (^Z), and right (^X). (There is no home key on the Mime.)

**Tabs and Initialization**

If the terminal needs to be in a special mode when running a program that uses *termcap* capabilities, the codes to enter and exit this mode can be given as ti and te. This is needed, for example, on terminals like the Concept with more than one page of memory. If the terminal has only memory-relative cursor addressing, a screen-sized window must be fixed into the display for cursor addressing to work properly. This is also used for the Tektronix 4025, where `ti` sets the command character to be the one used by `termcap`.

Other capabilities include `is`, an initialization string for the terminal, and `if`, the name of a file containing long initialization strings. These strings are expected to set the terminal into modes consistent with the rest of the `termcap` description. They should be printed in the following order: `is`; setting tabs using `ct` and `st`; and finally `if`. A pair of sequences that does a harder reset from a totally unknown state can be analogously given as `rs` and `if`. Commands are normally placed in `rs` and `rf` only if they produce annoying effects on the screen and are usually unnecessary. For example, the command to set the VT100 into 80-column mode would normally be part of `is`, but it causes an annoying glitch of the screen and is not normally needed since the terminal is usually in 80-column mode already.

If the terminal has hardware tabs, the command to advance to the next tab stop can be given as `ta` (usually `^I`). A "backtab" command which moves leftward to the previous tab stop can be given as `bt`. By convention, if the terminal driver modes indicate that tab stops are being expanded by the computer rather than being sent to the terminal, programs should not use `ta` or `bt` even if they are present, since the user may not have the tab stops properly set. If the terminal has hardware tabs that are initially set every *n* positions when the terminal is powered up, then the numeric parameter `it` should be given, showing the number of positions between tab stops. If the terminal has tab stops that can be saved in nonvolatile memory, the `termcap` description can assume that they are properly set.

If there are commands to set and clear tab stops, they can be given as `ct` (clear all tab stops) and `st` (set a tab stop in the current column of every row). If a more complex sequence is needed to set the tabs than can be described by this, the sequence can be placed in `is` or `if`.

### Delays

Certain capabilities control padding in the terminal driver. These are primarily needed by hardcopy terminals. The delays should be embedded as padding information in the `cr`, `sf`, `le`, `ff`, and `ta` capabilities. If the numeric capability `pb` (padding baud rate) is given, these values can be ignored at baud rates below the value of `pb`. The delays can also be given as (obsolete) numeric capabilities instead: `dC`, `dN`, `dB`, `dF`, and `dT`.

### Miscellaneous

If the terminal requires other than a NUL (zero) character as padding, this can be given as `pc`. Only the first character of the `pc` string is used.

If the terminal has commands to save and restore the position of the cursor, give them as `sc` and `rc`.

If the terminal has an extra "status line" that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, then the Boolean capability `hs` should be given. Special strings to go to a position in the status line and to return from the status line can be given as `ts` and `fs`. (`fs` must leave the cursor position in the same place that it was before `ts`. If necessary, the `sc` and `rc` strings can be included in `ts` and `fs` to get this effect.) The capability `ts` takes one parameter, which is the column number of the status line to which the cursor is to be moved. If escape sequences and other special commands such as tab work while in the status line, the flag `es` can be given. A string that turns off the status line (or otherwise erases its contents) should be given as `ds`. The status line is normally assumed to be the same width as the rest of the screen, that is, `co`. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded), then its width in columns can be indicated with the numeric parameter `ws`.

If the terminal can move up or down half a line, this can be indicated with `hu` (half-line up) or `hd` (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as `ff` (usually `^L`).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters), this can be indicated with the parameterized string `rp`. The first parameter is the character to be repeated and the second is the number of times to repeat it.

If the terminal has a settable command character, such as the Tektronix 4025, this can be indicated with `CC`. A prototype command character is chosen which is used in all capabilities. This character is given in the `CC` capability to identify it. The following convention is supported on some UNIX systems: The environment is searched for a `CC` variable, and if found, all occurrences of the prototype character are replaced by the character in the environment variable. This use of the `CC` environment variable is a very bad idea, however, because it conflicts with `make(1)`.

Terminal descriptions that do not represent a specific kind of known terminal, such as *switch*, *dialup*, *patch*, and *network*, should include the `gn` (generic) Boolean capability so that programs can complain that they do not know how to work with that terminal. (This capability does not apply to virtual terminal descriptions for which the escape sequences are known.)

If the terminal uses XOFF/XON (DC3/DC1) handshaking for flow control, give `xo`. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters will not be transmitted.

If the terminal has a "meta key" which acts as a shift key, setting the eighth bit of any character transmitted, then this fact can be indicated with `km`. Otherwise, software will assume that the eighth bit is parity and it will usually be cleared. If strings exist to turn this "meta mode" on and off, they can be given as `mm` and `mo`.

If the terminal has more lines of memory than will fit on the screen at once, the number of lines of memory can be indicated with `lm`. An explicit value of 0 indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

If the terminal is one of those supported by the UNIX system virtual terminal protocol, the terminal number can be given as `vt`.

Media copy strings which control an auxiliary printer connected to the terminal can be given as `ps`: print the contents of the screen; `pf`: turn off the printer; and `po`: turn on the printer. When the printer is on, all text sent to the terminal will be sent to the printer. It is undefined whether the text is also displayed on the terminal screen when the printer is on. A variation `pO` takes one parameter and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. All text, including `pf`, is transparently passed to the printer while `pO` is in effect.

## Glitches and Braindamage

Hazeltine terminals, which do not allow tilde (~) characters to be displayed, should indicate `hz`.

The `nc` capability, now obsolete, formerly indicated Datamedia terminals, which echo **\r \n** for carriage return then ignore a following linefeed.

Terminals that ignore a linefeed immediately after an am wrap, such as the Concept, should indicate xn.

If ce is required to get rid of standout text (instead of merely writing normal text on top of it), xs should be given.

Teleray terminals, where tabs turn all characters moved over to blanks, should indicate xt (destructive tabs). This glitch is also taken to mean that it is not possible to position the cursor on top of a "magic cookie", and that to erase standout mode it is necessary to use delete and insert line.

The Beehive Superbee, which is unable to correctly transmit the ESC or ^C characters, should specify xb, indicating that the "f1" key is used for ESC and "f2" for ^C. (Only certain Superbees have this problem, depending on the ROM.)

You may correct other specific terminal problems by adding more capabilities of the form x*x*.

## FILES

`/etc/termcap` file containing terminal descriptions

## SEE ALSO

make(1) and vi(1) in the *User's Reference for the DG/UX System*.
termcap(3X), curses(3X), printf(3S), term(5), terminfo(4), in the *Programmer's Reference for the DG/UX System*.
captoinfo(1M) and infocmp(1M) in *System Manager's Reference for the DG/UX System*.

## CAVEATS AND BUGS

Note: termcap is made obsolete by terminfo(4). The transition will be relatively painless if capabilities flagged as "obsolete" are avoided.

Lines and columns are now stored by the kernel as well as in the *termcap* entry.

The total length of a single entry (excluding only escaped newlines) may not exceed 1024 characters.

Not all programs support all entries.

## NAME

types – primitive system data types

## SYNOPSIS

```
#include <sys/types.h>
```

## DESCRIPTION

The data types defined in the include file are used in DG/UX system code; some data of these types are accessible to user code:

```
typedef struct { int r[1]; } *physadr;
typedef long    clock_t;
typedef long    daddr_t;
typedef char *      caddr_t;
typedef unsigned char  unchar;
typedef unsigned short  ushort;
typedef unsigned int   uint;
typedef unsigned long  ulong;
typedef unsigned long  ino_t;
typedef int     pid_t;
typedef int     uid_t;
typedef int     gid_t;
typedef ulong   nlink_t;
typedef ulong   mode_t;
typedef short   cnt_t;
typedef long    time_t;
typedef int     label_t[10];
typedef ulong   dev_t;
typedef long    off_t;
typedef long    pid_t;
typedef long    paddr_t;
typedef int     key_t;
typedef unsigned char  use_t;
typedef short   sysid_t;
typedef short   index_t;
typedef short   lock_t;
typedef unsigned int   size_t;
```

The form *daddr_t* is used for disk addresses except in an i-node on disk; see `fs`(4). Times are encoded in seconds since 00:00:00 GMT, January 1, 1970. The major and minor parts of a device code specify kind and unit number of a device and are installation-dependent. Offsets are measured in bytes from the beginning of a file. The *label_t* variables are used to save the processor state while another process is running.

## SEE ALSO

fs(4).

## NAME

`ucontext` – user context

## SYNOPSIS

```
#include <ucontext.h>
```

## DESCRIPTION

The `ucontext` structure defines the context of a thread of control within an executing process.

The ucontext_t structure is defined in <sys/ucontext.h>.

## SEE ALSO

`getcontext`(2), `setcontext`(2), `sigaction`(2), `sigprocmask`(2), `sigaltstack`(2),

## NAME

ustat – data returned by the ustat system call

## SYNOPSIS

```
#include <sys/types.h>
```

## DESCRIPTION

The system call `ustat` takes a parameter that is a pointer to the structure defined by this include file. This structure returns file system device statistics.

```
struct  ustat
{
   daddr_t            f_tfree;
   ino_t        f_tinode;
   char         f_fname [6];
   char         f_fpack [6];
};
```

The fields of this structure are defined as follows:

`f_tfree`
: The number of blocks with a size of DEV_BSIZ bytes that are available for allocation on the file system.

`f_tinode`
: The number of additional files that can be created on the file system.

`f_fname`
: The file system name. This field will be null unless a label has been added to the file system with `labelit`.

`f_fpack`
: The file system pack name. This field will be null unless a label has been added to the file system with `labelit`.

## FILES

/usr/include/sys/ustat.h
/usr/include/sys/types.h

## SEE ALSO

labelit(1M), ustat(2), types(5).

## NAME

values – machine-dependent values

## SYNOPSIS

```
#include <values.h>
```

## DESCRIPTION

This file contains a set of manifest constants, conditionally defined for particular processor architectures.

The model assumed for integers is binary representation (one's or two's complement), where the sign is represented by the value of the high-order bit.

| | |
|---|---|
| `BITS(`*type*`)` | The number of bits in a specified type (e.g., `int`). |
| `HIBITS` | The value of a short integer with only the high-order bit set. |
| `HIBITL` | The value of a long integer with only the high-order bit set. |
| `HIBITI` | The value of a regular integer with only the high-order bit set. |
| `MAXSHORT` | The maximum value of a signed short integer. |
| `MAXLONG` | The maximum value of a signed long integer. |
| `MAXINT` | The maximum value of a signed regular integer. |
| `MAXFLOAT, LN_MAXFLOAT` | The maximum value of a single-precision floating-point number, and its natural logarithm. |
| `MAXDOUBLE, LN_MAXDOUBLE` | The maximum value of a double-precision floating-point number, and its natural logarithm. |
| `MINFLOAT, LN_MINFLOAT` | The minimum positive value of a single-precision floating-point number, and its natural logarithm. |
| `MINDOUBLE, LN_MINDOUBLE` | The minimum positive value of a double-precision floating-point number, and its natural logarithm. |
| `FSIGNIF` | The number of significant bits in the mantissa of a single-precision floating-point number. |
| `DSIGNIF` | The number of significant bits in the mantissa of a double-precision floating-point number. |

## SEE ALSO

`intro`(3), `math`(5), `limits`(4).

## NAME

varargs – handle variable argument list

## SYNOPSIS

```
#include <varargs.h>

va_alist

va_dcl

va_list pvar;

void va_start(va_list pvar);

type va_arg(va_list pvar, type);

void va_end(va_list pvar);
```

## DESCRIPTION

This set of macros allows portable procedures that accept variable argument lists to be written. Routines that have variable argument lists [such as printf(3S)] but do not use varargs are inherently non-portable, as different machines use different argument-passing conventions.

va_alist is used as the parameter list in a function header.

va_dcl is a declaration for va_alist. No semicolon should follow va_dcl.

va_list is a type defined for the variable used to traverse the list.

va_start is called to initialize pvar to the beginning of the list.

va_arg will return the next argument in the list pointed to by pvar. *type* is the type the argument is expected to be. Different types can be mixed, but it is up to the routine to know what type of argument is expected, as it cannot be determined at runtime.

va_end is used to clean up.

Multiple traversals, each bracketed by va_start and va_end, are possible.

## EXAMPLE

This example is a possible implementation of execl [see exec(2)].

```
#include <unistd.h>
#include <varargs.h>
#define MAXARGS   100

/*    execl is called by
            execl(file, arg1, arg2, ..., (char *)0);
*/
execl(va_alist)
va_dcl
{
      va_list ap;
      char *file;
      char *args[MAXARGS];          /* assumed big enough*/
      int argno = 0;

      va_start(ap);
      file = va_arg(ap, char *);
      while ((args[argno++] = va_arg(ap, char *)) != 0)
            ;
```

```
                va_end(ap);
                return execv(file, args);
        }
```

## SEE ALSO

`exec`(2), `printf`(3S), `vprintf`(3S), `stdarg`(5).

## NOTES

It is up to the calling routine to specify in some manner how many arguments there are, since it is not always possible to determine the number of arguments from the stack frame. For example, `execl` is passed a zero pointer to signal the end of the list. `printf` can tell how many arguments are there by the format.

It is non-portable to specify a second argument of `char`, `short`, or `float` to `va_arg`, since arguments seen by the called function are not `char`, `short`, or `float`. C converts `char` and `short` arguments to `int` and converts `float` arguments to `double` before passing them to a function.

`stdarg` is the preferred interface.

## NAME
`wstat` – wait status

## SYNOPSIS
```
#include <sys/wait.h>
```

## DESCRIPTION
When a process waits for status from its children via either the `wait` or `waitpid` function, the status returned may be evaluated with macros, defined in `sys/wait.h`. These macros evaluate to integral expressions. The *stat* argument to these macros is the integer value returned from `wait` or `waitpid`.

See the `wait` man page for complete descriptions of these macros.

## SEE ALSO
`exit`(2), `wait`(2), `waitpid`(3C).

End of Chapter

# Chapter 6
# Communications Protocols

This chapter contains in printed form all the online manual entries for communications protocols. The entries in this chapter are generic to the DG/UX system; entries relating to a specific product such as TCP/IP or NFS are described in the documentation for that product.

## NAME

unix_ipc – piping communications within a host

## SYNOPSIS

```
#include <sys/types.h>
#include sys/un.h
```

## DESCRIPTION

The unix_ipc protocol is used for interprocess communications within a single host. It supports stream and datagram interfaces.

### Addressing

Endpoints can be named by entries in the file system:

```
        struct sockaddr_n {
              short sun_family;    /* AF_UNIX */
              char sun_path[109];  /* pathname */
};
```

## SEE ALSO

bind(2), pipe(2).

## NOTE

This implementation uses names in the file system; this is subject to change. See NOTES in bind(2).

End of Chapter

# Appendix A
# Contents and Permuted Index Man Pages

This is a printed copy of the table of contents and the permuted keyword in context index contained in the online **contents**(0) and **index**(0) manual pages. These man pages contain information extracted from the man pages in the DG/UX *Programmer's Reference* (Volumes 1 and 2), *System Manager's Reference*, and *User's Reference*.

The permuted index is a list of keywords, given in the second of three columns, together with the context in which the keyword is found. Keywords are either topical keywords or the names of manual entries. Entries are identified with their chapter numbers shown in parentheses. The right column lists the name of the manual page on which each keyword may be found. The left column contains useful information about the keyword.

## TABLE OF CONTENTS

This manual page contains the following sections:

## 3. Subroutines and Libraries

## PERMUTED INDEX

| | | |
|---|---|---|
| l3tol, ltol3: convert between | 3-byte integers and long integers . . . . . . . . | l3tol(3C) |
| berk_diff3: Berkeley | 3-way differential file comparison . . . . . . . | berk_diff3(1) |
| diff3: | 3-way differential file comparison . . . . . . . | diff3(1) |
| PostScript translator for tektronix | 4014 files /posttek: . . . . . . . . . . . . . | posttek(1) |
| module ttcompat: V7, | 4BSD and XENIX STREAMS compatibility . . | ttcompat(7) |
| PostScript translator for Diablo | 630 files /postdaisy: . . . . . . . . . . . . . | postdaisy(1) |
| and base-64 ASCII string | a64l, l64a: convert between long integer . . . . . | a64l(3C) |
| abort: generate an | abnormal termination signal . . . . . . . . . | abort(3C) |
| signal | abort: generate an abnormal termination . . . . . | abort(3C) |
| | abs, labs: return integer absolute value . . . . . . | abs(3C) |
| abs, labs: return integer | absolute value . . . . . . . . . . . . . . . . | abs(3C) |
| remainder: floor, ceiling, remainder, | absolute value functions /fabsf, rint, . . . . . . | floor(3M) |
| t_accept: | accept a connect request . . . . . . . . . . . | t_accept(3N) |
| accept: | accept a connection on a socket . . . . . . . . | accept(2) |
| | accept: accept a connection on a socket . . . . . | accept(2) |
| determine whether remote system can | accept binary messages /ckbinarsys: . . . . . . | ckbinarsys(1M) |
| accept, reject: | accept or reject print requests . . . . . . . . . | accept(1M) |
| requests | accept, reject: accept or reject print . . . . . . | accept(1M) |
| /touch: update | access and modification times of a file . . . . . . | touch(1) |
| utime: set file | access and modification times . . . . . . . . . | utime(2) |
| utimes: set file | access and modification times . . . . . . . . . | utimes(2) |
| sacadm: service | access controller administration . . . . . . . . | sacadm(1M) |
| sac: service | access controller . . . . . . . . . . . . . . | sac(1M) |
| file | access: determine the accessibility of a . . . . . | access(2) |
| elf_next: sequential archive member | access . . . . . . . . . . . . . . . . . . . | elf_next(3E) |
| elf_rand: random archive member | access . . . . . . . . . . . . . . . . . . . | elf_rand(3E) |
| misalign: handle misaligned memory | access faults . . . . . . . . . . . . . . . . | misalign(5) |
| elf: object file | access library . . . . . . . . . . . . . . . | elf(3E) |
| get or set supplementary group | access list IDs /getgroups, setgroups: . . . . . . | getgroups(2) |
| initialize the supplementary group | access list /initgroups: . . . . . . . . . . . . | initgroups(3C) |
| machine-independent/ sputl, sgetl: | access long integer data in a . . . . . . . . . | sputl(3X) |
| ldfcn: COFF executable file | access routines . . . . . . . . . . . . . . | ldfcn(4) |
| trelease: terminate | access to a tape . . . . . . . . . . . . . . | trelease(1) |
| taccess: initiate | access to labeled tape . . . . . . . . . . . | taccess(1) |
| device grantpt: grant | access to the slave pseudo-terminal . . . . . . | grantpt(3C) |
| pututline, setutent, endutent, utmpname: | access utmp file entry /getutline, . . . . . . . . | getut(3C) |
| access: determine the | accessibility of a file . . . . . . . . . . . . | access(2) |
| acct: enable or disable process | accounting . . . . . . . . . . . . . . . . . | acct(2) |
| acctcon1, acctcon2: connect-time | accounting . . . . . . . . . . . . . . . . . | acctcon(1M) |
| acctprc1, acctprc2: process | accounting . . . . . . . . . . . . . . . . . | acctprc(1M) |
| startup, turnacct: shell procedures for | accounting /prdaily, prtacct, shutacct, . . . . . | acctsh(1M) |
| /acctdusg, accton, acctwtmp: overview of | accounting and miscellaneous accounting/ . . . . | acct(1M) |
| overview of accounting and miscellaneous | accounting commands /accton, acctwtmp: . . . . | acct(1M) |
| diskusg: generate disk | accounting data by user id . . . . . . . . . . | diskusg(1M) |
| acct: per-process | accounting file format . . . . . . . . . . . | acct(4) |
| acctcom: search and print process | accounting file(s) . . . . . . . . . . . . . | acctcom(1) |
| acctmerg: merge or add total | accounting files . . . . . . . . . . . . . . | acctmerg(1M) |
| distinguish prime and/ holidays: | accounting information used to . . . . . . . . | holidays(4) |
| command summary from per-process | accounting records /acctcms: . . . . . . . . . . | acctcms(1M) |
| fwtmp, wtmpfix: manipulate connect | accounting records . . . . . . . . . . . . . | fwtmp(1M) |
| runacct: run daily | accounting . . . . . . . . . . . . . . . . . | runacct(1M) |
| /admaccounting: manage | accounting system . . . . . . . . . . . . . | admaccounting(1M) |
| accounting | acct: enable or disable process . . . . . . . . . | acct(2) |
| | acct: per-process accounting file format . . . . . | acct(4) |
| per-process accounting records | acctcms: command summary from . . . . . . . . | acctcms(1M) |
| accounting file(s) | acctcom: search and print process . . . . . . . . | acctcom(1) |
| accounting | acctcon1, acctcon2: connect-time . . . . . . . . | acctcon(1M) |
| acctcon1, | acctcon2: connect-time accounting . . . . . . . | acctcon(1M) |
| overview of accounting and/ | acctdisk, acctdusg, accton, acctwtmp: . . . . . . | acct(1M) |
| accounting and miscellaneous/ /acctdisk, | acctdusg, accton, acctwtmp: overview of . . . . . | acct(1M) |
| files | acctmerg: merge or add total accounting . . . . . | acctmerg(1M) |
| and miscellaneous/ /acctdisk, acctdusg, | accton, acctwtmp: overview of accounting . . . . | acct(1M) |
| | acctprc1, acctprc2: process accounting . . . . . . | acctprc(1M) |
| acctprc1, | acctprc2: process accounting . . . . . . . . . | acctprc(1M) |
| acctdisk, acctdusg, accton, | acctwtmp: overview of accounting and/ . . . . . | acct(1M) |
| pkgchk: check | accuracy of installation . . . . . . . . . . . | pkgchk(1M) |
| release indication t_rcvrel: | acknowledge receipt of an orderly . . . . . . . . | t_rcvrel(3N) |
| /sinf, cos, cosf, tan, tanf, asin, asinf, | acos, acosf, atan, atanf, atan2, atan2f:/ . . . . . | trig(3M) |
| /cos, cosf, tan, tanf, asin, asinf, acos, | acosf, atan, atanf, atan2, atan2f:/ . . . . . . . . | trig(3M) |

| | | |
|---|---|---|
| exportfs: make a directory | available for mounting via NFS | exportfs(2) |
| da: | AViiON family disk array subsystem | da(7) |
| cied: | AViiON family disk subsystem | cied(7) |
| cimd: | AViiON family disk subsystem | cimd(7) |
| cird: | AViiON family disk subsystem | cird(7) |
| sd: | AViiON family disk subsystem | sd(7) |
| Array adapter subsystem hada: | AViiON family High Availability Disk | hada(7) |
| controller /syac: | AViiON family intelligent asynchronous | syac(7) |
| /lp: DGC | AViiON family line printer special files | lp(7) |
| cisc: | AViiON family SCSI adapter subsystem | cisc(7) |
| insc: | AViiON family SCSI adapter subsystem | insc(7) |
| st: | AViiON family tape subsystem | st(7) |
| processor grfx: | AViiON series workstation graphics | grfx(7) |
| keyboard kbd: | AViiON series workstation system | kbd(7) |
| wait: | await completion of process | wait(1) |
| language nawk, | awk: pattern scanning and processing | nawk(1) |
| /mvgetch, mvwgetch, ungetch: get (or push | back) characters from curses terminal/ | curs_getch(3X) |
| ungetwc: push wchar_t character | back into input stream | ungetwc(3W) |
| ungetc: push character | back onto input stream | ungetc(3S) |
| /mvwgetwch, ungetwch: get (or push | back) wchar_t characters from curses/ | curs_getwch(3X) |
| /wbkgdset, bkgd, wbkgd: curses window | background manipulation routines | curs_bkgd(3X) |
| admbackup: manage | backup and recovery of file systems | admbackup(1M) |
| dump2: incremental file system | backup | dump2(1M) |
| tapesave: daily/weekly file system | backup /filesave, | filesave(1M) |
| frec: recover files from a | backup tape | frec(1M) |
| dumpcycle: dump cycle file for | backups | dumpcycle(4M) |
| pg: display file forward or | backward one screenful at a time | pg(1) |
| | banner: make posters | banner(1) |
| a text string from a message data | base /gettxt: retrieve | gettxt(1) |
| /elf_getbase: get the | base offset for an object file | elf_getbase(3E) |
| printcap: printer capability data | base | printcap(5) |
| software development environment data | base /sdetab: | sdetab(4) |
| signal: | base signals | signal(5) |
| store, delete, firstkey, nextkey: data | base subroutines /dbminit, fetch, | dbm(3X) |
| dbm_error, dbm_clearerr: data | base subroutines /dbm_nextkey, | ndbm(3C) |
| termcap: terminal capability data | base | termcap(5) |
| l64a: convert between long integer and | base-64 ASCII string /a64l, | a64l(3C) |
| forms: character | based forms package | forms(3X) |
| menus: character | based menus package | menus(3X) |
| getdev: lists devices | based on criteria | getdev(1M) |
| screen-oriented (visual) display editor | based on ex /vi, vedit, view: | vi(1) |
| panels: character | based panels package | panels(3X) |
| path names | basename, dirname: deliver portions of | basename(1) |
| path name | basename: return the last element of a | basename(3G) |
| for a text string in, message data | bases /display contents of, or search | srchtxt(1) |
| atrm: remove jobs spooled by at or | batch | atrm(1) |
| /at, | batch: execute commands at a later time | at(1) |
| cfgetospeed, cfsetispeed, cfsetospeed: | baud rate functions /cfgetispeed, | cfsetospeed(3C) |
| killchar, longname,/ /curs_termattrs: | baudrate, erasechar, has_ic, has_il, | curs_termattrs(3X) |
| language | bc: arbitrary-precision arithmetic | bc(1) |
| | bcmp: compare two areas of memory | bcmp(3C) |
| another | bcopy: copy bytes from one area to | bcopy(3C) |
| protocols, group or services/ | bcs_cat: type hosts, networks, passwd, | bcs_cat(1M) |
| | bdiff: big diff | bdiff(1) |
| cb: C program | beautifier | cb(1) |
| su: | become super-user or another user | su(1) |
| flash routines curs_beep: | beep, flash: curses bell and screen | curs_beep(3X) |
| /mvwinsstr, mvwinsnstr: insert string | before character under the cursor in a/ | curs_insstr(3X) |
| /mvwinsnwstr: insert wchar_t string | before character under the cursor in a/ | curs_inswstr(3X) |
| a/ /mvinsch, mvwinsch: insert a character | before the character under the cursor in | curs_insch(3X) |
| a/ /mvwinswch: insert a wchar_t character | before the character under the cursor in | curs_inswch(3X) |
| starter: information for | beginning users | starter(1) |
| forms field has off-screen data ahead or | behind /data_ahead, data_behind: tell if | form_data(3X) |
| curs_beep: beep, flash: curses | bell and screen flash routines | curs_beep(3X) |
| and directory comparator | berk_diff: Berkeley differential file | berk_diff(1) |
| file comparison | berk_diff3: Berkeley 3-way differential | berk_diff3(1) |
| comparison berk_diff3: | Berkeley 3-way differential file | berk_diff3(1) |
| comparator /berk_diff: | Berkeley differential file and directory | berk_diff(1) |
| handle regular expressions | berk_regex, regex, re_comp, re_exec: | berk_regex(3C) |
| signal facilities | berk_signal, signal: simplified software | berk_signal(3C) |
| suspend process until a signal is/ | berk_sigpause: set blocked signals and | berk_sigpause(2) |

| | | |
|---|---|---|
| cxref: generate | C program cross-reference | cxref(1) |
| cscope: interactively examine a | C program | cscope(1) |
| ctrace: trace a | C program to debug it | ctrace(1) |
| /xstr: extract strings from | C programs to implement shared strings | xstr(1) |
| an error message file by massaging | C source /mkstr: create | mkstr(1) |
| | cal: print calendar | cal(1) |
| dc: desk | calculator | dc(1) |
| cal: print | calendar | cal(1) |
| | calendar: reminder service | calendar(1) |
| cu: | call another UNIX system | cu(1) |
| data returned by the dg_mknod system | call /dg_mknod: | dg_mknod(5) |
| returned by dg_stat and dg_fstat system | call /dg_stat: data | dg_stat(5) |
| stat: data returned by stat system | call | stat(5) |
| data returned by the statfs system | call /statfs: | statfs(5) |
| ustat: data returned by the ustat system | call | ustat(5) |
| change the working directory of the | calling process /chdir: | chdir(2) |
| chroot: change the root directory of the | calling process | chroot(2) |
| change the working directory of the | calling process /fchdir: | fchdir(2) |
| /sighold: add a signal to the | calling process's set of blocked signals | sighold(2) |
| /sigrelse: remove a signal from the | calling process's set of blocked signals | sigrelse(2) |
| allocator malloc, free, realloc, | calloc, mallopt, mallinfo: memory | malloc(3X) |
| allocator malloc, free, realloc, | calloc, memalign, valloc,: memory | malloc(3C) |
| clnt_destroy,/ /authunix_create_default, | callrpc, clnt_broadcast, clnt_call, | rpc(3N) |
| intro: introduction to system | calls and error numbers | intro(2) |
| from source files, replace with catgets | calls. /catexstr: extract strings | catexstr(1) |
| unlink: exercise link and unlink system | calls /link, | link(1M) |
| library routines for remote procedure | calls /xprt_register, xprt_unregister: | rpc(3N) |
| determine whether remote system | can accept binary messages /ckbinarsys: | ckbinarsys(1M) |
| number of open files the current process | can have /getdtablesize: return the | getdtablesize(2) |
| print service /lp, | cancel: send/cancel requests to an LP | lp(1) |
| /init_pair, init_color, has_colors, | can_change_color, color_content,/ | curs_color(3X) |
| printcap: printer | capability data base | printcap(5) |
| termcap: terminal | capability data base | termcap(5) |
| terminfo: terminal and printer | capability database | terminfo(4) |
| a TERMINFO entry | captoinfo: convert a TERMCAP entry into | captoinfo(1M) |
| asa: interpret ASA | carriage control characters | asa(1) |
| edit: text editor (variant of ex for | casual users) | edit(1) |
| standard output | cat: concatenate and type files to | cat(1) |
| catgets: print message from message | catalog | catgets(1) |
| catopen, catclose: open/close a message | catalogue | catopen(3C) |
| gencat: generate a formatted message | catalogue | gencat(1) |
| /catopen, | catclose: open/close a message catalogue | catopen(3C) |
| files, replace with catgets calls. | catexstr: extract strings from source | catexstr(1) |
| strings from source files, replace with | catgets calls. /catexstr: extract | catexstr(1) |
| catalog | catgets: print message from message | catgets(1) |
| | catgets: read a program message | catgets(3C) |
| catalogue | catopen, catclose: open/close a message | catopen(3C) |
| and suspend process until a signal is | caught /set blocked signals | berk_sigpause(2) |
| pause: suspend process until a signal is | caught | pause(2) |
| suspend the process until a signal is | caught /clear a blocked signal and | sigpause(2) |
| | cb: C program beautifier | cb(1) |
| halfdelay, intrflush,/ /curs_inopts: | cbreak, nocbreak, echo, noecho, | curs_inopts(3X) |
| powf, sqrt, sqrtf:/ exp, expf, | cbrt, log, logf, log10, log10f, pow, | exp(3M) |
| | cc: C language compiler | cc(1) |
| | cd: change working directory | cd(1) |
| SCCS delta | cdc: change the delta commentary of an | cdc(1) |
| fabs, fabsf, rint,/ floor, floorf, | ceil, ceilf, copysign, fmod, fmodf, | floor(3M) |
| fabsf, rint,/ floor, floorf, ceil, | ceilf, copysign, fmod, fmodf, fabs, | floor(3M) |
| /fabs, fabsf, rint, remainder: floor, | ceiling, remainder, absolute value/ | floor(3M) |
| cfsetospeed: baud rate functions | cfgetispeed, cfgetospeed, cfsetispeed, | cfsetospeed(3C) |
| /tcdrain, tcflush, tcflow, cfgetospeed, | cfgetispeed, cfsetispeed, cfsetospeed,/ | termios(3C) |
| /tcsendbreak, tcdrain, tcflush, tcflow, | cfgetospeed, cfgetispeed, cfsetispeed,/ | termios(3C) |
| baud rate functions /cfgetispeed, | cfgetospeed, cfsetispeed, cfsetospeed: | cfsetospeed(3C) |
| | cflow: generate a C flow graph | cflow(1) |
| functions /cfgetispeed, cfgetospeed, | cfsetispeed, cfsetospeed: baud rate | cfsetospeed(3C) |
| /tcflow, cfgetospeed, cfgetispeed, | cfsetispeed, cfsetospeed, tcgetpgrp,/ | termios(3C) |
| /cfgetispeed, cfgetospeed, cfsetispeed, | cfsetospeed: baud rate functions | cfsetospeed(3C) |
| /cfgetospeed, cfgetispeed, cfsetispeed, | cfsetospeed, tcgetpgrp, tcsetpgrp,/ | termios(3C) |
| to string /strftime, | cftime, ascftime: convert date and time | strftime(3C) |
| sigprocmask: examine and | change blocked signals | sigprocmask(2) |
| deblock: | change blocking size | deblock(1) |

| | | |
|---|---|---|
| brk: | change data segment space allocation | brk(2) |
| sbrk: | change data segment space allocation | sbrk(2) |
| chmod: | change file mode | chmod(1) |
| passwd: | change login password | passwd(1) |
| chmod: | change mode of file | chmod(2) |
| fchmod: | change mode of file | fchmod(2) |
| /lseek: | change object pointer's current position | lseek(2) |
| putenv: | change or add value to environment | putenv(3C) |
| strchg, strconf: | change or query stream configuration | strchg(1) |
| a control point directory /cpd: | change or view the allocation limits for | cpd(1) |
| chown, chgrp: | change owner or group | chown(1) |
| nice: | change priority of a process | nice(2) |
| rcs: | change RCS file attributes | rcs(1) |
| chroot: | change root directory for a command | chroot(1M) |
| sigaction: examine and | change signal action | sigaction(2) |
| waitid: wait for child process to | change state | waitid(2) |
| shutdown: shut down system, | change system state | shutdown(1M) |
| delta /cdc: | change the delta commentary of an SCCS | cdc(1) |
| newform: | change the format of a text file | newform(1) |
| generate random numbers better, or | change the generator /setstate: | random(3C) |
| rename: | change the name of a file | rename(2) |
| point directory /dg_set_cpd_limits: | change the resource limits of a control | dg_set_cpd_limits(2) |
| process /chroot: | change the root directory of the calling | chroot(2) |
| calling process chdir: | change the working directory of the | chdir(2) |
| calling process fchdir: | change the working directory of the | fchdir(2) |
| delta: make a delta | (change) to an SCCS file | delta(1) |
| chown, lchown: | change user id and group id of a file | chown(2) |
| fchown: | change user id and group id of a file | fchown(2) |
| cd: | change working directory | cd(1) |
| helpadm: make | changes to the help facility database | helpadm(1M) |
| pipe: create an interprocess | channel | pipe(2) |
| /inch, winch, mvinch, mvwinch: get a | character and its attributes from a/ | curs_inch(3X) |
| /wstandend, standout, wstandout: curses | character and window attribute control/ | curs_attr(3X) |
| ungetwc: push wchar_t | character back into input stream | ungetwc(3W) |
| ungetc: push | character back onto input stream | ungetc(3S) |
| forms: | character based forms package | forms(3X) |
| menus: | character based menus package | menus(3X) |
| panels: | character based panels package | panels(3X) |
| /winsch, mvinsch, mvwinsch: insert a | character before the character under the/ | curs_insch(3X) |
| /mvinswch, mvwinswch: insert a wchar_t | character before the character under the/ | curs_inswch(3X) |
| isencrypt: determine whether a | character buffer is encrypted | isencrypt(3G) |
| tables /chrtbl: generate | character classification and conversion | chrtbl(1M) |
| tables /wchrtbl: generate | character classification and conversion | wchrtbl(1M) |
| mbchar: mbtowc, wctomb, mblen: multibyte | character conversion | mbchar(3W) |
| winwch, mvinwch, mvwinwch: get a wchar_t | character from a curses window /inwch, | curs_inwch(3X) |
| getwc, getwchar, fgetwc: get wchar_t | character from a stream | getwc(3W) |
| ispunct, isprint, isgraph, isascii: | character handling /isspace, iscntrl, | ctype(3C) |
| mbchar: mbtowc, mblen, wctomb: multibyte | character handling | mbchar(3C) |
| search for the first occurrence of a | character in a string /index: | index(3C) |
| search for the last occurrence of a | character in a string /rindex: | rindex(3C) |
| widec: multibyte | character I/O routines | widec(3W) |
| isalphanum: determine if a | character is alphanumeric | isalphanum(3C) |
| ishex: determine if a | character is hexadecimal | ishex(3C) |
| /sysv3_cuserid: get | character login name of the user | sysv3_cuserid(3S) |
| associated with effective/ cuserid: get | character login name or user name | cuserid(3S) |
| putwc, putwchar, fputwc: put wchar_t | character on a stream | putwc(3W) |
| getc, getchar, fgetc, getw: get | character or word from a stream | getc(3S) |
| putc, putchar, fputc, putw: put | character or word on a stream | putc(3S) |
| ascii: map of ASCII | character set | ascii(5) |
| rdsk: | character special disk interface | rdsk(7) |
| interface rmt: | character special magnetic tape | rmt(7) |
| fgrep: search a file for a | character string | fgrep(1) |
| itoa: convert an integer to an ASCII | character string | itoa(3C) |
| /mvgetnstr, mvwgetstr, mvwgetnstr: get | character strings from curses terminal/ | curs_getstr(3X) |
| /mvwgetwstr, mvwgetnwstr: get wchar_t | character strings from curses terminal/ | curs_getwstr(3X) |
| echowchar, wechowchar: add a wchar_t | character to a curses window /mvwaddwch, | curs_addwch(3X) |
| /delch, wdelch, mvdelch, mvwdelch: delete | character under cursor in a curses/ | curs_delch(3X) |
| /mvwinsch: insert a character before the | character under the cursor in a curses/ | curs_insch(3X) |
| window /mvwinsnstr: insert string before | character under the cursor in a curses | curs_insstr(3X) |
| /insert a wchar_t character before the | character under the cursor in a curses/ | curs_inswch(3X) |
| window /insert wchar_t string before | character under the cursor in a curses | curs_inswstr(3X) |

| | | |
|---|---|---|
| /mvwaddch, echochar, wechochar: add a | character (with attributes) to a curses/ | curs_addch(3X) |
| pkginfo: package | characteristics file | pkginfo(4) |
| dynamic_field_info: get forms field | characteristics /field_info, | form_field_info(3X) |
| /mvwinchstr, mvwinchnstr: get a string of | characters (and attributes) from a/ | curs_inchstr(3X) |
| /mvwaddchstr, mvwaddchnstr: add string of | characters (and attributes) to a curses/ | curs_addchst(3X) |
| /mvwaddchstr, mvwaddchnstr: add string of | characters (and attributes) to a curses/ | curs_addchstr(3X) |
| asa: interpret ASA carriage control | characters | asa(1) |
| _toupper, _tolower, toascii: translate | characters /conv: toupper, tolower, | conv(3C) |
| /mvwinstr, mvwinnstr: get a string of | characters from a curses window | curs_instr(3X) |
| /mvwinwchnstr: get a string of wchar_t | characters from a curses window | curs_inwchstr(3X) |
| /mvwinnwstr: get a string of wchar_t | characters from a curses window | curs_inwstr(3X) |
| /mvwgetch, ungetch: get (or push back) | characters from curses terminal keyboard | curs_getch(3X) |
| /ungetwch: get (or push back) wchar_t | characters from curses terminal keyboard | curs_getwch(3X) |
| rev: reverse order of | characters in each line of file | rev(1) |
| /mvwaddstr, mvwaddnstr: add a string of | characters to a curses window and/ | curs_addstr(3X) |
| /mvwaddwchnstr: add string of wchar_t | characters to a curses window | curs_addwchstr(3X) |
| /mvwaddnwstr: add a string of wchar_t | characters to a curses window | curs_addwstr(3X) |
| tr: translate | characters | tr(1) |
| wconv: towupper, towlower: translate | characters | wconv(3W) |
| classify ASCII and supplemetary code set | characters /isnumber, isspecial: | wctype(3W) |
| monacct, nulladm, prctmp, prdaily,/ | chargefee, ckpacct, dodisk, lastlogin, | acctsh(1M) |
| the calling process | chdir: change the working directory of | chdir(2) |
| pkgchk: | check accuracy of installation | pkgchk(1M) |
| repair them /fsck: | check file systems for consistency and | fsck(1M) |
| ci: | check in RCS revisions | ci(1) |
| get: | check out a version of an SCCS file | get(1) |
| co: | check out RCS revisions | co(1) |
| pwck, grpck: | check password or group file | pwck(1M) |
| permissions file uucheck: | check the uucp directories and | uucheck(1M) |
| lint: a C program | checker | lint(1) |
| labelit: copy file systems with label | checking /volcopy, | volcopy(1M) |
| processed by fsck and ncheck | checklist: list of file systems | checklist(4) |
| sum: print | checksum and block count of a file | sum(1) |
| chown, | chgrp: change owner or group | chown(1) |
| a TERMINFO entry | chgtinfo: create a temporary version of | chgtinfo(1) |
| times: get process and | child process times | times(2) |
| waitid: wait for | child process to change state | waitid(2) |
| wait3: wait for | child process to stop or terminate | wait3(2) |
| wait4: wait for the specified | child process to stop or terminate | wait4(2) |
| | chmod: change file mode | chmod(1) |
| | chmod: change mode of file | chmod(2) |
| | chown, chgrp: change owner or group | chown(1) |
| id of a file | chown, lchown: change user id and group | chown(2) |
| command | chroot: change root directory for a | chroot(1M) |
| calling process | chroot: change the root directory of the | chroot(2) |
| classification and conversion tables | chrtbl: generate character | chrtbl(1M) |
| | ci: check in RCS revisions | ci(1) |
| | cied: AViiON family disk subsystem | cied(7) |
| | cimd: AViiON family disk subsystem | cimd(7) |
| remque: remove an element from a | circular queue | remque(3C) |
| | cird: AViiON family disk subsystem | cird(7) |
| subsystem | cisc: AViiON family SCSI adapter | cisc(7) |
| system can accept binary messages | ckbinarsys: determine whether remote | ckbinarsys(1M) |
| prompt for and validate a date | ckdate, errdate, helpdate, valdate: | ckdate(1) |
| for and validate a group id | ckgid, errgid, helpgid, valgid: prompt | ckgid(1) |
| return an integer value | ckint: display a prompt; verify and | ckint(1) |
| return a menu item | ckitem: build a menu; prompt for and | ckitem(1) |
| keyword | ckkeywd: prompt for and validate a | ckkeywd(1) |
| nulladm, prctmp, prdaily,/ chargefee, | ckpacct, dodisk, lastlogin, monacct, | acctsh(1M) |
| return a pathname | ckpath: display a prompt; verify and | ckpath(1) |
| integer | ckrange: prompt for and validate an | ckrange(1) |
| return a string answer | ckstr: display a prompt; verify and | ckstr(1) |
| return a time of day | cktime: display a prompt; verify and | cktime(1) |
| | ckuid: prompt for and validate a user ID | ckuid(1) |
| | ckyorn: prompt for and validate yes/no | ckyorn(1) |
| /elf32_xlatetof, elf32_xlatetom: | class-dependent data translation | elf_xlate(3E) |
| /elf32_getehdr, elf32_newehdr: retrieve | class-dependent object file header | elf_getehdr(3E) |
| /elf32_getphdr, elf32_newphdr: retrieve | class-dependent program header table | elf_getphdr(3E) |
| /elf_getshdr: elf32_getshdr: retrieve | class-dependent section header | elf_getshdr(3E) |
| chrtbl: generate character | classification and conversion tables | chrtbl(1M) |
| wchrtbl: generate character | classification and conversion tables | wchrtbl(1M) |

| | | |
|---|---|---|
| ´/isenglish, isnumber, isspecial: | classify ASCII and supplemetary code set/ | wctype(3W) |
| strclean: STREAMS error logger | cleanup program | strclean(1M) |
| uucleanup: uucp spool directory | clean-up | uucleanup(1M) |
| process until a signal is/ /sigpause: | clear a blocked signal and suspend the | sigpause(2) |
| /wclrtobot, clrtoeol, wclrtoeol: | clear all or part of a curses window | curs_clear(3X) |
| | clear: clear terminal screen | clear(1) |
| clri: | clear inode | clri(1M) |
| clear: | clear terminal screen | clear(1) |
| clrtoeol,/ curs_clear: erase, werase, | clear, wclear, clrtobot, wclrtobot, | curs_clear(3X) |
| inquiries ferror, feof, | clearerr, fileno: stream status | ferror(3S) |
| setscrreg, wsetscrreg,/ /curs_outopts: | clearok, idlok, idcok immedok, leaveok, | curs_outopts(3X) |
| ypprot_err: Network Information Service | client interface /yperr_string, | ypclnt(3N) |
| admclient: manage operating system | clients | admclient(1M) |
| nlsgetcall: get | client's data passed via the listener | nlsgetcall(3N) |
| remove locks held by remote lock | clients /dg_lock_kill: | dg_lock_kill(2) |
| /dg_setsecretkey: store a | client's secret key in the keyserver | dg_setsecretkey(2) |
| /decrypt conversation key with the | client/server common key | dg_decryptsessionkey(2) |
| /encrypt conversation key with the | client/server common key | dg_encryptsessionkey(2) |
| a shell (command interpreter) having a | C-like syntax /csh: invoke | csh(1) |
| /authunix_create_default, callrpc, | clnt_broadcast, clnt_call, clnt_destroy,/ | rpc(3N) |
| clnt_control,/ /callrpc, clnt_broadcast, | clnt_call, clnt_destroy, clnt_create, | rpc(3N) |
| /clnt_call, clnt_destroy, clnt_create, | clnt_control, clnt_freeres, clnt_geterr,/ | rpc(3N) |
| /clnt_broadcast, clnt_call, clnt_destroy, | clnt_create, clnt_control, clnt_freeres,/ | rpc(3N) |
| /callrpc, clnt_broadcast, clnt_call, | clnt_destroy, clnt_create, clnt_control,/ | rpc(3N) |
| clnt_destroy, clnt_create, clnt_control, | clnt_freeres, clnt_geterr,/ /clnt_call, | rpc(3N) |
| /clnt_create, clnt_control, clnt_freeres, | clnt_geterr, clnt_pcreateerror,/ | rpc(3N) |
| /clnt_control, clnt_freeres, clnt_geterr, | clnt_pcreateerror, clnt_perrno,/ | rpc(3N) |
| clnt_geterr, clnt_pcreateerror, | clnt_perrno, clnt_perror,/ /clnt_freeres, | rpc(3N) |
| /clnt_pcreateerror, clnt_perrno, | clnt_perror, clnt_spcreateerror,/ | rpc(3N) |
| /clnt_sperrno, clnt_sperror, | clntraw_create, clnttcp_create,/ | rpc(3N) |
| clnt_sperror,/ /clnt_perrno, clnt_perror, | clnt_spcreateerror, clnt_sperrno, | rpc(3N) |
| /clnt_perror, clnt_spcreateerror, | clnt_sperrno, clnt_sperror,/ | rpc(3N) |
| /clnt_spcreateerror, clnt_sperrno, | clnt_sperror, clntraw_create,/ | rpc(3N) |
| /clnt_sperror, clntraw_create, | clnttcp_create, clntudp_create,/ | rpc(3N) |
| /clntraw_create, clnttcp_create, | clntudp_create, host2netname,/ | rpc(3N) |
| to allow synchronization of the system | clock /adjtime: correct the time | adjtime(2) |
| cron: | clock agent | cron(1M) |
| alarm: set a process alarm | clock | alarm(2) |
| | clock: report CPU time used | clock(3C) |
| STREAMS driver | clone: open any minor device on a | clone(7) |
| ldclose, ldaclose: | close a common object file | ldclose(3X) |
| t_close: | close a transport endpoint | t_close(3N) |
| descriptor /close: | close an object associated with a file | close(2) |
| file descriptor | close: close an object associated with a | close(2) |
| fclose, fflush: | close or flush a stream | fclose(3S) |
| p2open, p2close: open, | close pipes to and from a command | p2open(3G) |
| readdir, telldir, seekdir, rewinddir, | closedir: directory operations /opendir, | directory(3X) |
| /syslog, openlog, | closelog, setlogmask: control system log | syslog(3C) |
| | clri: clear inode | clri(1M) |
| /erase, werase, clear, wclear, | clrtobot, wclrtobot, clrtoeol,/ | curs_clear(3X) |
| of/ /clear, wclear, clrtobot, wclrtobot, | clrtoeol, wclrtoeol: clear all or part | curs_clear(3X) |
| | cmp: compare two files | cmp(1) |
| | co: check out RCS revisions | co(1) |
| dis: object | code disassembler | dis(1) |
| classify ASCII and supplemetary | code set characters /isspecial: | wctype(3W) |
| iconv: | code set conversion | iconv(1) |
| eucset: set or get EUC | code set widths | eucset(1) |
| get information of supplementary | code sets /getwidth: | getwidth(3W) |
| strings, compressing or expanding escape | codes /streadd, strcadd, strecpy: copy | strecpy(3G) |
| to ELF | cof2elf: translate object file from COFF | cof2elf(1) |
| read the archive header of a member of a | COFF archive file /ldahread: | ldahread(3X) |
| ldfcn: | COFF executable file access routines | ldfcn(4) |
| cof2elf: translate object file from | COFF to ELF | cof2elf(1) |
| ctl: | COFF-to-legend translator | ctl(1) |
| | col: filter reverse line-feeds | col(1) |
| colltbl: create | collation database | colltbl(1M) |
| strcoll: string | collation | strcoll(3C) |
| | colltbl: create collation database | colltbl(1M) |
| /color_content, pair_content: curses | color manipulation routines | curs_color(3X) |
| color/ /has_colors, can_change_color, | color_content, pair_content: curses | curs_color(3X) |
| set and get maximum numbers of rows and | columns in menus /menu_format: | menu_format(3X) |

| | | |
|---|---|---|
| | comb: combine SCCS deltas | comb(1) |
| comb: | combine SCCS deltas | comb(1) |
| two sorted files | comm: select or reject lines common to | comm(1) |
| rksh: KornShell, a standard/restricted | command and programming language /ksh, | ksh(1) |
| nice: run a | command at a higher or lower priority | nice(1) |
| chroot: change root directory for a | command | chroot(1M) |
| /usage: retrieve a | command description and usage examples | usage(1) |
| env: set environment for | command execution | env(1) |
| uux: UNIX-to-UNIX system | command execution | uux(1) |
| mail_pipe: invoke recipient | command for incoming mail | mail_pipe(1M) |
| nohup: run a | command immune to hangups and quits | nohup(1) |
| syntax /csh: invoke a shell | (command interpreter) having a C-like | csh(1) |
| editread: | command line editor | editread(5) |
| getopt: parse | command options | getopt(1) |
| getopts, getoptcvt: parse | command options | getopts(1) |
| p2close: open, close pipes to and from a | command /p2open, | p2open(3G) |
| subsystem /form_driver: | command processor for the forms | form_driver(3X) |
| subsystem /menu_driver: | command processor for the menus | menu_driver(3X) |
| sh, jsh, rsh, restsh: shell, the | command programming language | sh(1) |
| for returning a stream to a remote | command /rresvport, ruserok: routines | rcmd(3X) |
| activity /timex: time a | command; report process data and system | timex(1) |
| uuxqt: execute remote | command requests | uuxqt(1M) |
| rexec: return stream to a remote | command | rexec(3X) |
| accounting records acctcms: | command summary from per-process | acctcms(1M) |
| system: issue a shell | command | system(3S) |
| test: condition evaluation | command | test(1) |
| time: time a | command | time(1) |
| locate: identify a | command using keywords | locate(1) |
| construct argument list(s) and execute | command /xargs: | xargs(1) |
| accounting and miscellaneous accounting | commands /accton, acctwtmp: overview of | acct(1M) |
| intro: introduction to | commands and application programs | intro(1) |
| intro: introduction to | commands and application programs | intro(1) |
| /introduction to system maintenance | commands and application programs | intro(1M) |
| at, batch: execute | commands at a later time | at(1) |
| apropos: locate | commands by keyword lookup | apropos(1) |
| ANSI tapes /REELexchange: | commands for reading and writing IBM and | reelexchange_intro(1) |
| mail mailsurr: surrogate | commands for routing and transport of | mailsurr(4M) |
| install: install | commands | install(1M) |
| rcsintro: introduction to RCS | commands | rcsintro(1) |
| streamio: STREAMS ioctl | commands | streamio(7) |
| environment target /sde-target: print | commands to reset software development | sde-target(1) |
| mcs: manipulate the | comment section of an object file. | mcs(1) |
| cdc: change the delta | commentary of an SCCS delta | cdc(1) |
| ar: DG/UX | common archive file format | ar(4) |
| /a.out: | common assembler and link editor output | a.out(4) |
| conversation key with the client/server | common key /decrypt | dg_decryptsessionkey(2) |
| conversation key with the client/server | common key /encrypt | dg_encryptsessionkey(2) |
| cprs: compress a | common object file | cprs(1) |
| manipulate line number entries of a | common object file function /ldlitem: | ldlread(3X) |
| ldclose, ldaclose: close a | common object file | ldclose(3X) |
| ldfhread: read the file header of a | common object file | ldfhread(3X) |
| to line number entries of a section of a | common object file /ldnlseek: seek | ldlseek(3X) |
| to relocation entries of a section of a | common object file /ldnrseek: seek | ldrseek(3X) |
| an indexed/named section header of a | common object file /ldnshread: read | ldshread(3X) |
| seek to an indexed/named section of a | common object file /ldsseek, ldnsseek: | ldsseek(3X) |
| linenum: line number entries in a | common object file | linenum(4) |
| nm: print name list of | common object file | nm(1) |
| reloc: relocation information for a | common object file | reloc(4) |
| /syms: | common object file symbol table format | syms(4) |
| filehdr: file header for | common object files | filehdr(4) |
| ld: link editor for | common object files | ld-coff(1) |
| glossary: definitions of | common terms and symbols | glossary(1) |
| comm: select or reject lines | common to two sorted files | comm(1) |
| ipcs: report inter-process | communication facilities status | ipcs(1) |
| stdipc: ftok: standard interprocess | communication package | stdipc(3C) |
| socket: create an endpoint for | communication | socket(2) |
| unix_ipc: piping | communications within a host | unix_ipc(6F) |
| /admsnmpcommunity: manage the SNMP | community database | admsnmpcommunity(1M) |
| Berkeley differential file and directory | comparator /berk_diff: | berk_diff(1) |
| diff: differential file | comparator | diff(1) |
| /store_conditional: indivisible | compare and swap | store_conditional(2) |

| | | |
|---|---|---|
| descriptions infocmp: | compare or print out TERMINFO | infocmp(1M) |
| rcsdiff: | compare RCS revisions | rcsdiff(1) |
| bcmp: | compare two areas of memory | bcmp(3C) |
| dircmp: | compare two directories | dircmp(1) |
| cmp: | compare two files | cmp(1) |
| sccsdiff: | compare two versions of an SCCS file | sccsdiff(1) |
| Berkeley 3-way differential file | comparison /berk_diff3: | berk_diff3(1) |
| diff3: 3-way differential file | comparison | diff3(1) |
| ttcompat: V7, 4BSD and XENIX STREAMS | compatibility module | ttcompat(7) |
| compver: | compatible versions file | compver(4) |
| /regcmp, regex: | compile and execute regular expression | regcmp(3G) |
| /regcmp, regex: | compile and execute regular expression | regcmp(3X) |
| step, advance: regular expression | compile and match routines /compile, | regexp(5) |
| step, advance: regular expression | compile and match routines /compile, | regexpr(3G) |
| kbdcomp: | compile kbd tables | kbdcomp(1M) |
| regcmp: regular expression | compile | regcmp(1) |
| expression compile and match/ regexp: | compile, step, advance: regular | regexp(5) |
| expression compile and match/ regexpr: | compile, step, advance: regular | regexpr(3G) |
| cc: C language | compiler | cc(1) |
| gcc: GNU C language | compiler | gcc(1) |
| sno: SNOBOL interpreter and | compiler | sno(1) |
| tic: TERMINFO | compiler | tic(1M) |
| zic: time zone | compiler | zic(1M) |
| yacc: yet another | compiler-compiler | yacc(1) |
| erf, erfc: error function and | complementary error function | erf(3M) |
| for previously delayed lock requests to | complete /dg_lock_wait: wait | dg_lock_wait(2) |
| wait: await | completion of process | wait(1) |
| entry corresponding to NETPATH | component /get /etc/netconfig | getnetpath(3N) |
| cprs: | compress a common object file | cprs(1) |
| pack, pcat, unpack: | compress and expand files | pack(1) |
| files compress, uncompress, zcat: | compress, expand or display expanded | compress(1) |
| expand or display expanded files | compress, uncompress, zcat: compress, | compress(1) |
| /streadd, strcadd, strecpy: copy strings, | compressing or expanding escape codes | strecpy(3G) |
| elf_hash: | compute hash value | elf_hash(3E) |
| an object file /ldtbindex: | compute index of symbol table entry of | ldtbindex(3X) |
| div, ldiv: | compute the quotient and remainder | div(3C) |
| | compver: compatible versions file | compver(4) |
| output /cat: | concatenate and type files to standard | cat(1) |
| test: | condition evaluation command | test(1) |
| select: wait for I/O | conditions | select(2) |
| system log server syslog. | conf: configuration file for syslogd | syslog.conf(5) |
| | config: configure a system | config(1M) |
| pathconf, fpathconf: get | configurable pathname variables | pathconf(2) |
| sysconf: get | configurable system variables | sysconf(2) |
| dg_sysctl: perform system | configuration and control functions | dg_sysctl(2) |
| /getnetconfig: get network | configuration database entry | getnetconfig(3N) |
| netconfig: network | configuration database | netconfig(4) |
| log server syslog.conf: | configuration file for syslogd system | syslog.conf(5) |
| doconfig: execute a | configuration script | doconfig(3N) |
| strchg, strconf: change or query stream | configuration | strchg(1) |
| config: | configure a system | config(1M) |
| modules /autopush: | configure automatically pushed STREAMS | autopush(1M) |
| lpadmin: | configure the LP print service | lpadmin(1M) |
| /t_rcvconnect: receive the | confirmation from a connect request | t_rcvconnect(3N) |
| fwtmp, wtmpfix: manipulate | connect accounting records | fwtmp(1M) |
| /set_menu_items, menu_items, item_count: | connect and disconnect items to and from/ | menu_items(3X) |
| /form_fields, field_count, move_field: | connect fields to forms | form_field(3X) |
| socket | connect: initiate a connection on a | connect(2) |
| t_accept: accept a | connect request | t_accept(3N) |
| t_listen: listen for a | connect request | t_listen(3N) |
| receive the confirmation from a | connect request /t_rcvconnect: | t_rcvconnect(3N) |
| getpeername: get name of | connected peer | getpeername(2) |
| socketpair: create a pair of | connected sockets | socketpair(2) |
| establish an out-going terminal line | connection /dial: | dial(3C) |
| accept: accept a | connection on a socket | accept(2) |
| connect: initiate a | connection on a socket | connect(2) |
| shut down part of a full-duplex | connection /shutdown: | shutdown(2) |
| data or expedited data sent over a | connection /t_rcv: receive | t_rcv(3N) |
| send data or expedited data over a | connection /t_snd: | t_snd(3N) |
| /t_connect: establish a | connection with another transport user | t_connect(3N) |
| line discipline for unique stream | connections /connld: | connld(7) |

| | | |
|---|---|---|
| listen: listen for | connections on a socket | listen(2) |
| acctcon1, acctcon2: | connect-time accounting | acctcon(1M) |
| stream connections | connld: line discipline for unique | connld(7) |
| fsck: check file systems for | consistency and repair them | fsck(1M) |
| display a message on stderr or system | console /fmtmsg: | fmtmsg(1) |
| display a message on stderr or system | console /fmtmsg: | fmtmsg(3C) |
| syscon: DG/UX operating system | console pseudo-device | syscon(7) |
| langinfo: language information | constants | langinfo(5) |
| header file for implementation-specific | constants /limits: | limits(4) |
| math: math functions and | constants | math(5) |
| command /xargs: | construct argument list(s) and execute | xargs(1) |
| deroff: remove nroff/troff, tbl, and eqn | constructs | deroff(1) |
| control maximum system resource | consumption /getrlimit, setrlimit: | getrlimit(2) |
| vlimit: control maximum system resource | consumption | vlimit(3C) |
| /Uutry: try to | contact remote system with debugging on | uutry(1M) |
| getdgrp: lists device groups which | contain devices that match criteria | getdgrp(1M) |
| restore the process state to that | contained in a signal frame /sigret: | sigret(2) |
| the file handle of the export entry | containing filename /getfh: return | getfh(2) |
| pkgmap: package | contents description file | pkgmap(4) |
| /elf_rawfile: retrieve uninterpreted file | contents | elf_rawfile(3E) |
| readlink: read the | contents of a symbolic link | readlink(2) |
| ls: list | contents of directory | ls(1) |
| in, message data bases /srchtxt: display | contents of, or search for a text string | srchtxt(1) |
| register getpsr: return the current | contents of the processor status | getpsr(2) |
| tsniff: summary report of tape | contents | tsniff(1) |
| setcontext: get and set current user | context /getcontext, | getcontext(2) |
| set or get signal alternate stack | context /sigaltstack: | sigaltstack(2) |
| sigstack: set and/or get signal stack | context | sigstack(2) |
| csplit: | context split | csplit(1) |
| ucontext: user | context | ucontext(5) |
| /swapcontext: manipulate user | contexts | swapcontext(3C) |
| ioctl: | control a device | ioctl(2) |
| elf_cntl: | control a file descriptor | elf_cntl(3E) |
| asa: interpret ASA carriage | control characters | asa(1) |
| tcsendbreak, tcdrain, tcflush, tcflow: | control data transmission | tcflush(3C) |
| jobs: summary of DG/UX job | control facilities | jobs(3C) |
| fcntl: file descriptor | control | fcntl(2) |
| IEEE floating-point environment | control /fpgetsticky, fpsetsticky: | fpgetround(3C) |
| perform system configuration and | control functions /dg_sysctl: | dg_sysctl(2) |
| init, telinit: process | control initialization | init(1M) |
| consumption getrlimit, setrlimit: | control maximum system resource | getrlimit(2) |
| consumption vlimit: | control maximum system resource | vlimit(3C) |
| memcntl: memory management | control | memcntl(2) |
| /menu_grey, set_menu_pad, menu_pad: | control menus display attributes | menu_attributes(3X) |
| mt: magnetic tape | control | mt(1) |
| semctl: semaphore | control operations | semctl(2) |
| shmctl: shared memory | control operations | shmctl(2) |
| fcntl: file | control options | fcntl(5) |
| or view the allocation limits for a | control point directory /cpd: change | cpd(1) |
| /change the resource limits of a | control point directory | dg_set_cpd_limits(2) |
| lpc: line printer | control program | lpc(1M) |
| curses character and window attribute | control routines /standout, wstandout: | curs_attr(3X) |
| typeahead: curses terminal input option | control routines /timeout, wtimeout, | curs_inopts(3X) |
| nl, nonl: curses terminal output option | control routines /wsetscrreg, scrollok, | curs_outopts(3X) |
| is_wintouched: curses refresh | control routines /is_linetouched, | curs_touch(3X) |
| setpgid: set process group ID for job | control | setpgid(2) |
| dkctl: | control special disk operations | dkctl(1M) |
| syslog, openlog, closelog, setlogmask: | control system log | syslog(3C) |
| devtty: | control terminal pseudo-device | devtty(7) |
| vhangup: virtually hang up the current | control terminal | vhangup(2) |
| uadmin: administrative | control | uadmin(2) |
| uustat: uucp status inquiry and job | control | uustat(1) |
| vc: version | control | vc(1) |
| sacadm: service access | controller administration | sacadm(1M) |
| tcload: load terminal | controller devices | tcload(1M) |
| vitr: Vilya TokenRing | Controller interface | vitr(7) |
| verify that the VSC synchronous | controller is operable /vsccheck: | vsccheck(1M) |
| sac: service access | controller | sac(1M) |
| AViiON family intelligent asynchronous | controller /syac: | syac(7) |
| resident software onto VSC synchronous | controller /vscload: download board | vscload(1M) |
| _tolower, toascii: translate characters | conv: toupper, tolower, _toupper, | conv(3C) |

| | | |
|---|---|---|
| term: | conventional names for terminals | term(5) |
| common/ /dg_decryptsessionkey: decrypt | conversation key with the client/server | dg_decryptsessionkey(2) |
| common/ /dg_encryptsessionkey: encrypt | conversation key with the client/server | dg_encryptsessionkey(2) |
| iconv: code set | conversion | iconv(1) |
| wctomb, mblen: multibyte character | conversion /mbchar: mbtowc, | mbchar(3W) |
| mbstowcs, wctombs,: multibyte string | conversion /mbstring: | mbstring(3W) |
| units: | conversion program | units(1) |
| generate character classification and | conversion tables /chrtbl: | chrtbl(1M) |
| generate character classification and | conversion tables /wchrtbl: | wchrtbl(1M) |
| entry /captoinfo: | convert a TERMCAP entry into a TERMINFO | captoinfo(1M) |
| string /itoa: | convert an integer to an ASCII character | itoa(3C) |
| dd: | convert and copy a file | dd(1) |
| integers /l3tol, ltol3: | convert between 3-byte integers and long | l3tol(3C) |
| ASCII string /a64l, l64a: | convert between long integer and base-64 | a64l(3C) |
| localtime, gmtime, asctime, tzset: | convert date and time to string /ctime, | ctime(3C) |
| strftime, cftime, ascftime: | convert date and time to string | strftime(3C) |
| /ecvt, fcvt, gcvt: | convert floating-point number to string | ecvt(3C) |
| /wscanw, mvscanw, mvwscanw, vwscanw: | convert formatted input from a curses/ | curs_scanw(3X) |
| scanf, fscanf, sscanf: | convert formatted input | scanf(3S) |
| scanf, fscanf, sscanf: | convert formatted input | scanf(3W) |
| argument list vscanf, vfscanf, vsscanf: | convert formatted input using varargs | vscanf(3S) |
| number strtod, atof,: | convert string to double-precision | strtod(3C) |
| strtol, strtoul, atol, atoi: | convert string to integer | strtol(3C) |
| getdate, getdate_err: | convert user format date and time | getdate(3C) |
| byte order /htonl, htons, ntohl, ntohs: | convert values between host and network | byteorder(3N) |
| timod: Transport Interface | cooperating STREAMS module | timod(7) |
| versions /elf_version: | coordinate library and application | elf_version(3E) |
| getmaxyx: get curses cursor and window | coordinates /getyx, getparyx, getbegyx, | curs_getyx(3X) |
| dd: convert and | copy a file | dd(1) |
| copylist: | copy a file into memory | copylist(3G) |
| bcopy: | copy bytes from one area to another | bcopy(3C) |
| cpio: | copy file archives in and out | cpio(1) |
| volcopy, labelit: | copy file systems with label checking | volcopy(1M) |
| cp: | copy files | cp(1) |
| /strccpy: streadd, strcadd, strecpy: | copy strings, compressing or expanding/ | strccpy(3G) |
| uucp, uulog, uuname: UNIX-to-UNIX system | copy | uucp(1) |
| uupick: public UNIX-to-UNIX system file | copy /uuto, | uuto(1) |
| | copylist: copy a file into memory | copylist(3G) |
| | copyright: copyright information file | copyright(4) |
| copyright: | copyright information file | copyright(4) |
| rint,/ floor, floorf, ceil, ceilf, | copysign, fmod, fmodf, fabs, fabsf, | floor(3M) |
| /finite, unordered, | copysign: IEEE floating-point routines | ieeefp(3C) |
| /curs_overlay: overlay, overwrite, | copywin: overlap and manipulate/ | curs_overlay(3X) |
| | core: format of core image file | core(4) |
| core: format of | core image file | core(4) |
| synchronization of the system/ adjtime: | correct the time to allow | adjtime(2) |
| /menu_cursor: pos_menu_cursor: | correctly position a menus cursor | menu_cursor(3X) |
| getnetpath: get /etc/netconfig entry | corresponding to NETPATH component | getnetpath(3N) |
| acosf, atan, atanf,/ /trig: sin, sinf, | cos, cosf, tan, tanf, asin, asinf, acos, | trig(3M) |
| acosf, atan,/ trig: sin, sinf, cos, | cosf, tan, tanf, asin, asinf, acos, | trig(3M) |
| acosh, atanh:/ /sinh, sinhf, | cosh, coshf, tanh, tanhf, asinh, | sinh(3M) |
| atanh:/ /sinh, sinhf, cosh, | coshf, tanh, tanhf, asinh, acosh, | sinh(3M) |
| sum: print checksum and block | count of a file | sum(1) |
| wc: word | count | wc(1) |
| | cp: copy files | cp(1) |
| limits for a control point directory | cpd: change or view the allocation | cpd(1) |
| cpio: format of | cpio archive | cpio(4) |
| | cpio: copy file archives in and out | cpio(1) |
| | cpio: format of cpio archive | cpio(4) |
| | cpp: the C language preprocessor | cpp(1) |
| | cprs: compress a common object file | cprs(1) |
| clock: report | CPU time used | clock(3C) |
| | crash: examine system images | crash(1M) |
| crashes | crash: what to do when the DG/UX system | crash(8) |
| crash: what to do when the DG/UX system | crashes | crash(8) |
| existing one | creat: create a new file or rewrite an | creat(2) |
| mkdir: | create a directory file | mkdir(2) |
| /mknod: | create a file entry in the file system | mknod(2) |
| mkfs, newfs: | create a file system | mkfs(1M) |
| dg_mknod: | create a file system node | dg_mknod(2) |
| tmpnam, tempnam: | create a name for a temporary file | tmpnam(3S) |

| | | |
|---|---|---|
| getcontext, setcontext: get and set | current user context | getcontext(2) |
| find the slot in the utmp file of the | current user /ttyslot: | ttyslot(3C) |
| /replace_panel: get or set the | current window of a panels panel | panel_window(3X) |
| getcwd: get pathname of | current working directory | getcwd(3C) |
| getwd: get | current working directory pathname | getwd(3C) |
| current/ /form_page, set_current_field, | current_field, field_index: set forms | form_page(3X) |
| /menu_item_current: set_current_item, | current_item, set_top_row, top_row,/ | menu_item_current(3X) |
| /get information about the system's | currently active processes | dg_process_info(2) |
| mvwaddch, echochar, wechochar: add a/ | curs_addch: addch, waddch, mvaddch, | curs_addch(3X) |
| waddchstr, waddchnstr, mvaddchstr,/ | curs_addchstr: addchstr, addchnstr, | curs_addchst(3X) |
| waddchstr, waddchnstr, mvaddchstr,/ | curs_addchstr: addchstr, addchnstr, | curs_addchstr(3X) |
| waddnstr, mvaddstr, mvaddnstr,/ | curs_addstr: addstr, addnstr, waddstr, | curs_addstr(3X) |
| mvwaddwch, echowchar, wechowchar: add a/ | curs_addwch: addwch, waddwch, mvaddwch, | curs_addwch(3X) |
| waddwchstr, waddwchnstr, mvaddwchstr,/ | curs_addwchstr: addwchstr, addwchnstr, | curs_addwchstr(3X) |
| waddwstr, waddnwstr, mvaddwstr,/ | curs_addwstr: addwstr, addnwstr, | curs_addwstr(3X) |
| wattron, attrset, wattrset, standend,/ | curs_attr: attroff, wattroff, attron, | curs_attr(3X) |
| screen flash routines | curs_beep: beep, flash: curses bell and | curs_beep(3X) |
| wbkgd: curses window background/ | curs_bkgd: bkgdset, wbkgdset, bkgd, | curs_bkgd(3X) |
| whline, wvline: create curses borders,/ | curs_border: border, wborder, box, | curs_border(3X) |
| wclear, clrtobot, wclrtobot, clrtoeol,/ | curs_clear: erase, werase, clear, | curs_clear(3X) |
| init_color, has_colors,/ | curs_color: start_color, init_pair, | curs_color(3X) |
| mvwdelch: delete character under cursor/ | curs_delch: delch, wdelch, mvdelch, | curs_delch(3X) |
| insdelln, winsdelln, insertln,/ | curs_deleteln: deleteln, wdeleteln, | curs_deleteln(3X) |
| curs_beep: beep, flash: | curses bell and screen flash routines | curs_beep(3X) |
| /wborder, box, whline, wvline: create | curses borders, horizontal and vertical/ | curs_border(3X) |
| control/ /wstandend, standout, wstandout: | curses character and window attribute | curs_attr(3X) |
| /color_content, pair_content: | curses color manipulation routines | curs_color(3X) |
| optimization package | curses: CRT screen handling and | curses(3X) |
| /getyx, getparyx, getbegyx, getmaxyx: get | curses cursor and window coordinates | curs_getyx(3X) |
| /killchar, longname, termattrs, termname: | curses environment query routines | curs_termattrs(3X) |
| termcap/ /tgetnum, tgetstr, tgoto, tputs: | curses interfaces (emulated) to the | curs_termcap(3X) |
| /mvcur, tigetflag, tigetnum, tigetstr: | curses interfaces to terminfo database | curs_terminfo(3X) |
| pechowchar: create and display | curses pads /pnoutrefresh, pechochar, | curs_pad(3X) |
| /wtouchln, is_linetouched, is_wintouched: | curses refresh control routines | curs_touch(3X) |
| ripoffline, curs_set, napms: low-level | curses routines /getsyx, setsyx, | curs_kernel(3X) |
| /scr_init, scr_set: read (write) a | curses screen from (to) a file | curs_scr_dump(3X) |
| scr_dump: format of | curses screen image file | scr_dump(4) |
| /endwin, isendwin, set_term, delscreen: | curses screen initialization and/ | curs_initscr(3X) |
| slk_attron, slk_attrset, slk_attroff: | curses soft label routines /slk_touch, | curs_slk(3X) |
| /qiflush, timeout, wtimeout, typeahead: | curses terminal input option control/ | curs_inopts(3X) |
| get (or push back) characters from | curses terminal keyboard /ungetch: | curs_getch(3X) |
| mvwgetnstr: get character strings from | curses terminal keyboard /mvwgetstr, | curs_getstr(3X) |
| (or push back) wchar_t characters from | curses terminal keyboard /ungetwch: get | curs_getwch(3X) |
| get wchar_t character strings from | curses terminal keyboard /mvwgetnwstr: | curs_getwstr(3X) |
| /wsetscrreg, scrollok, nl, nonl: | curses terminal output option control/ | curs_outopts(3X) |
| delay_output, flushinp: miscellaneous | curses utility routines /putwin, getwin, | curs_util(3X) |
| vwscanw: convert formatted input from a | curses widow /wscanw, mvscanw, mvwscanw, | curs_scanw(3X) |
| /add a string of characters to a | curses window and advance cursor | curs_addstr(3X) |
| /bkgdset, wbkgdset, bkgd, wbkgd: | curses window background manipulation/ | curs_bkgd(3X) |
| add a character (with attributes) to a | curses window /echochar, wechochar: | curs_addch(3X) |
| of characters (and attributes) to a | curses window /mvwaddchnstr: add string | curs_addchst(3X) |
| of characters (and attributes) to a | curses window /mvwaddchnstr: add string | curs_addchstr(3X) |
| wechowchar: add a wchar_t character to a | curses window /mvwaddwch, echowchar, | curs_addwch(3X) |
| add string of wchar_t characters to a | curses window /mvwaddwchnstr: | curs_addwchstr(3X) |
| add a string of wchar_t characters to a | curses window /mvwaddwstr, mvwaddnwstr: | curs_addwstr(3X) |
| wclrtoeol: clear all or part of a | curses window /wclrtobot, clrtoeol, | curs_clear(3X) |
| delete character under cursor in a | curses window. /mvdelch, mvwdelch: | curs_delch(3X) |
| delete and insert lines in a | curses window /insertln, winsertln: | curs_deleteln(3X) |
| a character and its attributes from a | curses window /mvinch, mvwinch: get | curs_inch(3X) |
| of characters (and attributes) from a | curses window /mvwinchnstr: get a string | curs_inchstr(3X) |
| the character under the cursor in a | curses window /insert a character before | curs_insch(3X) |
| before character under the cursor in a | curses window /mvwinsnstr: insert string | curs_insstr(3X) |
| get a string of characters from a | curses window /mvwinstr, mvwinnstr: | curs_instr(3X) |
| the character under the cursor in a | curses window /wchar_t character before | curs_inswch(3X) |
| before character under the cursor in a | curses window /insert wchar_t string | curs_inswstr(3X) |
| mvwinwch: get a wchar_t character from a | curses window /inwch, winwch, mvinwch, | curs_inwch(3X) |
| a string of wchar_t characters from a | curses window /mvwinwchnstr: get | curs_inwchstr(3X) |
| a string of wchar_t characters from a | curses window /mvwinnwstr: get | curs_inwstr(3X) |
| curs_move: move, wmove: move | curses window cursor | curs_move(3X) |
| scroll, srcl, wscrl: scroll a | curses window /curs_scroll: | curs_scroll(3X) |
| doupdate, redrawwin, wredrawln: refresh | curses windows and lines /wnoutrefresh, | curs_refresh(3X) |

| | | |
|---|---|---|
| /dg_unbuffered_read: synchronously read | data from a file without system/ | dg_unbuffered_read(2) |
| sputl, sgetl: access long integer | data in a machine-independent fashion | sputl(3X) |
| /t_snd: send | data or expedited data over a connection | t_snd(3N) |
| connection t_rcv: receive | data or expedited data sent over a | t_rcv(3N) |
| t_snd: send data or expedited | data over a connection | t_snd(3N) |
| nlsgetcall: get client's | data passed via the listener | nlsgetcall(3N) |
| prof: display profile | data | prof(1) |
| library routines for external | data representation /xdr_wrapstring: | xdr(3N) |
| system call dg_stat: | data returned by dg_stat and dg_fstat | dg_stat(5) |
| stat: | data returned by stat system call | stat(5) |
| call dg_mknod: | data returned by the dg_mknod system | dg_mknod(5) |
| /statfs: | data returned by the statfs system call | statfs(5) |
| /ustat: | data returned by the ustat system call | ustat(5) |
| brk: change | data segment space allocation | brk(2) |
| sbrk: change | data segment space allocation | sbrk(2) |
| t_rcv: receive data or expedited | data sent over a connection | t_rcv(3N) |
| plock: lock | data, text, or both into memory | plock(2) |
| /dg_unbuffered_write: synchronously write | data to a file without system buffering | dg_unbuffered_write(2) |
| elf32_xlatetom: class-dependent | data translation /elf32_xlatetof, | elf_xlate(3E) |
| tkey: set label and | data translation parameters | tkey(1) |
| tcdrain, tcflush, tcflow: control | data transmission /tcsendbreak, | tcflush(3C) |
| field_type, field_arg: forms field | data type validation /set_field_type, | form_field_validation(3X) |
| nl_types: native language | data types | nl_types(5) |
| types: primitive system | data types | types(5) |
| t_rcvudata: receive a | data unit | t_rcvudata(3N) |
| t_sndudata: send a | data unit | t_sndudata(3N) |
| /panel_userptr: associate application | data with a panels panel | panel_userptr(3X) |
| field_userptr: associate application | data with forms /set_field_userptr, | form_field_userptr(3X) |
| form_userptr: associate application | data with forms /set_form_userptr, | form_userptr(3X) |
| item_userptr: associate application | data with menus items /set_item_userptr, | menu_item_userptr(3X) |
| menu_userptr: associate application | data with menus /set_menu_userptr, | menu_userptr(3X) |
| field has off-screen data/ /form_data: | data_ahead, data_behind: tell if forms | form_data(3X) |
| mail alias information in the aliases | database /admalias: manage | admalias(1M) |
| admether: manage ether | database | admether(1M) |
| manage group information in the group | database /admgroup: | admgroup(1M) |
| admhost: manage hosts | database | admhost(1M) |
| manage the TCP/IP network interfaces | database /admipinterface: | admipinterface(1M) |
| admnetwork: manage network | database | admnetwork(1M) |
| resolver's domain name and nameservers | database /admresolve: manage DNS | admresolve(1M) |
| admservice: manage service | database | admservice(1M) |
| manage the SNMP community | database /admsnmpcommunity: | admsnmpcommunity(1M) |
| /admsnmpobject: manage the snmpd object | database | admsnmpobject(1M) |
| /admsnmptrap: manage the SNMP traps | database | admsnmptrap(1M) |
| /admtrustedhost: manage the trusted hosts | database | admtrustedhost(1M) |
| manage user information in the password | database /admuser: | admuser(1M) |
| colltbl: create collation | database | colltbl(1M) |
| tigetstr: curses interfaces to terminfo | database /mvcur, tigetflag, tigetnum, | curs_terminfo(3X) |
| /getnetconfig: get network configuration | database entry | getnetconfig(3N) |
| make changes to the help facility | database /helpadm: | helpadm(1M) |
| add a file to the software installation | database /installf: | installf(1M) |
| montbl: create monetary | database | montbl(1M) |
| netconfig: network configuration | database | netconfig(4) |
| join: relational | database operator | join(1) |
| removef: remove a file from software | database | removef(1M) |
| /dg_lock_reset: reset remote file lock | database, start lock reclaim grace/ | dg_lock_reset(2) |
| terminal and printer capability | database /terminfo: | terminfo(4) |
| initialize a terminal or query terminfo | database /tput: | tput(1) |
| admroute: manage routing | databases | admroute(1M) |
| order for /etc/hosts, NIS, and DNS | databases /admsvcorder: manage search | admsvcorder(1M) |
| off-screen data/ form_data: data_ahead, | data_behind: tell if forms field has | form_data(3X) |
| ftime: get | date and time | ftime(3C) |
| getdate_err: convert user format | date and time /getdate, | getdate(3C) |
| /gettimeofday: get | date and time | gettimeofday(2) |
| /settimeofday: set | date and time | settimeofday(2) |
| gmtime, asctime, tzset: convert | date and time to string /localtime, | ctime(3C) |
| strftime, cftime, ascftime: convert | date and time to string | strftime(3C) |
| valdate: prompt for and validate a | date /ckdate, errdate, helpdate, | ckdate(1) |
| date: print and set the | date | date(1) |
| | date: print and set the date | date(1) |
| admdate: manipulate the system | date, time and time zone | admdate(1M) |
| a prompt; verify and return a time of | day /cktime: display | cktime(1) |

| | | |
|---|---|---|
| used to distinguish prime and non-prime | days /holidays: accounting information | holidays(4) |
| /dbm_firstkey, dbm_nextkey, dbm_error, | dbm_clearerr: data base subroutines | ndbm(3C) |
| dbm_delete, dbm_firstkey,/ dbm_open, | dbm_close, dbm_fetch, dbm_store, | ndbm(3C) |
| /dbm_close, dbm_fetch, dbm_store, | dbm_delete, dbm_firstkey, dbm_nextkey,/ | ndbm(3C) |
| /dbm_delete, dbm_firstkey, dbm_nextkey, | dbm_error, dbm_clearerr: data base/ | ndbm(3C) |
| dbm_firstkey,/ dbm_open, dbm_close, | dbm_fetch, dbm_store, dbm_delete, | ndbm(3C) |
| /dbm_fetch, dbm_store, dbm_delete, | dbm_firstkey, dbm_nextkey, dbm_error,/ | ndbm(3C) |
| nextkey: data base subroutines | dbminit, fetch, store, delete, firstkey, | dbm(3X) |
| /dbm_store, dbm_delete, dbm_firstkey, | dbm_nextkey, dbm_error, dbm_clearerr:/ | ndbm(3C) |
| dbm_store, dbm_delete, dbm_firstkey,/ | dbm_open, dbm_close, dbm_fetch, | ndbm(3C) |
| dbm_open, dbm_close, dbm_fetch, | dbm_store, dbm_delete, dbm_firstkey,/ | ndbm(3C) |
| | dbx: source level debugger | dbx(1) |
| | dc: desk calculator | dc(1) |
| | dd: convert and copy a file | dd(1) |
| | deblock: change blocking size | deblock(1) |
| ctrace: trace a C program to | debug it | ctrace(1) |
| dbx: source level | debugger | dbx(1) |
| dg_fsdb: file system | debugger | dg_fsdb(1M) |
| fsdb: file system | debugger | fsdb(1M) |
| sdb: symbolic | debugger | sdb(1) |
| syacdb: syac | debugger utility program | syacdb(1M) |
| legend: | Debugging information technology | legend(5) |
| Uutry: try to contact remote system with | debugging on | uutry(1M) |
| hide_panel, panel_hidden: panels | deck manipulation routines /show_panel, | panel_show(3X) |
| top_panel, bottom_panel: panels | deck manipulation routines /panel_top: | panel_top(3X) |
| panel_above, panel_below: panels | deck traversal primitives /panel_above: | panel_above(3X) |
| client/server/ /dg_decryptsessionkey: | decrypt conversation key with the | dg_decryptsessionkey(2) |
| kill: terminate a process by | default | kill(1) |
| admtape: manipulate the | default parameters for tapes | admtape(1M) |
| provide an interface to named | default sets /admdefault: | admdefault(1M) |
| timezone: set | default system time zone and locale | timezone(4) |
| default-gcc: set or query | default version of GNU C | default-gcc(1) |
| version of GNU C | default-gcc: set or query default | default-gcc(1) |
| groupdel: delete a group | definition from the system | groupdel(1M) |
| groupadd: add (create) a new group | definition on the system | groupadd(1M) |
| groupmod: modify a group | definition on the system | groupmod(1M) |
| sysdef: output system | definition | sysdef(1M) |
| testlocale: test locale | definition | testlocale(1M) |
| /glossary: | definitions of common terms and symbols | glossary(1) |
| reset_prog_mode,/ /curs_kernel: | def_prog_mode, def_shell_mode, | curs_kernel(3X) |
| /curs_kernel: def_prog_mode, | def_shell_mode, reset_prog_mode,/ | curs_kernel(3X) |
| /dg_lock_wait: wait for previously | delayed lock requests to complete | dg_lock_wait(2) |
| curses/ /filter, use_env, putwin, getwin, | delay_output, flushinp: miscellaneous | curs_util(3X) |
| character under cursor in a/ /curs_delch: | delch, wdelch, mvdelch, mvwdelch: delete | curs_delch(3X) |
| putp,/ /setupterm, setterm, set_curterm, | del_curterm, restartterm, tparm, tputs, | curs_terminfo(3X) |
| system groupdel: | delete a group definition from the | groupdel(1M) |
| userdel: | delete a user's login from the system | userdel(1M) |
| /winsdelln, insertln, winsertln: | delete and insert lines in a curses/ | curs_deleteln(3X) |
| /delch, wdelch, mvdelch, mvwdelch: | delete character under cursor in a/ | curs_delch(3X) |
| rm, rmdir: remove, | delete files or directories | rm(1) |
| subroutines dbminit, fetch, store, | delete, firstkey, nextkey: data base | dbm(3X) |
| winsdelln, insertln,/ /curs_deleteln: | deleteln, wdeleteln, insdelln, | curs_deleteln(3X) |
| bgets: read stream up to next | delimiter | bgets(3G) |
| basename, dirname: | deliver portions of path names | basename(1) |
| tail: | deliver the last part of a file | tail(1) |
| panel_new: new_panel, | del_panel: create and destroy panels | panel_new(3X) |
| /newterm, endwin, isendwin, set_term, | delscreen: curses screen initialization/ | curs_initscr(3X) |
| change the delta commentary of an SCCS | delta /cdc: | cdc(1) |
| delta: make a | delta (change) to an SCCS file | delta(1) |
| cdc: change the | delta commentary of an SCCS delta | cdc(1) |
| rmdel: remove a | delta from an SCCS file | rmdel(1) |
| file | delta: make a delta (change) to an SCCS | delta(1) |
| comb: combine SCCS | deltas | comb(1) |
| dupwin, wsyncup,/ /curs_window: newwin, | delwin, mvwin, subwin, derwin, mvderwin, | curs_window(3X) |
| swapon: add a swap device for | demand paging | swapon(2) |
| mesg: permit or | deny messages | mesg(1) |
| | depend: software dependencies files | depend(4) |
| depend: software | dependencies files | depend(4) |
| ldd: list dynamic | dependencies | ldd(1) |
| constructs | deroff: remove nroff/troff, tbl, and eqn | deroff(1) |
| syncok,/ /newwin, delwin, mvwin, subwin, | derwin, mvderwin, dupwin, wsyncup, | curs_window(3X) |

| | | |
|---|---|---|
| devfree: release | devices from exclusive use | devfree(1M) |
| probedev: probe system for | devices | probedev(1M) |
| dialups: | devices requiring a dial-up password. | dialups(4) |
| tcload: load terminal controller | devices | tcload(1M) |
| lists device groups which contain | devices that match criteria /getdgrp: | getdgrp(1M) |
| | devnm: device name | devnm(1M) |
| use | devreserv: reserve devices for exclusive | devreserv(1M) |
| | devtty: control terminal pseudo-device | devtty(7) |
| and inodes | df: report number of free disk blocks | df(1M) |
| | dfm: DOS file manager | dfm(4M) |
| processes attach shared descriptor/ | dg_allow_shared_descriptor_attach: let | dg_allow_shared_descriptor_attach(2) |
| another process's shared descriptor/ | dg_attach_to_shared_descriptors: attach | dg_attach_to_shared_descriptors(2) |
| /dg_seek, | dg_block_seek: extended seek functions | dg_seek(3C) |
| files /lp: | DGC AViiON family line printer special | lp(7) |
| conversation key with the client/server/ | dg_decryptsessionkey: decrypt | dg_decryptsessionkey(2) |
| functions | dg_devctl: perform device-control | dg_devctl(2) |
| conversation key with the client/server/ | dg_encryptsessionkey: encrypt | dg_encryptsessionkey(2) |
| for the current process | dg_ext_errno: return the extended errno | dg_ext_errno(2) |
| for process identified by process key | dg_file_info: get file usage information | dg_file_info(2) |
| lock on an open DG/UX file | dg_flock: apply or remove an advisory | dg_flock(3C) |
| | dg_fsdb: file system debugger | dg_fsdb(1M) |
| information | dg_fstat: get extended file status | dg_fstat(2) |
| dg_stat: data returned by dg_stat and | dg_fstat system call | dg_stat(5) |
| | dg_getrootkey: get root's secret key | dg_getrootkey(2) |
| u3b5, vax: provide truth value/ machid: | dghost, m68k, m88k, i386, pdp11, u3b, | machid(1) |
| current IPCs state | dg_ipc_info: get information about | dg_ipc_info(2) |
| | dg_kill: test for or terminate a process | dg_kill(1) |
| on a filehandle | dg_lcntl: process a record lock request | dg_lcntl(2) |
| remote lock clients | dg_lock_kill: remove locks held by | dg_lock_kill(2) |
| database, start lock reclaim grace/ | dg_lock_reset: reset remote file lock | dg_lock_reset(2) |
| delayed lock requests to complete | dg_lock_wait: wait for previously | dg_lock_wait(2) |
| | dg_mknod: create a file system node | dg_mknod(2) |
| system call | dg_mknod: data returned by the dg_mknod | dg_mknod(5) |
| dg_mknod: data returned by the | dg_mknod system call | dg_mknod(5) |
| | dg_mount: mount a file system | dg_mount(2) |
| | dg_mstat: get file status | dg_mstat(2) |
| memory pages | dg_paging_info: determine residency of | dg_paging_info(2) |
| the system's currently active processes | dg_process_info: get information about | dg_process_info(2) |
| functions | dg_seek, dg_block_seek: extended seek | dg_seek(3C) |
| limits of a control point directory | dg_set_cpd_limits: change the resource | dg_set_cpd_limits(2) |
| key in the keyserver | dg_setsecretkey: store a client's secret | dg_setsecretkey(2) |
| dg_stat: data returned by | dg_stat and dg_fstat system call | dg_stat(5) |
| dg_fstat system call | dg_stat: data returned by dg_stat and | dg_stat(5) |
| information | dg_stat: get extended file status | dg_stat(2) |
| | dg_sysctl: modify system parameters | dg_sysctl(1M) |
| and control functions | dg_sysctl: perform system configuration | dg_sysctl(2) |
| | dg_sys_info: get system information | dg_sys_info(2) |
| data from a file without system/ | dg_unbuffered_read: synchronously read | dg_unbuffered_read(2) |
| data to a file without system buffering | dg_unbuffered_write: synchronously write | dg_unbuffered_write(2) |
| ar: | DG/UX common archive file format | ar(4) |
| or remove an advisory lock on an open | DG/UX file /dg_flock: apply | dg_flock(3C) |
| hier: | DG/UX file system hierarchy | hier(5) |
| jobs: summary of | DG/UX job control facilities | jobs(3C) |
| pseudo-device syscon: | DG/UX operating system console | syscon(7) |
| crash: what to do when the | DG/UX system crashes | crash(8) |
| intro: introduction to | DG/UX System special files | intro(7) |
| admpackage: manage | DG/UX-style software packages | admpackage(1M) |
| | dg_xtrace: extended process trace | dg_xtrace(2) |
| postdaisy: PostScript translator for | Diablo 630 files | postdaisy(1) |
| line connection | dial: establish an out-going terminal | dial(3C) |
| ratfor: rational FORTRAN | dialect | ratfor(1) |
| log-in programs and passwords for | dial-up devices /d_passwd: | d_passwd(4) |
| dialups: devices requiring a | dial-up password. | dialups(4) |
| password. | dialups: devices requiring a dial-up | dialups(4) |
| bdiff: big | diff | bdiff(1) |
| | diff: differential file comparator | diff(1) |
| comparison | diff3: 3-way differential file | diff3(1) |
| sdiff: side-by-side | difference program | sdiff(1) |
| comparator berk_diff: Berkeley | differential file and directory | berk_diff(1) |
| diff: | differential file comparator | diff(1) |
| berk_diff3: Berkeley 3-way | differential file comparison | berk_diff3(1) |

| | | |
|---|---|---|
| diff3: 3-way | differential file comparison | diff3(1) |
| | dircmp: compare two directories | dircmp(1) |
| display information about files and | directories /admfsinfo: | admfsinfo(1M) |
| uucheck: check the uucp | directories and permissions file | uucheck(1M) |
| dircmp: compare two | directories | dircmp(1) |
| mkdirp, rmdirp: create, remove | directories in a path | mkdirp(3G) |
| pathfind: search for named file in named | directories | pathfind(3G) |
| rm, rmdir: remove, delete files or | directories | rm(1) |
| /exportfs: make a | directory available for mounting via NFS | exportfs(2) |
| cd: change working | directory | cd(1) |
| uucleanup: uucp spool | directory clean-up | uucleanup(1M) |
| Berkeley differential file and | directory comparator /berk_diff: | berk_diff(1) |
| allocation limits for a control point | directory /cpd: change or view the | cpd(1) |
| the resource limits of a control point | directory /dg_set_cpd_limits: change | dg_set_cpd_limits(2) |
| filesystem-independent/ getdents: get | directory entries in a | getdents(2) |
| dirent: file system independent | directory entry | dirent(4) |
| unlink: remove a | directory entry | unlink(2) |
| mkdir: create a | directory file | mkdir(2) |
| rmdir: remove a | directory file | rmdir(2) |
| chroot: change root | directory for a command | chroot(1M) |
| getcwd: get pathname of current working | directory | getcwd(3C) |
| ls: list contents of | directory | ls(1) |
| mkdir: make a | directory | mkdir(1) |
| mvdir: move a | directory | mvdir(1M) |
| dirname: report the parent | directory name of a file path name | dirname(3G) |
| pwd: print working | directory name | pwd(1) |
| chdir: change the working | directory of the calling process | chdir(2) |
| chroot: change the root | directory of the calling process | chroot(2) |
| fchdir: change the working | directory of the calling process | fchdir(2) |
| seekdir, rewinddir, closedir: directory/ | directory: opendir, readdir, telldir, | directory(3X) |
| telldir, seekdir, rewinddir, closedir: | directory operations /opendir, readdir, | directory(3X) |
| getwd: get current working | directory pathname | getwd(3C) |
| scandir, alphasort: scan a | directory | scandir(3C) |
| ttysrch: | directory search list for ttyname | ttysrch(4M) |
| directory entry | dirent: file system independent | dirent(4) |
| /basename, | dirname: deliver portions of path names | basename(1) |
| name of a file path name | dirname: report the parent directory | dirname(3G) |
| | dis: object code disassembler | dis(1) |
| t_unbind: | disable a transport endpoint | t_unbind(3N) |
| enable, | disable: enable/disable LP printers | enable(1) |
| acct: enable or | disable process accounting | acct(2) |
| dis: object code | disassembler | dis(1) |
| /connld: line | discipline for unique stream connections | connld(7) |
| terminal type, modes, speed, and line | discipline /getty: set | getty(1M) |
| ldterm: standard STREAMS terminal line | discipline module | ldterm(7) |
| /menu_items, item_count: connect and | disconnect items to and from menus | menu_items(3X) |
| t_snddis: send user-initiated | disconnect request | t_snddis(3N) |
| t_rcvdis: retrieve information from | disconnect | t_rcvdis(3N) |
| diskusg: generate | disk accounting data by user id | diskusg(1M) |
| information sync: synchronize | disk and memory resident file system | sync(2) |
| hada: AViiON family High Availability | Disk Array adapter subsystem | hada(7) |
| da: AViiON family | disk array subsystem | da(7) |
| for maintaining a High Availability | Disk Array subsystem /menu interface | gridman(1M) |
| df: report number of free | disk blocks and inodes | df(1M) |
| a file's in-core state with that on | disk /fsync: synchronize | fsync(2) |
| dsk: block special | disk interface | dsk(7) |
| rdsk: character special | disk interface | rdsk(7) |
| dkctl: control special | disk operations | dkctl(1M) |
| space: | disk space requirement file | space(4) |
| cied: AViiON family | disk subsystem | cied(7) |
| cimd: AViiON family | disk subsystem | cimd(7) |
| cird: AViiON family | disk subsystem | cird(7) |
| sd: AViiON family | disk subsystem | sd(7) |
| du: summarize | disk usage | du(1) |
| physical and logical disks | diskman: menu interface for managing | diskman(1M) |
| for managing physical and logical | disks /diskman: menu interface | diskman(1M) |
| by user id | diskusg: generate disk accounting data | diskusg(1M) |
| mount, umount: mount and | dismount filesystems | mount(1M) |
| group names | dispgid: display a list of all valid | dispgid(1) |
| /dispgid: | display a list of all valid group names | dispgid(1) |
| /dispuid: | display a list of all valid user names | dispuid(1) |

| | | |
|---|---|---|
| console fmtmsg: | display a message on stderr or system | fmtmsg(1) |
| console fmtmsg: | display a message on stderr or system | fmtmsg(3C) |
| /whatis: | display a one-line summary about a topic | whatis(1) |
| pathname ckpath: | display a prompt; verify and return a | ckpath(1) |
| string answer ckstr: | display a prompt; verify and return a | ckstr(1) |
| time of day cktime: | display a prompt; verify and return a | cktime(1) |
| integer value /ckint: | display a prompt; verify and return an | ckint(1) |
| set_menu_pad, menu_pad: control menus | display attributes /menu_grey, | menu_attributes(3X) |
| /field_pad: format the general | display attributes of forms | form_field_attributes(3X) |
| text string in, message data/ srchtxt: | display contents of, or search for a | srchtxt(1) |
| pechochar, pechowchar: create and | display curses pads /pnoutrefresh, | curs_pad(3X) |
| vedit, view: screen-oriented (visual) | display editor based on ex /vi, | vi(1) |
| uncompress, zcat: compress, expand or | display expanded files /compress, | compress(1) |
| fez: | display file element sizes | fez(1) |
| screenful at a time pg: | display file forward or backward one | pg(1) |
| more, page: | display file one screenful at a time | more(1) |
| directories admfsinfo: | display information about files and | admfsinfo(1M) |
| remote users finger: | display information about local and | finger(1) |
| settings tdisplay: | display label and record translation | tdisplay(1) |
| prof: | display profile data | prof(1) |
| /postmd: matrix | display program for PostScript printers | postmd(1) |
| pkginfo: | display software package information | pkginfo(1) |
| /admxterminal: manage serving of X | display terminals | admxterminal(1M) |
| specified times atq: | display the jobs queued to run at | atq(1) |
| systemid: | display the unique system identifier | systemid(1M) |
| pkgparam: | displays package parameter values | pkgparam(1) |
| user names | dispuid: display a list of all valid | dispuid(1) |
| hypot: Euclidean | distance function | hypot(3M) |
| holidays: accounting information used to | distinguish prime and non-prime days | holidays(4) |
| /seed48, lcong48: generate uniformly | distributed pseudo-random numbers | drand48(3C) |
| remainder | div, ldiv: compute the quotient and | div(3C) |
| | dkctl: control special disk operations | dkctl(1M) |
| postdmd: PostScript translator for | DMD bitmap files | postdmd(1) |
| res_mkquery, res_send, res_init, | dn_comp, dn_expand: make, send, and/ | resolver(3C) |
| packets to/ /res_send, res_init, dn_comp, | dn_expand: make, send, and interpret | resolver(3C) |
| search order for /etc/hosts, NIS, and | DNS databases /admsvcorder: manage | admsvcorder(1M) |
| nameservers database admresolve: manage | DNS resolver's domain name and | admresolve(1M) |
| | doconfig: execute a configuration script | doconfig(3N) |
| prctmp, prdaily,/ chargefee, ckpacct, | dodisk, lastlogin, monacct, nulladm, | acctsh(1M) |
| whodo: who is | doing what | whodo(1M) |
| /getdomainname: get name of current | domain | getdomainname(2) |
| admresolve: manage DNS resolver's | domain name and nameservers database | admresolve(1M) |
| send, and interpret packets to Internet | domain name servers /dn_expand: make, | resolver(3C) |
| /setdomainname: set name of current | domain | setdomainname(2) |
| dfm: | DOS file manager | dfm(4M) |
| strtod, atof,: convert string to | double-precision number | strtod(3C) |
| curses/ /refresh, wrefresh, wnoutrefresh, | doupdate, redrawwin, wredrawln: refresh | curs_refresh(3X) |
| putmsg, putpmsg: pass a message | down a stream | putmsg(2) |
| shutdown: shut | down part of a full-duplex connection | shutdown(2) |
| shutdown: shut | down system, change system state | shutdown(1M) |
| VSC synchronous controller vscload: | download board resident software onto | vscload(1M) |
| PostScript fonts | download: download host resident | download(1) |
| /download: | download host resident PostScript fonts | download(1) |
| for dial-up devices | d_passwd: log-in programs and passwords | d_passwd(4) |
| PostScript printers | dpost: troff postprocessor for | dpost(1) |
| mrand48, jrand48, srand48, seed48,/ | drand48, erand48, lrand48, nrand48, | drand48(3C) |
| | drem: IEEE floating-point remainder | drem(3M) |
| open any minor device on a STREAMS | driver /clone: | clone(7) |
| sad: STREAMS Administrative | Driver | sad(7) |
| ssid: Streams Synchronous Interface | Driver | ssid(7) |
| generic interface to EUC handling TTY | drivers and modules /eucioctl: | eucioctl(5) |
| | dsk: block special disk interface | dsk(7) |
| | du: summarize disk usage | du(1) |
| /duart: | Dual Asynchronous Receiver/Transmitter | duart(7) |
| Receiver/Transmitter | duart: Dual Asynchronous | duart(7) |
| dumpcycle: | dump cycle file for backups | dumpcycle(4M) |
| /admdumpcycle: manage | dump cycle tables | admdumpcycle(1M) |
| /admdumpdevice: manage the | dump device table | admdumpdevice(1M) |
| dump: incremental file system | dump | dump(1M) |
| dumpfs: | dump file system information | dumpfs(1M) |
| lsd: load a system | dump from tape | lsd(1M) |

| | | |
|---|---|---|
| | dump: incremental file system dump | dump(1M) |
| od: octal | dump | od(1) |
| archive file att_dump: | dump parts of an object or object | att_dump(1) |
| dump2label: read and write labels for | dump tapes | dump2label(1M) |
| dumptab: tape table file for | dump2 | dumptab(4) |
| | dump2: incremental file system backup | dump2(1M) |
| dump tapes | dump2label: read and write labels for | dump2label(1M) |
| | dumpcycle: dump cycle file for backups | dumpcycle(4M) |
| zdump: time zone | dumper | zdump(1M) |
| | dumpfs: dump file system information | dumpfs(1M) |
| | dumptab: tape table file for dump2 | dumptab(4) |
| | dup: duplicate an open file descriptor | dup(2) |
| onto a specific descriptor | dup2: duplicate an open file descriptor | dup2(2) |
| create and/ /form_field_new: new_field, | dup_field, link_field, free_field,: | form_field_new(3X) |
| dup: | duplicate an open file descriptor | dup(2) |
| specific descriptor /dup2: | duplicate an open file descriptor onto a | dup2(2) |
| /delwin, mvwin, subwin, derwin, mvderwin, | dupwin, wsyncup, syncok, wcursyncup,/ | curs_window(3X) |
| ldd: list | dynamic dependencies | ldd(1) |
| /form_field_info: field_info, | dynamic_field_info: get forms field/ | form_field_info(3X) |
| cut: cut out selected fields of | each line of a file | cut(1) |
| rev: reverse order of characters in | each line of file | rev(1) |
| echo: | echo arguments | echo(1) |
| | echo: echo arguments | echo(1) |
| keypad,/ /curs_inopts: cbreak, nocbreak, | echo, noecho, halfdelay, intrflush, | curs_inopts(3X) |
| (with/ /addch, waddch, mvaddch, mvwaddch, | echochar, wechochar: add a character | curs_addch(3X) |
| /addwch, waddwch, mvaddwch, mvwaddwch, | echowchar, wechowchar: add a wchar_t/ | curs_addwch(3X) |
| number to string | ecvt, fcvt, gcvt: convert floating-point | ecvt(3C) |
| | ed, red: text editor | ed(1) |
| end, etext, | edata: last locations in program | end(3C) |
| putdgrp: | edit device group table | putdgrp(1M) |
| putdev: | edit device table | putdev(1M) |
| casual users) | edit: text editor (variant of ex for | edit(1) |
| vipw: | edit the system password file | vipw(1M) |
| sact: print current SCCS file | editing activity | sact(1) |
| view: screen-oriented (visual) display | editor based on ex /vi, vedit, | vi(1) |
| ed, red: text | editor | ed(1) |
| editread: command line | editor | editread(5) |
| ex: text | editor | ex(1) |
| ld: link | editor for common object files | ld-coff(1) |
| ld: link | editor for object files | ld(1) |
| a.out: common assembler and link | editor output | a.out(4) |
| sed: stream | editor | sed(1) |
| /edit: text | editor (variant of ex for casual users) | edit(1) |
| | editread: command line editor | editread(5) |
| setgid: set the real-, | effective-, and saved-group-ids | setgid(2) |
| setregid: set the real-, | effective-, and saved-group-ids | setregid(2) |
| setreuid: set the real-, | effective-, and saved-user-ids | setreuid(2) |
| setuid: set the real-, | effective-, and saved-user-ids | setuid(2) |
| process setegid: set the | effective group id of the current | setegid(2) |
| login name or user name associated with | effective UID /cuserid: get character | cuserid(3S) |
| /seteuid: set the | effective user id of the current process | seteuid(2) |
| getegid: get the | effective-group-id | getegid(2) |
| geteuid: get the | effective-user-id | geteuid(2) |
| spawn new process in a virtual memory | efficient way /vfork: | vfork(2) |
| full regular expressions | egrep: search a file for a pattern using | egrep(1) |
| remque: remove an | element from a circular queue | remque(3C) |
| insque, remque: insert/remove | element from a queue | insque(3C) |
| basename: return the last | element of a path name | basename(3G) |
| fez: display file | element sizes | fez(1) |
| translate object file from COFF to | ELF /cof2elf: | cof2elf(1) |
| elf_update: update an | ELF descriptor | elf_update(3E) |
| | elf: object file access library | elf(3E) |
| object file type elf_fsize: | elf32_fsize: return the size of an | elf_fsize(3E) |
| class-dependent object/ /elf_getehdr: | elf32_getehdr, elf32_newehdr: retrieve | elf_getehdr(3E) |
| class-dependent program/ /elf_getphdr: | elf32_getphdr, elf32_newphdr: retrieve | elf_getphdr(3E) |
| section header /elf_getshdr: | elf32_getshdr: retrieve class-dependent | elf_getshdr(3E) |
| object file/ /elf_getehdr: elf32_getehdr, | elf32_newehdr: retrieve class-dependent | elf_getehdr(3E) |
| program/ /elf_getphdr: elf32_getphdr, | elf32_newphdr: retrieve class-dependent | elf_getphdr(3E) |
| class-dependent data/ elf_xlate: | elf32_xlatetof, elf32_xlatetom: | elf_xlate(3E) |
| translation elf_xlate: elf32_xlatetof, | elf32_xlatetom: class-dependent data | elf_xlate(3E) |
| | elf_begin: make a file descriptor | elf_begin(3E) |

| | | |
|---|---|---|
| | elf_cntl: control a file descriptor | elf_cntl(3E) |
| | elf_end: finish using an object file | elf_end(3E) |
| | elf_errmsg, elf_errno: error handling | elf_error(3E) |
| elf_errmsg, | elf_errno: error handling | elf_error(3E) |
| | elf_fill: set fill byte | elf_fill(3E) |
| elf_flagphdr, elf_flagscn,/ | elf_flagdata, elf_flagehdr, elf_flagelf, | elf_flag(3E) |
| elf_flagscn,/ /elf_flagdata, | elf_flagehdr, elf_flagelf, elf_flagphdr, | elf_flag(3E) |
| /elf_flagdata, elf_flagehdr, | elf_flagelf, elf_flagphdr, elf_flagscn,/ | elf_flag(3E) |
| /elf_flagdata, elf_flagehdr, elf_flagelf, | elf_flagphdr, elf_flagscn, elf_flagshdr:/ | elf_flag(3E) |
| /elf_flagehdr, elf_flagelf, elf_flagphdr, | elf_flagscn, elf_flagshdr: manipulate/ | elf_flag(3E) |
| /elf_flagelf, elf_flagphdr, elf_flagscn, | elf_flagshdr: manipulate flags | elf_flag(3E) |
| of an object file type | elf_fsize: elf32_fsize: return the size | elf_fsize(3E) |
| header | elf_getarhdr: retrieve archive member | elf_getarhdr(3E) |
| table | elf_getarsym: retrieve archive symbol | elf_getarsym(3E) |
| object file | elf_getbase: get the base offset for an | elf_getbase(3E) |
| get section data | elf_getdata, elf_newdata, elf_rawdata: | elf_getdata(3E) |
| elf32_newehdr: retrieve class-dependent/ | elf_getehdr: elf32_getehdr, | elf_getehdr(3E) |
| identification data | elf_getident: retrieve file | elf_getident(3E) |
| elf32_newphdr: retrieve class-dependent/ | elf_getphdr: elf32_getphdr, | elf_getphdr(3E) |
| elf_nextscn: get section information | elf_getscn, elf_ndxscn, elf_newscn, | elf_getscn(3E) |
| class-dependent section header | elf_getshdr: elf32_getshdr: retrieve | elf_getshdr(3E) |
| | elf_hash: compute hash value | elf_hash(3E) |
| | elf_kind: determine file type | elf_kind(3E) |
| section information /elf_getscn, | elf_ndxscn, elf_newscn, elf_nextscn: get | elf_getscn(3E) |
| data /elf_getdata, | elf_newdata, elf_rawdata: get section | elf_getdata(3E) |
| information elf_getscn, elf_ndxscn, | elf_newscn, elf_nextscn: get section | elf_getscn(3E) |
| access | elf_next: sequential archive member | elf_next(3E) |
| elf_getscn, elf_ndxscn, elf_newscn, | elf_nextscn: get section information | elf_getscn(3E) |
| | elf_rand: random archive member access | elf_rand(3E) |
| /elf_getdata, elf_newdata, | elf_rawdata: get section data | elf_getdata(3E) |
| contents | elf_rawfile: retrieve uninterpreted file | elf_rawfile(3E) |
| | elf_strptr: make a string pointer | elf_strptr(3E) |
| | elf_update: update an ELF descriptor | elf_update(3E) |
| application versions | elf_version: coordinate library and | elf_version(3E) |
| elf32_xlatetom: class-dependent data/ | elf_xlate: elf32_xlatetof, | elf_xlate(3E) |
| file link | elink: Environment variable sensitive | elink(5) |
| /tgetstr, tgoto, tputs: curses interfaces | (emulated) to the termcap library | curs_termcap(3X) |
| ptem: STREAMS Pseudo Terminal | Emulation module | ptem(7) |
| printers | enable, disable: enable/disable LP | enable(1) |
| acct: | enable or disable process accounting | acct(2) |
| enable, disable: | enable/disable LP printers | enable(1) |
| transmission via/ uuencode, uudecode: | encode/decode a binary file for | uuencode(1) |
| crypt: | encode/decode | crypt(1) |
| client/server/ /dg_encryptsessionkey: | encrypt conversation key with the | dg_encryptsessionkey(2) |
| crypt, setkey, | encrypt: generate encryption | crypt(3C) |
| determine whether a character buffer is | encrypted /isencrypt: | isencrypt(3G) |
| crypt, setkey, encrypt: generate | encryption | crypt(3C) |
| crypt: password and file | encryption functions | crypt(3X) |
| makekey: generate | encryption key | makekey(1) |
| program | end, etext, edata: last locations in | end(3C) |
| file system/ /addexportent, remexportent, | endexportent, getexportopt: get exported | exportent(3C) |
| entry /getfsfile, getfstype, setfsent, | endfsent: get filesystem descriptor file | getfsent(3C) |
| getgrent, getgrgid, getgrnam, setgrent, | endgrent, fgetgrent: get group file/ | getgrent(3C) |
| /gethostbyname, sethostent, | endhostent: get network host entry | gethostent(3N) |
| getmntent, setmntent, addmntent, | endmntent, hasmntopt: get file system/ | getmntent(3C) |
| getnetbyaddr, getnetbyname, setnetent, | endnetent: get network entry /getnetent, | getnetent(3N) |
| entry /getnetgrent, setnetgrent, | endnetgrent, innetgr: get network group | getnetgrent(3N) |
| socket: create an | endpoint for communication | socket(2) |
| t_bind: bind an address to a transport | endpoint | t_bind(3N) |
| t_close: close a transport | endpoint | t_close(3N) |
| look at the current event on a transport | endpoint /t_look: | t_look(3N) |
| t_open: establish a transport | endpoint | t_open(3N) |
| manage options for a transport | endpoint /t_optmgmt: | t_optmgmt(3N) |
| t_unbind: disable a transport | endpoint | t_unbind(3N) |
| /getprotobyname, setprotoent, | endprotoent: get protocol entry | getprotoent(3N) |
| getpwent, getpwuid, getpwnam, setpwent, | endpwent, setpwfile, fgetpwent:/ | getpwent(3C) |
| getrpcbyname, getrpcbynumber, setrpcent, | endrpcent: get RPC entry /getrpcent, | getrpcent(3N) |
| /getservbyname, setservent, | endservent: get service entry | getservent(3N) |
| /getspent, getspnam, setspent, | endspent, fgetspent, lckpwdf, ulckpwdf:/ | getspent(3C) |
| /getutid, getutline, pututline, setutent, | endutent, utmpname: access utmp file/ | getut(3C) |
| curses/ /curs_initscr: initscr, newterm, | endwin, isendwin, set_term, delscreen: | curs_initscr(3X) |

| | | |
|---|---|---|
| strsave, strnsave: allocate area large | enough to hold string and move string/ | strsave(3C) |
| main: | enter a C main program | main(3C) |
| nlist: get | entries from name list | nlist(3C) |
| man: locate and print | entries from the reference manuals | man(1) |
| linenum: line number | entries in a common object file | linenum(4) |
| format getdents: get directory | entries in a filesystem-independent | getdents(2) |
| logger: make | entries in the system log | logger(1) |
| /ldlinit, ldlitem: manipulate line number | entries of a common object file function | ldlread(3X) |
| /ldlseek, ldnlseek: seek to line number | entries of a section of a common object/ | ldlseek(3X) |
| /ldrseek, ldnrseek: seek to relocation | entries of a section of a common object/ | ldrseek(3X) |
| convert a TERMCAP entry into a TERMINFO | entry /captoinfo: | captoinfo(1M) |
| create a temporary version of a TERMINFO | entry /chgtinfo: | chgtinfo(1) |
| return the file handle of the export | entry containing filename /getfh: | getfh(2) |
| /getnetpath: get /etc/netconfig | entry corresponding to NETPATH component | getnetpath(3N) |
| file system independent directory | entry /dirent: | dirent(4) |
| utmp, wtmp: utmp and wtmp | entry formats | utmp(4) |
| endfsent: get filesystem descriptor file | entry /getfsfile, getfstype, setfsent, | getfsent(3C) |
| endgrent, fgetgrent: get group file | entry /getgrgid, getgrnam, setgrent, | getgrent(3C) |
| sethostent, endhostent: get network host | entry /gethostbyaddr, gethostbyname, | gethostent(3N) |
| get file system descriptor file | entry /addmntent, endmntent, hasmntopt: | getmntent(3C) |
| get network configuration database | entry /getnetconfig: | getnetconfig(3N) |
| setnetent, endnetent: get network | entry /getnetbyaddr, getnetbyname, | getnetent(3N) |
| endnetgrent, innetgr: get network group | entry /getnetgrent, setnetgrent, | getnetgrent(3N) |
| setprotoent, endprotoent: get protocol | entry /getprotobynumber, getprotobyname, | getprotoent(3N) |
| fgetpwent: manipulate password file | entry /setpwent, endpwent, setpwfile, | getpwent(3C) |
| setrpcent, endrpcent: get RPC | entry /getrpcbyname, getrpcbynumber, | getrpcent(3N) |
| setservent, endservent: get service | entry /getservbyport, getservbyname, | getservent(3N) |
| manipulate shadow password file | entry /fgetspent, lckpwdf, ulckpwdf: | getspent(3C) |
| endutent, utmpname: access utmp file | entry /getutline, pututline, setutent, | getut(3C) |
| mknod: create a file | entry in the file system | mknod(2) |
| captoinfo: convert a TERMCAP | entry into a TERMINFO entry | captoinfo(1M) |
| symbol name for object file symbol table | entry /ldgetname: retrieve | ldgetname(3X) |
| ldtbindex: compute index of symbol table | entry of an object file | ldtbindex(3X) |
| ldtbread: read an indexed symbol table | entry of an object file | ldtbread(3X) |
| putpwent: write password file | entry | putpwent(3C) |
| putspent: write shadow password file | entry | putspent(3C) |
| unlink: remove a directory | entry | unlink(2) |
| execution | env: set environment for command | env(1) |
| | environ: user environment | environ(5) |
| profile: setting up an | environment at login time | profile(4) |
| fpsetsticky: IEEE floating-point | environment control /fpgetsticky, | fpgetround(3C) |
| sdetab: software development | environment data base | sdetab(4) |
| environ: user | environment | environ(5) |
| env: set | environment for command execution | env(1) |
| getenv: return value for | environment name | getenv(3C) |
| printenv: print out the | environment | printenv(1) |
| putenv: change or add value to | environment | putenv(3C) |
| longname, termattrs, termname: curses | environment query routines /killchar, | curs_termattrs(3X) |
| sde: software development | environment | sde(5) |
| commands to reset software development | environment target /sde-target: print | sde-target(1) |
| /elink: | Environment variable sensitive file link | elink(5) |
| sde-chooser: execute | environment-sensitive tool | sde-chooser(4) |
| deroff: remove nroff/troff, tbl, and | eqn constructs | deroff(1) |
| jrand48, srand48, seed48,/ drand48, | erand48, lrand48, nrand48, mrand48, | drand48(3C) |
| /post_form, unpost_form: write or | erase forms from associated subwindows | form_post(3X) |
| /post_menu, unpost_menu: write or | erase menus from associated subwindows | menu_post(3X) |
| wclrtobot, clrtoeol,/ /curs_clear: | erase, werase, clear, wclear, clrtobot, | curs_clear(3X) |
| longname,/ /curs_termattrs: baudrate, | erasechar, has_ic, has_il, killchar, | curs_termattrs(3X) |
| complementary error function | erf, erfc: error function and | erf(3M) |
| error function /erf, | erfc: error function and complementary | erf(3M) |
| | err: error-logging interface | err(7) |
| and validate a date /ckdate, | errdate, helpdate, valdate: prompt for | ckdate(1) |
| validate a group id /ckgid, | errgid, helpgid, valgid: prompt for and | ckgid(1) |
| /dg_ext_errno: return the extended | errno for the current process | dg_ext_errno(2) |
| print an error message to standard | error /extended_perror: | extended_perror(3C) |
| function /erf, erfc: | error function and complementary error | erf(3M) |
| erfc: error function and complementary | error function /erf, | erf(3M) |
| elf_errmsg, elf_errno: | error handling | elf_error(3E) |
| t_rcvuderr: receive a unit data | error indication | t_rcvuderr(3N) |
| strclean: STREAMS | error logger cleanup program | strclean(1M) |
| strerr: STREAMS | error logger server | strerr(1M) |

| | | |
|---|---|---|
| log: interface to STREAMS | error logging and event tracing | log(7) |
| /mkstr: create an | error message file by massaging C source | mkstr(1) |
| /extended_strerror: get extended | error message string | extended_strerror(3C) |
| strerror: get | error message string | strerror(3C) |
| t_error: produce | error message | t_error(3N) |
| /extended_perror: print an | error message to standard error | extended_perror(3C) |
| perror: print system | error messages | perror(3C) |
| intro: introduction to system calls and | error numbers | intro(2) |
| matherr: | error-handling function | matherr(3M) |
| err: | error-logging interface | err(7) |
| spellin, hashcheck: find spelling | errors /spell, hashmake, | spell(1) |
| copy strings, compressing or expanding | escape codes /streadd, strcadd, strecpy: | strccpy(3G) |
| transport user t_connect: | establish a connection with another | t_connect(3N) |
| t_open: | establish a transport endpoint | t_open(3N) |
| connection dial: | establish an out-going terminal line | dial(3C) |
| setmnt: | establish mount table | setmnt(1M) |
| /admsvcorder: manage search order for | /etc/hosts, NIS, and DNS databases | admsvcorder(1M) |
| NETPATH component getnetpath: get | /etc/netconfig entry corresponding to | getnetpath(3N) |
| /end, | etext, edata: last locations in program | end(3C) |
| admether: manage | ether database | admether(1M) |
| ether_hostton,/ ethers, ether_ntoa, | ether_aton, ether_ntohost, | ethers(3N) |
| /ether_ntoa, ether_aton, ether_ntohost, | ether_hostton, ether_line: Ethernet/ | ethers(3N) |
| /ether_ntohost, ether_hostton, | ether_line: Ethernet address mapping/ | ethers(3N) |
| /ether_hostton, ether_line: | Ethernet address mapping operations | ethers(3N) |
| hken: Hawk | Ethernet interface | hken(7) |
| inen: integrated | Ethernet interface | inen(7) |
| ether_hostton, ether_line:/ /ethers, | ether_ntoa, ether_aton, ether_ntohost, | ethers(3N) |
| ethers, ether_ntoa, ether_aton, | ether_ntohost, ether_hostton,/ | ethers(3N) |
| ether_ntohost, ether_hostton,/ | ethers, ether_ntoa, ether_aton, | ethers(3N) |
| eucset: set or get | EUC code set widths | eucset(1) |
| eucioctl: generic interface to | EUC handling TTY drivers and modules | eucioctl(5) |
| handling TTY drivers and modules | eucioctl: generic interface to EUC | eucioctl(5) |
| hypot: | Euclidean distance function | hypot(3M) |
| | eucset: set or get EUC code set widths | eucset(1) |
| expr: | evaluate arguments as an expression | expr(1) |
| test: condition | evaluation command | test(1) |
| t_look: look at the current | event on a transport endpoint | t_look(3N) |
| interface to STREAMS error logging and | event tracing /log: | log(7) |
| edit: text editor (variant of | ex for casual users) | edit(1) |
| | ex: text editor | ex(1) |
| (visual) display editor based on | ex /vi, vedit, view: screen-oriented | vi(1) |
| cscope: interactively | examine a C program | cscope(1) |
| sigprocmask: | examine and change blocked signals | sigprocmask(2) |
| sigaction: | examine and change signal action | sigaction(2) |
| sigpending: | examine pending signals | sigpending(2) |
| crash: | examine system images | crash(1M) |
| lpq: | examine the spool queue | lpq(1) |
| retrieve a command description and usage | examples /usage: | usage(1) |
| devfree: release devices from | exclusive use | devfree(1M) |
| devreserv: reserve devices for | exclusive use | devreserv(1M) |
| execlp, execvp: execute a file | exec: execl, execv, execle, execve, | exec(2) |
| execvp: execute a file exec: | execl, execv, execle, execve, execlp, | exec(2) |
| a file /exec: execl, execv, | execle, execve, execlp, execvp: execute | exec(2) |
| exec: execl, execv, execle, execve, | execlp, execvp: execute a file | exec(2) |
| ldfcn: COFF | executable file access routines | ldfcn(4) |
| doconfig: | execute a configuration script | doconfig(3N) |
| execv, execle, execve, execlp, execvp: | execute a file /exec: execl, | exec(2) |
| xargs: construct argument list(s) and | execute command | xargs(1) |
| at, batch: | execute commands at a later time | at(1) |
| sde-chooser: | execute environment-sensitive tool | sde-chooser(4) |
| regcmp, regex: compile and | execute regular expression | regcmp(3G) |
| regcmp, regex: compile and | execute regular expression | regcmp(3X) |
| uuxqt: | execute remote command requests | uuxqt(1M) |
| env: set environment for command | execution | env(1) |
| sleep: suspend | execution for an interval | sleep(1) |
| sleep: suspend | execution for interval | sleep(3C) |
| monitor: prepare | execution profile | monitor(3C) |
| /profil: set up | execution time profiling for a process | profil(2) |
| uux: UNIX-to-UNIX system command | execution | uux(1) |
| execute a file /exec: execl, | execv, execle, execve, execlp, execvp: | exec(2) |
| /exec: execl, execv, execle, | execve, execlp, execvp: execute a file | exec(2) |

| | | |
|---|---|---|
| execl, execv, execle, execve, execlp, | execvp: execute a file /exec: | exec(2) |
| link, unlink: | exercise link and unlink system calls | link(1M) |
| tunefs: tune an | existing file system | tunefs(1M) |
| creat: create a new file or rewrite an | existing one | creat(2) |
| | exit, _exit: terminate process | exit(2) |
| exit, | _exit: terminate process | exit(2) |
| log10f, pow, powf, sqrt, sqrtf:/ | exp, expf, cbrt, log, logf, log10, | exp(3M) |
| pack, pcat, unpack: compress and | expand files | pack(1) |
| compress, uncompress, zcat: compress, | expand or display expanded files | compress(1) |
| zcat: compress, expand or display | expanded files /compress, uncompress, | compress(1) |
| strecpy: copy strings, compressing or | expanding escape codes /strcadd, | strccpy(3G) |
| t_snd: send data or | expedited data over a connection | t_snd(3N) |
| t_rcv: receive data or | expedited data sent over a connection | t_rcv(3N) |
| pow, powf, sqrt, sqrtf:/ exp, | expf, cbrt, log, logf, log10, log10f, | exp(3M) |
| /log10, log10f, pow, powf, sqrt, sqrtf: | exponential, logarithm, power, square/ | exp(3M) |
| getfh: return the file handle of the | export entry containing filename | getfh(2) |
| /endexportent, getexportopt: get | exported file system information | exportent(3C) |
| addexportent, remexportent,/ | exportent, getexportent, setexportent, | exportent(3C) |
| mounting via NFS | exportfs: make a directory available for | exportfs(2) |
| expression | expr: evaluate arguments as an | expr(1) |
| regexp: compile, step, advance: regular | expression compile and match routines | regexp(5) |
| regexpr: compile, step, advance: regular | expression compile and match routines | regexpr(3G) |
| regcmp: regular | expression compile | regcmp(1) |
| expr: evaluate arguments as an | expression | expr(1) |
| regex: compile and execute regular | expression /regcmp, | regcmp(3G) |
| regex: compile and execute regular | expression /regcmp, | regcmp(3X) |
| regex, re_comp, re_exec: handle regular | expressions /berk_regex, | berk_regex(3C) |
| a file for a pattern using full regular | expressions /egrep: search | egrep(1) |
| /dg_ext_errno: return the | extended errno for the current process | dg_ext_errno(2) |
| /extended_strerror: get | extended error message string | extended_strerror(3C) |
| dg_fstat: get | extended file status information | dg_fstat(2) |
| dg_stat: get | extended file status information | dg_stat(2) |
| termiox: | extended general terminal interface | termiox(7) |
| dg_xtrace: | extended process trace | dg_xtrace(2) |
| dg_seek, dg_block_seek: | extended seek functions | dg_seek(3C) |
| to standard error | extended_perror: print an error message | extended_perror(3C) |
| message string | extended_strerror: get extended error | extended_strerror(3C) |
| xdr_wrapstring: library routines for | external data representation /xdr_void, | xdr(3N) |
| implement shared strings xstr: | extract strings from C programs to | xstr(1) |
| replace with catgets calls. catexstr: | extract strings from source files, | catexstr(1) |
| fsplit: split | f77 or ratfor files | fsplit(1) |
| /ceil, ceilf, copysign, fmod, fmodf, | fabs, fabsf, rint, remainder: floor,/ | floor(3M) |
| /ceilf, copysign, fmod, fmodf, fabs, | fabsf, rint, remainder: floor, ceiling,/ | floor(3M) |
| signal: simplified software signal | facilities /berk_signal, | berk_signal(3C) |
| jobs: summary of DG/UX job control | facilities | jobs(3C) |
| ipcs: report inter-process communication | facilities status | ipcs(1) |
| helpadm: make changes to the help | facility database | helpadm(1M) |
| help: help | facility | help(1) |
| factor: | factor a number | factor(1) |
| | factor: factor a number | factor(1) |
| true, | false: provide truth values | true(1) |
| da: AViiON | family disk array subsystem | da(7) |
| cied: AViiON | family disk subsystem | cied(7) |
| cimd: AViiON | family disk subsystem | cimd(7) |
| cird: AViiON | family disk subsystem | cird(7) |
| sd: AViiON | family disk subsystem | sd(7) |
| adapter subsystem hada: AViiON | family High Availability Disk Array | hada(7) |
| controller syac: AViiON | family intelligent asynchronous | syac(7) |
| lp: DGC AViiON | family line printer special files | lp(7) |
| cisc: AViiON | family SCSI adapter subsystem | cisc(7) |
| insc: AViiON | family SCSI adapter subsystem | insc(7) |
| st: AViiON | family tape subsystem | st(7) |
| integer data in a machine-independent | fashion /sputl, sgetl: access long | sputl(3X) |
| descriptor to object in file system/ | fattach: attach STREAMS-based file | fattach(3C) |
| handle misaligned memory access | faults /misalign: | misalign(5) |
| the calling process | fchdir: change the working directory of | fchdir(2) |
| | fchmod: change mode of file | fchmod(2) |
| file | fchown: change user id and group id of a | fchown(2) |
| | fclose, fflush: close or flush a stream | fclose(3S) |
| | fcntl: file control options | fcntl(5) |
| | fcntl: file descriptor control | fcntl(2) |

| | | |
|---|---|---|
| number to string ecvt, | fcvt, gcvt: convert floating-point | ecvt(3C) |
| STREAMS-based file descriptor | fdetach: detach a name from a | fdetach(3C) |
| fopen, freopen, | fdopen: open a stream | fopen(3S) |
| inquiries ferror, | feof, clearerr, fileno: stream status | ferror(3S) |
| status inquiries | ferror, feof, clearerr, fileno: stream | ferror(3S) |
| /fetch_and_add: indivisible | fetch and add to memory location | fetch_and_add(2) |
| data base subroutines /dbminit, | fetch, store, delete, firstkey, nextkey: | dbm(3X) |
| to memory location | fetch_and_add: indivisible fetch and add | fetch_and_add(2) |
| head: give the first | few lines | head(1) |
| | fez: display file element sizes | fez(1) |
| fclose, | fflush: close or flush a stream | fclose(3S) |
| | ffs: find first set bit | ffs(3C) |
| a stream /getc, getchar, | fgetc, getw: get character or word from | getc(3S) |
| /getgrgid, getgrnam, setgrent, endgrent, | fgetgrent: get group file entry | getgrent(3C) |
| /getpwnam, setpwent, endpwent, setpwfile, | fgetpwent: manipulate password file/ | getpwent(3C) |
| gets, | fgets: get a string from a stream | gets(3S) |
| /getspent, getspnam, setspent, endspent, | fgetspent, lckpwdf, ulckpwdf: manipulate/ | getspent(3C) |
| stream getwc, getwchar, | fgetwc: get wchar_t character from a | getwc(3W) |
| stream getws, | fgetws: get a wchar_t string from a | getws(3W) |
| string | fgrep: search a file for a character | fgrep(1) |
| set_max_field: set and get forms | field attributes /field_status, | form_field_buffer(3X) |
| dynamic_field_info: get forms | field characteristics /field_info, | form_field_info(3X) |
| /field_type, field_arg: forms | field data type validation | form_field_validation(3X) |
| field_index: set forms current page and | field /set_current_field, current_field, | form_page(3X) |
| /data_ahead, data_behind: tell if forms | field has off-screen data ahead or/ | form_data(3X) |
| field_opts_off, field_opts: forms | field option routines /field_opts_on, | form_field_opts(3X) |
| validation /set_field_type, field_type, | field_arg: forms field data type | form_field_validation(3X) |
| format the/ /field_fore, set_field_back, | field_back, set_field_pad, field_pad: | form_field_attributes(3X) |
| /form_field_buffer: set_field_buffer, | field_buffer, set_field_status,/ | form_field_buffer(3X) |
| to forms /set_form_fields, form_fields, | field_count, move_field: connect fields | form_field(3X) |
| /form_field_attributes: set_field_fore, | field_fore, set_field_back, field_back,/ | form_field_attributes(3X) |
| field /set_current_field, current_field, | field_index: set forms current page and | form_page(3X) |
| forms field/ /form_field_info: | field_info, dynamic_field_info: get | form_field_info(3X) |
| assign/ /form_term, set_field_init, | field_init, set_field_term, field_term: | form_hook(3X) |
| /form_field_just: set_field_just, | field_just: format the general/ | form_field_just(3X) |
| /field_opts_on, field_opts_off, | field_opts: forms field option routines | form_field_opts(3X) |
| option/ /set_field_opts, field_opts_on, | field_opts_off, field_opts: forms field | form_field_opts(3X) |
| /form_field_opts: set_field_opts, | field_opts_on, field_opts_off,/ | form_field_opts(3X) |
| attributes/ /field_back, set_field_pad, | field_pad: format the general display | form_field_attributes(3X) |
| bufsplit: split buffer into | fields | bufsplit(3G) |
| free_field,: create and destroy forms | fields /dup_field, link_field, | form_field_new(3X) |
| cut: cut out selected | fields of each line of a file | cut(1) |
| field_count, move_field: connect | fields to forms /form_fields, | form_field(3X) |
| forms/ /field_buffer, set_field_status, | field_status, set_max_field: set and get | form_field_buffer(3X) |
| routines/ /field_init, set_field_term, | field_term: assign application-specific | form_hook(3X) |
| /form_field_validation: set_field_type, | field_type, field_arg: forms field data/ | form_field_validation(3X) |
| /link_fieldtype: forms | fieldtype routines | form_fieldtype(3X) |
| /form_field_userptr: set_field_userptr, | field_userptr: associate application/ | form_field_userptr(3X) |
| mkfifo: create a new | FIFO | mkfifo(3C) |
| mkfifo: make | FIFO special file | mkfifo(1M) |
| utime: set | file access and modification times | utime(2) |
| utimes: set | file access and modification times | utimes(2) |
| elf: object | file access library | elf(3E) |
| ldfcn: COFF executable | file access routines | ldfcn(4) |
| access: determine the accessibility of a | file | access(2) |
| berk_diff: Berkeley differential | file and directory comparator | berk_diff(1) |
| tar: tape | file archiver | tar(1) |
| cpio: copy | file archives in and out | cpio(1) |
| parts of an object or object archive | file /att_dump: dump | att_dump(1) |
| rcs: change RCS | file attributes | rcs(1) |
| mkstr: create an error message | file by massaging C source | mkstr(1) |
| chmod: change mode of | file | chmod(2) |
| lchown: change user id and group id of a | file /chown, | chown(2) |
| diff: differential | file comparator | diff(1) |
| berk_diff3: Berkeley 3-way differential | file comparison | berk_diff3(1) |
| diff3: 3-way differential | file comparison | diff3(1) |
| compver: compatible versions | file | compver(4) |
| /elf_rawfile: retrieve uninterpreted | file contents | elf_rawfile(3E) |
| fcntl: | file control options | fcntl(5) |
| uuto, uupick: public UNIX-to-UNIX system | file copy | uuto(1) |
| copyright: copyright information | file | copyright(4) |

| | | |
|---|---|---|
| core: format of core image | file | core(4) |
| cprs: compress a common object | file | cprs(1) |
| umask: set and get | file creation mask | umask(2) |
| crontab: user crontab | file | crontab(1) |
| ctags: create a tags | file | ctags(1) |
| read (write) a curses screen from (to) a | file /scr_restore, scr_init, scr_set: | curs_scr_dump(3X) |
| out selected fields of each line of a | file /cut: cut | cut(1) |
| dd: convert and copy a | file | dd(1) |
| delta: make a delta (change) to an SCCS | file | delta(1) |
| close: close an object associated with a | file descriptor | close(2) |
| fcntl: | file descriptor control | fcntl(2) |
| dup: duplicate an open | file descriptor | dup(2) |
| elf_begin: make a | file descriptor | elf_begin(3E) |
| elf_cntl: control a | file descriptor | elf_cntl(3E) |
| detach a name from a STREAMS-based | file descriptor /fdetach: | fdetach(3C) |
| isastream: test a | file descriptor | isastream(3C) |
| descriptor dup2: duplicate an open | file descriptor onto a specific | dup2(2) |
| name/ /fattach: attach STREAMS-based | file descriptor to object in file system | fattach(3C) |
| | file: determine file type | file(1) |
| remove an advisory lock on an open DG/UX | file /dg_flock: apply or | dg_flock(3C) |
| sact: print current SCCS | file editing activity | sact(1) |
| fez: display | file element sizes | fez(1) |
| elf_end: finish using an object | file | elf_end(3E) |
| get the base offset for an object | file /elf_getbase: | elf_getbase(3E) |
| crypt: password and | file encryption functions | crypt(3X) |
| endfsent: get filesystem descriptor | file entry /getfstype, setfsent, | getfsent(3C) |
| setgrent, endgrent, fgetgrent: get group | file entry /getgrgid, getgrnam, | getgrent(3C) |
| hasmntopt: get file system descriptor | file entry /addmntent, endmntent, | getmntent(3C) |
| fgetpwent: manipulate password | file entry /endpwent, setpwfile, | getpwent(3C) |
| ulckpwdf: manipulate shadow password | file entry /fgetspent, lckpwdf, | getspent(3C) |
| endutent, utmpname: access utmp | file entry /pututline, setutent, | getut(3C) |
| mknod: create a | file entry in the file system | mknod(2) |
| putpwent: write password | file entry | putpwent(3C) |
| putspent: write shadow password | file entry | putspent(3C) |
| execve, execlp, execvp: execute a | file /exec: execl, execv, execle, | exec(2) |
| fchmod: change mode of | file | fchmod(2) |
| fchown: change user id and group id of a | file | fchown(2) |
| dumptab: tape table | file for dump2 | dumptab(4) |
| fgrep: search a | file for a character string | fgrep(1) |
| grep: search a | file for a pattern | grep(1) |
| expressions egrep: search a | file for a pattern using full regular | egrep(1) |
| dumpcycle: dump cycle | file for backups | dumpcycle(4M) |
| which: locate a program | file for csh(1) users | which(1) |
| constants limits: header | file for implementation-specific | limits(4) |
| ldopen, ldaopen: open an object | file for reading | ldopen(3X) |
| open: open | file for reading or writing | open(2) |
| syslog.conf: configuration | file for syslogd system log server | syslog.conf(5) |
| /uudecode: encode/decode a binary | file for transmission via mail | uuencode(1) |
| acct: per-process accounting | file format | acct(4) |
| ar: DG/UX common archive | file format | ar(4) |
| tar: tape archive | file format | tar(5) |
| intro: introduction to | file formats | intro(4) |
| intro: introduction to | file formats | intro(4M) |
| at a time /pg: display | file forward or backward one screenful | pg(1) |
| cof2elf: translate object | file from COFF to ELF | cof2elf(1) |
| sccstorcs: build RCS | file from SCCS file | sccstorcs(1) |
| removef: remove a | file from software database | removef(1M) |
| ftruncate: truncate a | file | ftruncate(2) |
| line number entries of a common object | file function /ldlitem: manipulate | ldlread(3X) |
| get: check out a version of an SCCS | file | get(1) |
| group: group | file | group(4) |
| containing filename getfh: return the | file handle of the export entry | getfh(2) |
| retrieve class-dependent object | file header /elf32_newehdr: | elf_getehdr(3E) |
| filehdr: | file header for common object files | filehdr(4) |
| ldfhread: read the | file header of a common object file | ldfhread(3X) |
| ldohseek: seek to the optional | file header of an object file | ldohseek(3X) |
| /elf_getident: retrieve | file identification data | elf_getident(3E) |
| pathfind: search for named | file in named directories | pathfind(3G) |
| copylist: copy a | file into memory | copylist(3G) |
| split: split a | file into pieces | split(1) |
| issue: issue identification | file | issue(4) |

| | | |
|---|---|---|
| header of a member of a COFF archive | file /ldahread: read the archive | ldahread(3X) |
| ldclose, ldaclose: close a common object | file | ldclose(3X) |
| read the file header of a common object | file /ldfhread: | ldfhread(3X) |
| entries of a section of a common object | file /ldnlseek: seek to line number | ldlseek(3X) |
| to the optional file header of an object | file /ldohseek: seek | ldohseek(3X) |
| entries of a section of a common object | file /ldnrseek: seek to relocation | ldrseek(3X) |
| section header of a common object | file /ldnshread: read an indexed/named | ldshread(3X) |
| section of a common object | file /seek to an indexed/named | ldsseek(3X) |
| index of symbol table entry of an object | file /ldtbindex: compute | ldtbindex(3X) |
| indexed symbol table entry of an object | file /ldtbread: read an | ldtbread(3X) |
| seek to the symbol table of an object | file /ldtbseek: | ldtbseek(3X) |
| line number entries in a common object | file /linenum: | linenum(4) |
| elink: Environment variable sensitive | file link | elink(5) |
| link: create a new link to a | file | link(2) |
| grace/ /dg_lock_reset: reset remote | file lock database, start lock reclaim | dg_lock_reset(2) |
| dfm: DOS | file manager | dfm(4M) |
| hfm: high sierra | file manager | hfm(4) |
| master: format of a master | file | master(4) |
| the comment section of an object | file. /mcs: manipulate | mcs(1) |
| merge: three-way | file merge | merge(1) |
| mkdir: create a directory | file | mkdir(2) |
| mkfifo: make FIFO special | file | mkfifo(1M) |
| mknod: build a special | file | mknod(1M) |
| chmod: change | file mode | chmod(1) |
| ctermid: generate | file name for terminal | ctermid(3S) |
| mkstemp: make a unique | file name | mkstemp(3C) |
| mktemp: make a unique | file name | mktemp(3C) |
| realpath: returns the real | file name | realpath(3C) |
| newform: change the format of a text | file | newform(1) |
| nm: print name list of common object | file | nm(1) |
| inode: | file node structure | inode(4) |
| null: the null | file | null(7) |
| ttyslot: find the slot in the utmp | file of the current user | ttyslot(3C) |
| more, page: display | file one screenful at a time | more(1) |
| fuser: identify processes using a | file or file structure | fuser(1M) |
| creat: create a new | file or rewrite an existing one | creat(2) |
| passwd: password | file | passwd(4) |
| report the parent directory name of a | file path name /dirname: | dirname(3G) |
| pkginfo: package characteristics | file | pkginfo(4) |
| pkgmap: package contents description | file | pkgmap(4) |
| pkgproto: generate a prototype | file | pkgproto(1) |
| fseek, rewind, ftell: reposition a | file pointer in a stream | fseek(3S) |
| reverse the page order in a PostScript | file /postreverse: | postreverse(1) |
| prototype: package information | file | prototype(4) |
| prs: print an SCCS | file | prs(1) |
| pwck, grpck: check password or group | file | pwck(1M) |
| rcsfile: format of RCS | file | rcsfile(4) |
| readv: read from | file | readv(2) |
| information for a common object | file /reloc: relocation | reloc(4) |
| remove: remove | file | remove(3C) |
| rename: change the name of a | file | rename(2) |
| order of characters in each line of | file /rev: reverse | rev(1) |
| rmdel: remove a delta from an SCCS | file | rmdel(1) |
| rmdir: remove a directory | file | rmdir(2) |
| bfs: big | file scanner | bfs(1) |
| compare two versions of an SCCS | file /sccsdiff: | sccsdiff(1) |
| sccsfile: format of SCCS | file | sccsfile(4) |
| sccstorcs: build RCS file from SCCS | file | sccstorcs(1) |
| scr_dump: format of curses screen image | file | scr_dump(4) |
| space: disk space requirement | file | space(4) |
| dg_mstat: get | file status | dg_mstat(2) |
| fstat: get | file status | fstat(2) |
| dg_fstat: get extended | file status information | dg_fstat(2) |
| dg_stat: get extended | file status information | dg_stat(2) |
| lstat: get | file status | lstat(2) |
| stat: get | file status | stat(2) |
| strings in an object or other binary | file /strings: find the printable | strings(1) |
| information from an object | file /strip: strip non-executable | strip(1) |
| identify processes using a file or | file structure /fuser: | fuser(1M) |
| sum: print checksum and block count of a | file | sum(1) |
| retrieve symbol name for object | file symbol table entry /ldgetname: | ldgetname(3X) |

| | | |
|---|---|---|
| syms: common object | file symbol table format | syms(4) |
| symlink: create a symbolic link | file | symlink(2) |
| dump2: incremental | file system backup | dump2(1M) |
| filesave, tapesave: daily/weekly | file system backup | filesave(1M) |
| dg_fsdb: | file system debugger | dg_fsdb(1M) |
| fsdb: | file system debugger | fsdb(1M) |
| /addmntent, endmntent, hasmntopt: get | file system descriptor file entry | getmntent(3C) |
| ustat: get | file system device statistics | ustat(2) |
| umount: remove a | file system device | umount(2) |
| dg_mount: mount a | file system | dg_mount(2) |
| dump: incremental | file system dump | dump(1M) |
| fs: | file system format | fs(4) |
| fstatfs: get information about a mounted | file system | fstatfs(2) |
| fstatvfs: return information about a | file system | fstatvfs(2) |
| hier: DG/UX | file system hierarchy | hier(5) |
| /dirent: | file system independent directory entry | dirent(4) |
| dumpfs: dump | file system information | dumpfs(1M) |
| endexportent, getexportopt: get exported | file system information /remexportent, | exportent(3C) |
| synchronize disk and memory resident | file system information /sync: | sync(2) |
| mfs: memory | file system | mfs(4) |
| mkfs, newfs: create a | file system | mkfs(1M) |
| mknod: create a file entry in the | file system | mknod(2) |
| mount: mount a | file system | mount(2) |
| file descriptor to object in | file system name space /STREAMS-based | fattach(3C) |
| dg_mknod: create a | file system node | dg_mknod(2) |
| filesystem: | file system organization | filesystem(7) |
| restore: incrementally restore a | file system | restore(1M) |
| statfs: get information about a mounted | file system | statfs(2) |
| statvfs: return information about a | file system | statvfs(2) |
| mnttab: mounted | file system table | mnttab(4) |
| tunefs: tune an existing | file system | tunefs(1M) |
| sysfs: returns information about | file system types | sysfs(2) |
| system: format of a kernel description | file | system(4) |
| admbackup: manage backup and recovery of | file systems | admbackup(1M) |
| /admfilesystem: manage | file systems | admfilesystem(1M) |
| them /fsck: check | file systems for consistency and repair | fsck(1M) |
| fstab: static information about | file systems | fstab(4) |
| ncheck checklist: list of | file systems processed by fsck and | checklist(4) |
| volcopy, labelit: copy | file systems with label checking | volcopy(1M) |
| tail: deliver the last part of a | file | tail(1) |
| tmpfile: create a temporary | file | tmpfile(3S) |
| tempnam: create a name for a temporary | file /tmpnam, | tmpnam(3S) |
| truncate: truncate a | file to a specified length | truncate(2) |
| twrite: writes a | file to tape | twrite(1) |
| database installf: add a | file to the software installation | installf(1M) |
| access and modification times of a | file /touch: update | touch(1) |
| tposn: position tape to specified | file | tposn(1) |
| system uucico: | file transport program for the uucp | uucico(1M) |
| uusched: the scheduler for the uucp | file transport program | uusched(1M) |
| ftw, nftw: walk a | file tree | ftw(3C) |
| return the size of an object | file type /elf_fsize: elf32_fsize: | elf_fsize(3E) |
| elf_kind: determine | file type | elf_kind(3E) |
| file: determine | file type | file(1) |
| unget: undo a previous get of an SCCS | file | unget(1) |
| uniq: report repeated lines in a | file | uniq(1) |
| identified by process/ /dg_file_info: get | file usage information for process | dg_file_info(2) |
| termprinter: print a | file using the 40014A Terminal Server | termprinter(1) |
| the uucp directories and permissions | file /uucheck: check | uucheck(1M) |
| val: validate SCCS | file | val(1) |
| vipw: edit the system password | file | vipw(1M) |
| /synchronously read data from a | file without system buffering | dg_unbuffered_read(2) |
| /synchronously write data to a | file without system buffering | dg_unbuffered_write(2) |
| writev: write on a | file | writev(2) |
| umask: set | file-creation mode mask | umask(1) |
| process a record lock request on a | filehandle /dg_lcntl: | dg_lcntl(2) |
| files | filehdr: file header for common object | filehdr(4) |
| handle of the export entry containing | filename /getfh: return the file | getfh(2) |
| ferror, feof, clearerr, | fileno: stream status inquiries | ferror(3S) |
| search and print process accounting | file(s) /acctcom: | acctcom(1) |
| acctmerg: merge or add total accounting | files | acctmerg(1M) |
| admin: create and administer SCCS | files | admin(1) |

| | | |
|---|---|---|
| admfsinfo: display information about | files and directories | admfsinfo(1M) |
| cmp: compare two | files | cmp(1) |
| or reject lines common to two sorted | files /comm: select | comm(1) |
| compress, expand or display expanded | files /compress, uncompress, zcat: | compress(1) |
| cp: copy | files | cp(1) |
| depend: software dependencies | files | depend(4) |
| filehdr: file header for common object | files | filehdr(4) |
| find: find | files | find(1) |
| mkmsgs: create message | files for use by gettxt | mkmsgs(1) |
| frec: recover | files from a backup tape | frec(1M) |
| tread: read | file(s) from tape | tread(1) |
| fspec: format specification in text | files | fspec(4) |
| fsplit: split f77 or ratfor | files | fsplit(1) |
| ident: identify | files | ident(1) |
| /fsync: synchronize a | file's in-core state with that on disk | fsync(2) |
| postprint: translate text | files into PostScript | postprint(1) |
| introduction to DG/UX System special | files /intro: | intro(7) |
| ld: link editor for object | files | ld(1) |
| ld: link editor for common object | files | ld-coff(1) |
| ln: link | files | ln(1) |
| lockf: record locking on | files | lockf(3C) |
| DGC AViiON family line printer special | files /lp: | lp(7) |
| passmgmt: password | files management | passmgmt(1M) |
| mv: move | files | mv(1) |
| rm, rmdir: remove, delete | files or directories | rm(1) |
| pack, pcat, unpack: compress and expand | files | pack(1) |
| PostScript translator for Diablo 630 | files /postdaisy: | postdaisy(1) |
| PostScript translator for DMD bitmap | files /postdmd: | postdmd(1) |
| translator for plot(4) graphics | files /postplot: PostScript | postplot(1) |
| PostScript translator for tektronix 4014 | files /posttek: | posttek(1) |
| pr: print | files | pr(1) |
| catexstr: extract strings from source | files, replace with catgets calls. | catexstr(1) |
| messages and other information about RCS | files /rlog: print log | rlog(1) |
| size: print section sizes of object | files | size(1) |
| sort: sort and/or merge | files | sort(1) |
| /getdtablesize: return the number of open | files the current process can have | getdtablesize(2) |
| cat: concatenate and type | files to standard output | cat(1) |
| what: identify SCCS | files | what(1) |
| system backup | filesave, tapesave: daily/weekly file | filesave(1M) |
| /getfstype, setfsent, endfsent: get | filesystem descriptor file entry | getfsent(3C) |
| | filesystem: file system organization | filesystem(7) |
| getdents: get directory entries in a | filesystem-independent format | getdents(2) |
| mount, umount: mount and dismount | filesystems | mount(1M) |
| elf_fill: set | fill byte | elf_fill(3E) |
| implementation-defined/ sigfillset: | fill in the set of | sigfillset(2) |
| nl: line numbering | filter | nl(1) |
| col: | filter reverse line-feeds | col(1) |
| curs_util: unctrl, keyname, | filter, use_env, putwin, getwin,/ | curs_util(3X) |
| /lpfilter: administer | filters used with the LP print service | lpfilter(1M) |
| find: | find files | find(1) |
| | find: find files | find(1) |
| ffs: | find first set bit | ffs(3C) |
| ttyname, isatty: | find name of a terminal | ttyname(3C) |
| library lorder: | find ordering relation for an object | lorder(1) |
| spell, hashmake, spellin, hashcheck: | find spelling errors | spell(1) |
| or other binary file /strings: | find the printable strings in an object | strings(1) |
| current user ttyslot: | find the slot in the utmp file of the | ttyslot(3C) |
| and remote users | finger: display information about local | finger(1) |
| information server | fingerd, in.fingerd: remote user | fingerd(1M) |
| /fingerd, in. | fingerd: remote user information server | fingerd(1M) |
| elf_end: | finish using an object file | elf_end(3E) |
| type of/ isnan, isnand, isnanf, | finite, fpclass, unordered: determine | isnan(3C) |
| floating-point routines | finite, unordered, copysign: IEEE | ieeefp(3C) |
| fold: fold long lines for | finite width output device | fold(1) |
| head: give the | first few lines | head(1) |
| string index: search for the | first occurrence of a character in a | index(3C) |
| ffs: find | first set bit | ffs(3C) |
| /dbminit, fetch, store, delete, | firstkey, nextkey: data base subroutines | dbm(3X) |
| tee: pipe | fitting | tee(1) |
| elf_flagscn, elf_flagshdr: manipulate | flags /elf_flagelf, elf_flagphdr, | elf_flag(3E) |
| routines curs_beep: beep, | flash: curses bell and screen flash | curs_beep(3X) |

| | | |
|---|---|---|
| dynamic_field_info: get forms field/ | form_field_info: field_info, | form_field_info(3X) |
| field_just: format the general/ | form_field_just: set_field_just, | form_field_just(3X) |
| link_field, free_field,: create and/ | form_field_new: new_field, dup_field, | form_field_new(3X) |
| field_opts_on, field_opts_off,/ | form_field_opts: set_field_opts, | form_field_opts(3X) |
| connect/ form_field: set_form_fields, | form_fields, field_count, move_field: | form_field(3X) |
| free_fieldtype, set_fieldtype_arg,/ | form_fieldtype: new_fieldtype, | form_fieldtype(3X) |
| field_userptr: associate application/ | form_field_userptr: set_field_userptr, | form_field_userptr(3X) |
| field_type, field_arg: forms field data/ | form_field_validation: set_field_type, | form_field_validation(3X) |
| set_form_term, form_term,/ | form_hook: set_form_init, form_init, | form_hook(3X) |
| form_hook: set_form_init, | form_init, set_form_term, form_term,/ | form_hook(3X) |
| and destroy forms | form_new: new_form, free_form: create | form_new(3X) |
| forms pagination | form_new_page: set_new_page, new_page: | form_new_page(3X) |
| form_opts_off, form_opts: forms option/ | form_opts: set_form_opts, form_opts_on, | form_opts(3X) |
| /form_opts_on, form_opts_off, | form_opts: forms option routines | form_opts(3X) |
| /form_opts: set_form_opts, form_opts_on, | form_opts_off, form_opts: forms option/ | form_opts(3X) |
| forms option/ /form_opts: set_form_opts, | form_opts_on, form_opts_off, form_opts: | form_opts(3X) |
| set_current_field, current_field,/ | form_page: set_form_page, form_page, | form_page(3X) |
| form_page: set_form_page, | form_page, set_current_field,/ | form_page(3X) |
| write or erase forms from associated/ | form_post: post_form, unpost_form: | form_post(3X) |
| | forms: character based forms package | forms(3X) |
| /current_field, field_index: set | forms current page and field | form_page(3X) |
| /field_status, set_max_field: set and get | forms field attributes | form_field_buffer(3X) |
| /field_info, dynamic_field_info: get | forms field characteristics | form_field_info(3X) |
| /set_field_type, field_type, field_arg: | forms field data type validation | form_field_validation(3X) |
| behind /data_ahead, data_behind: tell if | forms field has off-screen data ahead or | form_data(3X) |
| /field_opts_off, field_opts: | forms field option routines | form_field_opts(3X) |
| free_field,: create and destroy | forms fields /dup_field, link_field, | form_field_new(3X) |
| /set_fieldtype_choice, link_fieldtype: | forms fieldtype routines | form_fieldtype(3X) |
| move_field: connect fields to | forms /form_fields, field_count, | form_field(3X) |
| format the general display attributes of | forms /set_field_pad, field_pad: | form_field_attributes(3X) |
| format the general appearance of | forms /set_field_just, field_just: | form_field_just(3X) |
| associate application data with | forms /set_field_userptr, field_userptr: | form_field_userptr(3X) |
| routines for invocation by | forms /assign application-specific | form_hook(3X) |
| new_form, free_form: create and destroy | forms /form_new: | form_new(3X) |
| associate application data with | forms /set_form_userptr, form_userptr: | form_userptr(3X) |
| /post_form, unpost_form: write or erase | forms from associated subwindows | form_post(3X) |
| form_opts_on, form_opts_off, form_opts: | forms option routines /set_form_opts, | form_opts(3X) |
| forms: character based | forms package | forms(3X) |
| /form_new_page: set_new_page, new_page: | forms pagination | form_new_page(3X) |
| /form_driver: command processor for the | forms subsystem | form_driver(3X) |
| lpforms: administer | forms used with the LP print service | lpforms(1M) |
| /set_form_sub, form_sub, scale_form: | forms window and subwindow association/ | form_win(3X) |
| /form_cursor: pos_form_cursor: position | forms window cursor | form_cursor(3X) |
| /set_form_win, form_win, set_form_sub, | form_sub, scale_form: forms window and/ | form_win(3X) |
| /set_form_init, form_init, set_form_term, | form_term, set_field_init, field_init,/ | form_hook(3X) |
| form_userptr: associate application/ | form_userptr: set_form_userptr, | form_userptr(3X) |
| with/ /form_userptr: set_form_userptr, | form_userptr: associate application data | form_userptr(3X) |
| set_form_sub, form_sub, scale_form:/ | form_win: set_form_win, form_win, | form_win(3X) |
| scale_form:/ form_win: set_form_win, | form_win, set_form_sub, form_sub, | form_win(3X) |
| ratfor: rational | FORTRAN dialect | ratfor(1) |
| time /pg: display file | forward or backward one screenful at a | pg(1) |
| variables pathconf, | fpathconf: get configurable pathname | pathconf(2) |
| isnan, isnand, isnanf, finite, | fpclass, unordered: determine type of/ | isnan(3C) |
| fpsetsticky:/ fpgetround, fpsetround, | fpgetmask, fpsetmask, fpgetsticky, | fpgetround(3C) |
| fpsetmask, fpgetsticky, fpsetsticky:/ | fpgetround, fpsetround, fpgetmask, | fpgetround(3C) |
| /fpsetround, fpgetmask, fpsetmask, | fpgetsticky, fpsetsticky: IEEE/ | fpgetround(3C) |
| /printf, | fprintf, sprintf: print formatted output | printf(3S) |
| /printf, | fprintf, sprintf: print formatted output | printf(3W) |
| IEEE/ fpgetround, fpsetround, fpgetmask, | fpsetmask, fpgetsticky, fpsetsticky: | fpgetround(3C) |
| fpgetsticky, fpsetsticky:/ fpgetround, | fpsetround, fpgetmask, fpsetmask, | fpgetround(3C) |
| /fpgetmask, fpsetmask, fpgetsticky, | fpsetsticky: IEEE floating-point/ | fpgetround(3C) |
| stream /putc, putchar, | fputc, putw: put character or word on a | putc(3S) |
| puts, | fputs: put a string on a stream | puts(3S) |
| stream putwc, putwchar, | fputwc: put wchar_t character on a | putwc(3W) |
| /putws, | fputws: put a wchar_t string on a stream | putws(3W) |
| state to that contained in a signal | frame /sigret: restore the process | sigret(2) |
| | fread, fwrite: binary input/output | fread(3S) |
| | frec: recover files from a backup tape | frec(1M) |
| t_free: | free a library structure | t_free(3N) |
| df: report number of | free disk blocks and inodes | df(1M) |
| mallinfo: memory allocator malloc, | free, realloc, calloc, mallopt, | malloc(3X) |

| | | |
|---|---|---|
| /diskusg: | generate disk accounting data by user id | diskusg(1M) |
| crypt, setkey, encrypt: | generate encryption | crypt(3C) |
| makekey: | generate encryption key | makekey(1) |
| ctermid: | generate file name for terminal | ctermid(3S) |
| ncheck: | generate names from i-numbers | ncheck(1M) |
| tasks lex: | generate programs for simple lexical | lex(1) |
| random, srandom, initstate, setstate: | generate random numbers better, or/ | random(3C) |
| jrand48, srand48, seed48, lcong48: | generate uniformly distributed/ /mrand48, | drand48(3C) |
| siginfo: signal | generation information | siginfo(5) |
| rand, srand: simple random-number | generator | rand(3C) |
| random numbers better, or change the | generator /initstate, setstate: generate | random(3C) |
| drivers and modules euciocti: | generic interface to EUC handling TTY | eucioctl(5) |
| /netdir_perror, netdir_sperror: | generic transport name-to-address/ | netdir(3N) |
| a curses/ /inch, winch, mvinch, mvwinch: | get a character and its attributes from | curs_inch(3X) |
| getmsg, getpmsg: | get a message from a stream | getmsg(2) |
| semget: | get a set of semaphores | semget(2) |
| gets, fgets: | get a string from a stream | gets(3S) |
| /mvinchnstr, mvwinchstr, mvwinchnstr: | get a string of characters (and/ | curs_inchstr(3X) |
| /mvinstr, mvinnstr, mvwinstr, mvwinnstr: | get a string of characters from a curses/ | curs_instr(3X) |
| /mvinwchnstr, mvwinwchstr, mvwinwchnstr: | get a string of wchar_t characters from/ | curs_inwchstr(3X) |
| a/ /mvinnwstr, mvwinwstr, mvwinnwstr: | get a string of wchar_t characters from | curs_inwstr(3X) |
| /inwch, winwch, mvinwch, mvwinwch: | get a wchar_t character from a curses/ | curs_inwch(3X) |
| getws, fgetws: | get a wchar_t string from a stream | getws(3W) |
| getcontext, setcontext: | get and set current user context | getcontext(2) |
| tcgetattr, tcsetattr: | get and set state | tcsetattr(3C) |
| /sysinfo: | get and set system information strings | sysinfo(2) |
| ulimit: | get and set user limits | ulimit(2) |
| /sysv3_cuserid: | get character login name of the user | sysv3_cuserid(3S) |
| associated with effective UID cuserid: | get character login name or user name | cuserid(3S) |
| getc, getchar, fgetc, getw: | get character or word from a stream | getc(3S) |
| /mvgetnstr, mvwgetstr, mvwgetnstr: | get character strings from curses/ | curs_getstr(3X) |
| | get: check out a version of an SCCS file | get(1) |
| listener nlsgetcall: | get client's data passed via the | nlsgetcall(3N) |
| pathconf, fpathconf: | get configurable pathname variables | pathconf(2) |
| sysconf: | get configurable system variables | sysconf(2) |
| top_row, item_index: set and | get current menus items /set_top_row, | menu_item_current(3X) |
| /getwd: | get current working directory pathname | getwd(3C) |
| /getyx, getparyx, getbegyx, getmaxyx: | get curses cursor and window coordinates | curs_getyx(3X) |
| ftime: | get date and time | ftime(3C) |
| /gettimeofday: | get date and time | gettimeofday(2) |
| filesystem-independent format getdents: | get directory entries in a | getdents(2) |
| nlist: | get entries from name list | nlist(3C) |
| strerror: | get error message string | strerror(3C) |
| to NETPATH component /getnetpath: | get /etc/netconfig entry corresponding | getnetpath(3N) |
| eucset: set or | get EUC code set widths | eucset(1) |
| /endexportent, getexportopt: | get exported file system information | exportent(3C) |
| /extended_strerror: | get extended error message string | extended_strerror(3C) |
| dg_fstat: | get extended file status information | dg_fstat(2) |
| dg_stat: | get extended file status information | dg_stat(2) |
| umask: set and | get file creation mask | umask(2) |
| dg_mstat: | get file status | dg_mstat(2) |
| fstat: | get file status | fstat(2) |
| lstat: | get file status | lstat(2) |
| stat: | get file status | stat(2) |
| /addmntent, endmntent, hasmntopt: | get file system descriptor file entry | getmntent(3C) |
| ustat: | get file system device statistics | ustat(2) |
| identified by process key /dg_file_info: | get file usage information for process | dg_file_info(2) |
| /getfstype, setfsent, endfsent: | get filesystem descriptor file entry | getfsent(3C) |
| tcgetpgrp: | get foreground process group ID | tcgetpgrp(3C) |
| /field_status, set_max_field: set and | get forms field attributes | form_field_buffer(3X) |
| /field_info, dynamic_field_info: | get forms field characteristics | form_field_info(3X) |
| getgrnam, setgrent, endgrent, fgetgrent: | get group file entry /getgrgid, | getgrent(3C) |
| system fstatfs: | get information about a mounted file | fstatfs(2) |
| system statfs: | get information about a mounted file | statfs(2) |
| /dg_ipc_info: | get information about current IPCs state | dg_ipc_info(2) |
| vtimes: | get information about resource usage | vtimes(3C) |
| utilization getrusage: | get information about resource | getrusage(2) |
| currently active/ /dg_process_info: | get information about the system's | dg_process_info(2) |
| sets getwidth: | get information of supplementary code | getwidth(3W) |
| getlogin: | get login name | getlogin(3C) |
| logname: | get login name | logname(1) |

| | | |
|---|---|---|
| /set_menu_format, menu_format: set and | get maximum numbers of rows and columns/ . . . | menu_format(3X) |
| /item_name, item_description: | get menus item name and description . . . . . . | menu_item_name(3X) |
| set_item_value, item_value: set and | get menus item values /menu_item_value: . . . | menu_item_value(3X) |
| /set_menu_pattern, menu_pattern: set and | get menus pattern match buffer . . . . . . . . . . | menu_pattern(3X) |
| msgget: | get message queue identifier . . . . . . . . . . | msgget(2) |
| getpw: | get name from UID . . . . . . . . . . . . . | getpw(3C) |
| getpeername: | get name of connected peer . . . . . . . . . . | getpeername(2) |
| /getdomainname: | get name of current domain . . . . . . . . . . . | getdomainname(2) |
| gethostname: | get name of current host . . . . . . . . . . . | gethostname(2) |
| uname, nuname: | get name of current UNIX system . . . . . . . | uname(2) |
| device ptsname: | get name of the slave pseudo-terminal . . . . . | ptsname(3C) |
| /nlsprovider: | get name of transport provider . . . . . . . . . | nlsprovider(3N) |
| /getnetconfig: | get network configuration database entry . . . . . | getnetconfig(3N) |
| getnetbyname, setnetent, endnetent: | get network entry /getnetbyaddr, . . . . . . . . | getnetent(3N) |
| setnetgrent, endnetgrent, innetgr: | get network group entry /getnetgrent, . . . . . . | getnetgrent(3N) |
| gethostbyname, sethostent, endhostent: | get network host entry /gethostbyaddr, . . . . . | gethostent(3N) |
| localeconv: | get numeric formatting information . . . . . . | localeconv(3C) |
| unget: undo a previous | get of an SCCS file . . . . . . . . . . . . | unget(1) |
| /getopt: | get option letter from argument vector . . . . . . | getopt(3C) |
| getsockopt: | get options on a socket . . . . . . . . . . . . | getsockopt(2) |
| /wgetch, mvgetch, mvwgetch, ungetch: | get (or push back) characters from/ . . . . . . | curs_getch(3X) |
| /wgetwch, mvgetwch, mvwgetwch, ungetwch: | get (or push back) wchar_t characters/ . . . . . | curs_getwch(3X) |
| destroy a message queue /msgctl: | get or set message queue attributes or . . . . . . | msgctl(2) |
| list IDs getgroups, setgroups: | get or set supplementary group access . . . . . . | getgroups(2) |
| panels/ /panel_window, replace_panel: | get or set the current window of a . . . . . . . . | panel_window(3X) |
| getitimer, setitimer: | get or set value of interval timer . . . . . . . . | getitimer(2) |
| getppid: | get parent process-id . . . . . . . . . . . . | getppid(2) |
| directory getcwd: | get pathname of current working . . . . . . . | getcwd(3C) |
| times: | get process and child process times . . . . . . | times(2) |
| getpgrp2: | get process group . . . . . . . . . . . . . . | getpgrp2(2) |
| getpgrp: | get process group ID . . . . . . . . . . . . | getpgrp(2) |
| /getpid, getpgrp, getppid, getpgid: | get process, process group, and parent/ . . . . . | getpid(2) |
| getpriority: | get process scheduling priority . . . . . . . . | getpriority(2) |
| setprotoent, endprotoent: | get protocol entry /getprotobyname, . . . . . . | getprotoent(3N) |
| information t_getinfo: | get protocol-specific service . . . . . . . . . | t_getinfo(3N) |
| rtime: | get remote time . . . . . . . . . . . . . . | rtime(3N) |
| /dg_getrootkey: | get root's secret key . . . . . . . . . . . . | dg_getrootkey(2) |
| getrpcbynumber, setrpcent, endrpcent: | get RPC entry /getrpcent, getrpcbyname, . . . . | getrpcent(3N) |
| getrpcport: | get RPC port number . . . . . . . . . . . . | getrpcport(3R) |
| /elf_getdata, elf_newdata, elf_rawdata: | get section data . . . . . . . . . . . . . . | elf_getdata(3E) |
| elf_ndxscn, elf_newscn, elf_nextscn: | get section information /elf_getscn, . . . . . . . | elf_getscn(3E) |
| getservbyname, setservent, endservent: | get service entry /getservbyport, . . . . . . . . | getservent(3N) |
| getsid: | get session ID . . . . . . . . . . . . . . . | getsid(2) |
| shmget: | get shared memory segment . . . . . . . . . . | shmget(2) |
| sigaltstack: set or | get signal alternate stack context . . . . . . . | sigaltstack(2) |
| sigstack: set and/or | get signal stack context . . . . . . . . . . . | sigstack(2) |
| getsockname: | get socket name . . . . . . . . . . . . . . | getsockname(2) |
| dg_sys_info: | get system information . . . . . . . . . . . | dg_sys_info(2) |
| time: | get system time . . . . . . . . . . . . . . | time(2) |
| /elf_getbase: | get the base offset for an object file . . . . . . . | elf_getbase(3E) |
| t_getstate: | get the current state . . . . . . . . . . . . | t_getstate(3N) |
| getegid: | get the effective-group-id . . . . . . . . . . . | getegid(2) |
| geteuid: | get the effective-user-id . . . . . . . . . . . | geteuid(2) |
| tty: | get the name of the terminal . . . . . . . . . . | tty(1) |
| getgid: | get the real-group-id . . . . . . . . . . . . | getgid(2) |
| getuid: | get the real-user-id . . . . . . . . . . . . . | getuid(2) |
| getpagesize: | get the system page size . . . . . . . . . . . | getpagesize(2) |
| gethostid: | get unique identifier of current host . . . . . . | gethostid(2) |
| getwc, getwchar, fgetwc: | get wchar_t character from a stream . . . . . . | getwc(3W) |
| /mvgetnwstr, mvwgetwstr, mvwgetnwstr: | get wchar_t character strings from/ . . . . . . . | curs_getwstr(3X) |
| and/ curs_getyx: getyx, getparyx, | getbegyx, getmaxyx: get curses cursor . . . . . | curs_getyx(3X) |
| character or word from a stream | getc, getchar, fgetc, getw: get . . . . . . . . . | getc(3S) |
| ungetch: get (or push back)/ curs_getch: | getch, wgetch, mvgetch, mvwgetch, . . . . . . | curs_getch(3X) |
| word from a stream /getc, | getchar, fgetc, getw: get character or . . . . . . | getc(3S) |
| current user context | getcontext, setcontext: get and set . . . . . . . . | getcontext(2) |
| directory | getcwd: get pathname of current working . . . . | getcwd(3C) |
| format date and time | getdate, getdate_err: convert user . . . . . . . . | getdate(3C) |
| and time getdate, | getdate_err: convert user format date . . . . . | getdate(3C) |
| filesystem-independent format | getdents: get directory entries in a . . . . . . . . | getdents(2) |
| | getdev: lists devices based on criteria . . . . . . | getdev(1M) |
| contain devices that match criteria | getdgrp: lists device groups which . . . . . . . . | getdgrp(1M) |

| | | |
|---|---|---|
| domain | getdomainname: get name of current | getdomainname(2) |
| files the current process can have | getdtablesize: return the number of open | getdtablesize(2) |
| | getegid: get the effective-group-id | getegid(2) |
| name | getenv: return value for environment | getenv(3C) |
| | geteuid: get the effective-user-id | geteuid(2) |
| addexportent, remexportent,/ exportent, | getexportent, setexportent, | exportent(3C) |
| information /remexportent, endexportent, | getexportopt: get exported file system | exportent(3C) |
| export entry containing filename | getfh: return the file handle of the | getfh(2) |
| getfstype, setfsent, endfsent: get/ | getfsent, getfsspec, getfsfile, | getfsent(3C) |
| endfsent: get/ getfsent, getfsspec, | getfsfile, getfstype, setfsent, | getfsent(3C) |
| setfsent, endfsent: get/ getfsent, | getfsspec, getfsfile, getfstype, | getfsent(3C) |
| getfsent, getfsspec, getfsfile, | getfstype, setfsent, endfsent: get/ | getfsent(3C) |
| | getgid: get the real-group-id | getgid(2) |
| endgrent, fgetgrent: get group file/ | getgrent, getgrgid, getgrnam, setgrent, | getgrent(3C) |
| fgetgrent: get group file/ /getgrent, | getgrgid, getgrnam, setgrent, endgrent, | getgrent(3C) |
| get group file/ /getgrent, getgrgid, | getgrnam, setgrent, endgrent, fgetgrent: | getgrent(3C) |
| supplementary group access list IDs | getgroups, setgroups: get or set | getgroups(2) |
| sethostent, endhostent: get/ gethostent, | gethostbyaddr, gethostbyname, | gethostent(3N) |
| get network/ /gethostent, gethostbyaddr, | gethostbyname, sethostent, endhostent: | gethostent(3N) |
| gethostbyname, sethostent, endhostent:/ | gethostent, gethostbyaddr, | gethostent(3N) |
| current host | gethostid: get unique identifier of | gethostid(2) |
| | gethostname: get name of current host | gethostname(2) |
| of interval timer | getitimer, setitimer: get or set value | getitimer(2) |
| | getlogin: get login name | getlogin(3C) |
| /curs_getyx: getyx, getparyx, getbegyx, | getmaxyx: get curses cursor and window/ | curs_getyx(3X) |
| endmntent, hasmntopt: get file system/ | getmntent, setmntent, addmntent, | getmntent(3C) |
| stream | getmsg, getpmsg: get a message from a | getmsg(2) |
| /key_gendes, key_setsecret, | get_myaddress, getnetname, netname2host,/ | rpc(3N) |
| endnetent: get network entry /getnetent, | getnetbyaddr, getnetbyname, setnetent, | getnetent(3N) |
| network entry /getnetent, getnetbyaddr, | getnetbyname, setnetent, endnetent: get | getnetent(3N) |
| database entry | getnetconfig: get network configuration | getnetconfig(3N) |
| setnetent, endnetent: get network entry | getnetent, getnetbyaddr, getnetbyname, | getnetent(3N) |
| innetgr: get network group entry | getnetgrent, setnetgrent, endnetgrent, | getnetgrent(3N) |
| /key_setsecret, get_myaddress, | getnetname, netname2host, netname2user,/ | rpc(3N) |
| corresponding to NETPATH component | getnetpath: get /etc/netconfig entry | getnetpath(3N) |
| mvgetnstr,/ /curs_getstr: getstr, | getnstr, wgetstr, wgetnstr, mvgetstr, | curs_getstr(3X) |
| mvgetwstr,/ /curs_getwstr: getwstr, | getnwstr, wgetwstr, wgetnwstr, | curs_getwstr(3X) |
| vector | getopt: get option letter from argument | getopt(3C) |
| | getopt: parse command options | getopt(1) |
| getopts, | getoptcvt: parse command options | getopts(1) |
| options | getopts, getoptcvt: parse command | getopts(1) |
| | getpagesize: get the system page size | getpagesize(2) |
| cursor and window/ /curs_getyx: getyx, | getparyx, getbegyx, getmaxyx: get curses | curs_getyx(3X) |
| | getpass: read a password | getpass(3C) |
| | getpeername: get name of connected peer | getpeername(2) |
| parent/ /getpid, getpgrp, getppid, | getpgid: get process, process group, and | getpid(2) |
| | getpgrp: get process group ID | getpgrp(2) |
| process group, and parent/ /getpid, | getpgrp, getppid, getpgid: get process, | getpid(2) |
| | getpgrp2: get process group | getpgrp2(2) |
| process, process group, and parent/ | getpid, getpgrp, getppid, getpgid: get | getpid(2) |
| getmsg, | getpmsg: get a message from a stream | getmsg(2) |
| | getppid: get parent process-id | getppid(2) |
| group, and parent/ /getpid, getpgrp, | getppid, getpgid: get process, process | getpid(2) |
| priority | getpriority: get process scheduling | getpriority(2) |
| /getprotoent, getprotobynumber, | getprotobyname, setprotoent,/ | getprotoent(3N) |
| setprotoent, endprotoent:/ /getprotoent, | getprotobynumber, getprotobyname, | getprotoent(3N) |
| getprotobyname, setprotoent,/ | getprotoent, getprotobynumber, | getprotoent(3N) |
| the processor status register | getpsr: return the current contents of | getpsr(2) |
| | getpw: get name from UID | getpw(3C) |
| endpwent, setpwfile, fgetpwent:/ | getpwent, getpwuid, getpwnam, setpwent, | getpwent(3C) |
| fgetpwent:/ /getpwent, getpwuid, | getpwnam, setpwent, endpwent, setpwfile, | getpwent(3C) |
| setpwfile, fgetpwent:/ /getpwent, | getpwuid, getpwnam, setpwent, endpwent, | getpwent(3C) |
| system resource consumption | getrlimit, setrlimit: control maximum | getrlimit(2) |
| endrpcent: get RPC entry /getrpcent, | getrpcbyname, getrpcbynumber, setrpcent, | getrpcent(3N) |
| get RPC entry getrpcent, getrpcbyname, | getrpcbynumber, setrpcent, endrpcent: | getrpcent(3N) |
| setrpcent, endrpcent: get RPC entry | getrpcent, getrpcbyname, getrpcbynumber, | getrpcent(3N) |
| | getrpcport: get RPC port number | getrpcport(3R) |
| resource utilization | getrusage: get information about | getrusage(2) |
| | gets, fgets: get a string from a stream | gets(3S) |
| get service/ /getservent, getservbyport, | getservbyname, setservent, endservent: | getservent(3N) |
| setservent, endservent: get/ getservent, | getservbyport, getservbyname, | getservent(3N) |

| | | |
|---|---|---|
| getservbyname, setservent, endservent:/ | getservent, getservbyport, | getservent(3N) |
| | getsid: get session ID | getsid(2) |
| | getsockname: get socket name | getsockname(2) |
| | getsockopt: get options on a socket | getsockopt(2) |
| fgetspent, lckpwdf, ulckpwdf:/ | getspent, getspnam, setspent, endspent, | getspent(3C) |
| lckpwdf, ulckpwdf: manipulate/ /getspent, | getspnam, setspent, endspent, fgetspent, | getspent(3C) |
| mvgetstr, mvgetnstr,/ /curs_getstr: | getstr, getnstr, wgetstr, wgetnstr, | curs_getstr(3X) |
| string | getsubopt: parse suboptions from a | getsubopt(3C) |
| /reset_shell_mode, resetty, savetty, | getsyx, setsyx, ripoffline, curs_set,/ | curs_kernel(3X) |
| | gettimeofday: get date and time | gettimeofday(2) |
| mkmsgs: create message files for use by | gettxt | mkmsgs(1) |
| message data base | gettxt: retrieve a text string from a | gettxt(1) |
| | gettxt: retrieve a text string | gettxt(3C) |
| and line discipline | getty: set terminal type, modes, speed, | getty(1M) |
| ct: spawn | getty to a remote terminal | ct(1) |
| | getuid: get the real-user-id | getuid(2) |
| pututline, setutent, endutent,/ | getut: getutent, getutid, getutline, | getut(3C) |
| setutent, endutent, utmpname:/ /getut: | getutent, getutid, getutline, pututline, | getut(3C) |
| endutent, utmpname:/ /getut: getutent, | getutid, getutline, pututline, setutent, | getut(3C) |
| endutent,/ getut: getutent, getutid, | getutline, pututline, setutent, | getut(3C) |
| stream getc, getchar, fgetc, | getw: get character or word from a | getc(3S) |
| character from a stream | getwc, getwchar, fgetwc: get wchar_t | getwc(3W) |
| ungetwch: get (or push/ /curs_getwch: | getwch, wgetwch, mvgetwch, mvwgetwch, | curs_getwch(3X) |
| from a stream /getwc, | getwchar, fgetwc: get wchar_t character | getwc(3W) |
| pathname | getwd: get current working directory | getwd(3C) |
| supplementary code sets | getwidth: get information of | getwidth(3W) |
| keyname, filter, use_env, putwin, | getwin, delay_output, flushinp:/ /unctrl, | curs_util(3X) |
| a stream | getws, fgetws: get a wchar_t string from | getws(3W) |
| mvgetwstr, mvgetnwstr,/ /curs_getwstr: | getwstr, getnwstr, wgetwstr, wgetnwstr, | curs_getwstr(3X) |
| curses cursor and window/ /curs_getyx: | getyx, getparyx, getbegyx, getmaxyx: get | curs_getyx(3X) |
| head: | give the first few lines | head(1) |
| gmatch: shell | global pattern matching | gmatch(3G) |
| and symbols | glossary: definitions of common terms | glossary(1) |
| | gmatch: shell global pattern matching | gmatch(3G) |
| time to string /ctime, localtime, | gmtime, asctime, tzset: convert date and | ctime(3C) |
| set or query default version of | GNU C /default-gcc: | default-gcc(1) |
| gcc: | GNU C language compiler | gcc(1) |
| setjmp, longjmp: non-local | goto | setjmp(3C) |
| sigsetjmp, siglongjmp: a non-local | goto with signal state | sigsetjmp(3C) |
| file lock database, start lock reclaim | grace period /reset remote | dg_lock_reset(2) |
| pseudo-terminal device grantpt: | grant access to the slave | grantpt(3C) |
| pseudo-terminal device | grantpt: grant access to the slave | grantpt(3C) |
| cflow: generate a C flow | graph | cflow(1) |
| PostScript translator for plot(4) | graphics files /postplot: | postplot(1) |
| grfx: AViiON series workstation | graphics processor | grfx(7) |
| | grep: search a file for a pattern | grep(1) |
| processor | grfx: AViiON series workstation graphics | grfx(7) |
| a High Availability Disk Array/ | gridman: menu interface for maintaining | gridman(1M) |
| setgroups: get or set supplementary | group access list IDs /getgroups, | getgroups(2) |
| initgroups: initialize the supplementary | group access list | initgroups(3C) |
| getppid, getpgid: get process, process | group, and parent process IDs /getpgrp, | getpid(2) |
| chown, chgrp: change owner or | group | chown(1) |
| manage group information in the | group database /admgroup: | admgroup(1M) |
| groupdel: delete a | group definition from the system | groupdel(1M) |
| groupadd: add (create) a new | group definition on the system | groupadd(1M) |
| groupmod: modify a | group definition on the system | groupmod(1M) |
| endnetgrent, innetgr: get network | group entry /getnetgrent, setnetgrent, | getnetgrent(3N) |
| setgrent, endgrent, fgetgrent: get | group file entry /getgrgid, getgrnam, | getgrent(3C) |
| group: | group file | group(4) |
| pwck, grpck: check password or | group file | pwck(1M) |
| getpgrp2: get process | group | getpgrp2(2) |
| | group: group file | group(4) |
| valgid: prompt for and validate a | group id /ckgid, errgid, helpgid, | ckgid(1) |
| setpgid: set process | group ID for job control | setpgid(2) |
| getpgrp: get process | group ID | getpgrp(2) |
| chown, lchown: change user id and | group id of a file | chown(2) |
| fchown: change user id and | group id of a file | fchown(2) |
| setegid: set the effective | group id of the current process | setegid(2) |
| setsid: create session and set process | group ID | setsid(2) |
| tcgetpgrp: get foreground process | group ID | tcgetpgrp(3C) |
| set terminal foreground process | group id /tcsetpgrp: | tcsetpgrp(3C) |

| | | |
|---|---|---|
| nice: run a command at a | higher or lower priority | nice(1) |
| | hken: Hawk Ethernet interface | hken(7) |
| /strnsave: allocate area large enough to | hold string and move string into it | strsave(3C) |
| distinguish prime and non-prime days | holidays: accounting information used to | holidays(4) |
| whline, wvline: create curses borders, | horizontal and vertical lines /box, | curs_border(3X) |
| ntohl, ntohs: convert values between | host and network byte order /htons, | byteorder(3N) |
| sethostent, endhostent: get network | host entry /gethostbyname, | gethostent(3N) |
| get unique identifier of current | host /gethostid: | gethostid(2) |
| gethostname: get name of current | host | gethostname(2) |
| /admtcpipparams: manage the TCP/IP | host parameters | admtcpipparams(1M) |
| download: download | host resident PostScript fonts | download(1) |
| set unique identifier of current | host /sethostid: | sethostid(2) |
| sethostname: set name of current | host | sethostname(2) |
| unix_ipc: piping communications within a | host | unix_ipc(6F) |
| /clnttcp_create, clntudp_create, | host2netname, key_decryptsession,/ | rpc(3N) |
| admhost: manage | hosts database | admhost(1M) |
| /admtrustedhost: manage the trusted | hosts database | admtrustedhost(1M) |
| group or services/ bcs_cat: type | hosts, networks, passwd, protocols, | bcs_cat(1M) |
| search tables | hsearch, hcreate, hdestroy: manage hash | hsearch(3C) |
| values between host and network byte/ | htonl, htons, ntohl, ntohs: convert | byteorder(3N) |
| between host and network byte/ htonl, | htons, ntohl, ntohs: convert values | byteorder(3N) |
| sttydefs: maintain line and | hunt settings for TTY ports | sttydefs(1M) |
| tanh, tanhf, asinh, acosh, atanh: | hyperbolic functions /cosh, coshf, | sinh(3M) |
| | hypot: Euclidean distance function | hypot(3M) |
| truth value/ machid: dghost, m68k, m88k, | i386, pdp11, u3b, u3b5, vax: provide | machid(1) |
| commands for reading and writing | IBM and ANSI tapes /REELexchange: | reelexchange_intro(1) |
| | iconv: code set conversion | iconv(1) |
| chown, lchown: change user | id and group id of a file | chown(2) |
| fchown: change user | id and group id of a file | fchown(2) |
| id: print the user name and | ID, and group name and ID | id(1) |
| valgid: prompt for and validate a group | id /ckgid, errgid, helpgid, | ckgid(1) |
| ckuid: prompt for and validate a user | ID | ckuid(1) |
| generate disk accounting data by user | id /diskusg: | diskusg(1M) |
| setpgid: set process group | ID for job control | setpgid(2) |
| getpgrp: get process group | ID | getpgrp(2) |
| getsid: get session | ID | getsid(2) |
| the user name and ID, and group name and | ID /id: print | id(1) |
| queue, semaphore set, or shared memory | ID /ipcrm: remove a message | ipcrm(1) |
| chown, lchown: change user id and group | id of a file | chown(2) |
| fchown: change user id and group | id of a file | fchown(2) |
| setegid: set the effective group | id of the current process | setegid(2) |
| seteuid: set the effective user | id of the current process | seteuid(2) |
| group name and ID | id: print the user name and ID, and | id(1) |
| create session and set process group | ID /setsid: | setsid(2) |
| tcgetpgrp: get foreground process group | ID | tcgetpgrp(3C) |
| set terminal foreground process group | id /tcsetpgrp: | tcsetpgrp(3C) |
| /curs_outopts: clearok, idlok, | idcok immedok, leaveok, setscrreg,/ | curs_outopts(3X) |
| | ident: identify files | ident(1) |
| /elf_getident: retrieve file | identification data | elf_getident(3E) |
| issue: issue | identification file | issue(4) |
| get file usage information for process | identified by process key /dg_file_info: | dg_file_info(2) |
| msgget: get message queue | identifier | msgget(2) |
| gethostid: get unique | identifier of current host | gethostid(2) |
| sethostid: set unique | identifier of current host | sethostid(2) |
| systemid: display the unique system | identifier | systemid(1M) |
| locate: | identify a command using keywords | locate(1) |
| ident: | identify files | ident(1) |
| structure /fuser: | identify processes using a file or file | fuser(1M) |
| what: | identify SCCS files | what(1) |
| | idi: interface description interpreter | idi(1) |
| interface description interpreter | idi_tools: tools for use with the | idi_tools(1) |
| | idl: interface description language | idl(4) |
| setscrreg,/ /curs_outopts: clearok, | idlok, idcok immedok, leaveok, | curs_outopts(3X) |
| or set supplementary group access list | IDs /getgroups, setgroups: get | getgroups(2) |
| process group, and parent process | IDs /getppid, getpgid: get process, | getpid(2) |
| /fpsetmask, fpgetsticky, fpsetsticky: | IEEE floating-point environment control | fpgetround(3C) |
| drem: | IEEE floating-point remainder | drem(3M) |
| finite, unordered, copysign: | IEEE floating-point routines | ieeefp(3C) |
| isalphanum: determine | if a character is alphanumeric | isalphanum(3C) |
| ishex: determine | if a character is hexadecimal | ishex(3C) |
| or behind /data_ahead, data_behind: tell | if forms field has off-screen data ahead | form_data(3X) |

| | | |
|---|---|---|
| /menu_item_visible: item_visible: tell | if menus item is visible | menu_item_visible(3X) |
| set the signal action of a signal to | 'ignore' /sigignore: | sigignore(2) |
| core: format of core | image file | core(4) |
| scr_dump: format of curses screen | image file | scr_dump(4) |
| crash: examine system | images | crash(1M) |
| /curs_outopts: clearok, idlok, idcok | immedok, leaveok, setscrreg, wsetscrreg,/ | curs_outopts(3X) |
| nohup: run a command | immune to hangups and quits | nohup(1) |
| xstr: extract strings from C programs to | implement shared strings | xstr(1) |
| sigfillset: fill in the set of | implementation-defined signals | sigfillset(2) |
| limits: header file for | implementation-specific constants | limits(4) |
| character and its attributes/ curs_inch: | inch, winch, mvinch, mvwinch: get a | curs_inch(3X) |
| mvinchstr,/ /curs_inchstr: inchstr, | inchnstr, winchstr, winchnstr, | curs_inchstr(3X) |
| mvinchstr, mvinchnstr,/ /curs_inchstr: | inchstr, inchnstr, winchstr, winchnstr, | curs_inchstr(3X) |
| mail_pipe: invoke recipient command for | incoming mail | mail_pipe(1M) |
| vacation: automatically respond to | incoming mail messages | vacation(1) |
| fsync: synchronize a file's | in-core state with that on disk | fsync(2) |
| dump2: | incremental file system backup | dump2(1M) |
| dump: | incremental file system dump | dump(1M) |
| restore: | incrementally restore a file system | restore(1M) |
| dirent: file system | independent directory entry | dirent(4) |
| /tgetstr, tgoto, tputs: terminal | independent operation routines | termcap(3X) |
| file /ldtbindex: compute | index of symbol table entry of an object | ldtbindex(3X) |
| of a character in a string | index: search for the first occurrence | index(3C) |
| file /ldtbread: read an | indexed symbol table entry of an object | ldtbread(3X) |
| common/ ldshread, ldnshread: read an | indexed/named section header of a | ldshread(3X) |
| object/ ldsseek, ldnsseek: seek to an | indexed/named section of a common | ldsseek(3X) |
| last: | indicate last user or terminal logins | last(1) |
| receipt of an orderly release | indication /t_rcvrel: acknowledge | t_rcvrel(3N) |
| t_rcvuderr: receive a unit data error | indication | t_rcvuderr(3N) |
| /store_conditional: | indivisible compare and swap | store_conditional(2) |
| location /fetch_and_add: | indivisible fetch and add to memory | fetch_and_add(2) |
| | inen: integrated Ethernet interface | inen(7) |
| inet_makeaddr, inet_lnaof, inet_netof:/ | inet_addr, inet_network, inet_ntoa, | inet(3N) |
| /inet_network, inet_ntoa, inet_makeaddr, | inet_lnaof, inet_netof: Internet address/ | inet(3N) |
| /inet_addr, inet_network, inet_ntoa, | inet_makeaddr, inet_lnaof, inet_netof:/ | inet(3N) |
| /inet_ntoa, inet_makeaddr, inet_lnaof, | inet_netof: Internet address/ | inet(3N) |
| inet_lnaof, inet_netof:/ /inet_addr, | inet_network, inet_ntoa, inet_makeaddr, | inet(3N) |
| inet_netof:/ inet_addr, inet_network, | inet_ntoa, inet_makeaddr, inet_lnaof, | inet(3N) |
| descriptions | infocmp: compare or print out TERMINFO | infocmp(1M) |
| fstatvfs: return | information about a file system | fstatvfs(2) |
| statvfs: return | information about a file system | statvfs(2) |
| /fstatfs: get | information about a mounted file system | fstatfs(2) |
| /statfs: get | information about a mounted file system | statfs(2) |
| dg_ipc_info: get | information about current IPCs state | dg_ipc_info(2) |
| sysfs: returns | information about file system types | sysfs(2) |
| fstab: static | information about file systems | fstab(4) |
| /admfsinfo: display | information about files and directories | admfsinfo(1M) |
| /finger: display | information about local and remote users | finger(1) |
| rlog: print log messages and other | information about RCS files | rlog(1) |
| vtimes: get | information about resource usage | vtimes(3C) |
| /getrusage: get | information about resource utilization | getrusage(2) |
| print service /lpstat: print | information about the status of the LP | lpstat(1) |
| active processes /dg_process_info: get | information about the system's currently | dg_process_info(2) |
| passwd, protocols, group or services | information /type hosts, networks, | bcs_cat(1M) |
| langinfo: language | information constants | langinfo(5) |
| dg_fstat: get extended file status | information | dg_fstat(2) |
| dg_stat: get extended file status | information | dg_stat(2) |
| dg_sys_info: get system | information | dg_sys_info(2) |
| dumpfs: dump file system | information | dumpfs(1M) |
| elf_newscn, elf_nextscn: get section | information /elf_getscn, elf_ndxscn, | elf_getscn(3E) |
| getexportopt: get exported file system | information /remexportent, endexportent, | exportent(3C) |
| copyright: copyright | information file | copyright(4) |
| prototype: package | information file | prototype(4) |
| reloc: relocation | information for a common object file | reloc(4) |
| starter: | information for beginning users | starter(1) |
| mailcnfg: initialization | information for mail and rmail | mailcnfg(4M) |
| process/ /dg_file_info: get file usage | information for process identified by | dg_file_info(2) |
| ttydefs: terminal line settings | information for ttymon | ttydefs(4M) |
| strip: strip non-executable | information from an object file | strip(1) |
| t_rcvdis: retrieve | information from disconnect | t_rcvdis(3N) |
| admalias: manage mail alias | information in the aliases database | admalias(1M) |

| | | |
|---|---|---|
| admgroup: manage group | information in the group database | admgroup(1M) |
| admuser: manage user | information in the password database | admuser(1M) |
| listusers: list user login | information | listusers(1) |
| localeconv: get numeric formatting | information | localeconv(3C) |
| logins: list user and system login | information | logins(1M) |
| /nl_langinfo: language | information | nl_langinfo(3C) |
| /getwidth: get | information of supplementary code sets | getwidth(3W) |
| usermod: modify a user's login | information on the system | usermod(1M) |
| pkginfo: display software package | information | pkginfo(1) |
| fingerd, in.fingerd: remote user | information server | fingerd(1M) |
| /yperr_string, ypprot_err: Network | Information Service client interface | ypclnt(3N) |
| siginfo: signal generation | information | siginfo(5) |
| sysinfo: get and set system | information strings | sysinfo(2) |
| disk and memory resident file system | information /sync: synchronize | sync(2) |
| legend: Debugging | information technology | legend(5) |
| t_getinfo: get protocol-specific service | information | t_getinfo(3N) |
| format and output TTY port monitor | information /ttyadm: | ttyadm(1M) |
| and non-prime days holidays: accounting | information used to distinguish prime | holidays(4) |
| inittab: script for | init | inittab(4) |
| initialization | init, telinit: process control | init(1M) |
| curs_color: start_color, init_pair, | init_color, has_colors,/ | curs_color(3X) |
| group access list | initgroups: initialize the supplementary | initgroups(3C) |
| /set_term, delscreen: curses screen | initialization and manipulation routines | curs_initscr(3X) |
| rmail /mailcnfg: | initialization information for mail and | mailcnfg(4M) |
| init, telinit: process control | initialization | init(1M) |
| tlabel: | initialize a tape with a volume label | tlabel(1) |
| database /tput: | initialize a terminal or query terminfo | tput(1) |
| access list initgroups: | initialize the supplementary group | initgroups(3C) |
| connect: | initiate a connection on a socket | connect(2) |
| taccess: | initiate access to labeled tape | taccess(1) |
| t_sndrel: | initiate an orderly release | t_sndrel(3N) |
| popen, pclose: | initiate pipe to/from a process | popen(3S) |
| curs_color: start_color, | init_pair, init_color, has_colors,/ | curs_color(3X) |
| set_term, delscreen:/ /curs_initscr: | initscr, newterm, endwin, isendwin, | curs_initscr(3X) |
| numbers better, or/ random, srandom, | initstate, setstate: generate random | random(3C) |
| | inittab: script for init | inittab(4) |
| /getnetgrent, setnetgrent, endnetgrent, | innetgr: get network group entry | getnetgrent(3N) |
| mvinnstr, mvwinstr,/ curs_instr: instr, | innstr, winstr, winnstr, mvinstr, | curs_instr(3X) |
| mvinnwstr,/ /curs_inwstr: inwstr, | innwstr, winwstr, winnwstr, mvinwstr, | curs_inwstr(3X) |
| clri: clear | inode | clri(1M) |
| | inode: file node structure | inode(4) |
| report number of free disk blocks and | inodes /df: | df(1M) |
| mvwscanw, vwscanw: convert formatted | input from a curses widow /mvscanw, | curs_scanw(3X) |
| wtimeout, typeahead: curses terminal | input option control routines /timeout, | curs_inopts(3X) |
| scanf, fscanf, sscanf: convert formatted | input | scanf(3S) |
| scanf, fscanf, sscanf: convert formatted | input | scanf(3W) |
| ungetc: push character back onto | input stream | ungetc(3S) |
| push wchar_t character back into | input stream /ungetwc: | ungetwc(3W) |
| /vfscanf, vsscanf: convert formatted | input using varargs argument list | vscanf(3S) |
| fread, fwrite: binary | input/output | fread(3S) |
| poll: | input/output multiplexing | poll(2) |
| stdio: standard buffered | input/output package | stdio(3S) |
| feof, clearerr, fileno: stream status | inquiries /ferror, | ferror(3S) |
| uustat: uucp status | inquiry and job control | uustat(1) |
| subsystem | insc: AViiON family SCSI adapter | insc(7) |
| a character before the/ /curs_insch: | insch, winsch, mvinsch, mvwinsch: insert | curs_insch(3X) |
| /curs_deleteln: deleteln, wdeleteln, | insdelln, winsdelln, insertln,/ | curs_deleteln(3X) |
| under/ /insch, winsch, mvinsch, mvwinsch: | insert a character before the character | curs_insch(3X) |
| /inswch, winswch, mvinswch, mvwinswch: | insert a wchar_t character before the/ | curs_inswch(3X) |
| /insertln, winsertln: delete and | insert lines in a curses window | curs_deleteln(3X) |
| /mvinsnstr, mvwinsstr, mvwinsnstr: | insert string before character under the/ | curs_insstr(3X) |
| /mvinsnwstr, mvwinswstr, mvwinsnwstr: | insert wchar_t string before character/ | curs_inswstr(3X) |
| lines/ /wdeleteln, insdelln, winsdelln, | insertln, winsertln: delete and insert | curs_deleteln(3X) |
| insque, remque: | insert/remove element from a queue | insque(3C) |
| mvinsnstr,/ /curs_instr: insstr, | insnstr, winsstr, winsnstr, mvinsstr, | curs_insstr(3X) |
| mvinswstr,/ /curs_instr: inswstr, | insnwstr, winswstr, winsnwstr, | curs_inswstr(3X) |
| from a queue | insque, remque: insert/remove element | insque(3C) |
| mvinsstr, mvinsnstr,/ /curs_instr: | insstr, insnstr, winsstr, winsnstr, | curs_insstr(3X) |
| install: | install commands | install(1M) |
| | install: install commands | install(1M) |
| pkgmk: produce an | installable package | pkgmk(1) |

| | | |
|---|---|---|
| enough to hold string and move string | into it /strnsave: allocate area large | strsave(3C) |
| copylist: copy a file | into memory | copylist(3G) |
| plock: lock data, text, or both | into memory | plock(2) |
| split: split a file | into pieces | split(1) |
| postprint: translate text files | into PostScript | postprint(1) |
| /nocbreak, echo, noecho, halfdelay, | intrflush, keypad, meta, nodelay,/ | curs_inopts(3X) |
| application programs | intro: introduction to commands and | intro(1) |
| application programs | intro: introduction to commands and | intro(1) |
| special files | intro: introduction to DG/UX System | intro(7) |
| | intro: introduction to file formats | intro(4) |
| | intro: introduction to file formats | intro(4M) |
| | intro: introduction to miscellany | intro(5) |
| functions | intro: introduction to network library | intro(3N) |
| libraries | intro: introduction to subroutines and | intro(3) |
| error numbers | intro: introduction to system calls and | intro(2) |
| maintenance commands and application/ | intro: introduction to system | intro(1M) |
| maintenance procedures | intro: introduction to system | intro(8) |
| programs /intro: | introduction to commands and application | intro(1) |
| programs /intro: | introduction to commands and application | intro(1) |
| files intro: | introduction to DG/UX System special | intro(7) |
| intro: | introduction to file formats | intro(4) |
| intro: | introduction to file formats | intro(4M) |
| intro: | introduction to miscellany | intro(5) |
| functions intro: | introduction to network library | intro(3N) |
| rcsintro: | introduction to RCS commands | rcsintro(1) |
| libraries intro: | introduction to subroutines and | intro(3) |
| numbers /intro: | introduction to system calls and error | intro(2) |
| commands and application/ intro: | introduction to system maintenance | intro(1M) |
| procedures intro: | introduction to system maintenance | intro(8) |
| valtools: | introduction to validation tools | valtools(1) |
| ncheck: generate names from | i-numbers | ncheck(1M) |
| /mp: madd, msub, mult, mdiv, pow, gcd, | invert, rpow, msqrt, mcmp, move, min,/ | mp(3X) |
| assign application-specific routines for | invocation by forms /field_term: | form_hook(3X) |
| /routines for automatic | invocation by menus | menu_hook(3X) |
| having a C-like syntax csh: | invoke a shell (command interpreter) | csh(1) |
| mail mail_pipe: | invoke recipient command for incoming | mail_pipe(1M) |
| wchar_t character from a/ /curs_inwch: | inwch, winwch, mvinwch, mvwinwch: get a | curs_inwch(3X) |
| mvinwchstr,/ /curs_inwchstr: inwchstr, | inwchnstr, winwchstr, winwchnstr, | curs_inwchstr(3X) |
| winwchnstr, mvinwchstr,/ /curs_inwchstr: | inwchstr, inwchnstr, winwchstr, | curs_inwchstr(3X) |
| mvinwstr, mvinnwstr,/ /curs_inwstr: | inwstr, innwstr, winwstr, winnwstr, | curs_inwstr(3X) |
| select: wait for | I/O conditions | select(2) |
| start a BIOD server for asynchronous | I/O requests /async_daemon: | async_daemon(2) |
| widec: multibyte character | I/O routines | widec(3W) |
| biod: start block | I/O servers | biod(1M) |
| streamio: STREAMS | ioctl commands | streamio(7) |
| | ioctl: control a device | ioctl(2) |
| set, or shared memory ID | ipcrm: remove a message queue, semaphore | ipcrm(1) |
| facilities status | ipcs: report inter-process communication | ipcs(1) |
| get information about current | IPCs state /dg_ipc_info: | dg_ipc_info(2) |
| /isxdigit, islower, isupper, isalpha, | isalnum, isspace, iscntrl, ispunct,/ | ctype(3C) |
| /isdigit, isxdigit, islower, isupper, | isalpha, isalnum, isspace, iscntrl,/ | ctype(3C) |
| alphanumeric | isalphanum: determine if a character is | isalphanum(3C) |
| iscntrl, ispunct, isprint, isgraph, | isascii: character handling /isspace, | ctype(3C) |
| | isastream: test a file descriptor | isastream(3C) |
| ttyname, | isatty: find name of a terminal | ttyname(3C) |
| /isupper, isalpha, isalnum, isspace, | iscntrl, ispunct, isprint, isgraph,/ | ctype(3C) |
| isalpha, isalnum, isspace,/ ctype: | isdigit, isxdigit, islower, isupper, | ctype(3C) |
| buffer is encrypted | isencrypt: determine whether a character | isencrypt(3G) |
| /curs_initscr: initscr, newterm, endwin, | isendwin, set_term, delscreen: curses/ | curs_initscr(3X) |
| /iswascii, isphonogram, isideogram, | isenglish, isnumber, isspecial: classify/ | wctype(3W) |
| /isspace, iscntrl, ispunct, isprint, | isgraph, isascii: character handling | ctype(3C) |
| hexadecimal | ishex: determine if a character is | ishex(3C) |
| /iswcntrl, iswascii, isphonogram, | isideogram, isenglish, isnumber,/ | wctype(3W) |
| /touchline, untouchwin, wtouchln, | is_linetouched, is_wintouched: curses/ | curs_touch(3X) |
| isspace,/ ctype: isdigit, isxdigit, | islower, isupper, isalpha, isalnum, | ctype(3C) |
| unordered: determine type of/ | isnan, isnand, isnanf, finite, fpclass, | isnan(3C) |
| unordered: determine type of/ isnan, | isnand, isnanf, finite, fpclass, | isnan(3C) |
| determine type of/ isnan, isnand, | isnanf, finite, fpclass, unordered: | isnan(3C) |
| /isphonogram, isideogram, isenglish, | isnumber, isspecial: classify ASCII and/ | wctype(3W) |
| /iswprint, iswgraph, iswcntrl, iswascii, | isphonogram, isideogram, isenglish,/ | wctype(3W) |
| /isalnum, isspace, iscntrl, ispunct, | isprint, isgraph, isascii: character/ | ctype(3C) |

| | | |
|---|---|---|
| /isalpha, isalnum, isspace, iscntrl, | ispunct, isprint, isgraph, isascii:/ | ctype(3C) |
| /islower, isupper, isalpha, isalnum, | isspace, iscntrl, ispunct, isprint,/ | ctype(3C) |
| /isideogram, isenglish, isnumber, | isspecial: classify ASCII and/ | wctype(3W) |
| system: | issue a shell command | system(3S) |
| issue: | issue identification file | issue(4) |
| | issue: issue identification file | issue(4) |
| ctype: isdigit, isxdigit, islower, | isupper, isalpha, isalnum, isspace,/ | ctype(3C) |
| /iswupper, iswlower, iswdigit, iswxdigit, | iswalnum, iswspace, iswpunct, iswprint,/ | wctype(3W) |
| iswxdigit, iswalnum, iswspace,/ /wctype: | iswalpha, iswupper, iswlower, iswdigit, | wctype(3W) |
| /iswpunct, iswprint, iswgraph, iswcntrl, | iswascii, isphonogram, isideogram,/ | wctype(3W) |
| /iswspace, iswpunct, iswprint, iswgraph, | iswcntrl, iswascii, isphonogram,/ | wctype(3W) |
| /wctype: iswalpha, iswupper, iswlower, | iswdigit, iswxdigit, iswalnum, iswspace,/ | wctype(3W) |
| /iswalnum, iswspace, iswpunct, iswprint, | iswgraph, iswcntrl, iswascii,/ | wctype(3W) |
| /untouchwin, wtouchln, is_linetouched, | is_wintouched: curses refresh control/ | curs_touch(3X) |
| iswspace,/ /wctype: iswalpha, iswupper, | iswlower, iswdigit, iswxdigit, iswalnum, | wctype(3W) |
| /iswxdigit, iswalnum, iswspace, iswpunct, | iswprint, iswgraph, iswcntrl, iswascii,/ | wctype(3W) |
| /iswdigit, iswxdigit, iswalnum, iswspace, | iswpunct, iswprint, iswgraph, iswcntrl,/ | wctype(3W) |
| /iswlower, iswdigit, iswxdigit, iswalnum, | iswspace, iswpunct, iswprint, iswgraph,/ | wctype(3W) |
| iswalnum, iswspace,/ /wctype: iswalpha, | iswupper, iswlower, iswdigit, iswxdigit, | wctype(3W) |
| /iswalpha, iswupper, iswlower, iswdigit, | iswxdigit, iswalnum, iswspace, iswpunct,/ | wctype(3W) |
| isalnum, isspace,/ ctype: isdigit, | isxdigit, islower, isupper, isalpha, | ctype(3C) |
| a menu; prompt for and return a menu | item /ckitem: build | ckitem(1) |
| item_visible: tell if menus | item is visible /menu_item_visible: | menu_item_visible(3X) |
| /item_name, item_description: get menus | item name and description | menu_item_name(3X) |
| item_opts_off, item_opts: menus | item option routines /item_opts_on, | menu_item_opts(3X) |
| item_value: set and get menus | item values /set_item_value, | menu_item_value(3X) |
| /menu_items: set_menu_items, menu_items, | item_count: connect and disconnect items/ | menu_items(3X) |
| and/ /menu_item_name: item_name, | item_description: get menus item name | menu_item_name(3X) |
| /current_item, set_top_row, top_row, | item_index: set and get current menus/ | menu_item_current(3X) |
| menu_hook: set_item_init, | item_init, set_item_term, item_term,/ | menu_hook(3X) |
| item name and/ /menu_item_name: | item_name, item_description: get menus | menu_item_name(3X) |
| /item_opts_on, item_opts_off, | item_opts: menus item option routines | menu_item_opts(3X) |
| option/ /set_item_opts, item_opts_on, | item_opts_off, item_opts: menus item | menu_item_opts(3X) |
| menus/ /menu_item_opts: set_item_opts, | item_opts_on, item_opts_off, item_opts: | menu_item_opts(3X) |
| item_index: set and get current menus | items /set_top_row, top_row, | menu_item_current(3X) |
| free_item: create and destroy menus | items /menu_item_new: new_item, | menu_item_new(3X) |
| associate application data with menus | items /set_item_userptr, item_userptr: | menu_item_userptr(3X) |
| news: print news | items | news(1) |
| item_count: connect and disconnect | items to and from menus /menu_items, | menu_items(3X) |
| /set_item_init, item_init, set_item_term, | item_term, set_menu_init, menu_init,/ | menu_hook(3X) |
| /menu_item_userptr: set_item_userptr, | item_userptr: associate application data/ | menu_item_userptr(3X) |
| /menu_item_value: set_item_value, | item_value: set and get menus item/ | menu_item_value(3X) |
| visible /menu_item_visible: | item_visible: tell if menus item is | menu_item_visible(3X) |
| character string | itoa: convert an integer to an ASCII | itoa(3C) |
| m_in, mout, omout, fmout, m_out, sdiv, | itom: multiple precision integer/ /fmin, | mp(3X) |
| /bessel: | j0, j1, jn, y0, y1, yn: Bessel functions | bessel(3M) |
| bessel: j0, | j1, jn, y0, y1, yn: Bessel functions | bessel(3M) |
| bessel: j0, j1, | jn, y0, y1, yn: Bessel functions | bessel(3M) |
| jobs: summary of DG/UX | job control facilities | jobs(3C) |
| setpgid: set process group ID for | job control | setpgid(2) |
| uustat: uucp status inquiry and | job control | uustat(1) |
| queue lprm: remove | jobs from the line printer spooling | lprm(1) |
| /atq: display the | jobs queued to run at specified times | atq(1) |
| atrm: remove | jobs spooled by at or batch | atrm(1) |
| facilities | jobs: summary of DG/UX job control | jobs(3C) |
| | join: relational database operator | join(1) |
| /erand48, lrand48, nrand48, mrand48, | jrand48, srand48, seed48, lcong48:/ | drand48(3C) |
| programming language sh, | jsh, rsh, restsh: shell, the command | sh(1) |
| keyboard | kbd: AViiON series workstation system | kbd(7) |
| kbdset: attach to | kbd mapping tables, set modes | kbdset(1) |
| kbdpipe: use the | KBD module in a pipeline | kbdpipe(1) |
| kbdcomp: compile | kbd tables | kbdcomp(1M) |
| kbdload: load or link | kbd tables | kbdload(1M) |
| | kbdcomp: compile kbd tables | kbdcomp(1M) |
| | kbdload: load or link kbd tables | kbdload(1M) |
| pipeline | kbdpipe: use the KBD module in a | kbdpipe(1) |
| set modes | kbdset: attach to kbd mapping tables, | kbdset(1) |
| admkernel: manipulate the system's | kernel | admkernel(1M) |
| system: format of a | kernel description file | system(4) |
| kmem: | kernel logical memory | kmem(7) |
| key with the client/server common | key /decrypt conversation | dg_decryptsessionkey(2) |

| | | |
|---|---|---|
| | ld: link editor for common object files | ld-coff(1) |
| | ld: link editor for object files | ld(1) |
| ldclose, | ldaclose: close a common object file | ldclose(3X) |
| member of a COFF archive file | ldahread: read the archive header of a | ldahread(3X) |
| /ldopen, | ldaopen: open an object file for reading | ldopen(3X) |
| file | ldclose, ldaclose: close a common object | ldclose(3X) |
| | ldd: list dynamic dependencies | ldd(1) |
| scalb: manipulate parts of/ frexp, | ldexp, logb, modf, modff, nextafter, | frexp(3C) |
| routines | ldfcn: COFF executable file access | ldfcn(4) |
| common object file | ldfhread: read the file header of a | ldfhread(3X) |
| object file symbol table entry | ldgetname: retrieve symbol name for | ldgetname(3X) |
| /div, | ldiv: compute the quotient and remainder | div(3C) |
| entries of a common object/ /ldlread, | ldlinit, ldlitem: manipulate line number | ldlread(3X) |
| of a common object/ /ldlread, ldlinit, | ldlitem: manipulate line number entries | ldlread(3X) |
| line number entries of a common object/ | ldlread, ldlinit, ldlitem: manipulate | ldlread(3X) |
| entries of a section of a common object/ | ldlseek, ldnlseek: seek to line number | ldlseek(3X) |
| a section of a common object/ /ldlseek, | ldnlseek: seek to line number entries of | ldlseek(3X) |
| a section of a common object/ /ldrseek, | ldnrseek: seek to relocation entries of | ldrseek(3X) |
| section header of a common/ ldshread, | ldnshread: read an indexed/named | ldshread(3X) |
| section of a common object/ ldsseek, | ldnsseek: seek to an indexed/named | ldsseek(3X) |
| header of an object file | ldohseek: seek to the optional file | ldohseek(3X) |
| reading | ldopen, ldaopen: open an object file for | ldopen(3X) |
| entries of a section of a common object/ | ldrseek, ldnrseek: seek to relocation | ldrseek(3X) |
| indexed/named section header of a/ | ldshread, ldnshread: read an | ldshread(3X) |
| indexed/named section of a common/ | ldsseek, ldnsseek: seek to an | ldsseek(3X) |
| entry of an object file | ldtbindex: compute index of symbol table | ldtbindex(3X) |
| entry of an object file | ldtbread: read an indexed symbol table | ldtbread(3X) |
| object file | ldtbseek: seek to the symbol table of an | ldtbseek(3X) |
| discipline module | ldterm: standard STREAMS terminal line | ldterm(7) |
| /clearok, idlok, idcok immedok, | leaveok, setscrreg, wsetscrreg,/ | curs_outopts(3X) |
| | legend: Debugging information technology | legend(5) |
| truncate: truncate a file to a specified | length | truncate(2) |
| /dg_allow_shared_descriptor_attach: | let processes attach shared descriptor/ | dg_allow_shared_descriptor_attach(2 |
| getopt: get option | letter from argument vector | getopt(3C) |
| dbx: source | level debugger | dbx(1) |
| /addseverity: build list of severity | levels for application to be used with/ | addseverity(3C) |
| lexical tasks | lex: generate programs for simple | lex(1) |
| lex: generate programs for simple | lexical tasks | lex(1) |
| lsearch, | lfind: linear search and update | lsearch(3C) |
| gamma, | lgamma: log gamma function | gamma(3M) |
| intro: introduction to subroutines and | libraries | intro(3) |
| /elf_version: coordinate | library and application versions | elf_version(3E) |
| interfaces (emulated) to the termcap | library /tgetstr, tgoto, tputs: curses | curs_termcap(3X) |
| elf: object file access | library | elf(3E) |
| intro: introduction to network | library functions | intro(3N) |
| find ordering relation for an object | library /lorder: | lorder(1) |
| /ar: archive and | library maintainer for portable archives | ar(1) |
| /xdr_vector, xdr_void, xdr_wrapstring: | library routines for external data/ | xdr(3N) |
| calls /xprt_register, xprt_unregister: | library routines for remote procedure | rpc(3N) |
| t_alloc: allocate a | library structure | t_alloc(3N) |
| t_free: free a | library structure | t_free(3N) |
| t_sync: synchronize transport | library | t_sync(3N) |
| cpd: change or view the allocation | limits for a control point directory | cpd(1) |
| implementation-specific constants | limits: header file for | limits(4) |
| /dg_set_cpd_limits: change the resource | limits of a control point directory | dg_set_cpd_limits(2) |
| ulimit: get and set user | limits | ulimit(2) |
| sttydefs: maintain | line and hunt settings for TTY ports | sttydefs(1M) |
| setlinebuf: assign | line buffering for a specified stream | setlinebuf(3C) |
| dial: establish an out-going terminal | line connection | dial(3C) |
| connections connld: | line discipline for unique stream | connld(7) |
| set terminal type, modes, speed, and | line discipline /getty: | getty(1M) |
| ldterm: standard STREAMS terminal | line discipline module | ldterm(7) |
| editread: command | line editor | editread(5) |
| line: read one | line | line(1) |
| file /linenum: | line number entries in a common object | linenum(4) |
| /ldlread, ldlinit, ldlitem: manipulate | line number entries of a common object/ | ldlread(3X) |
| common/ ldlseek, ldnlseek: seek to | line number entries of a section of a | ldlseek(3X) |
| nl: | line numbering filter | nl(1) |
| cut: cut out selected fields of each | line of a file | cut(1) |
| rev: reverse order of characters in each | line of file | rev(1) |
| lpc: | line printer control program | lpc(1M) |

| | | |
|---|---|---|
| lp: DGC AViiON family | line printer special files | lp(7) |
| lpd: | line printer spooler | lpd(1M) |
| lpr: send print requests to a | line printer spooler | lpr(1) |
| lprm: remove jobs from the | line printer spooling queue | lprm(1) |
| | line: read one line | line(1) |
| ttydefs: terminal | line settings information for ttymon | ttydefs(4M) |
| lsearch, lfind: | linear search and update | lsearch(3C) |
| col: filter reverse | line-feeds | col(1) |
| object file | linenum: line number entries in a common | linenum(4) |
| comm: select or reject | lines common to two sorted files | comm(1) |
| curses borders, horizontal and vertical | lines /box, whline, wvline: create | curs_border(3X) |
| wredrawln: refresh curses windows and | lines /doupdate, redrawwin, | curs_refresh(3X) |
| fold: fold long | lines for finite width output device | fold(1) |
| head: give the first few | lines | head(1) |
| insertln, winsertln: delete and insert | lines in a curses window /winsdelln, | curs_deleteln(3X) |
| uniq: report repeated | lines in a file | uniq(1) |
| paste: merge | lines | paste(1) |
| link, unlink: exercise | link and unlink system calls | link(1M) |
| | link: create a new link to a file | link(2) |
| ld: | link editor for common object files | ld-coff(1) |
| ld: | link editor for object files | ld(1) |
| a.out: common assembler and | link editor output | a.out(4) |
| Environment variable sensitive file | link /elink: | elink(5) |
| symlink: create a symbolic | link file | symlink(2) |
| ln: | link files | ln(1) |
| kbdload: load or | link kbd tables | kbdload(1M) |
| read the contents of a symbolic | link /readlink: | readlink(2) |
| link: create a new | link to a file | link(2) |
| system calls | link, unlink: exercise link and unlink | link(1M) |
| /form_field_new: new_field, dup_field, | link_field, free_field,: create and/ | form_field_new(3X) |
| /set_fieldtype_arg, set_fieldtype_choice, | link_fieldtype: forms fieldtype routines | form_fieldtype(3X) |
| | lint: a C program checker | lint(1) |
| ls: | list contents of directory | ls(1) |
| ldd: | list dynamic dependencies | ldd(1) |
| ttysrch: directory search | list for ttyname | ttysrch(4M) |
| get or set supplementary group access | list IDs /getgroups, setgroups: | getgroups(2) |
| the supplementary group access | list /initgroups: initialize | initgroups(3C) |
| nlist: get entries from name | list | nlist(3C) |
| dispgid: display a | list of all valid group names | dispgid(1) |
| dispuid: display a | list of all valid user names | dispuid(1) |
| nm: print name | list of common object file | nm(1) |
| and ncheck /checklist: | list of file systems processed by fsck | checklist(4) |
| to be used with/ /addseverity: build | list of severity levels for application | addseverity(3C) |
| stdarg: handle variable argument | list | stdarg(5) |
| /logins: | list user and system login information | logins(1M) |
| listusers: | list user login information | listusers(1) |
| varargs: handle variable argument | list | varargs(5) |
| formatted output of a variable argument | list /vprintf, vfprintf, vsprintf: print | vprintf(3S) |
| formatted output of a variable argument | list /vprintf, vfprintf, vsprintf: print | vprintf(3W) |
| formatted input using varargs argument | list /vscanf, vfscanf, vsscanf: convert | vscanf(3S) |
| group | listdgrp: lists members of a device | listdgrp(1M) |
| t_listen: | listen for a connect request | t_listen(3N) |
| listen: | listen for connections on a socket | listen(2) |
| socket | listen: listen for connections on a | listen(2) |
| | listen: network listener server | listen(1M) |
| get client's data passed via the | listener /nlsgetcall: | nlsgetcall(3N) |
| listen: network | listener server | listen(1M) |
| nlsadmin: network | listener service administration | nlsadmin(1M) |
| nlsrequest: format and send | listener service request message | nlsrequest(3N) |
| xargs: construct argument | list(s) and execute command | xargs(1) |
| devattr: | lists device attributes | devattr(1M) |
| devices that match criteria getdgrp: | lists device groups which contain | getdgrp(1M) |
| getdev: | lists devices based on criteria | getdev(1M) |
| listdgrp: | lists members of a device group | listdgrp(1M) |
| | listusers: list user login information | listusers(1) |
| | ln: link files | ln(1) |
| lsd: | load a system dump from tape | lsd(1M) |
| kbdload: | load or link kbd tables | kbdload(1M) |
| tcload: | load terminal controller devices | tcload(1M) |
| finger: display information about | local and remote users | finger(1) |
| testlocale: test | locale definition | testlocale(1M) |

| | | |
|---|---|---|
| setlocale: modify and query a program's | locale | setlocale(3C) |
| set default system time zone and | locale /timezone: | timezone(4) |
| information | localeconv: get numeric formatting | localeconv(3C) |
| convert date and time to string ctime, | localtime, gmtime, asctime, tzset: | ctime(3C) |
| /which: | locate a program file for csh(1) users | which(1) |
| reference manuals man: | locate and print entries from the | man(1) |
| apropos: | locate commands by keyword lookup | apropos(1) |
| keywords | locate: identify a command using | locate(1) |
| program /whereis: | locate source, binary, and or manual for | whereis(1) |
| indivisible fetch and add to memory | location /fetch_and_add: | fetch_and_add(2) |
| end, etext, edata: last | locations in program | end(3C) |
| remove locks held by remote | lock clients /dg_lock_kill: | dg_lock_kill(2) |
| plock: | lock data, text, or both into memory | plock(2) |
| period /dg_lock_reset: reset remote file | lock database, start lock reclaim grace | dg_lock_reset(2) |
| plm: pseudo | lock manager device interface | plm(7) |
| dg_flock: apply or remove an advisory | lock on an open DG/UX file | dg_flock(3C) |
| mlockall, munlockall: | lock or unlock address space | mlockall(3C) |
| mlock, munlock: | lock (or unlock) pages in memory | mlock(3C) |
| /reset remote file lock database, start | lock reclaim grace period | dg_lock_reset(2) |
| dg_lcntl: process a record | lock request on a filehandle | dg_lcntl(2) |
| wait for previously delayed | lock requests to complete /dg_lock_wait: | dg_lock_wait(2) |
| | lockf: record locking on files | lockf(3C) |
| lockf: record | locking on files | lockf(3C) |
| /dg_lock_kill: remove | locks held by remote lock clients | dg_lock_kill(2) |
| gamma, lgamma: | log gamma function | gamma(3M) |
| newgrp: | log in to a new group | newgrp(1) |
| and event tracing | log: interface to STREAMS error logging | log(7) |
| sqrt, sqrtf:/ exp, expf, cbrt, | log, logf, log10, log10f, pow, powf, | exp(3M) |
| logger: make entries in the system | log | logger(1) |
| RCS files /rlog: print | log messages and other information about | rlog(1) |
| configuration file for syslogd system | log server /syslog.conf: | syslog.conf(5) |
| closelog, setlogmask: control system | log /syslog, openlog, | syslog(3C) |
| syslogd: | log systems messages | syslogd(1M) |
| /exp, expf, cbrt, log, logf, | log10, log10f, pow, powf, sqrt, sqrtf:/ | exp(3M) |
| exp, expf, cbrt, log, logf, log10, | log10f, pow, powf, sqrt, sqrtf:/ | exp(3M) |
| /pow, powf, sqrt, sqrtf: exponential, | logarithm, power, square root functions | exp(3M) |
| manipulate parts of/ frexp, ldexp, | logb, modf, modff, nextafter, scalb: | frexp(3C) |
| sqrtf:/ exp, expf, cbrt, log, | logf, log10, log10f, pow, powf, sqrt, | exp(3M) |
| strclean: STREAMS error | logger cleanup program | strclean(1M) |
| | logger: make entries in the system log | logger(1) |
| strerr: STREAMS error | logger server | strerr(1M) |
| log: interface to STREAMS error | logging and event tracing | log(7) |
| menu interface for managing physical and | logical disks /diskman: | diskman(1M) |
| kmem: kernel | logical memory | kmem(7) |
| userdel: delete a user's | login from the system | userdel(1M) |
| listusers: list user | login information | listusers(1) |
| logins: list user and system | login information | logins(1M) |
| usermod: modify a user's | login information on the system | usermod(1M) |
| getlogin: get | login name | getlogin(3C) |
| logname: get | login name | logname(1) |
| /sysv3_cuserid: get character | login name of the user | sysv3_cuserid(3S) |
| logname: return | login name of user | logname(3X) |
| effective UID /cuserid: get character | login name or user name associated with | cuserid(3S) |
| useradd: administer a new user | login on the system | useradd(1M) |
| passwd: change | login password | passwd(1) |
| dial-up devices d_passwd: | log-in programs and passwords for | d_passwd(4) |
| | login: sign on | login(1) |
| profile: setting up an environment at | login time | profile(4) |
| last: indicate last user or terminal | logins | last(1) |
| information | logins: list user and system login | logins(1M) |
| | logname: get login name | logname(1) |
| | logname: return login name of user | logname(3X) |
| a64l, l64a: convert between | long integer and base-64 ASCII string | a64l(3C) |
| sputl, sgetl: access | long integer data in a/ | sputl(3X) |
| convert between 3-byte integers and | long integers /l3tol, ltol3: | l3tol(3C) |
| device fold: fold | long lines for finite width output | fold(1) |
| setjmp, | longjmp: non-local goto | setjmp(3C) |
| /erasechar, has_ic, has_il, killchar, | longname, termattrs, termname: curses/ | curs_termattrs(3X) |
| endpoint /t_look: | look at the current event on a transport | t_look(3N) |
| apropos: locate commands by keyword | lookup | apropos(1) |
| object library | lorder: find ordering relation for an | lorder(1) |

| | | |
|---|---|---|
| nice: run a command at a higher or | lower priority | nice(1) |
| setsyx, ripoffline, curs_set, napms: | low-level curses routines /getsyx, | curs_kernel(3X) |
| LP print service | lp, cancel: send/cancel requests to an | lp(1) |
| special files | lp: DGC AViiON family line printer | lp(7) |
| lpprint, xlpprint: menu-driven | lp interface | lpprint(1M) |
| lpsched, lpshut, lpmove: start/stop the | LP print service and move requests | lpsched(1M) |
| lp, cancel: send/cancel requests to an | LP print service | lp(1) |
| lpadmin: configure the | LP print service | lpadmin(1M) |
| administer filters used with the | LP print service /lpfilter: | lpfilter(1M) |
| lpforms: administer forms used with the | LP print service | lpforms(1M) |
| information about the status of the | LP print service /lpstat: print | lpstat(1) |
| enable, disable: enable/disable | LP printers | enable(1) |
| | lpadmin: configure the LP print service | lpadmin(1M) |
| | lpc: line printer control program | lpc(1M) |
| | lpd: line printer spooler | lpd(1M) |
| the LP print service | lpfilter: administer filters used with | lpfilter(1M) |
| LP print service | lpforms: administer forms used with the | lpforms(1M) |
| and move requests /lpsched, lpshut, | lpmove: start/stop the LP print service | lpsched(1M) |
| interface | lpprint, xlpprint: menu-driven lp | lpprint(1M) |
| | lpq: examine the spool queue | lpq(1) |
| printer spooler | lpr: send print requests to a line | lpr(1) |
| spooling queue | lprm: remove jobs from the line printer | lprm(1) |
| LP print service and move requests | lpsched, lpshut, lpmove: start/stop the | lpsched(1M) |
| service and move requests /lpsched, | lpshut, lpmove: start/stop the LP print | lpsched(1M) |
| status of the LP print service | lpstat: print information about the | lpstat(1) |
| the print service | lpsystem: register remote systems with | lpsystem(1M) |
| with 40014A Terminal Server | lptermprinter: start printer session | lptermprinter(1) |
| | lpusers: set printing queue priorities | lpusers(1M) |
| srand48, seed48,/ drand48, erand48, | lrand48, nrand48, mrand48, jrand48, | drand48(3C) |
| | ls: list contents of directory | ls(1) |
| | lsd: load a system dump from tape | lsd(1M) |
| | lsearch, lfind: linear search and update | lsearch(3C) |
| position | lseek: change object pointer's current | lseek(2) |
| | lstat: get file status | lstat(2) |
| and long integers /l3tol, | ltol3: convert between 3-byte integers | l3tol(3C) |
| | m4: macro processor | m4(1) |
| provide truth value/ /machid: dghost, | m68k, m88k, i386, pdp11, u3b, u3b5, vax: | machid(1) |
| provide truth/ machid: dghost, m68k, | m88k, i386, pdp11, u3b, u3b5, vax: | machid(1) |
| u3b, u3b5, vax: provide truth value/ | machid: dghost, m68k, m88k, i386, pdp11, | machid(1) |
| values: | machine-dependent values | values(5) |
| sgetl: access long integer data in a | machine-independent fashion /sputl, | sputl(3X) |
| m4: | macro processor | m4(1) |
| invert, rpow, msqrt, mcmp, move, mp: | madd, msub, mult, mdiv, pow, gcd, | mp(3X) |
| rmt: start the remote | mag tape server | rmt(1M) |
| mt: | magnetic tape control | mt(1) |
| wmtd: start the WORM | magnetic tape device server | wmtd(1M) |
| rmt: character special | magnetic tape interface | rmt(7) |
| Once Read Multiple optical device) as | magtape interface /pseudo WORM (Write | wmt(7) |
| database admalias: manage | mail alias information in the aliases | admalias(1M) |
| mailalias: translate | mail alias names | mailalias(1) |
| mailcnfg: initialization information for | mail and rmail | mailcnfg(4M) |
| invoke recipient command for incoming | mail /mail_pipe: | mail_pipe(1M) |
| commands for routing and transport of | mail /mailsurr: surrogate | mailsurr(4M) |
| automatically respond to incoming | mail messages /vacation: | vacation(1) |
| notify user of the arrival of new | mail /notify: | notify(1) |
| mail, rmail: read | mail or send mail to users | mail(1) |
| users | mail, rmail: read mail or send mail to | mail(1) |
| mail, rmail: read mail or send | mail to users | mail(1) |
| a binary file for transmission via | mail /uuencode, uudecode: encode/decode | uuencode(1) |
| | mailalias: translate mail alias names | mailalias(1) |
| mail and rmail | mailcnfg: initialization information for | mailcnfg(4M) |
| incoming mail | mail_pipe: invoke recipient command for | mail_pipe(1M) |
| and transport of mail | mailsurr: surrogate commands for routing | mailsurr(4M) |
| system | mailx: interactive message processing | mailx(1) |
| | main: enter a C main program | main(3C) |
| main: enter a C | main program | main(3C) |
| mem: | main system memory | mem(7) |
| ports /sttydefs: | maintain line and hunt settings for TTY | sttydefs(1M) |
| of programs /make: | maintain, update, and regenerate groups | make(1) |
| ar: archive and library | maintainer for portable archives | ar(1) |
| Array/ gridman: menu interface for | maintaining a High Availability Disk | gridman(1M) |

| | | |
|---|---|---|
| operations /memory: memccpy, memchr, | memcmp, memcpy, memmove, memset: memory | memory(3C) |
| | memcntl: memory management control | memcntl(2) |
| memory: memccpy, memchr, memcmp, | memcpy, memmove, memset: memory/ | memory(3C) |
| | memctl: set protection of memory mapping | memctl(2) |
| memory: memccpy, memchr, memcmp, memcpy, | memmove, memset: memory operations | memory(3C) |
| misalign: handle misaligned | memory access faults | misalign(5) |
| realloc, calloc, memalign, valloc,: | memory allocator /malloc, free, | malloc(3C) |
| realloc, calloc, mallopt, mallinfo: | memory allocator /malloc, free, | malloc(3X) |
| bcmp: compare two areas of | memory | bcmp(3C) |
| bzero: zero a portion of | memory | bzero(3C) |
| shmctl: shared | memory control operations | shmctl(2) |
| copylist: copy a file into | memory | copylist(3G) |
| vfork: spawn new process in a virtual | memory efficient way | vfork(2) |
| mfs: | memory file system | mfs(4) |
| message queue, semaphore set, or shared | memory ID /ipcrm: remove a | ipcrm(1) |
| kmem: kernel logical | memory | kmem(7) |
| indivisible fetch and add to | memory location /fetch_and_add: | fetch_and_add(2) |
| memcntl: | memory management control | memcntl(2) |
| memctl: set protection of | memory mapping | memctl(2) |
| mprotect: set protection of | memory mapping | mprotect(2) |
| mem: main system | memory | mem(7) |
| memmove, memset: memory operations | memory: memccpy, memchr, memcmp, memcpy, | memory(3C) |
| munlock: lock (or unlock) pages in | memory /mlock, | mlock(3C) |
| mmap: map pages of | memory | mmap(2) |
| munmap: unmap pages of | memory | munmap(2) |
| shmsys: perform a shared | memory operation | shmsys(2) |
| memchr, memcmp, memcpy, memmove, memset: | memory operations /memory: memccpy, | memory(3C) |
| /dg_paging_info: determine residency of | memory pages | dg_paging_info(2) |
| mincore: determine residency of | memory pages | mincore(2) |
| plock: lock data, text, or both into | memory | plock(2) |
| /sync: synchronize disk and | memory resident file system information | sync(2) |
| shmat: attach a shared | memory segment | shmat(2) |
| shmdt: detach a shared | memory segment | shmdt(2) |
| shmget: get shared | memory segment | shmget(2) |
| msync: synchronize | memory with physical storage | msync(3C) |
| memchr, memcmp, memcpy, memmove, | memset: memory operations /memccpy, | memory(3C) |
| Availability Disk Array/ gridman: | menu interface for maintaining a High | gridman(1M) |
| logical disks /diskman: | menu interface for managing physical and | diskman(1M) |
| build a menu; prompt for and return a | menu item /ckitem: | ckitem(1) |
| /ckitem: build a | menu; prompt for and return a menu item | ckitem(1) |
| menu_fore, set_menu_back, menu_back,/ | menu_attributes: set_menu_fore, | menu_attributes(3X) |
| /set_menu_fore, menu_fore, set_menu_back, | menu_back, set_menu_grey, menu_grey,/ | menu_attributes(3X) |
| position a menus cursor | menu_cursor: pos_menu_cursor: correctly | menu_cursor(3X) |
| lpprint, xlpprint: | menu-driven lp interface | lpprint(1M) |
| interface sysadm, xsysadm: | menu-driven system administration | sysadm(1M) |
| program osysadm: | menu-driven system administration | osysadm(1M) |
| menus subsystem | menu_driver: command processor for the | menu_driver(3X) |
| /menu_attributes: set_menu_fore, | menu_fore, set_menu_back, menu_back,/ | menu_attributes(3X) |
| menu_format: set and get maximum/ | menu_format: set_menu_format, | menu_format(3X) |
| of rows/ /menu_format: set_menu_format, | menu_format: set and get maximum numbers | menu_format(3X) |
| /set_menu_back, menu_back, set_menu_grey, | menu_grey, set_menu_pad, menu_pad:/ | menu_attributes(3X) |
| set_item_term, item_term,/ | menu_hook: set_item_init, item_init, | menu_hook(3X) |
| /set_item_term, item_term, set_menu_init, | menu_init, set_menu_term, menu_term:/ | menu_hook(3X) |
| current_item, set_top_row, top_row,/ | menu_item_current: set_current_item, | menu_item_current(3X) |
| item_description: get menus item name/ | menu_item_name: item_name, | menu_item_name(3X) |
| create and destroy menus items | menu_item_new: new_item, free_item: | menu_item_new(3X) |
| item_opts_on, item_opts_off, item_opts:/ | menu_item_opts: set_item_opts, | menu_item_opts(3X) |
| item_count: connect and disconnect/ | menu_items: set_menu_items, menu_items, | menu_items(3X) |
| disconnect/ menu_items: set_menu_items, | menu_items, item_count: connect and | menu_items(3X) |
| item_userptr: associate application/ | menu_item_userptr: set_item_userptr, | menu_item_userptr(3X) |
| item_value: set and get menus item/ | menu_item_value: set_item_value, | menu_item_value(3X) |
| if menus item is visible | menu_item_visible: item_visible: tell | menu_item_visible(3X) |
| menus mark string routines | menu_mark: set_menu_mark, menu_mark: | menu_mark(3X) |
| menu_mark: set_menu_mark, | menu_mark: menus mark string routines | menu_mark(3X) |
| and destroy menus | menu_new: new_menu, free_menu: create | menu_new(3X) |
| menu_opts_off, menu_opts: menus option/ | menu_opts: set_menu_opts, menu_opts_on, | menu_opts(3X) |
| /menu_opts_on, menu_opts_off, | menu_opts: menus option routines | menu_opts(3X) |
| /menu_opts: set_menu_opts, menu_opts_on, | menu_opts_off, menu_opts: menus option/ | menu_opts(3X) |
| menus option/ /menu_opts: set_menu_opts, | menu_opts_on, menu_opts_off, menu_opts: | menu_opts(3X) |
| /set_menu_grey, menu_grey, set_menu_pad, | menu_pad: control menus display/ | menu_attributes(3X) |
| menu_pattern: set and get menus pattern/ | menu_pattern: set_menu_pattern, | menu_pattern(3X) |

| | | |
|---|---|---|
| match/ /menu_pattern: set_menu_pattern, | menu_pattern: set and get menus pattern | menu_pattern(3X) |
| write or erase menus from associated/ | menu_post: post_menu, unpost_menu: | menu_post(3X) |
| | menus: character based menus package | menus(3X) |
| pos_menu_cursor: correctly position a | menus cursor /menu_cursor: | menu_cursor(3X) |
| set_menu_pad, menu_pad: control | menus display attributes /menu_grey, | menu_attributes(3X) |
| /post_menu, unpost_menu: write or erase | menus from associated subwindows | menu_post(3X) |
| /item_visible: tell if | menus item is visible | menu_item_visible(3X) |
| /item_name, item_description: get | menus item name and description | menu_item_name(3X) |
| /item_opts_on, item_opts_off, item_opts: | menus item option routines | menu_item_opts(3X) |
| set_item_value, item_value: set and get | menus item values /menu_item_value: | menu_item_value(3X) |
| top_row, item_index: set and get current | menus items /current_item, set_top_row, | menu_item_current(3X) |
| new_item, free_item: create and destroy | menus items /menu_item_new: | menu_item_new(3X) |
| associate application data with | menus items /item_userptr: | menu_item_userptr(3X) |
| menu_mark: set_menu_mark, menu_mark: | menus mark string routines | menu_mark(3X) |
| maximum numbers of rows and columns in | menus /menu_format: set and get | menu_format(3X) |
| routines for automatic invocation by | menus /assign application-specific | menu_hook(3X) |
| connect and disconnect items to and from | menus /menu_items, item_count: | menu_items(3X) |
| new_menu, free_menu: create and destroy | menus /menu_new: | menu_new(3X) |
| associate application data with | menus /set_menu_userptr, menu_userptr: | menu_userptr(3X) |
| menu_opts_on, menu_opts_off, menu_opts: | menus option routines /set_menu_opts, | menu_opts(3X) |
| menus: character based | menus package | menus(3X) |
| /menu_pattern: set and get | menus pattern match buffer | menu_pattern(3X) |
| /menu_driver: command processor for the | menus subsystem | menu_driver(3X) |
| /set_menu_sub, menu_sub, scale_menu: | menus window and subwindow association/ | menu_win(3X) |
| /set_menu_win, menu_win, set_menu_sub, | menu_sub, scale_menu: menus window and/ | menu_win(3X) |
| /set_menu_init, menu_init, set_menu_term, | menu_term: assign application-specific/ | menu_hook(3X) |
| menu_userptr: associate application/ | menu_userptr: set_menu_userptr, | menu_userptr(3X) |
| with/ /menu_userptr: set_menu_userptr, | menu_userptr: associate application data | menu_userptr(3X) |
| set_menu_sub, menu_sub, scale_menu:/ | menu_win: set_menu_win, menu_win, | menu_win(3X) |
| scale_menu:/ menu_win: set_menu_win, | menu_win, set_menu_sub, menu_sub, | menu_win(3X) |
| sort: sort and/or | merge files | sort(1) |
| paste: | merge lines | paste(1) |
| merge: three-way file | merge | merge(1) |
| acctmerg: | merge or add total accounting files | acctmerg(1M) |
| rcsmerge: | merge RCS revisions | rcsmerge(1) |
| | merge: three-way file merge | merge(1) |
| | mesg: permit or deny messages | mesg(1) |
| catgets: print message from | message catalog | catgets(1) |
| catopen, catclose: open/close a | message catalogue | catopen(3C) |
| gencat: generate a formatted | message catalogue | gencat(1) |
| catgets: read a program | message | catgets(3C) |
| gettxt: retrieve a text string from a | message data base | gettxt(1) |
| of, or search for a text string in, | message data bases /display contents | srchtxt(1) |
| putmsg, putpmsg: pass a | message down a stream | putmsg(2) |
| mkstr: create an error | message file by massaging C source | mkstr(1) |
| mkmsgs: create | message files for use by gettxt | mkmsgs(1) |
| recv: receive a | message from a socket | recv(2) |
| recvfrom: receive a | message from a socket | recvfrom(2) |
| recvmsg: receive a | message from a socket | recvmsg(2) |
| send: send a | message from a socket | send(2) |
| sendmsg: send a | message from a socket | sendmsg(2) |
| sendto: send a | message from a socket | sendto(2) |
| getmsg, getpmsg: get a | message from a stream | getmsg(2) |
| catgets: print | message from message catalog | catgets(1) |
| msgrcv: receive a | message | msgrcv(2) |
| msgsnd: send a | message | msgsnd(2) |
| format and send listener service request | message /nlsrequest: | nlsrequest(3N) |
| fmtmsg: display a | message on stderr or system console | fmtmsg(1) |
| fmtmsg: display a | message on stderr or system console | fmtmsg(3C) |
| mailx: interactive | message processing system | mailx(1) |
| message queue msgctl: get or set | message queue attributes or destroy a | msgctl(2) |
| msgget: get | message queue identifier | msgget(2) |
| message queue attributes or destroy a | message queue /msgctl: get or set | msgctl(2) |
| msgsys: perform a | message queue operation | msgsys(2) |
| memory ID /ipcrm: remove a | message queue, semaphore set, or shared | ipcrm(1) |
| /extended_strerror: get extended error | message string | extended_strerror(3C) |
| strerror: get error | message string | strerror(3C) |
| t_error: produce error | message | t_error(3N) |
| /extended_perror: print an error | message to standard error | extended_perror(3C) |
| files /rlog: print log | messages and other information about RCS | rlog(1) |
| whether remote system can accept binary | messages /ckbinarsys: determine | ckbinarsys(1M) |

| | | |
|---|---|---|
| mesg: permit or deny | messages | mesg(1) |
| perror: print system error | messages | perror(3C) |
| psignal, psiginfo: system signal | messages | psignal(3C) |
| strace: print STREAMS trace | messages | strace(1M) |
| syslogd: log systems | messages | syslogd(1M) |
| automatically respond to incoming mail | messages /vacation: | vacation(1) |
| /noecho, halfdelay, intrflush, keypad, | meta, nodelay, notimeout, raw, noraw,/ | curs_inopts(3X) |
| | mfs: memory file system | mfs(4) |
| /msqrt, mcmp, move, min, omin, fmin, | m_in, mout, omout, fmout, m_out, sdiv,/ | mp(3X) |
| /gcd, invert, rpow, msqrt, mcmp, move, | min, omin, fmin, m_in, mout, omout,/ | mp(3X) |
| pages | mincore: determine residency of memory | mincore(2) |
| clone: open any | minor device on a STREAMS driver | clone(7) |
| access faults | misalign: handle misaligned memory | misalign(5) |
| misalign: handle | misaligned memory access faults | misalign(5) |
| /acctwtmp: overview of accounting and | miscellaneous accounting commands | acct(1M) |
| /putwin, getwin, delay_output, flushinp: | miscellaneous curses utility routines | curs_util(3X) |
| intro: introduction to | miscellany | intro(5) |
| | mkdir: create a directory file | mkdir(2) |
| | mkdir: make a directory | mkdir(1) |
| directories in a path | mkdirp, rmdirp: create, remove | mkdirp(3G) |
| | mkfifo: create a new FIFO | mkfifo(3C) |
| | mkfifo: make FIFO special file | mkfifo(1M) |
| | mkfs, newfs: create a file system | mkfs(1M) |
| gettxt | mkmsgs: create message files for use by | mkmsgs(1) |
| | mknod: build a special file | mknod(1M) |
| system | mknod: create a file entry in the file | mknod(2) |
| | mkstemp: make a unique file name | mkstemp(3C) |
| massaging C source | mkstr: create an error message file by | mkstr(1) |
| | mktemp: make a unique file name | mktemp(3C) |
| in memory | mlock, munlock: lock (or unlock) pages | mlock(3C) |
| address space | mlockall, munlockall: lock or unlock | mlockall(3C) |
| | mmap: map pages of memory | mmap(2) |
| | mnttab: mounted file system table | mnttab(4) |
| chmod: change file | mode | chmod(1) |
| umask: set file-creation | mode mask | umask(1) |
| pckt: STREAMS Packet | Mode module | pckt(7) |
| chmod: change | mode of file | chmod(2) |
| fchmod: change | mode of file | fchmod(2) |
| attach to kbd mapping tables, set | modes /kbdset: | kbdset(1) |
| getty: set terminal type, | modes, speed, and line discipline | getty(1M) |
| manipulate parts of/ frexp, ldexp, logb, | modf, modff, nextafter, scalb: | frexp(3C) |
| parts of/ frexp, ldexp, logb, modf, | modff, nextafter, scalb: manipulate | frexp(3C) |
| touch: update access and | modification times of a file | touch(1) |
| utime: set file access and | modification times | utime(2) |
| utimes: set file access and | modification times | utimes(2) |
| /groupmod: | modify a group definition on the system | groupmod(1M) |
| system /usermod: | modify a user's login information on the | usermod(1M) |
| setlocale: | modify and query a program's locale | setlocale(3C) |
| dg_sysctl: | modify system parameters | dg_sysctl(1M) |
| alp: Algorithm Pool management | module | alp(7) |
| alpq: query the ALP STREAMS | module | alpq(1) |
| att_kbd: generalized string translation | module | att_kbd(7) |
| kbdpipe: use the KBD | module in a pipeline | kbdpipe(1) |
| STREAMS terminal line discipline | module /ldterm: standard | ldterm(7) |
| pckt: STREAMS Packet Mode | module | pckt(7) |
| ptem: STREAMS Pseudo Terminal Emulation | module | ptem(7) |
| Transport Interface cooperating STREAMS | module /timod: | timod(7) |
| Interface read/write interface STREAMS | module /tirdwr: Transport | tirdwr(7) |
| V7, 4BSD and XENIX STREAMS compatibility | module /ttcompat: | ttcompat(7) |
| configure automatically pushed STREAMS | modules /autopush: | autopush(1M) |
| to EUC handling TTY drivers and | modules /eucioctl: generic interface | eucioctl(5) |
| chargefee, ckpacct, dodisk, lastlogin, | monacct, nulladm, prctmp, prdaily,/ | acctsh(1M) |
| montbl: create | monetary database | montbl(1M) |
| pmadm: port | monitor administration | pmadm(1M) |
| ttyadm: format and output TTY port | monitor information | ttyadm(1M) |
| | monitor: prepare execution profile | monitor(3C) |
| /admportservice: manage port | monitor services | admportservice(1M) |
| ttymon: | monitor terminal ports | ttymon(1M) |
| admsar: manage system activity | monitoring and reporting | admsar(1M) |
| /admportmonitor: manage port | monitors | admportmonitor(1M) |
| | montbl: create monetary database | montbl(1M) |

| | | |
|---|---|---|
| at a time | more, page: display file one screenful | more(1) |
| dg_mount: | mount a file system | dg_mount(2) |
| mount: | mount a file system | mount(2) |
| mount, umount: | mount and dismount filesystems | mount(1M) |
| | mount: mount a file system | mount(2) |
| setmnt: establish | mount table | setmnt(1M) |
| filesystems | mount, umount: mount and dismount | mount(1M) |
| fstatfs: get information about a | mounted file system | fstatfs(2) |
| statfs: get information about a | mounted file system | statfs(2) |
| mnttab: | mounted file system table | mnttab(4) |
| exportfs: make a directory available for | mounting via NFS | exportfs(2) |
| mouse: | mouse device | mouse(7) |
| | mouse: mouse device | mouse(7) |
| /mcmp, move, min, omin, fmin, m_in, | mout, omout, fmout, m_out, sdiv, itom:/ | mp(3X) |
| /omin, fmin, m_in, mout, omout, fmout, | m_out, sdiv, itom: multiple precision/ | mp(3X) |
| mvdir: | move a directory | mvdir(1M) |
| screen panel_move: move_panel: | move a panels window on the virtual | panel_move(3X) |
| curs_move: move, wmove: | move curses window cursor | curs_move(3X) |
| mv: | move files | mv(1) |
| /pow, gcd, invert, rpow, msqrt, mcmp, | move, min, omin, fmin, m_in, mout,/ | mp(3X) |
| start/stop the LP print service and | move requests /lpsched, lpshut, lpmove: | lpsched(1M) |
| area large enough to hold string and | move string into it /strnsave: allocate | strsave(3C) |
| /curs_move: | move, wmove: move curses window cursor | curs_move(3X) |
| /form_fields, field_count, | move_field: connect fields to forms | form_field(3X) |
| virtual screen /panel_move: | move_panel: move a panels window on the | panel_move(3X) |
| invert, rpow, msqrt, mcmp, move, min,/ | mp: madd, msub, mult, mdiv, pow, gcd, | mp(3X) |
| mapping | mprotect: set protection of memory | mprotect(2) |
| drand48, erand48, lrand48, nrand48, | mrand48, jrand48, srand48, seed48,/ | drand48(3C) |
| attributes or destroy a message queue | msgctl: get or set message queue | msgctl(2) |
| | msgget: get message queue identifier | msgget(2) |
| | msgrcv: receive a message | msgrcv(2) |
| | msgsnd: send a message | msgsnd(2) |
| operation | msgsys: perform a message queue | msgsys(2) |
| /mult, mdiv, pow, gcd, invert, rpow, | msqrt, mcmp, move, min, omin, fmin,/ | mp(3X) |
| rpow, msqrt, mcmp, move, min,/ mp: madd, | msub, mult, mdiv, pow, gcd, invert, | mp(3X) |
| storage | msync: synchronize memory with physical | msync(3C) |
| | mt: magnetic tape control | mt(1) |
| msqrt, mcmp, move, min,/ mp: madd, msub, | mult, mdiv, pow, gcd, invert, rpow, | mp(3X) |
| mbchar: mbtowc, wctomb, mblen: | multibyte character conversion | mbchar(3W) |
| mbchar: mbtowc, mblen, wctomb: | multibyte character handling | mbchar(3C) |
| widec: | multibyte character I/O routines | widec(3W) |
| mbstring: mbstowcs, wctombs,: | multibyte string conversion | mbstring(3W) |
| mbstring: mbstowcs, wcstombs: | multibyte string functions | mbstring(3C) |
| wmt: pseudo WORM (Write Once Read | Multiple optical device) as magtape/ | wmt(7) |
| /mout, omout, fmout, m_out, sdiv, itom: | multiple precision integer arithmetic | mp(3X) |
| poll: input/output | multiplexing | poll(2) |
| memory mlock, | munlock: lock (or unlock) pages in | mlock(3C) |
| /mlockall, | munlockall: lock or unlock address space | mlockall(3C) |
| | munmap: unmap pages of memory | munmap(2) |
| | mv: move files | mv(1) |
| add a/ /curs_addch: addch, waddch, | mvaddch, mvwaddch, echochar, wechochar: | curs_addch(3X) |
| add/ /waddchstr, waddchnstr, mvaddchstr, | mvaddchnstr, mvwaddchstr, mvwaddchnstr: | curs_addchst(3X) |
| add/ /waddchstr, waddchnstr, mvaddchstr, | mvaddchnstr, mvwaddchstr, mvwaddchnstr: | curs_addchstr(3X) |
| /addchnstr, waddchstr, waddchnstr, | mvaddchstr, mvaddchnstr, mvwaddchstr,/ | curs_addchst(3X) |
| /addchnstr, waddchstr, waddchnstr, | mvaddchstr, mvaddchnstr, mvwaddchstr,/ | curs_addchstr(3X) |
| /addnstr, waddstr, waddnstr, mvaddstr, | mvaddnstr, mvwaddstr, mvwaddnstr: add a/ | curs_addstr(3X) |
| a/ /waddwstr, waddnwstr, mvaddwstr, | mvaddnwstr, mvwaddwstr, mvwaddnwstr: add | curs_addwstr(3X) |
| /addstr, addnstr, waddstr, waddnstr, | mvaddstr, mvaddnstr, mvwaddstr,/ | curs_addstr(3X) |
| /curs_addwch: addwch, waddwch, | mvaddwch, mvwaddwch, echowchar,/ | curs_addwch(3X) |
| waddwchstr, waddwchnstr, mvaddwchstr, | mvaddwchnstr, mvwaddwchstr,/ /addwchnstr, | curs_addwchstr(3X) |
| /addwchnstr, waddwchstr, waddwchnstr, | mvaddwchstr, mvaddwchnstr, mvwaddwchstr,/ | curs_addwchstr(3X) |
| /addwstr, addnwstr, waddwstr, waddnwstr, | mvaddwstr, mvaddnwstr, mvwaddwstr,/ | curs_addwstr(3X) |
| /tparm, tputs, putp, vidputs, vidattr, | mvcur, tigetflag, tigetnum, tigetstr:/ | curs_terminfo(3X) |
| under/ curs_delch: delch, wdelch, | mvdelch, mvwdelch: delete character | curs_delch(3X) |
| /newwin, delwin, mvwin, subwin, derwin, | mvderwin, dupwin, wsyncup, syncok,/ | curs_window(3X) |
| | mvdir: move a directory | mvdir(1M) |
| back)/ /curs_getch: getch, wgetch, | mvgetch, mvwgetch, ungetch: get (or push | curs_getch(3X) |
| /getnstr, wgetstr, wgetnstr, mvgetstr, | mvgetnstr, mvwgetstr, mvwgetnstr: get/ | curs_getstr(3X) |
| wchar_t/ /wgetwstr, wgetnwstr, mvgetwstr, | mvgetnwstr, mvwgetwstr, mvwgetnwstr: get | curs_getwstr(3X) |
| /getstr, getnstr, wgetstr, wgetnstr, | mvgetstr, mvgetnstr, mvwgetstr,/ | curs_getstr(3X) |
| push/ /curs_getwch: getwch, wgetwch, | mvgetwch, mvwgetwch, ungetwch: get (or | curs_getwch(3X) |

| | | |
|---|---|---|
| /p2open, p2close: | open, close pipes to and from a command | p2open(3G) |
| apply or remove an advisory lock on an | open DG/UX file /dg_flock: | dg_flock(3C) |
| dup: duplicate an | open file descriptor | dup(2) |
| descriptor dup2: duplicate an | open file descriptor onto a specific | dup2(2) |
| open: | open file for reading or writing | open(2) |
| /getdtablesize: return the number of | open files the current process can have | getdtablesize(2) |
| | open: open file for reading or writing | open(2) |
| catopen, catclose: | open/close a message catalogue | catopen(3C) |
| rewinddir, closedir:/ directory: | opendir, readdir, telldir, seekdir, | directory(3X) |
| system log /syslog, | openlog, closelog, setlogmask: control | syslog(3C) |
| that the VSC synchronous controller is | operable /vsccheck: verify | vsccheck(1M) |
| admclient: manage | operating system clients | admclient(1M) |
| /syscon: DG/UX | operating system console pseudo-device | syscon(7) |
| prf: | operating system profiler | prf(7) |
| prfld, prfstat, prfdc, prfsnap, prfpr: | operating system profiler | profiler(1M) |
| reboot: restart the | operating system | reboot(1M) |
| msgsys: perform a message queue | operation | msgsys(2) |
| tgoto, tputs: terminal independent | operation routines /tgetflag, tgetstr, | termcap(3X) |
| semsys: perform a semaphore | operation | semsys(2) |
| shmsys: perform a shared memory | operation | shmsys(2) |
| /wstok, wstostr, strtows: wchar_t string | operations and type transformation | wstring(3W) |
| seekdir, rewinddir, closedir: directory | operations /opendir, readdir, telldir, | directory(3X) |
| dkctl: control special disk | operations | dkctl(1M) |
| ether_line: Ethernet address mapping | operations /ether_hostton, | ethers(3N) |
| memcmp, memcpy, memmove, memset: memory | operations /memory: memccpy, memchr, | memory(3C) |
| semctl: semaphore control | operations | semctl(2) |
| semop: semaphore | operations | semop(2) |
| shmctl: shared memory control | operations | shmctl(2) |
| strspn, strcspn, strtok, strstr: string | operations /strchr, strrchr, strpbrk, | string(3C) |
| join: relational database | operator | join(1) |
| /pseudo WORM (Write Once Read Multiple | optical device) as magtape interface | wmt(7) |
| curses: CRT screen handling and | optimization package | curses(3X) |
| typeahead: curses terminal input | option control routines /wtimeout, | curs_inopts(3X) |
| nl, nonl: curses terminal output | option control routines /scrollok, | curs_outopts(3X) |
| getopt: get | option letter from argument vector | getopt(3C) |
| field_opts_off, field_opts: forms field | option routines /field_opts_on, | form_field_opts(3X) |
| form_opts_off, form_opts: forms | option routines /form_opts_on, | form_opts(3X) |
| item_opts_off, item_opts: menus item | option routines /item_opts_on, | menu_item_opts(3X) |
| menu_opts_off, menu_opts: menus | option routines /menu_opts_on, | menu_opts(3X) |
| /ldohseek: seek to the | optional file header of an object file | ldohseek(3X) |
| processor(s) reboot: reboot halts and | optionally reboots the system | reboot(2) |
| fcntl: file control | options | fcntl(5) |
| stty: set the | options for a terminal | stty(1) |
| t_optmgmt: manage | options for a transport endpoint | t_optmgmt(3N) |
| getopt: parse command | options | getopt(1) |
| getopts, getoptcvt: parse command | options | getopts(1) |
| getsockopt: get | options on a socket | getsockopt(2) |
| setsockopt: set | options on sockets | setsockopt(2) |
| values between host and network byte | order /htons, ntohl, ntohs: convert | byteorder(3N) |
| databases /admsvcorder: manage search | order for /etc/hosts, NIS, and DNS | admsvcorder(1M) |
| postreverse: reverse the page | order in a PostScript file | postreverse(1) |
| /rev: reverse | order of characters in each line of file | rev(1) |
| /lorder: find | ordering relation for an object library | lorder(1) |
| t_rcvrel: acknowledge receipt of an | orderly release indication | t_rcvrel(3N) |
| t_sndrel: initiate an | orderly release | t_sndrel(3N) |
| filesystem: file system | organization | filesystem(7) |
| administration program | osysadm: menu-driven system | osysadm(1M) |
| dial: establish an | out-going terminal line connection | dial(3C) |
| a.out: common assembler and link editor | output | a.out(4) |
| concatenate and type files to standard | output /cat: | cat(1) |
| fold: fold long lines for finite width | output device | fold(1) |
| mvwprintw, vwprintw: print formatted | output in curses windows /mvprintw, | curs_printw(3X) |
| /vfprintf, vsprintf: print formatted | output of a variable argument list | vprintf(3S) |
| /vfprintf, vsprintf: print formatted | output of a variable argument list | vprintf(3W) |
| /scrollok, nl, nonl: curses terminal | output option control routines | curs_outopts(3X) |
| printf: print formatted | output | printf(1) |
| fprintf, sprintf: print formatted | output /printf, | printf(3S) |
| fprintf, sprintf: print formatted | output /printf, | printf(3W) |
| sysdef: | output system definition | sysdef(1M) |
| ttyadm: format and | output TTY port monitor information | ttyadm(1M) |
| windows /overlay, overwrite, copywin: | overlap and manipulate overlapped curses | curs_overlay(3X) |

| | | |
|---|---|---|
| copywin: overlap and manipulate | overlapped curses windows /overwrite, | curs_overlay(3X) |
| manipulate overlapped/ /curs_overlay: | overlay, overwrite, copywin: overlap and | curs_overlay(3X) |
| /acctdisk, acctdusg, accton, acctwtmp: | overview of accounting and miscellaneous/ | acct(1M) |
| manipulate/ /curs_overlay: overlay, | overwrite, copywin: overlap and | curs_overlay(3X) |
| chown, chgrp: change | owner or group | chown(1) |
| command /p2open, | p2close: open, close pipes to and from a | p2open(3G) |
| and from a command | p2open, p2close: open, close pipes to | p2open(3G) |
| files | pack, pcat, unpack: compress and expand | pack(1) |
| pkginfo: | package characteristics file | pkginfo(4) |
| pkgmap: | package contents description file | pkgmap(4) |
| CRT screen handling and optimization | package /curses: | curses(3X) |
| pkgtrans: translate | package format | pkgtrans(1) |
| forms: character based forms | package | forms(3X) |
| pkgrm: removes a | package from the system | pkgrm(1M) |
| prototype: | package information file | prototype(4) |
| pkginfo: display software | package information | pkginfo(1) |
| menus: character based menus | package | menus(3X) |
| panels: character based panels | package | panels(3X) |
| pkgparam: displays | package parameter values | pkgparam(1) |
| pkgmk: produce an installable | package | pkgmk(1) |
| sa1, sa2, sadc: system activity report | package /sar: | sar(1M) |
| stdio: standard buffered input/output | package | stdio(3S) |
| standard interprocess communication | package /stdipc: ftok: | stdipc(3C) |
| pkgadd: transfer software | package to the system | pkgadd(1M) |
| admpackage: manage DG/UX-style software | packages | admpackage(1M) |
| pckt: STREAMS | Packet Mode module | pckt(7) |
| /dn_expand: make, send, and interpret | packets to Internet domain name servers | resolver(3C) |
| pechowchar: create and display curses | pads /prefresh, pnoutrefresh, pechochar, | curs_pad(3X) |
| field_index: set forms current | page and field /current_field, | form_page(3X) |
| time more, | page: display file one screenful at a | more(1) |
| postreverse: reverse the | page order in a PostScript file | postreverse(1) |
| getpagesize: get the system | page size | getpagesize(2) |
| determine residency of memory | pages /dg_paging_info: | dg_paging_info(2) |
| mlock, munlock: lock (or unlock) | pages in memory | mlock(3C) |
| mincore: determine residency of memory | pages | mincore(2) |
| mmap: map | pages of memory | mmap(2) |
| munmap: unmap | pages of memory | munmap(2) |
| set_new_page, new_page: forms | pagination /form_new_page: | form_new_page(3X) |
| specify additional devices for system | paging /swapon: | swapon(1M) |
| swapon: add a swap device for demand | paging | swapon(2) |
| socketpair: create a | pair of connected sockets | socketpair(2) |
| master/slave pseudo-device | pair /pty, pts, ptc: pseudo-terminal | pty(7) |
| unlock a pseudo-terminal master/slave | pair /unlockpt: | unlockpt(3C) |
| /can_change_color, color_content, | pair_content: curses color manipulation/ | curs_color(3X) |
| associate application data with a panels | panel /set_panel_userptr, panel_userptr | panel_userptr(3X) |
| or set the current window of a panels | panel /panel_window, replace_panel: g | panel_window(3X) |
| panels deck traversal primitives | panel_above: panel_above, panel_below | panel_above(3X) |
| traversal primitives /panel_above: | panel_above, panel_below: panels deck | panel_above(3X) |
| primitives /panel_above: panel_above, | panel_below: panels deck traversal | panel_above(3X) |
| /panel_show: show_panel, hide_panel, | panel_hidden: panels deck manipulation/ | panel_show(3X) |
| window on the virtual screen | panel_move: move_panel: move a panels | panel_move(3X) |
| and destroy panels | panel_new: new_panel, del_panel: create | panel_new(3X) |
| | panels: character based panels package | panels(3X) |
| /show_panel, hide_panel, panel_hidden: | panels deck manipulation routines | panel_show(3X) |
| panel_top: top_panel, bottom_panel: | panels deck manipulation routines | panel_top(3X) |
| /panel_above: panel_above, panel_below: | panels deck traversal primitives | panel_above(3X) |
| panels: character based | panels package | panels(3X) |
| associate application data with a | panels panel /panel_userptr: | panel_userptr(3X) |
| get or set the current window of a | panels panel /replace_panel: | panel_window(3X) |
| new_panel, del_panel: create and destroy | panels /panel_new: | panel_new(3X) |
| /panel_update: update_panels: | panels virtual screen refresh routine | panel_update(3X) |
| panel_move: move_panel: move a | panels window on the virtual screen | panel_move(3X) |
| panel_hidden: panels deck manipulation/ | panel_show: show_panel, hide_panel, | panel_show(3X) |
| panels deck manipulation routines | panel_top: top_panel, bottom_panel: | panel_top(3X) |
| virtual screen refresh routine | panel_update: update_panels: panels | panel_update(3X) |
| panel_userptr: associate application/ | panel_userptr: set_panel_userptr, | panel_userptr(3X) |
| data/ /panel_userptr: set_panel_userptr, | panel_userptr: associate application | panel_userptr(3X) |
| replace_panel: get or set the current/ | panel_window: panel_window, | panel_window(3X) |
| the current window of a/ /panel_window: | panel_window, replace_panel: get or set | panel_window(3X) |
| pkgparam: displays package | parameter values | pkgparam(1) |
| /admtcpipparams: manage the TCP/IP host | parameters | admtcpipparams(1M) |

| | | |
|---|---|---|
| dg_sysctl: modify system | parameters | dg_sysctl(1M) |
| admtape: manipulate the default | parameters for tapes | admtape(1M) |
| tkey: set label and data translation | parameters | tkey(1) |
| name dirname: report the | parent directory name of a file path | dirname(3G) |
| getpgid: get process, process group, and | parent process IDs /getpgrp, getppid, | getpid(2) |
| getppid: get | parent process-id | getppid(2) |
| getopt: | parse command options | getopt(1) |
| getopts, getoptcvt: | parse command options | getopts(1) |
| getsubopt: | parse suboptions from a string | getsubopt(3C) |
| clrtoeol, wclrtoeol: clear all or | part of a curses window /wclrtobot, | curs_clear(3X) |
| tail: deliver the last | part of a file | tail(1) |
| shutdown: shut down | part of a full-duplex connection | shutdown(2) |
| file att_dump: dump | parts of an object or object archive | att_dump(1) |
| modff, nextafter, scalb: manipulate | parts of floating-point numbers /modf, | frexp(3C) |
| putmsg, putpmsg: | pass a message down a stream | putmsg(2) |
| nlsgetcall: get client's data | passed via the listener | nlsgetcall(3N) |
| | passmgmt: password files management | passmgmt(1M) |
| | passwd: change login password | passwd(1) |
| | passwd: password file | passwd(4) |
| bcs_cat: type hosts, networks, | passwd, protocols, group or services/ | bcs_cat(1M) |
| /crypt: | password and file encryption functions | crypt(3X) |
| admuser: manage user information in the | password database | admuser(1M) |
| dialups: devices requiring a dial-up | password. | dialups(4) |
| setpwfile, fgetpwent: manipulate | password file entry /setpwent, endpwent, | getpwent(3C) |
| lckpwdf, ulckpwdf: manipulate shadow | password file entry /fgetspent, | getspent(3C) |
| putpwent: write | password file entry | putpwent(3C) |
| putspent: write shadow | password file entry | putspent(3C) |
| passwd: | password file | passwd(4) |
| vipw: edit the system | password file | vipw(1M) |
| passmgmt: | password files management | passmgmt(1M) |
| getpass: read a | password | getpass(3C) |
| pwck, grpck: check | password or group file | pwck(1M) |
| passwd: change login | password | passwd(1) |
| d_passwd: log-in programs and | passwords for dial-up devices | d_passwd(4) |
| | paste: merge lines | paste(1) |
| rmdirp: create, remove directories in a | path /mkdirp, | mkdirp(3G) |
| basename: return the last element of a | path name | basename(3G) |
| the parent directory name of a file | path name /dirname: report | dirname(3G) |
| basename, dirname: deliver portions of | path names | basename(1) |
| pathname variables | pathconf, fpathconf: get configurable | pathconf(2) |
| directories | pathfind: search for named file in named | pathfind(3G) |
| display a prompt; verify and return a | pathname /ckpath: | ckpath(1) |
| getwd: get current working directory | pathname | getwd(3C) |
| getcwd: get | pathname of current working directory | getcwd(3C) |
| pathconf, fpathconf: get configurable | pathname variables | pathconf(2) |
| grep: search a file for a | pattern | grep(1) |
| menu_pattern: set and get menus | pattern match buffer /set_menu_pattern, | menu_pattern(3X) |
| gmatch: shell global | pattern matching | gmatch(3G) |
| /nawk, awk: | pattern scanning and processing language | nawk(1) |
| /oawk: old | pattern scanning and processing language | oawk(1) |
| /egrep: search a file for a | pattern using full regular expressions | egrep(1) |
| caught | pause: suspend process until a signal is | pause(2) |
| /pack, | pcat, unpack: compress and expand files | pack(1) |
| | pckt: STREAMS Packet Mode module | pckt(7) |
| /popen, | pclose: initiate pipe to/from a process | popen(3S) |
| value/ machid: dghost, m68k, m88k, i386, | pdp11, u3b, u3b5, vax: provide truth | machid(1) |
| /newpad, subpad, prefresh, pnoutrefresh, | pechochar, pechowchar: create and/ | curs_pad(3X) |
| pads /prefresh, pnoutrefresh, pechochar, | pechowchar: create and display curses | curs_pad(3X) |
| getpeername: get name of connected | peer | getpeername(2) |
| sigpending: examine | pending signals | sigpending(2) |
| lock database, start lock reclaim grace | period /dg_lock_reset: reset remote file | dg_lock_reset(2) |
| uucheck: check the uucp directories and | permissions file | uucheck(1M) |
| mesg: | permit or deny messages | mesg(1) |
| acct: | per-process accounting file format | acct(4) |
| acctcms: command summary from | per-process accounting records | acctcms(1M) |
| | perror: print system error messages | perror(3C) |
| screenful at a time | pg: display file forward or backward one | pg(1) |
| diskman: menu interface for managing | physical and logical disks | diskman(1M) |
| msync: synchronize memory with | physical storage | msync(3C) |
| split: split a file into | pieces | split(1) |
| | pipe: create an interprocess channel | pipe(2) |

| | | |
|---|---|---|
| files /posttek: | PostScript translator for tektronix 4014 | posttek(1) |
| tektronix 4014 files | posttek: PostScript translator for | posttek(1) |
| move, min,/ mp: madd, msub, mult, mdiv, | pow, gcd, invert, rpow, msqrt, mcmp, | mp(3X) |
| /expf, cbrt, log, logf, log10, log10f, | pow, powf, sqrt, sqrtf: exponential,/ | exp(3M) |
| sqrt, sqrtf: exponential, logarithm, | power, square root functions /pow, powf, | exp(3M) |
| cbrt, log, logf, log10, log10f, pow, | powf, sqrt, sqrtf: exponential,/ /expf, | exp(3M) |
| | pr: print files | pr(1) |
| /dodisk, lastlogin, monacct, nulladm, | prctmp, prdaily, prtacct, shutacct,/ | acctsh(1M) |
| /lastlogin, monacct, nulladm, prctmp, | prdaily, prtacct, shutacct, startup,/ | acctsh(1M) |
| fmout, m_out, sdiv, itom: multiple | precision integer arithmetic /omout, | mp(3X) |
| pechowchar:/ curs_pad: newpad, subpad, | prefresh, pnoutrefresh, pechochar, | curs_pad(3X) |
| monitor: | prepare execution profile | monitor(3C) |
| sifilter: | preprocess MC88100 assembly language | sifilter(1) |
| cpp: the C language | preprocessor | cpp(1) |
| signal: specify what to do upon | presentation of a signal | signal(2) |
| sigset: specify what to do upon | presentation of a signal | sigset(2) |
| sigvec: specify what to do upon | presentation of a signal | sigvec(2) |
| unget: undo a | previous get of an SCCS file | unget(1) |
| complete /dg_lock_wait: wait for | previously delayed lock requests to | dg_lock_wait(2) |
| | prf: operating system profiler | prf(7) |
| profiler /prfld, prfstat, | prfdc, prfsnap, prfpr: operating system | profiler(1M) |
| operating system profiler | prfld, prfstat, prfdc, prfsnap, prfpr: | profiler(1M) |
| prfld, prfstat, prfdc, prfsnap, | prfpr: operating system profiler | profiler(1M) |
| profiler prfld, prfstat, prfdc, | prfsnap, prfpr: operating system | profiler(1M) |
| operating system profiler prfld, | prfstat, prfdc, prfsnap, prfpr: | profiler(1M) |
| information used to distinguish | prime and non-prime days /accounting | holidays(4) |
| types: | primitive system data types | types(5) |
| panel_below: panels deck traversal | primitives /panel_above: panel_above, | panel_above(3X) |
| Server /termprinter: | print a file using the 40014A Terminal | termprinter(1) |
| /extended_perror: | print an error message to standard error | extended_perror(3C) |
| prs: | print an SCCS file | prs(1) |
| date: | print and set the date | date(1) |
| cal: | print calendar | cal(1) |
| /sum: | print checksum and block count of a file | sum(1) |
| development environment/ sde-target: | print commands to reset software | sde-target(1) |
| /sact: | print current SCCS file editing activity | sact(1) |
| /man: locate and | print entries from the reference manuals | man(1) |
| pr: | print files | pr(1) |
| /wprintw, mvprintw, mvwprintw, vwprintw: | print formatted output in curses windows | curs_printw(3X) |
| argument/ vprintf, vfprintf, vsprintf: | print formatted output of a variable | vprintf(3S) |
| argument/ vprintf, vfprintf, vsprintf: | print formatted output of a variable | vprintf(3W) |
| printf: | print formatted output | printf(1) |
| printf, fprintf, sprintf: | print formatted output | printf(3S) |
| printf, fprintf, sprintf: | print formatted output | printf(3W) |
| the LP print service lpstat: | print information about the status of | lpstat(1) |
| about RCS files /rlog: | print log messages and other information | rlog(1) |
| catgets: | print message from message catalog | catgets(1) |
| /nm: | print name list of common object file | nm(1) |
| uname: | print name of current system | uname(1) |
| news: | print news items | news(1) |
| infocmp: compare or | print out TERMINFO descriptions | infocmp(1M) |
| printenv: | print out the environment | printenv(1) |
| acctcom: search and | print process accounting file(s) | acctcom(1) |
| accept, reject: accept or reject | print requests | accept(1M) |
| /lpr: send | print requests to a line printer spooler | lpr(1) |
| size: | print section sizes of object files | size(1) |
| /lpshut, lpmove: start/stop the LP | print service and move requests | lpsched(1M) |
| cancel: send/cancel requests to an LP | print service /lp, | lp(1) |
| lpadmin: configure the LP | print service | lpadmin(1M) |
| administer filters used with the LP | print service /lpfilter: | lpfilter(1M) |
| administer forms used with the LP | print service /lpforms: | lpforms(1M) |
| information about the status of the LP | print service /lpstat: print | lpstat(1) |
| register remote systems with the | print service /lpsystem: | lpsystem(1M) |
| strace: | print STREAMS trace messages | strace(1M) |
| perror: | print system error messages | perror(3C) |
| name and ID /id: | print the user name and ID, and group | id(1) |
| pwd: | print working directory name | pwd(1) |
| binary file /strings: find the | printable strings in an object or other | strings(1) |
| | printcap: printer capability data base | printcap(5) |
| | printenv: print out the environment | printenv(1) |
| printcap: | printer capability data base | printcap(5) |

| | | |
|---|---|---|
| terminfo: terminal and | printer capability database | terminfo(4) |
| lpc: line | printer control program | lpc(1M) |
| Server /lptermprinter: start | printer session with 40014A Terminal | lptermprinter(1) |
| lp: DGC AViiON family line | printer special files | lp(7) |
| lpd: line | printer spooler | lpd(1M) |
| lpr: send print requests to a line | printer spooler | lpr(1) |
| lprm: remove jobs from the line | printer spooling queue | lprm(1) |
| troff postprocessor for PostScript | printers /dpost: | dpost(1) |
| enable, disable: enable/disable LP | printers | enable(1) |
| postio: serial interface for PostScript | printers | postio(1) |
| matrix display program for PostScript | printers /postmd: | postmd(1) |
| formatted output | printf, fprintf, sprintf: print | printf(3S) |
| formatted output | printf, fprintf, sprintf: print | printf(3W) |
| | printf: print formatted output | printf(1) |
| lpusers: set | printing queue priorities | lpusers(1M) |
| vwprintw: print formatted/ /curs_printw: | printw, wprintw, mvprintw, mvwprintw, | curs_printw(3X) |
| lpusers: set printing queue | priorities | lpusers(1M) |
| getpriority: get process scheduling | priority | getpriority(2) |
| nice: run a command at a higher or lower | priority | nice(1) |
| nice: change | priority of a process | nice(2) |
| renice: alter | priority of running processes | renice(1) |
| setpriority: set process scheduling | priority | setpriority(2) |
| probedev: | probe system for devices | probedev(1M) |
| | probedev: probe system for devices | probedev(1M) |
| library routines for remote | procedure calls /xprt_unregister: | rpc(3N) |
| shutacct, startup, turnacct: shell | procedures for accounting /prtacct, | acctsh(1M) |
| introduction to system maintenance | procedures /intro: | intro(8) |
| filehandle dg_lcntl: | process a record lock request on a | dg_lcntl(2) |
| acct: enable or disable | process accounting | acct(2) |
| acctprc1, acctprc2: | process accounting | acctprc(1M) |
| acctcom: search and print | process accounting file(s) | acctcom(1) |
| alarm: set a | process alarm clock | alarm(2) |
| times: get | process and child process times | times(2) |
| kill: terminate a | process by default | kill(1) |
| the number of open files the current | process can have /getdtablesize: return | getdtablesize(2) |
| the working directory of the calling | process /chdir: change | chdir(2) |
| change the root directory of the calling | process /chroot: | chroot(2) |
| init, telinit: | process control initialization | init(1M) |
| timex: time a command; report | process data and system activity | timex(1) |
| the extended errno for the current | process /dg_ext_errno: return | dg_ext_errno(2) |
| dg_kill: test for or terminate a | process | dg_kill(1) |
| exit, _exit: terminate | process | exit(2) |
| the working directory of the calling | process /fchdir: change | fchdir(2) |
| fork: create a new | process | fork(2) |
| /getpgrp, getppid, getpgid: get process, | process group, and parent process IDs | getpid(2) |
| getpgrp2: get | process group | getpgrp2(2) |
| setpgid: set | process group ID for job control | setpgid(2) |
| getpgrp: get | process group ID | getpgrp(2) |
| setsid: create session and set | process group ID | setsid(2) |
| tcgetpgrp: get foreground | process group ID | tcgetpgrp(3C) |
| tcsetpgrp: set terminal foreground | process group id | tcsetpgrp(3C) |
| killpg: send signal to a process or a | process group | killpg(2) |
| /get file usage information for | process identified by process key | dg_file_info(2) |
| get process, process group, and parent | process IDs /getpgrp, getppid, getpgid: | getpid(2) |
| way vfork: spawn new | process in a virtual memory efficient | vfork(2) |
| information for process identified by | process key /get file usage | dg_file_info(2) |
| kill: send a signal to a | process | kill(2) |
| nice: change priority of a | process | nice(2) |
| sigsend, sigsendset: send a signal to a | process or a group of processes | sigsend(2) |
| killpg: send signal to a | process or a process group | killpg(2) |
| popen, pclose: initiate pipe to/from a | process | popen(3S) |
| getpid, getpgrp, getppid, getpgid: get | process, process group, and parent/ | getpid(2) |
| set up execution time profiling for a | process /profil: | profil(2) |
| getpriority: get | process scheduling priority | getpriority(2) |
| setpriority: set | process scheduling priority | setpriority(2) |
| the effective group id of the current | process /setegid: set | setegid(2) |
| set the effective user id of the current | process /seteuid: | seteuid(2) |
| signal frame sigret: restore the | process state to that contained in a | sigret(2) |
| ps: report | process status | ps(1) |
| admlock: manage simple | process synchronization | admlock(1M) |
| wait, waitpid: wait for | process termination | wait(2) |

| | | |
|---|---|---|
| putmsg, | putpmsg: pass a message down a stream | putmsg(2) |
| | putpwent: write password file entry | putpwent(3C) |
| | puts, fputs: put a string on a stream | puts(3S) |
| entry | putspent: write shadow password file | putspent(3C) |
| /getut: getutent, getutid, getutline, | pututline, setutent, endutent, utmpname:/ | getut(3C) |
| /putc, putchar, fputc, | putw: put character or word on a stream | putc(3S) |
| character on a stream | putwc, putwchar, fputwc: put wchar_t | putwc(3W) |
| on a stream /putwc, | putwchar, fputwc: put wchar_t character | putwc(3W) |
| /unctrl, keyname, filter, use_env, | putwin, getwin, delay_output, flushinp:/ | curs_util(3X) |
| stream | putws, fputws: put a wchar_t string on a | putws(3W) |
| file | pwck, grpck: check password or group | pwck(1M) |
| | pwd: print working directory name | pwd(1) |
| /notimeout, raw, noraw, noqiflush, | qiflush, timeout, wtimeout, typeahead:/ | curs_inopts(3X) |
| | qsort: quicker sort | qsort(3C) |
| setlocale: modify and | query a program's locale | setlocale(3C) |
| default-gcc: set or | query default version of GNU C | default-gcc(1) |
| termattrs, termname: curses environment | query routines /killchar, longname, | curs_termattrs(3X) |
| strchg, strconf: change or | query stream configuration | strchg(1) |
| tput: initialize a terminal or | query terminfo database | tput(1) |
| alpq: | query the ALP STREAMS module | alpq(1) |
| queue msgctl: get or set message | queue attributes or destroy a message | msgctl(2) |
| msgget: get message | queue identifier | msgget(2) |
| remque: insert/remove element from a | queue /insque, | insque(3C) |
| lpq: examine the spool | queue | lpq(1) |
| jobs from the line printer spooling | queue /lprm: remove | lprm(1) |
| queue attributes or destroy a message | queue /msgctl: get or set message | msgctl(2) |
| msgsys: perform a message | queue operation | msgsys(2) |
| lpusers: set printing | queue priorities | lpusers(1M) |
| remove an element from a circular | queue /remque: | remque(3C) |
| ID /ipcrm: remove a message | queue, semaphore set, or shared memory | ipcrm(1) |
| atq: display the jobs | queued to run at specified times | atq(1) |
| qsort: | quicker sort | qsort(3C) |
| run a command immune to hangups and | quits /nohup: | nohup(1) |
| div, ldiv: compute the | quotient and remainder | div(3C) |
| | raise: send signal to program | raise(3C) |
| generator | rand, srand: simple random-number | rand(3C) |
| elf_rand: | random archive member access | elf_rand(3E) |
| /srandom, initstate, setstate: generate | random numbers better, or change the/ | random(3C) |
| generate random numbers better, or/ | random, srandom, initstate, setstate: | random(3C) |
| rand, srand: simple | random-number generator | rand(3C) |
| cfsetispeed, cfsetospeed: baud | rate functions /cfgetospeed, | cfsetospeed(3C) |
| fsplit: split f77 or | ratfor files | fsplit(1) |
| | ratfor: rational FORTRAN dialect | ratfor(1) |
| ratfor: | rational FORTRAN dialect | ratfor(1) |
| /keypad, meta, nodelay, notimeout, | raw, noraw, noqiflush, qiflush, timeout,/ | curs_inopts(3X) |
| returning a stream to a remote command | rcmd, rresvport, ruserok: routines for | rcmd(3X) |
| | rcs: change RCS file attributes | rcs(1) |
| rcsintro: introduction to | RCS commands | rcsintro(1) |
| rcs: change | RCS file attributes | rcs(1) |
| sccstorcs: build | RCS file from SCCS file | sccstorcs(1) |
| rcsfile: format of | RCS file | rcsfile(4) |
| log messages and other information about | RCS files /rlog: print | rlog(1) |
| ci: check in | RCS revisions | ci(1) |
| co: check out | RCS revisions | co(1) |
| rcsdiff: compare | RCS revisions | rcsdiff(1) |
| rcsmerge: merge | RCS revisions | rcsmerge(1) |
| | rcsdiff: compare RCS revisions | rcsdiff(1) |
| | rcsfile: format of RCS file | rcsfile(4) |
| | rcsintro: introduction to RCS commands | rcsintro(1) |
| | rcsmerge: merge RCS revisions | rcsmerge(1) |
| | rdsk: character special disk interface | rdsk(7) |
| getpass: | read a password | getpass(3C) |
| catgets: | read a program message | catgets(3C) |
| object file /ldtbread: | read an indexed symbol table entry of an | ldtbread(3X) |
| a common object/ /ldshread, ldnshread: | read an indexed/named section header of | ldshread(3X) |
| dump2label: | read and write labels for dump tapes | dump2label(1M) |
| /dg_unbuffered_read: synchronously | read data from a file without system/ | dg_unbuffered_read(2) |
| tread: | read file(s) from tape | tread(1) |
| read: | read from an object | read(2) |
| readv: | read from file | readv(2) |
| mail, rmail: | read mail or send mail to users | mail(1) |

| | | |
|---|---|---|
| facilities status ipcs: | report inter-process communication | ipcs(1) |
| inodes /df: | report number of free disk blocks and | df(1M) |
| tsniff: summary | report of tape contents | tsniff(1) |
| sar: sa1, sa2, sadc: system activity | report package | sar(1M) |
| /timex: time a command; | report process data and system activity | timex(1) |
| ps: | report process status | ps(1) |
| uniq: | report repeated lines in a file | uniq(1) |
| file path name dirname: | report the parent directory name of a | dirname(3G) |
| sar: system activity | reporter | sar(1) |
| manage system activity monitoring and | reporting /admsar: | admsar(1M) |
| fseek, rewind, ftell: | reposition a file pointer in a stream | fseek(3S) |
| library routines for external data | representation /xdr_wrapstring: | xdr(3N) |
| format and send listener service | request message /nlsrequest: | nlsrequest(3N) |
| dg_lcntl: process a record lock | request on a filehandle | dg_lcntl(2) |
| pkgask: stores answers to a | request script | pkgask(1M) |
| t_accept: accept a connect | request | t_accept(3N) |
| t_listen: listen for a connect | request | t_listen(3N) |
| receive the confirmation from a connect | request /t_rcvconnect: | t_rcvconnect(3N) |
| t_snddis: send user-initiated disconnect | request | t_snddis(3N) |
| accept, reject: accept or reject print | requests | accept(1M) |
| start a BIOD server for asynchronous I/O | requests /async_daemon: | async_daemon(2) |
| start/stop the LP print service and move | requests /lpsched, lpshut, lpmove: | lpsched(1M) |
| lpr: send print | requests to a line printer spooler | lpr(1) |
| lp, cancel: send/cancel | requests to an LP print service | lp(1) |
| wait for previously delayed lock | requests to complete /dg_lock_wait: | dg_lock_wait(2) |
| uuxqt: execute remote command | requests | uuxqt(1M) |
| space: disk space | requirement file | space(4) |
| dialups: devices | requiring a dial-up password. | dialups(4) |
| devreserv: | reserve devices for exclusive use | devreserv(1M) |
| lock reclaim grace/ /dg_lock_reset: | reset remote file lock database, start | dg_lock_reset(2) |
| sensible state | reset: reset the teletype bits to a | reset(1) |
| target /sde-target: print commands to | reset software development environment | sde-target(1) |
| state reset: | reset the teletype bits to a sensible | reset(1) |
| resetty,/ /def_prog_mode, def_shell_mode, | reset_prog_mode, reset_shell_mode, | curs_kernel(3X) |
| /def_shell_mode, reset_prog_mode, | reset_shell_mode, resetty, savetty,/ | curs_kernel(3X) |
| /reset_prog_mode, reset_shell_mode, | resetty, savetty, getsyx, setsyx,/ | curs_kernel(3X) |
| /dg_paging_info: determine | residency of memory pages | dg_paging_info(2) |
| mincore: determine | residency of memory pages | mincore(2) |
| sync: synchronize disk and memory | resident file system information | sync(2) |
| download: download host | resident PostScript fonts | download(1) |
| controller /vscload: download board | resident software onto VSC synchronous | vscload(1M) |
| send, and/ res_mkquery, res_send, | res_init, dn_comp, dn_expand: make, | resolver(3C) |
| dn_comp, dn_expand: make, send, and/ | res_mkquery, res_send, res_init, | resolver(3C) |
| database /admresolve: manage DNS | resolver's domain name and nameservers | admresolve(1M) |
| setrlimit: control maximum system | resource consumption /getrlimit, | getrlimit(2) |
| vlimit: control maximum system | resource consumption | vlimit(3C) |
| directory /dg_set_cpd_limits: change the | resource limits of a control point | dg_set_cpd_limits(2) |
| vtimes: get information about | resource usage | vtimes(3C) |
| getrusage: get information about | resource utilization | getrusage(2) |
| vacation: automatically | respond to incoming mail messages | vacation(1) |
| make, send, and interpret/ /res_mkquery, | res_send, res_init, dn_comp, dn_expand: | resolver(3C) |
| reboot: | restart the operating system | reboot(1M) |
| /setterm, set_curterm, del_curterm, | restartterm, tparm, tputs, putp,/ | curs_terminfo(3X) |
| restore: incrementally | restore a file system | restore(1M) |
| system | restore: incrementally restore a file | restore(1M) |
| contained in a signal frame sigret: | restore the process state to that | sigret(2) |
| admrshell: manage the remote and | restricted shell names | admrshell(1M) |
| language /sh, jsh, rsh, | restsh: shell, the command programming | sh(1) |
| examples /usage: | retrieve a command description and usage | usage(1) |
| data base gettxt: | retrieve a text string from a message | gettxt(1) |
| gettxt: | retrieve a text string | gettxt(3C) |
| /elf_getarhdr: | retrieve archive member header | elf_getarhdr(3E) |
| /elf_getarsym: | retrieve archive symbol table | elf_getarsym(3E) |
| header /elf32_getehdr, elf32_newehdr: | retrieve class-dependent object file | elf_getehdr(3E) |
| table /elf32_getphdr, elf32_newphdr: | retrieve class-dependent program header | elf_getphdr(3E) |
| /elf_getshdr: elf32_getshdr: | retrieve class-dependent section header | elf_getshdr(3E) |
| /elf_getident: | retrieve file identification data | elf_getident(3E) |
| t_rcvdis: | retrieve information from disconnect | t_rcvdis(3N) |
| symbol table entry ldgetname: | retrieve symbol name for object file | ldgetname(3X) |
| /elf_rawfile: | retrieve uninterpreted file contents | elf_rawfile(3E) |
| ckitem: build a menu; prompt for and | return a menu item | ckitem(1) |

| | | |
|---|---|---|
| ckpath: display a prompt; verify and | return a pathname | ckpath(1) |
| ckstr: display a prompt; verify and | return a string answer | ckstr(1) |
| cktime: display a prompt; verify and | return a time of day | cktime(1) |
| ckint: display a prompt; verify and | return an integer value | ckint(1) |
| /fstatvfs: | return information about a file system | fstatvfs(2) |
| /statvfs: | return information about a file system | statvfs(2) |
| abs, labs: | return integer absolute value | abs(3C) |
| logname: | return login name of user | logname(3X) |
| rexec: | return stream to a remote command | rexec(3X) |
| processor status register getpsr: | return the current contents of the | getpsr(2) |
| current process /dg_ext_errno: | return the extended errno for the | dg_ext_errno(2) |
| entry containing filename getfh: | return the file handle of the export | getfh(2) |
| /basename: | return the last element of a path name | basename(3G) |
| current process can have /getdtablesize: | return the number of open files the | getdtablesize(2) |
| /elf_fsize: elf32_fsize: | return the size of an object file type | elf_fsize(3E) |
| getenv: | return value for environment name | getenv(3C) |
| call /dg_stat: data | returned by dg_stat and dg_fstat system | dg_stat(5) |
| stat: data | returned by stat system call | stat(5) |
| dg_mknod: data | returned by the dg_mknod system call | dg_mknod(5) |
| statfs: data | returned by the statfs system call | statfs(5) |
| ustat: data | returned by the ustat system call | ustat(5) |
| /rcmd, rresvport, ruserok: routines for | returning a stream to a remote command | rcmd(3X) |
| types sysfs: | returns information about file system | sysfs(2) |
| realpath: | returns the real file name | realpath(3C) |
| line of file | rev: reverse order of characters in each | rev(1) |
| col: filter | reverse line-feeds | col(1) |
| of file /rev: | reverse order of characters in each line | rev(1) |
| file /postreverse: | reverse the page order in a PostScript | postreverse(1) |
| ci: check in RCS | revisions | ci(1) |
| co: check out RCS | revisions | co(1) |
| rcsdiff: compare RCS | revisions | rcsdiff(1) |
| rcsmerge: merge RCS | revisions | rcsmerge(1) |
| in a stream /fseek, | rewind, ftell: reposition a file pointer | fseek(3S) |
| /opendir, readdir, telldir, seekdir, | rewinddir, closedir: directory/ | directory(3X) |
| creat: create a new file or | rewrite an existing one | creat(2) |
| | rexec: return stream to a remote command | rexec(3X) |
| of a character in a string | rindex: search for the last occurrence | rindex(3C) |
| copysign, fmod, fmodf, fabs, fabsf, | rint, remainder: floor, ceiling,/ /ceilf, | floor(3M) |
| /resetty, savetty, getsyx, setsyx, | ripoffline, curs_set, napms: low-level/ | curs_kernel(3X) |
| command and programming language /ksh, | rksh: KornShell, a standard/restricted | ksh(1) |
| information about RCS files | rlog: print log messages and other | rlog(1) |
| directories | rm, rmdir: remove, delete files or | rm(1) |
| initialization information for mail and | rmail /mailcnfg: | mailcnfg(4M) |
| /mail, | rmail: read mail or send mail to users | mail(1) |
| | rmdel: remove a delta from an SCCS file | rmdel(1) |
| | rmdir: remove a directory file | rmdir(2) |
| directories rm, | rmdir: remove, delete files or | rm(1) |
| path /mkdirp, | rmdirp: create, remove directories in a | mkdirp(3G) |
| interface | rmt: character special magnetic tape | rmt(7) |
| | rmt: start the remote mag tape server | rmt(1M) |
| chroot: change | root directory for a command | chroot(1M) |
| chroot: change the | root directory of the calling process | chroot(2) |
| exponential, logarithm, power, square | root functions /pow, powf, sqrt, sqrtf: | exp(3M) |
| /dg_getrootkey: get | root's secret key | dg_getrootkey(2) |
| atexit: add program termination | routine | atexit(3C) |
| panels virtual screen refresh | routine /panel_update: update_panels: | panel_update(3X) |
| character and window attribute control | routines /standout, wstandout: curses | curs_attr(3X) |
| flash: curses bell and screen flash | routines /curs_beep: beep, | curs_beep(3X) |
| curses window background manipulation | routines /wbkgdset, bkgd, wbkgd: | curs_bkgd(3X) |
| pair_content: curses color manipulation | routines /color_content, | curs_color(3X) |
| screen initialization and manipulation | routines /set_term, delscreen: curses | curs_initscr(3X) |
| curses terminal input option control | routines /timeout, wtimeout, typeahead: | curs_inopts(3X) |
| curs_set, napms: low-level curses | routines /getsyx, setsyx, ripoffline, | curs_kernel(3X) |
| curses terminal output option control | routines /scrollok, nl, nonl: | curs_outopts(3X) |
| slk_attroff: curses soft label | routines /slk_attron, slk_attrset, | curs_slk(3X) |
| termname: curses environment query | routines /killchar, longname, termattrs, | curs_termattrs(3X) |
| is_wintouched: curses refresh control | routines /wtouchln, is_linetouched, | curs_touch(3X) |
| flushinp: miscellaneous curses utility | routines /putwin, getwin, delay_output, | curs_util(3X) |
| /menu_term: assign application-specific | routines for automatic invocation by/ | menu_hook(3X) |
| xdr_void, xdr_wrapstring: library | routines for external data/ /xdr_vector, | xdr(3N) |
| /field_term: assign application-specific | routines for invocation by forms | form_hook(3X) |

| | | |
|---|---|---|
| /xprt_register, xprt_unregister: library | routines for remote procedure calls | rpc(3N) |
| remote/ rcmd, rresvport, ruserok: | routines for returning a stream to a | rcmd(3X) |
| field_opts: forms field option | routines /field_opts_on, field_opts_off, | form_field_opts(3X) |
| link_fieldtype: forms fieldtype | routines /set_fieldtype_choice, | form_fieldtype(3X) |
| form_opts_off, form_opts: forms option | routines /set_form_opts, form_opts_on, | form_opts(3X) |
| forms window and subwindow association | routines /form_sub, scale_form: | form_win(3X) |
| unordered, copysign: IEEE floating-point | routines /finite, | ieeefp(3C) |
| Internet address manipulation | routines /inet_lnaof, inet_netof: | inet(3N) |
| ldfcn: COFF executable file access | routines | ldfcn(4) |
| item_opts: menus item option | routines /item_opts_on, item_opts_off, | menu_item_opts(3X) |
| menu_mark: menus mark string | routines /menu_mark: set_menu_mark, | menu_mark(3X) |
| menu_opts_off, menu_opts: menus option | routines /set_menu_opts, menu_opts_on, | menu_opts(3X) |
| menus window and subwindow association | routines /menu_sub, scale_menu: | menu_win(3X) |
| panel_hidden: panels deck manipulation | routines /show_panel, hide_panel, | panel_show(3X) |
| bottom_panel: panels deck manipulation | routines /panel_top: top_panel, | panel_top(3X) |
| regular expression compile and match | routines /compile, step, advance: | regexp(5) |
| regular expression compile and match | routines /compile, step, advance: | regexpr(3G) |
| tputs: terminal independent operation | routines /tgetflag, tgetstr, tgoto, | termcap(3X) |
| widec: multibyte character I/O | routines | widec(3W) |
| mailsurr: surrogate commands for | routing and transport of mail | mailsurr(4M) |
| admroute: manage | routing databases | admroute(1M) |
| set and get maximum numbers of | rows and columns in menus /menu_format: | menu_format(3X) |
| setrpcent, endrpcent: get | RPC entry /getrpcbyname, getrpcbynumber, | getrpcent(3N) |
| getrpcport: get | RPC port number | getrpcport(3R) |
| /msub, mult, mdiv, pow, gcd, invert, | rpow, msqrt, mcmp, move, min, omin,/ | mp(3X) |
| returning a stream to a remote/ rcmd, | rresvport, ruserok: routines for | rcmd(3X) |
| programming language sh, jsh, | rsh, restsh: shell, the command | sh(1) |
| | rtime: get remote time | rtime(3N) |
| priority nice: | run a command at a higher or lower | nice(1) |
| quits nohup: | run a command immune to hangups and | nohup(1) |
| atq: display the jobs queued to | run at specified times | atq(1) |
| runacct: | run daily accounting | runacct(1M) |
| | runacct: run daily accounting | runacct(1M) |
| renice: alter priority of | running processes | renice(1) |
| to a remote command /rcmd, rresvport, | ruserok: routines for returning a stream | rcmd(3X) |
| package /sar: | sa1, sa2, sadc: system activity report | sar(1M) |
| package sar: sa1, | sa2, sadc: system activity report | sar(1M) |
| | sac: service access controller | sac(1M) |
| administration | sacadm: service access controller | sacadm(1M) |
| activity | sact: print current SCCS file editing | sact(1) |
| | sad: STREAMS Administrative Driver | sad(7) |
| sar: sa1, sa2, | sadc: system activity report package | sar(1M) |
| report package | sar: sa1, sa2, sadc: system activity | sar(1M) |
| | sar: system activity reporter | sar(1) |
| setgid: set the real-, effective-, and | saved-group-ids | setgid(2) |
| setregid: set the real-, effective-, and | saved-group-ids | setregid(2) |
| setreuid: set the real-, effective-, and | saved-user-ids | setreuid(2) |
| setuid: set the real-, effective-, and | saved-user-ids | setuid(2) |
| curs_set,/ /reset_shell_mode, resetty, | savetty, getsyx, setsyx, ripoffline, | curs_kernel(3X) |
| allocation | sbrk: change data segment space | sbrk(2) |
| ldexp, logb, modf, modff, nextafter, | scalb: manipulate parts of/ /frexp, | frexp(3C) |
| /form_win, set_form_sub, form_sub, | scale_form: forms window and subwindow/ | form_win(3X) |
| /menu_win, set_menu_sub, menu_sub, | scale_menu: menus window and subwindow/ | menu_win(3X) |
| scandir, alphasort: | scan a directory | scandir(3C) |
| | scandir, alphasort: scan a directory | scandir(3C) |
| input | scanf, fscanf, sscanf: convert formatted | scanf(3S) |
| input | scanf, fscanf, sscanf: convert formatted | scanf(3W) |
| bfs: big file | scanner | bfs(1) |
| nawk, awk: pattern | scanning and processing language | nawk(1) |
| oawk: old pattern | scanning and processing language | oawk(1) |
| vwscanw: convert formatted/ curs_scanw: | scanw, wscanw, mvscanw, mvwscanw, | curs_scanw(3X) |
| cdc: change the delta commentary of an | SCCS delta | cdc(1) |
| comb: combine | SCCS deltas | comb(1) |
| delta: make a delta (change) to an | SCCS file | delta(1) |
| sact: print current | SCCS file editing activity | sact(1) |
| get: check out a version of an | SCCS file | get(1) |
| prs: print an | SCCS file | prs(1) |
| rmdel: remove a delta from an | SCCS file | rmdel(1) |
| sccsdiff: compare two versions of an | SCCS file | sccsdiff(1) |
| sccsfile: format of | SCCS file | sccsfile(4) |
| sccstorcs: build RCS file from | SCCS file | sccstorcs(1) |

| | | |
|---|---|---|
| unget: undo a previous get of an | SCCS file | unget(1) |
| val: validate | SCCS file | val(1) |
| admin: create and administer | SCCS files | admin(1) |
| what: identify | SCCS files | what(1) |
| SCCS file | sccsdiff: compare two versions of an | sccsdiff(1) |
| | sccsfile: format of SCCS file | sccsfile(4) |
| | sccstorcs: build RCS file from SCCS file | sccstorcs(1) |
| program uusched: the | scheduler for the uucp file transport | uusched(1M) |
| getpriority: get process | scheduling priority | getpriority(2) |
| setpriority: set process | scheduling priority | setpriority(2) |
| file | scr_dump: format of curses screen image | scr_dump(4) |
| scr_set: read (write) a/ /curs_scr_dump: | scr_dump, scr_restore, scr_init, | curs_scr_dump(3X) |
| clear: clear terminal | screen | clear(1) |
| curs_beep: beep, flash: curses bell and | screen flash routines | curs_beep(3X) |
| scr_init, scr_set: read (write) a curses | screen from (to) a file /scr_restore, | curs_scr_dump(3X) |
| /curses: CRT | screen handling and optimization package | curses(3X) |
| scr_dump: format of curses | screen image file | scr_dump(4) |
| /isendwin, set_term, delscreen: curses | screen initialization and manipulation/ | curs_initscr(3X) |
| move a panels window on the virtual | screen /panel_move: move_panel: | panel_move(3X) |
| update_panels: panels virtual | screen refresh routine /panel_update: | panel_update(3X) |
| more, page: display file one | screenful at a time | more(1) |
| pg: display file forward or backward one | screenful at a time | pg(1) |
| based on ex /vi, vedit, view: | screen-oriented (visual) display editor | vi(1) |
| /curs_scr_dump: scr_dump, scr_restore, | scr_init, scr_set: read (write) a curses/ | curs_scr_dump(3X) |
| doconfig: execute a configuration | script | doconfig(3N) |
| inittab: | script for init | inittab(4) |
| session | script: make typescript of a terminal | script(1) |
| pkgask: stores answers to a request | script | pkgask(1M) |
| /curs_scroll: scroll, srcl, wscrl: | scroll a curses window | curs_scroll(3X) |
| window /curs_scroll: | scroll, srcl, wscrl: scroll a curses | curs_scroll(3X) |
| /immedok, leaveok, setscrreg, wsetscrreg, | scrollok, nl, nonl: curses terminal/ | curs_outopts(3X) |
| (write) a/ /curs_scr_dump: scr_dump, | scr_restore, scr_init, scr_set: read | curs_scr_dump(3X) |
| from/ /scr_dump, scr_restore, scr_init, | scr_set: read (write) a curses screen | curs_scr_dump(3X) |
| cisc: AViiON family | SCSI adapter subsystem | cisc(7) |
| insc: AViiON family | SCSI adapter subsystem | insc(7) |
| | sd: AViiON family disk subsystem | sd(7) |
| | sdb: symbolic debugger | sdb(1) |
| | sde: software development environment | sde(5) |
| environment-sensitive tool | sde-chooser: execute | sde-chooser(4) |
| data base | sdetab: software development environment | sdetab(4) |
| software development environment target | sde-target: print commands to reset | sde-target(1) |
| | sdiff: side-by-side difference program | sdiff(1) |
| /fmin, m_in, mout, omout, fmout, m_out, | sdiv, itom: multiple precision integer/ | mp(3X) |
| fgrep: | search a file for a character string | fgrep(1) |
| grep: | search a file for a pattern | grep(1) |
| regular expressions /egrep: | search a file for a pattern using full | egrep(1) |
| bsearch: binary | search a sorted table | bsearch(3C) |
| file(s) acctcom: | search and print process accounting | acctcom(1) |
| lsearch, lfind: linear | search and update | lsearch(3C) |
| data/ srchtxt: display contents of, or | search for a text string in, message | srchtxt(1) |
| directories pathfind: | search for named file in named | pathfind(3G) |
| character in a string index: | search for the first occurrence of a | index(3C) |
| character in a string rindex: | search for the last occurrence of a | rindex(3C) |
| ttysrch: directory | search list for ttyname | ttysrch(4M) |
| DNS databases /admsvcorder: manage | search order for /etc/hosts, NIS, and | admsvcorder(1M) |
| hsearch, hcreate, hdestroy: manage hash | search tables | hsearch(3C) |
| tfind, tdelete, twalk: manage binary | search trees /tsearch, | tsearch(3C) |
| /dg_getrootkey: get root's | secret key | dg_getrootkey(2) |
| /dg_setsecretkey: store a client's | secret key in the keyserver | dg_setsecretkey(2) |
| elf_newdata, elf_rawdata: get | section data /elf_getdata, | elf_getdata(3E) |
| elf32_getshdr: retrieve class-dependent | section header /elf_getshdr: | elf_getshdr(3E) |
| /ldnshread: read an indexed/named | section header of a common object file | ldshread(3X) |
| elf_ndxscn, elf_newscn, elf_nextscn: get | section information /elf_getscn, | elf_getscn(3E) |
| /seek to line number entries of a | section of a common object file | ldlseek(3X) |
| /seek to relocation entries of a | section of a common object file | ldrseek(3X) |
| /ldnsseek: seek to an indexed/named | section of a common object file | ldsseek(3X) |
| mcs: manipulate the comment | section of an object file. | mcs(1) |
| size: print | section sizes of object files | size(1) |
| | sed: stream editor | sed(1) |
| /nrand48, mrand48, jrand48, srand48, | seed48, lcong48: generate uniformly/ | drand48(3C) |
| dg_seek, dg_block_seek: extended | seek functions | dg_seek(3C) |

| | | |
|---|---|---|
| common object file /ldsseek, ldnsseek: | seek to an indexed/named section of a | ldsseek(3X) |
| of a common object/ /ldlseek, ldnlseek: | seek to line number entries of a section | ldlseek(3X) |
| of a common object/ /ldrseek, ldnrseek: | seek to relocation entries of a section | ldrseek(3X) |
| object file /ldohseek: | seek to the optional file header of an | ldohseek(3X) |
| file ldtbseek: | seek to the symbol table of an object | ldtbseek(3X) |
| /directory: opendir, readdir, telldir, | seekdir, rewinddir, closedir: directory/ | directory(3X) |
| shmat: attach a shared memory | segment | shmat(2) |
| shmdt: detach a shared memory | segment | shmdt(2) |
| shmget: get shared memory | segment | shmget(2) |
| brk: change data | segment space allocation | brk(2) |
| sbrk: change data | segment space allocation | sbrk(2) |
| sorted files comm: | select or reject lines common to two | comm(1) |
| | select: wait for I/O conditions | select(2) |
| /cut: cut out | selected fields of each line of a file | cut(1) |
| semctl: | semaphore control operations | semctl(2) |
| semsys: perform a | semaphore operation | semsys(2) |
| semop: | semaphore operations | semop(2) |
| ipcrm: remove a message queue, | semaphore set, or shared memory ID | ipcrm(1) |
| semget: get a set of | semaphores | semget(2) |
| | semctl: semaphore control operations | semctl(2) |
| | semget: get a set of semaphores | semget(2) |
| | semop: semaphore operations | semop(2) |
| | semsys: perform a semaphore operation | semsys(2) |
| t_sndudata: | send a data unit | t_sndudata(3N) |
| send: | send a message from a socket | send(2) |
| sendmsg: | send a message from a socket | sendmsg(2) |
| sendto: | send a message from a socket | sendto(2) |
| msgsnd: | send a message | msgsnd(2) |
| kill: | send a signal to a process | kill(2) |
| processes /sigsend, sigsendset: | send a signal to a process or a group of | sigsend(2) |
| /res_init, dn_comp, dn_expand: make, | send, and interpret packets to Internet/ | resolver(3C) |
| connection t_snd: | send data or expedited data over a | t_snd(3N) |
| nlsrequest: format and | send listener service request message | nlsrequest(3N) |
| mail, rmail: read mail or | send mail to users | mail(1) |
| spooler /lpr: | send print requests to a line printer | lpr(1) |
| | send: send a message from a socket | send(2) |
| group killpg: | send signal to a process or a process | killpg(2) |
| raise: | send signal to program | raise(3C) |
| /t_snddis: | send user-initiated disconnect request | t_snddis(3N) |
| service lp, cancel: | send/cancel requests to an LP print | lp(1) |
| | sendmsg: send a message from a socket | sendmsg(2) |
| | sendto: send a message from a socket | sendto(2) |
| reset: reset the teletype bits to a | sensible state | reset(1) |
| elink: Environment variable | sensitive file link | elink(5) |
| t_rcv: receive data or expedited data | sent over a connection | t_rcv(3N) |
| elf_next: | sequential archive member access | elf_next(3E) |
| /postio: | serial interface for PostScript printers | postio(1) |
| grfx: AViiON | series workstation graphics processor | grfx(7) |
| kbd: AViiON | series workstation system keyboard | kbd(7) |
| in.fingerd: remote user information | server /fingerd, | fingerd(1M) |
| /async_daemon: start a BIOD | server for asynchronous I/O requests | async_daemon(2) |
| listen: network listener | server | listen(1M) |
| printer session with 40014A Terminal | Server /lptermprinter: start | lptermprinter(1) |
| nfssvc: start an NFS | server on a specified socket | nfssvc(2) |
| rmt: start the remote mag tape | server | rmt(1M) |
| strerr: STREAMS error logger | server | strerr(1M) |
| file for syslogd system log | server /syslog.conf: configuration | syslog.conf(5) |
| print a file using the 40014A Terminal | Server /termprinter: | termprinter(1) |
| start the WORM magnetic tape device | server /wmtd: | wmtd(1M) |
| /admtcpipdaemon: manage the TCP/IP | servers | admtcpipdaemon(1M) |
| biod: start block I/O | servers | biod(1M) |
| packets to Internet domain name | servers /make, send, and interpret | resolver(3C) |
| /sacadm: | service access controller administration | sacadm(1M) |
| sac: | service access controller | sac(1M) |
| nlsadmin: network listener | service administration | nlsadmin(1M) |
| lpshut, lpmove: start/stop the LP print | service and move requests /lpsched, | lpsched(1M) |
| calendar: reminder | service | calendar(1) |
| ypprot_err: Network Information | Service client interface /yperr_string, | ypclnt(3N) |
| admservice: manage | service database | admservice(1M) |
| setservent, endservent: get | service entry /getservbyname, | getservent(3N) |
| t_getinfo: get protocol-specific | service information | t_getinfo(3N) |

| | | |
|---|---|---|
| send/cancel requests to an LP print | service /lp, cancel: | lp(1) |
| lpadmin: configure the LP print | service | lpadmin(1M) |
| filters used with the LP print | service /lpfilter: administer | lpfilter(1M) |
| administer forms used with the LP print | service /lpforms: | lpforms(1M) |
| about the status of the LP print | service /lpstat: print information | lpstat(1) |
| register remote systems with the print | service /lpsystem: | lpsystem(1M) |
| nlsrequest: format and send listener | service request message | nlsrequest(3N) |
| /admportservice: manage port monitor | services | admportservice(1M) |
| networks, passwd, protocols, group or | services information /type hosts, | bcs_cat(1M) |
| /admxterminal: manage | serving of X display terminals | admxterminal(1M) |
| setsid: create | session and set process group ID | setsid(2) |
| getsid: get | session ID | getsid(2) |
| script: make typescript of a terminal | session | script(1) |
| /lptermprinter: start printer | session with 40014A Terminal Server | lptermprinter(1) |
| alarm: | set a process alarm clock | alarm(2) |
| /set_top_row, top_row, item_index: | set and get current menus items | menu_item_current(3X) |
| umask: | set and get file creation mask | umask(2) |
| /field_status, set_max_field: | set and get forms field attributes | form_field_buffer(3X) |
| columns/ /set_menu_format, menu_format: | set and get maximum numbers of rows and | menu_format(3X) |
| /set_item_value, item_value: | set and get menus item values | menu_item_value(3X) |
| /set_menu_pattern, menu_pattern: | set and get menus pattern match buffer | menu_pattern(3X) |
| sigstack: | set and/or get signal stack context | sigstack(2) |
| ascii: map of ASCII character | set | ascii(5) |
| ffs: find first | set bit | ffs(3C) |
| until a signal is caught /berk_sigpause: | set blocked signals and suspend process | berk_sigpause(2) |
| classify ASCII and supplemetary code | set characters /isnumber, isspecial: | wctype(3W) |
| iconv: code | set conversion | iconv(1) |
| getcontext, setcontext: get and | set current user context | getcontext(2) |
| /settimeofday: | set date and time | settimeofday(2) |
| /timezone: | set default system time zone and locale | timezone(4) |
| /env: | set environment for command execution | env(1) |
| /utime: | set file access and modification times | utime(2) |
| /utimes: | set file access and modification times | utimes(2) |
| umask: | set file-creation mode mask | umask(1) |
| elf_fill: | set fill byte | elf_fill(3E) |
| /current_field, field_index: | set forms current page and field | form_page(3X) |
| parameters tkey: | set label and data translation | tkey(1) |
| a message queue /msgctl: get or | set message queue attributes or destroy | msgctl(2) |
| kbdset: attach to kbd mapping tables, | set modes | kbdset(1) |
| /setdomainname: | set name of current domain | setdomainname(2) |
| sethostname: | set name of current host | sethostname(2) |
| sigblock: add to | set of blocked signals | sigblock(2) |
| add a signal to the calling process's | set of blocked signals /sighold: | sighold(2) |
| a signal from the calling process's | set of blocked signals /sigrelse: remove | sigrelse(2) |
| sigsetmask: specify | set of blocked signals | sigsetmask(2) |
| sigfillset: fill in the | set of implementation-defined signals | sigfillset(2) |
| semget: get a | set of semaphores | semget(2) |
| setsockopt: | set options on sockets | setsockopt(2) |
| eucset: | set or get EUC code set widths | eucset(1) |
| context sigaltstack: | set or get signal alternate stack | sigaltstack(2) |
| default-gcc: | set or query default version of GNU C | default-gcc(1) |
| ipcrm: remove a message queue, semaphore | set, or shared memory ID | ipcrm(1) |
| lpusers: | set printing queue priorities | lpusers(1M) |
| setpgid: | set process group ID for job control | setpgid(2) |
| setsid: create session and | set process group ID | setsid(2) |
| setpriority: | set process scheduling priority | setpriority(2) |
| setpgrp: | set process-group-id | setpgrp(2) |
| setpgrp2: | set process-group-id | setpgrp2(2) |
| memctl: | set protection of memory mapping | memctl(2) |
| mprotect: | set protection of memory mapping | mprotect(2) |
| tcgetattr, tcsetattr: get and | set state | tcsetattr(3C) |
| /getgroups, setgroups: get or | set supplementary group access list IDs | getgroups(2) |
| sysinfo: get and | set system information strings | sysinfo(2) |
| tabs: | set tabs on a terminal | tabs(1) |
| /tcsetpgrp: | set terminal foreground process group id | tcsetpgrp(3C) |
| line discipline getty: | set terminal type, modes, speed, and | getty(1M) |
| /panel_window, replace_panel: get or | set the current window of a panels panel | panel_window(3X) |
| date: print and | set the date | date(1) |
| current process setegid: | set the effective group id of the | setegid(2) |
| process /seteuid: | set the effective user id of the current | seteuid(2) |
| stty: | set the options for a terminal | stty(1) |

| | | |
|---|---|---|
| setpsr: | set the processor status register | setpsr(2) |
| saved-group-ids setgid: | set the real-, effective-, and | setgid(2) |
| saved-group-ids setregid: | set the real-, effective-, and | setregid(2) |
| saved-user-ids setreuid: | set the real-, effective-, and | setreuid(2) |
| saved-user-ids setuid: | set the real-, effective-, and | setuid(2) |
| 'ignore' sigignore: | set the signal action of a signal to | sigignore(2) |
| stime: | set time | stime(2) |
| sethostid: | set unique identifier of current host | sethostid(2) |
| process profil: | set up execution time profiling for a | profil(2) |
| ulimit: get and | set user limits | ulimit(2) |
| getitimer, setitimer: get or | set value of interval timer | getitimer(2) |
| eucset: set or get EUC code | set widths | eucset(1) |
| stream | setbuf, setvbuf: assign buffering to a | setbuf(3S) |
| specified stream | setbuffer: assign a buffer to a | setbuffer(3C) |
| context getcontext, | setcontext: get and set current user | getcontext(2) |
| form_page: set_form_page, form_page, | set_current_field, current_field,/ | form_page(3X) |
| set_top_row,/ /menu_item_current: | set_current_item, current_item, | menu_item_current(3X) |
| /curs_terminfo: setupterm, setterm, | set_curterm, del_curterm, restartterm,/ | curs_terminfo(3X) |
| domain | setdomainname: set name of current | setdomainname(2) |
| the current process | setegid: set the effective group id of | setegid(2) |
| the current process | seteuid: set the effective user id of | seteuid(2) |
| remexportent,/ exportent, getexportent, | setexportent, addexportent, | exportent(3C) |
| /set_field_fore, field_fore, | set_field_back, field_back,/ | form_field_attributes(3X) |
| set_field_status,/ /form_field_buffer: | set_field_buffer, field_buffer, | form_field_buffer(3X) |
| set_field_back,/ /form_field_attributes: | set_field_fore, field_fore, | form_field_attributes(3X) |
| /form_init, set_form_term, form_term, | set_field_init, field_init,/ | form_hook(3X) |
| general appearance of/ /form_field_just: | set_field_just, field_just: format the | form_field_just(3X) |
| field_opts_off,/ /form_field_opts: | set_field_opts, field_opts_on, | form_field_opts(3X) |
| /field_fore, set_field_back, field_back, | set_field_pad, field_pad: format the/ | form_field_attributes(3X) |
| /set_field_buffer, field_buffer, | set_field_status, field_status,/ | form_field_buffer(3X) |
| /form_term, set_field_init, field_init, | set_field_term, field_term: assign/ | form_hook(3X) |
| forms field data/ /form_field_validation: | set_field_type, field_type, field_arg: | form_field_validation(3X) |
| /new_fieldtype, free_fieldtype, | set_fieldtype_arg, set_fieldtype_choice,/ | form_fieldtype(3X) |
| /free_fieldtype, set_fieldtype_arg, | set_fieldtype_choice, link_fieldtype:/ | form_fieldtype(3X) |
| associate/ /form_field_userptr: | set_field_userptr, field_userptr: | form_field_userptr(3X) |
| field_count, move_field:/ form_field: | set_form_fields, form_fields, | form_field(3X) |
| form_term, set_field_init,/ /form_hook: | set_form_init, form_init, set_form_term, | form_hook(3X) |
| form_opts_off, form_opts:/ form_opts: | set_form_opts, form_opts_on, | form_opts(3X) |
| set_current_field,/ form_page: | set_form_page, form_page, | form_page(3X) |
| form_win: set_form_win, form_win, | set_form_sub, form_sub, scale_form:/ | form_win(3X) |
| form_hook: set_form_init, form_init, | set_form_term, form_term,/ | form_hook(3X) |
| associate application/ /form_userptr: | set_form_userptr, form_userptr: | form_userptr(3X) |
| form_sub, scale_form: forms/ form_win: | set_form_win, form_win, set_form_sub, | form_win(3X) |
| /getfsspec, getfsfile, getfstype, | setfsent, endfsent: get filesystem/ | getfsent(3C) |
| saved-group-ids | setgid: set the real-, effective-, and | setgid(2) |
| file/ /getgrent, getgrgid, getgrnam, | setgrent, endgrent, fgetgrent: get group | getgrent(3C) |
| group access list IDs getgroups, | setgroups: get or set supplementary | getgroups(2) |
| entry /gethostbyaddr, gethostbyname, | sethostent, endhostent: get network host | gethostent(3N) |
| current host | sethostid: set unique identifier of | sethostid(2) |
| | sethostname: set name of current host | sethostname(2) |
| item_term, set_menu_init,/ /menu_hook: | set_item_init, item_init, set_item_term, | menu_hook(3X) |
| item_opts_off,/ /menu_item_opts: | set_item_opts, item_opts_on, | menu_item_opts(3X) |
| /menu_hook: set_item_init, item_init, | set_item_term, item_term, set_menu_init,/ | menu_hook(3X) |
| associate/ /menu_item_userptr: | set_item_userptr, item_userptr: | menu_item_userptr(3X) |
| menus item values /menu_item_value: | set_item_value, item_value: set and get | menu_item_value(3X) |
| timer /getitimer, | setitimer: get or set value of interval | getitimer(2) |
| | setjmp, longjmp: non-local goto | setjmp(3C) |
| crypt, | setkey, encrypt: generate encryption | crypt(3C) |
| specified stream | setlinebuf: assign line buffering for a | setlinebuf(3C) |
| locale | setlocale: modify and query a program's | setlocale(3C) |
| syslog, openlog, closelog, | setlogmask: control system log | syslog(3C) |
| /set_field_status, field_status, | set_max_field: set and get forms field/ | form_field_buffer(3X) |
| menu_grey,/ /set_menu_fore, menu_fore, | set_menu_back, menu_back, set_menu_grey, | menu_attributes(3X) |
| menu_back,/ /menu_attributes: | set_menu_fore, menu_fore, set_menu_back, | menu_attributes(3X) |
| get maximum numbers of/ /menu_format: | set_menu_format, menu_format: set and | menu_format(3X) |
| /menu_fore, set_menu_back, menu_back, | set_menu_grey, menu_grey, set_menu_pad,/ | menu_attributes(3X) |
| /item_init, set_item_term, item_term, | set_menu_init, menu_init, set_menu_term,/ | menu_hook(3X) |
| connect and disconnect/ /menu_items: | set_menu_items, menu_items, item_count: | menu_items(3X) |
| string routines menu_mark: | set_menu_mark, menu_mark: menus mark | menu_mark(3X) |
| menu_opts_off, menu_opts:/ menu_opts: | set_menu_opts, menu_opts_on, | menu_opts(3X) |
| /menu_back, set_menu_grey, menu_grey, | set_menu_pad, menu_pad: control menus/ | menu_attributes(3X) |

| | | |
|---|---|---|
| get menus pattern match/ /menu_pattern: | set_menu_pattern, menu_pattern: set and | menu_pattern(3X) |
| menu_win: set_menu_win, menu_win, | set_menu_sub, menu_sub, scale_menu:/ | menu_win(3X) |
| /item_term, set_menu_init, menu_init, | set_menu_term, menu_term: assign/ | menu_hook(3X) |
| associate application/ /menu_userptr: | set_menu_userptr, menu_userptr: | menu_userptr(3X) |
| menu_sub, scale_menu: menus/ menu_win: | set_menu_win, menu_win, set_menu_sub, | menu_win(3X) |
| | setmnt: establish mount table | setmnt(1M) |
| hasmntopt: get file system/ getmntent, | setmntent, addmntent, endmntent, | getmntent(3C) |
| /getnetent, getnetbyaddr, getnetbyname, | setnetent, endnetent: get network entry | getnetent(3N) |
| network group entry /getnetgrent, | setnetgrent, endnetgrent, innetgr: get | getnetgrent(3N) |
| /form_new_page: | set_new_page, new_page: forms pagination | form_new_page(3X) |
| associate application/ /panel_userptr: | set_panel_userptr, panel_userptr: | panel_userptr(3X) |
| control | setpgid: set process group ID for job | setpgid(2) |
| | setpgrp: set process-group-id | setpgrp(2) |
| | setpgrp2: set process-group-id | setpgrp2(2) |
| priority | setpriority: set process scheduling | setpriority(2) |
| entry /getprotobynumber, getprotobyname, | setprotoent, endprotoent: get protocol | getprotoent(3N) |
| register | setpsr: set the processor status | setpsr(2) |
| getpwent, getpwuid, getpwnam, | setpwent, endpwent, setpwfile,/ | getpwent(3C) |
| /getpwuid, getpwnam, setpwent, endpwent, | setpwfile, fgetpwent: manipulate/ | getpwent(3C) |
| saved-group-ids | setregid: set the real-, effective-, and | setregid(2) |
| saved-user-ids | setreuid: set the real-, effective-, and | setreuid(2) |
| resource consumption getrlimit, | setrlimit: control maximum system | getrlimit(2) |
| getrpcent, getrpcbyname, getrpcbynumber, | setrpcent, endrpcent: get RPC entry | getrpcent(3N) |
| provide an interface to named default | sets /admdefault: | admdefault(1M) |
| get information of supplementary code | sets /getwidth: | getwidth(3W) |
| sigdelset, sigismember: manipulate | sets of signals. /sigfillset, sigaddset, | sigsetops(3C) |
| /clearok, idlok, idcok immedok, leaveok, | setscrreg, wsetscrreg, scrollok, nl,/ | curs_outopts(3X) |
| entry /getservbyport, getservbyname, | setservent, endservent: get service | getservent(3N) |
| group ID | setsid: create session and set process | setsid(2) |
| | setsockopt: set options on sockets | setsockopt(2) |
| ulckpwdf:/ /getspent, getspnam, | setspent, endspent, fgetspent, lckpwdf, | getspent(3C) |
| better, or/ random, srandom, initstate, | setstate: generate random numbers | random(3C) |
| low-level/ /resetty, savetty, getsyx, | setsyx, ripoffline, curs_set, napms: | curs_kernel(3X) |
| /initscr, newterm, endwin, isendwin, | set_term, delscreen: curses screen/ | curs_initscr(3X) |
| restartterm,/ /curs_terminfo: setupterm, | setterm, set_curterm, del_curterm, | curs_terminfo(3X) |
| | settimeofday: set date and time | settimeofday(2) |
| /profile: | setting up an environment at login time | profile(4) |
| sttydefs: maintain line and hunt | settings for TTY ports | sttydefs(1M) |
| ttydefs: terminal line | settings information for ttymon | ttydefs(4M) |
| display label and record translation | settings /tdisplay: | tdisplay(1) |
| and get/ /set_current_item, current_item, | set_top_row, top_row, item_index: set | menu_item_current(3X) |
| saved-user-ids | setuid: set the real-, effective-, and | setuid(2) |
| del_curterm,/ /curs_terminfo: | setupterm, setterm, set_curterm, | curs_terminfo(3X) |
| /getutent, getutid, getutline, pututline, | setutent, endutent, utmpname: access/ | getut(3C) |
| setbuf, | setvbuf: assign buffering to a stream | setbuf(3S) |
| used with/ /addseverity: build list of | severity levels for application to be | addseverity(3C) |
| machine-independent fashion sputl, | sgetl: access long integer data in a | sputl(3X) |
| programming language | sh, jsh, rsh, restsh: shell, the command | sh(1) |
| fgetspent, lckpwdf, ulckpwdf: manipulate | shadow password file entry /endspent, | getspent(3C) |
| putspent: write | shadow password file entry | putspent(3C) |
| /let processes attach | shared descriptor array | dg_allow_shared_descriptor_attach(2) |
| /attach another process's | shared descriptor array | dg_attach_to_shared_descriptors(2) |
| shmctl: | shared memory control operations | shmctl(2) |
| a message queue, semaphore set, or | shared memory ID /ipcrm: remove | ipcrm(1) |
| shmsys: perform a | shared memory operation | shmsys(2) |
| shmat: attach a | shared memory segment | shmat(2) |
| shmdt: detach a | shared memory segment | shmdt(2) |
| shmget: get | shared memory segment | shmget(2) |
| strings from C programs to implement | shared strings /xstr: extract | xstr(1) |
| C-like syntax csh: invoke a | shell (command interpreter) having a | csh(1) |
| system: issue a | shell command | system(3S) |
| gmatch: | shell global pattern matching | gmatch(3G) |
| shl: | shell layer manager | shl(1) |
| manage the remote and restricted | shell names /admrshell: | admrshell(1M) |
| /prtacct, shutacct, startup, turnacct: | shell procedures for accounting | acctsh(1M) |
| /sh, jsh, rsh, restsh: | shell, the command programming language | sh(1) |
| | shl: shell layer manager | shl(1) |
| | shmat: attach a shared memory segment | shmat(2) |
| | shmctl: shared memory control operations | shmctl(2) |
| | shmdt: detach a shared memory segment | shmdt(2) |
| | shmget: get shared memory segment | shmget(2) |

| | | |
|---|---|---|
| operation | shmsys: perform a shared memory | shmsys(2) |
| groups: | show group memberships | groups(1) |
| panels deck manipulation/ panel_show: | show_panel, hide_panel, panel_hidden: | panel_show(3X) |
| connection shutdown: | shut down part of a full-duplex | shutdown(2) |
| shutdown: | shut down system, change system state | shutdown(1M) |
| /nulladm, prctmp, prdaily, prtacct, | shutacct, startup, turnacct: shell/ | acctsh(1M) |
| full-duplex connection | shutdown: shut down part of a | shutdown(2) |
| system state | shutdown: shut down system, change | shutdown(1M) |
| sdiff: | side-by-side difference program | sdiff(1) |
| hfm: high | sierra file manager | hfm(4) |
| language | sifilter: preprocess MC88100 assembly | sifilter(1) |
| action | sigaction: examine and change signal | sigaction(2) |
| manipulate/ sigemptyset, sigfillset, | sigaddset, sigdelset, sigismember: | sigsetops(3C) |
| stack context | sigaltstack: set or get signal alternate | sigaltstack(2) |
| | sigblock: add to set of blocked signals | sigblock(2) |
| of/ /sigemptyset, sigfillset, sigaddset, | sigdelset, sigismember: manipulate sets | sigsetops(3C) |
| sigdelset, sigismember: manipulate sets/ | sigemptyset, sigfillset, sigaddset, | sigsetops(3C) |
| implementation-defined signals | sigfillset: fill in the set of | sigfillset(2) |
| sigismember: manipulate/ sigemptyset, | sigfillset, sigaddset, sigdelset, | sigsetops(3C) |
| process's set of blocked signals | sighold: add a signal to the calling | sighold(2) |
| signal to 'ignore' | sigignore: set the signal action of a | sigignore(2) |
| | siginfo: signal generation information | siginfo(5) |
| /sigfillset, sigaddset, sigdelset, | sigismember: manipulate sets of signals. | sigsetops(3C) |
| state /sigsetjmp, | siglongjmp: a non-local goto with signal | sigsetjmp(3C) |
| login: | sign on | login(1) |
| abort: generate an abnormal termination | signal | abort(3C) |
| sigignore: set the | signal action of a signal to 'ignore' | sigignore(2) |
| sigaction: examine and change | signal action | sigaction(2) |
| sigaltstack: set or get | signal alternate stack context | sigaltstack(2) |
| signal is/ /sigpause: clear a blocked | signal and suspend the process until a | sigpause(2) |
| | signal: base signals | signal(5) |
| /berk_signal, signal: simplified software | signal facilities | berk_signal(3C) |
| the process state to that contained in a | signal frame /sigret: restore | sigret(2) |
| blocked signals /sigrelse: remove a | signal from the calling process's set of | sigrelse(2) |
| siginfo: | signal generation information | siginfo(5) |
| signals and suspend process until a | signal is caught /set blocked | berk_sigpause(2) |
| pause: suspend process until a | signal is caught | pause(2) |
| signal and suspend the process until a | signal is caught /clear a blocked | sigpause(2) |
| psignal, psiginfo: system | signal messages | psignal(3C) |
| what to do upon presentation of a | signal /signal: specify | signal(2) |
| what to do upon presentation of a | signal /sigset: specify | sigset(2) |
| sigsuspend: wait for a | signal | sigsuspend(2) |
| what to do upon presentation of a | signal /sigvec: specify | sigvec(2) |
| facilities /berk_signal, | signal: simplified software signal | berk_signal(3C) |
| presentation of a signal | signal: specify what to do upon | signal(2) |
| sigstack: set and/or get | signal stack context | sigstack(2) |
| siglongjmp: a non-local goto with | signal state /sigsetjmp, | sigsetjmp(3C) |
| kill: send a | signal to a process | kill(2) |
| processes sigsend, sigsendset: send a | signal to a process or a group of | sigsend(2) |
| /killpg: send | signal to a process or a process group | killpg(2) |
| sigignore: set the signal action of a | signal to 'ignore' | sigignore(2) |
| raise: send | signal to program | raise(3C) |
| blocked signals /sighold: add a | signal to the calling process's set of | sighold(2) |
| signal is/ /berk_sigpause: set blocked | signals and suspend process until a | berk_sigpause(2) |
| sigblock: add to set of blocked | signals | sigblock(2) |
| in the set of implementation-defined | signals /sigfillset: fill | sigfillset(2) |
| to the calling process's set of blocked | signals /sighold: add a signal | sighold(2) |
| signal: base | signals | signal(5) |
| sigpending: examine pending | signals | sigpending(2) |
| sigprocmask: examine and change blocked | signals | sigprocmask(2) |
| the calling process's set of blocked | signals /sigrelse: remove a signal from | sigrelse(2) |
| sigsetmask: specify set of blocked | signals | sigsetmask(2) |
| sigismember: manipulate sets of | signals. /sigaddset, sigdelset, | sigsetops(3C) |
| ssignal, gsignal: software | signals | ssignal(3C) |
| suspend the process until a signal is/ | sigpause: clear a blocked signal and | sigpause(2) |
| | sigpending: examine pending signals | sigpending(2) |
| signals | sigprocmask: examine and change blocked | sigprocmask(2) |
| calling process's set of blocked/ | sigrelse: remove a signal from the | sigrelse(2) |
| that contained in a signal frame | sigret: restore the process state to | sigret(2) |
| process or a group of processes | sigsend, sigsendset: send a signal to a | sigsend(2) |
| or a group of processes /sigsend, | sigsendset: send a signal to a process | sigsend(2) |

| | | |
|---|---|---|
| presentation of a signal | sigset: specify what to do upon | sigset(2) |
| with signal state | sigsetjmp, siglongjmp: a non-local goto | sigsetjmp(3C) |
| signals | sigsetmask: specify set of blocked | sigsetmask(2) |
| context | sigstack: set and/or get signal stack | sigstack(2) |
| | sigsuspend: wait for a signal | sigsuspend(2) |
| presentation of a signal | sigvec: specify what to do upon | sigvec(2) |
| lex: generate programs for | simple lexical tasks | lex(1) |
| admlock: manage | simple process synchronization | admlock(1M) |
| rand, srand: | simple random-number generator | rand(3C) |
| fmt: | simple text formatter | fmt(1) |
| /berk_signal, signal: | simplified software signal facilities | berk_signal(3C) |
| asinf, acos, acosf, atan, atanf,/ /trig: | sin, sinf, cos, cosf, tan, tanf, asin, | trig(3M) |
| acos, acosf, atan, atanf,/ /trig: sin, | sinf, cos, cosf, tan, tanf, asin, asinf, | trig(3M) |
| tanhf, asinh, acosh, atanh:/ | sinh, sinhf, cosh, coshf, tanh, | sinh(3M) |
| asinh, acosh, atanh:/ sinh, | sinhf, cosh, coshf, tanh, tanhf, | sinh(3M) |
| deblock: change blocking | size | deblock(1) |
| getpagesize: get the system page | size | getpagesize(2) |
| elf_fsize: elf32_fsize: return the | size of an object file type | elf_fsize(3E) |
| files | size: print section sizes of object | size(1) |
| fez: display file element | sizes | fez(1) |
| size: print section | sizes of object files | size(1) |
| grantpt: grant access to the | slave pseudo-terminal device | grantpt(3C) |
| ptsname: get name of the | slave pseudo-terminal device | ptsname(3C) |
| | sleep: suspend execution for an interval | sleep(1) |
| | sleep: suspend execution for interval | sleep(3C) |
| /slk_touch, slk_attron, slk_attrset, | slk_attroff: curses soft label routines | curs_slk(3X) |
| /slk_clear, slk_restore, slk_touch, | slk_attron, slk_attrset, slk_attroff:/ | curs_slk(3X) |
| /slk_restore, slk_touch, slk_attron, | slk_attrset, slk_attroff: curses soft/ | curs_slk(3X) |
| /slk_refresh, slk_noutrefresh, slk_label, | slk_clear, slk_restore, slk_touch,/ | curs_slk(3X) |
| slk_noutrefresh, slk_label,/ curs_slk: | slk_init, slk_set, slk_refresh, | curs_slk(3X) |
| /slk_set, slk_refresh, slk_noutrefresh, | slk_label, slk_clear, slk_restore,/ | curs_slk(3X) |
| /slk_init, slk_set, slk_refresh, | slk_noutrefresh, slk_label, slk_clear,/ | curs_slk(3X) |
| /curs_slk: slk_init, slk_set, | slk_refresh, slk_noutrefresh, slk_label,/ | curs_slk(3X) |
| /slk_noutrefresh, slk_label, slk_clear, | slk_restore, slk_touch, slk_attron,/ | curs_slk(3X) |
| slk_label,/ /curs_slk: slk_init, | slk_set, slk_refresh, slk_noutrefresh, | curs_slk(3X) |
| /slk_label, slk_clear, slk_restore, | slk_touch, slk_attron, slk_attrset,/ | curs_slk(3X) |
| user ttyslot: find the | slot in the utmp file of the current | ttyslot(3C) |
| spline: interpolate | smooth curve | spline(1G) |
| /admsnmpcommunity: manage the | SNMP community database | admsnmpcommunity(1M) |
| /admsnmptrap: manage the | SNMP traps database | admsnmptrap(1M) |
| /admsnmpobject: manage the | snmpd object database | admsnmpobject(1M) |
| | sno: SNOBOL interpreter and compiler | sno(1) |
| sno: | SNOBOL interpreter and compiler | sno(1) |
| accept: accept a connection on a | socket | accept(2) |
| bind: bind a name to a | socket | bind(2) |
| connect: initiate a connection on a | socket | connect(2) |
| communication | socket: create an endpoint for | socket(2) |
| getsockopt: get options on a | socket | getsockopt(2) |
| listen: listen for connections on a | socket | listen(2) |
| getsockname: get | socket name | getsockname(2) |
| start an NFS server on a specified | socket /nfssvc: | nfssvc(2) |
| recv: receive a message from a | socket | recv(2) |
| recvfrom: receive a message from a | socket | recvfrom(2) |
| recvmsg: receive a message from a | socket | recvmsg(2) |
| send: send a message from a | socket | send(2) |
| sendmsg: send a message from a | socket | sendmsg(2) |
| sendto: send a message from a | socket | sendto(2) |
| sockets | socketpair: create a pair of connected | socketpair(2) |
| setsockopt: set options on | sockets | setsockopt(2) |
| socketpair: create a pair of connected | sockets | socketpair(2) |
| slk_attrset, slk_attroff: curses | soft label routines /slk_attron, | curs_slk(3X) |
| removef: remove a file from | software database | removef(1M) |
| depend: | software dependencies files | depend(4) |
| base sdetab: | software development environment data | sdetab(4) |
| sde: | software development environment | sde(5) |
| /sde-target: print commands to reset | software development environment target | sde-target(1) |
| installf: add a file to the | software installation database | installf(1M) |
| /vscload: download board resident | software onto VSC synchronous controller | vscload(1M) |
| pkginfo: display | software package information | pkginfo(1) |
| pkgadd: transfer | software package to the system | pkgadd(1M) |
| admpackage: manage DG/UX-style | software packages | admpackage(1M) |

| | | |
|---|---|---|
| /curs_scroll: scroll, | srcl, wscrl: scroll a curses window | curs_scroll(3X) |
| scanf, fscanf, | sscanf: convert formatted input | scanf(3S) |
| scanf, fscanf, | sscanf: convert formatted input | scanf(3W) |
| Driver | ssid: Streams Synchronous Interface | ssid(7) |
| | ssignal, gsignal: software signals | ssignal(3C) |
| | st: AViiON family tape subsystem | st(7) |
| sigaltstack: set or get signal alternate | stack context | sigaltstack(2) |
| sigstack: set and/or get signal | stack context | sigstack(2) |
| /stdio: | standard buffered input/output package | stdio(3S) |
| print an error message to | standard error /extended_perror: | extended_perror(3C) |
| package stdipc: ftok: | standard interprocess communication | stdipc(3C) |
| cat: concatenate and type files to | standard output | cat(1) |
| discipline module ldterm: | standard STREAMS terminal line | ldterm(7) |
| programming/ ksh, rksh: KornShell, a | standard/restricted command and | ksh(1) |
| /attron, wattron, attrset, wattrset, | standend, wstandend, standout,/ | curs_attr(3X) |
| and/ /wattrset, standend, wstandend, | standout, wstandout: curses character | curs_attr(3X) |
| requests /async_daemon: | start a BIOD server for asynchronous I/O | async_daemon(2) |
| socket nfssvc: | start an NFS server on a specified | nfssvc(2) |
| biod: | start block I/O servers | biod(1M) |
| /reset remote file lock database, | start lock reclaim grace period | dg_lock_reset(2) |
| Terminal Server /lptermprinter: | start printer session with 40014A | lptermprinter(1) |
| rmt: | start the remote mag tape server | rmt(1M) |
| server wmtd: | start the WORM magnetic tape device | wmtd(1M) |
| has_colors,/ curs_color: | start_color, init_pair, init_color, | curs_color(3X) |
| | starter: information for beginning users | starter(1) |
| requests /lpsched, lpshut, lpmove: | start/stop the LP print service and move | lpsched(1M) |
| /prctmp, prdaily, prtacct, shutacct, | startup, turnacct: shell procedures for/ | acctsh(1M) |
| | stat: data returned by stat system call | stat(5) |
| | stat: get file status | stat(2) |
| stat: data returned by | stat system call | stat(5) |
| get information about current IPCs | state /dg_ipc_info: | dg_ipc_info(2) |
| reset the teletype bits to a sensible | state /reset: | reset(1) |
| shut down system, change system | state /shutdown: | shutdown(1M) |
| siglongjmp: a non-local goto with signal | state /sigsetjmp, | sigsetjmp(3C) |
| tcgetattr, tcsetattr: get and set | state | tcsetattr(3C) |
| t_getstate: get the current | state | t_getstate(3N) |
| frame sigret: restore the process | state to that contained in a signal | sigret(2) |
| waitid: wait for child process to change | state | waitid(2) |
| fsync: synchronize a file's in-core | state with that on disk | fsync(2) |
| system call | statfs: data returned by the statfs | statfs(5) |
| file system | statfs: get information about a mounted | statfs(2) |
| statfs: data returned by the | statfs system call | statfs(5) |
| fstab: | static information about file systems | fstab(4) |
| ustat: get file system device | statistics | ustat(2) |
| dg_mstat: get file | status | dg_mstat(2) |
| fstat: get file | status | fstat(2) |
| dg_fstat: get extended file | status information | dg_fstat(2) |
| dg_stat: get extended file | status information | dg_stat(2) |
| ferror, feof, clearerr, fileno: stream | status inquiries | ferror(3S) |
| uustat: uucp | status inquiry and job control | uustat(1) |
| inter-process communication facilities | status /ipcs: report | ipcs(1) |
| lstat: get file | status | lstat(2) |
| lpstat: print information about the | status of the LP print service | lpstat(1) |
| ps: report process | status | ps(1) |
| the current contents of the processor | status register /getpsr: return | getpsr(2) |
| setpsr: set the processor | status register | setpsr(2) |
| stat: get file | status | stat(2) |
| wstat: wait | status | wstat(5) |
| system | statvfs: return information about a file | statvfs(2) |
| | stdarg: handle variable argument list | stdarg(5) |
| fmtmsg: display a message on | stderr or system console | fmtmsg(1) |
| fmtmsg: display a message on | stderr or system console | fmtmsg(3C) |
| package | stdio: standard buffered input/output | stdio(3S) |
| communication package | stdipc: ftok: standard interprocess | stdipc(3C) |
| compile and match/ regexp: compile, | step, advance: regular expression | regexp(5) |
| compile and match/ regexpr: compile, | step, advance: regular expression | regexpr(3G) |
| | stime: set time | stime(2) |
| wait3: wait for child process to | stop or terminate | wait3(2) |
| wait for the specified child process to | stop or terminate /wait4: | wait4(2) |
| halt: | stop the system processor | halt(1M) |
| msync: synchronize memory with physical | storage | msync(3C) |

| | | |
|---|---|---|
| admswap: manage | swap areas . . . . . . . . . . . . . . . . . . . | admswap(1M) |
| swab: | swap bytes . . . . . . . . . . . . . . . . . . . | swab(3C) |
| swapon: add a | swap device for demand paging . . . . . . . . . | swapon(2) |
| indivisible compare and | swap /store_conditional: . . . . . . . . . . . . | store_conditional(2) |
| | swapcontext: manipulate user contexts . . . . . . | swapcontext(3C) |
| paging | swapon: add a swap device for demand . . . . . | swapon(2) |
| system paging | swapon: specify additional devices for . . . . . . | swapon(1M) |
| asynchronous controller | syac: AViiON family intelligent . . . . . . . . . | syac(7) |
| syacdb: | syac debugger utility program . . . . . . . . . | syacdb(1M) |
| | syacdb: syac debugger utility program . . . . . . | syacdb(1M) |
| entry /ldgetname: retrieve | symbol name for object file symbol table . . . . . | ldgetname(3X) |
| /elf_getarsym: retrieve archive | symbol table . . . . . . . . . . . . . . . . . | elf_getarsym(3E) |
| retrieve symbol name for object file | symbol table entry /ldgetname: . . . . . . . . . . | ldgetname(3X) |
| ldtbindex: compute index of | symbol table entry of an object file . . . . . . | ldtbindex(3X) |
| ldtbread: read an indexed | symbol table entry of an object file . . . . . . | ldtbread(3X) |
| syms: common object file | symbol table format . . . . . . . . . . . . . | syms(4) |
| ldtbseek: seek to the | symbol table of an object file . . . . . . . . . | ldtbseek(3X) |
| sdb: | symbolic debugger . . . . . . . . . . . . . . . | sdb(1) |
| symlink: create a | symbolic link file . . . . . . . . . . . . . . . | symlink(2) |
| readlink: read the contents of a | symbolic link . . . . . . . . . . . . . . . . . | readlink(2) |
| definitions of common terms and | symbols /glossary: . . . . . . . . . . . . . . . | glossary(1) |
| | symlink: create a symbolic link file . . . . . . . | symlink(2) |
| format | syms: common object file symbol table . . . . . | syms(4) |
| resident file system information | sync: synchronize disk and memory . . . . . . . | sync(2) |
| | sync: update the super-block . . . . . . . . . . | sync(1M) |
| admlock: manage simple process | synchronization . . . . . . . . . . . . . . . . | admlock(1M) |
| adjtime: correct the time to allow | synchronization of the system clock . . . . . . | adjtime(2) |
| that on disk /fsync: | synchronize a file's in-core state with . . . . . . | fsync(2) |
| file system information sync: | synchronize disk and memory resident . . . . . | sync(2) |
| /msync: | synchronize memory with physical storage . . . . | msync(3C) |
| t_sync: | synchronize transport library . . . . . . . . . | t_sync(3N) |
| vsccheck: verify that the VSC | synchronous controller is operable . . . . . . | vsccheck(1M) |
| board resident software onto VSC | synchronous controller /download . . . . . . . | vscload(1M) |
| ssid: Streams | Synchronous Interface Driver . . . . . . . . . | ssid(7) |
| without system/ /dg_unbuffered_read: | synchronously read data from a file . . . . . . . | dg_unbuffered_read(2) |
| without system/ /dg_unbuffered_write: | synchronously write data to a file . . . . . . . . | dg_unbuffered_write(2) |
| /derwin, mvderwin, dupwin, wsyncup, | syncok, wcursyncup, wsyncdown : create/ . . . . | curs_window(3X) |
| (command interpreter) having a C-like | syntax /csh: invoke a shell . . . . . . . . . . | csh(1) |
| administration interface | sysadm, xsysadm: menu-driven system . . . . . . | sysadm(1M) |
| pseudo-device | syscon: DG/UX operating system console . . . . | syscon(7) |
| variables | sysconf: get configurable system . . . . . . . . . | sysconf(2) |
| | sysdef: output system definition . . . . . . . . . | sysdef(1M) |
| system types | sysfs: returns information about file . . . . . . . | sysfs(2) |
| strings | sysinfo: get and set system information . . . . . | sysinfo(2) |
| control system log | syslog, openlog, closelog, setlogmask: . . . . . . | syslog(3C) |
| syslogd system log server | syslog.conf: configuration file for . . . . . . . . | syslog.conf(5) |
| | syslogd: log systems messages . . . . . . . . . . | syslogd(1M) |
| syslog.conf: configuration file for | syslogd system log server . . . . . . . . . . . | syslog.conf(5) |
| /admsar: manage | system activity monitoring and reporting . . . . . | admsar(1M) |
| sar: sa1, sa2, sadc: | system activity report package . . . . . . . . . | sar(1M) |
| sar: | system activity reporter . . . . . . . . . . . . | sar(1) |
| time a command; report process data and | system activity /timex: . . . . . . . . . . . . | timex(1) |
| /admaccounting: manage accounting | system . . . . . . . . . . . . . . . . . . . . | admaccounting(1M) |
| sysadm, xsysadm: menu-driven | system administration interface . . . . . . . . | sysadm(1M) |
| osysadm: menu-driven | system administration program . . . . . . . . | osysadm(1M) |
| dump2: incremental file | system backup . . . . . . . . . . . . . . . . . | dump2(1M) |
| filesave, tapesave: daily/weekly file | system backup . . . . . . . . . . . . . . . . . | filesave(1M) |
| read data from a file without | system buffering /synchronously . . . . . . . . | dg_unbuffered_read(2) |
| write data to a file without | system buffering /synchronously . . . . . . . . | dg_unbuffered_write(2) |
| dg_mknod: data returned by the dg_mknod | system call . . . . . . . . . . . . . . . . . . | dg_mknod(5) |
| data returned by dg_stat and dg_fstat | system call /dg_stat: . . . . . . . . . . . . . | dg_stat(5) |
| stat: data returned by stat | system call . . . . . . . . . . . . . . . . . . | stat(5) |
| statfs: data returned by the statfs | system call . . . . . . . . . . . . . . . . . . | statfs(5) |
| ustat: data returned by the ustat | system call . . . . . . . . . . . . . . . . . . | ustat(5) |
| intro: introduction to | system calls and error numbers . . . . . . . . . | intro(2) |
| link, unlink: exercise link and unlink | system calls . . . . . . . . . . . . . . . . . . | link(1M) |
| ckbinarsys: determine whether remote | system can accept binary messages . . . . . . . | ckbinarsys(1M) |
| shutdown: shut down | system, change system state . . . . . . . . . . | shutdown(1M) |
| admclient: manage operating | system clients . . . . . . . . . . . . . . . . | admclient(1M) |
| the time to allow synchronization of the | system clock /adjtime: correct . . . . . . . . . | adjtime(2) |
| uux: UNIX-to-UNIX | system command execution . . . . . . . . . . | uux(1) |

| | | |
|---|---|---|
| config: configure a | system | config(1M) |
| functions dg_sysctl: perform | system configuration and control | dg_sysctl(2) |
| fmtmsg: display a message on stderr or | system console | fmtmsg(1) |
| fmtmsg: display a message on stderr or | system console | fmtmsg(3C) |
| syscon: DG/UX operating | system console pseudo-device | syscon(7) |
| uucp, uulog, uuname: UNIX-to-UNIX | system copy | uucp(1) |
| crash: what to do when the DG/UX | system crashes | crash(8) |
| cu: call another UNIX | system | cu(1) |
| types: primitive | system data types | types(5) |
| admdate: manipulate the | system date, time and time zone | admdate(1M) |
| dg_fsdb: file | system debugger | dg_fsdb(1M) |
| fsdb: file | system debugger | fsdb(1M) |
| sysdef: output | system definition | sysdef(1M) |
| endmntent, hasmntopt: get file | system descriptor file entry /addmntent, | getmntent(3C) |
| ustat: get file | system device statistics | ustat(2) |
| umount: remove a file | system device | umount(2) |
| dg_mount: mount a file | system | dg_mount(2) |
| dump: incremental file | system dump | dump(1M) |
| lsd: load a | system dump from tape | lsd(1M) |
| perror: print | system error messages | perror(3C) |
| uuto, uupick: public UNIX-to-UNIX | system file copy | uuto(1) |
| probedev: probe | system for devices | probedev(1M) |
| fs: file | system format | fs(4) |
| file | system: format of a kernel description | system(4) |
| get information about a mounted file | system /fstatfs: | fstatfs(2) |
| return information about a file | system /fstatvfs: | fstatvfs(2) |
| (create) a new group definition on the | system /groupadd: add | groupadd(1M) |
| delete a group definition from the | system /groupdel: | groupdel(1M) |
| modify a group definition on the | system /groupmod: | groupmod(1M) |
| hier: DG/UX file | system hierarchy | hier(5) |
| systemid: display the unique | system identifier | systemid(1M) |
| crash: examine | system images | crash(1M) |
| dirent: file | system independent directory entry | dirent(4) |
| dg_sys_info: get | system information | dg_sys_info(2) |
| dumpfs: dump file | system information | dumpfs(1M) |
| getexportopt: get exported file | system information /endexportent, | exportent(3C) |
| sysinfo: get and set | system information strings | sysinfo(2) |
| disk and memory resident file | system information /sync: synchronize | sync(2) |
| installman: manage | system installation | installman(1M) |
| | system: issue a shell command | system(3S) |
| kbd: AViiON series workstation | system keyboard | kbd(7) |
| logger: make entries in the | system log | logger(1) |
| configuration file for syslogd | system log server /syslog.conf: | syslog.conf(5) |
| openlog, closelog, setlogmask: control | system log /syslog, | syslog(3C) |
| logins: list user and | system login information | logins(1M) |
| mailx: interactive message processing | system | mailx(1) |
| application/ intro: introduction to | system maintenance commands and | intro(1M) |
| intro: introduction to | system maintenance procedures | intro(8) |
| mem: main | system memory | mem(7) |
| mfs: memory file | system | mfs(4) |
| mkfs, newfs: create a file | system | mkfs(1M) |
| mknod: create a file entry in the file | system | mknod(2) |
| mount: mount a file | system | mount(2) |
| file descriptor to object in file | system name space /attach STREAMS-based | fattach(3C) |
| dg_mknod: create a file | system node | dg_mknod(2) |
| filesystem: file | system organization | filesystem(7) |
| getpagesize: get the | system page size | getpagesize(2) |
| swapon: specify additional devices for | system paging | swapon(1M) |
| dg_sysctl: modify | system parameters | dg_sysctl(1M) |
| vipw: edit the | system password file | vipw(1M) |
| pkgadd: transfer software package to the | system | pkgadd(1M) |
| pkgrm: removes a package from the | system | pkgrm(1M) |
| halt: stop the | system processor | halt(1M) |
| reboot halts and optionally reboots the | system processor(s) /reboot: | reboot(2) |
| prf: operating | system profiler | prf(7) |
| prfdc, prfsnap, prfpr: operating | system profiler /prfld, prfstat, | profiler(1M) |
| reboot: restart the operating | system | reboot(1M) |
| getrlimit, setrlimit: control maximum | system resource consumption | getrlimit(2) |
| vlimit: control maximum | system resource consumption | vlimit(3C) |
| restore: incrementally restore a file | system | restore(1M) |
| psignal, psiginfo: | system signal messages | psignal(3C) |

| | | |
|---|---|---|
| rmt: start the remote mag | tape server | rmt(1M) |
| st: AViiON family | tape subsystem | st(7) |
| dumptab: | tape table file for dump2 | dumptab(4) |
| taccess: initiate access to labeled | tape | taccess(1) |
| tposn: position | tape to specified file | tposn(1) |
| tread: read file(s) from | tape | tread(1) |
| trelease: terminate access to a | tape | trelease(1) |
| twrite: writes a file to | tape | twrite(1) |
| tlabel: initialize a | tape with a volume label | tlabel(1) |
| manipulate the default parameters for | tapes /admtape: | admtape(1M) |
| read and write labels for dump | tapes /dump2label: | dump2label(1M) |
| for reading and writing IBM and ANSI | tapes /REELexchange: commands | reelexchange_intro(1) |
| backup filesave, | tapesave: daily/weekly file system | filesave(1M) |
| | tar: tape archive file format | tar(5) |
| | tar: tape file archiver | tar(1) |
| reset software development environment | target /sde-target: print commands to | sde-target(1) |
| generate programs for simple lexical | tasks /lex: | lex(1) |
| endpoint | t_bind: bind an address to a transport | t_bind(3N) |
| deroff: remove nroff/troff, | tbl, and eqn constructs | deroff(1) |
| /tcgetattr, tcsetattr, tcsendbreak, | tcdrain, tcflush, tcflow, cfgetospeed,/ | termios(3C) |
| transmission /tcsendbreak, | tcdrain, tcflush, tcflow: control data | tcflush(3C) |
| /tcsendbreak, tcdrain, tcflush, | tcflow, cfgetospeed, cfgetispeed,/ | termios(3C) |
| tcsendbreak, tcdrain, tcflush, | tcflow: control data transmission | tcflush(3C) |
| /tcsetattr, tcsendbreak, tcdrain, | tcflush, tcflow, cfgetospeed,/ | termios(3C) |
| transmission tcsendbreak, tcdrain, | tcflush, tcflow: control data | tcflush(3C) |
| | tcgetattr, tcsetattr: get and set state | tcsetattr(3C) |
| tcdrain, tcflush, tcflow,/ termios: | tcgetattr, tcsetattr, tcsendbreak, | termios(3C) |
| ID | tcgetpgrp: get foreground process group | tcgetpgrp(3C) |
| /cfgetispeed, cfsetispeed, cfsetospeed, | tcgetpgrp, tcsetpgrp, tcgetsid: general/ | termios(3C) |
| /cfsetospeed, tcgetpgrp, tcsetpgrp, | tcgetsid: general terminal interface | termios(3C) |
| | tcload: load terminal controller devices | tcload(1M) |
| | t_close: close a transport endpoint | t_close(3N) |
| another transport user | t_connect: establish a connection with | t_connect(3N) |
| /admtcpipparams: manage the | TCP/IP host parameters | admtcpipparams(1M) |
| /admipinterface: manage the | TCP/IP network interfaces database | admipinterface(1M) |
| /admtcpipdaemon: manage the | TCP/IP servers | admtcpipdaemon(1M) |
| /termios: tcgetattr, tcsetattr, | tcsendbreak, tcdrain, tcflush, tcflow,/ | termios(3C) |
| control data transmission | tcsendbreak, tcdrain, tcflush, tcflow: | tcflush(3C) |
| tcgetattr, | tcsetattr: get and set state | tcsetattr(3C) |
| tcflush, tcflow,/ termios: tcgetattr, | tcsetattr, tcsendbreak, tcdrain, | termios(3C) |
| process group id | tcsetpgrp: set terminal foreground | tcsetpgrp(3C) |
| /cfsetispeed, cfsetospeed, tcgetpgrp, | tcsetpgrp, tcgetsid: general terminal/ | termios(3C) |
| trees tsearch, tfind, | tdelete, twalk: manage binary search | tsearch(3C) |
| translation settings | tdisplay: display label and record | tdisplay(1) |
| legend: Debugging information | technology | legend(5) |
| | tee: pipe fitting | tee(1) |
| posttek: PostScript translator for | tektronix 4014 files | posttek(1) |
| reset: reset the | teletype bits to a sensible state | reset(1) |
| /init, | telinit: process control initialization | init(1M) |
| /form_data: data_ahead, data_behind: | tell if forms field has off-screen data/ | form_data(3X) |
| /menu_item_visible: item_visible: | tell if menus item is visible | menu_item_visible(3X) |
| directory/ /directory: opendir, readdir, | telldir, seekdir, rewinddir, closedir: | directory(3X) |
| file /tmpnam, | tempnam: create a name for a temporary | tmpnam(3S) |
| tmpfile: create a | temporary file | tmpfile(3S) |
| tmpnam, tempnam: create a name for a | temporary file | tmpnam(3S) |
| chgtinfo: create a | temporary version of a TERMINFO entry | chgtinfo(1) |
| | term: conventional names for terminals | term(5) |
| /has_ic, has_il, killchar, longname, | termattrs, termname: curses environment/ | curs_termattrs(3X) |
| captoinfo: convert a | TERMCAP entry into a TERMINFO entry | captoinfo(1M) |
| curses interfaces (emulated) to the | termcap library /tgetstr, tgoto, tputs: | curs_termcap(3X) |
| | termcap: terminal capability data base | termcap(5) |
| /terminfo: | terminal and printer capability database | terminfo(4) |
| termcap: | terminal capability data base | termcap(5) |
| tcload: load | terminal controller devices | tcload(1M) |
| ct: spawn getty to a remote | terminal | ct(1) |
| ctermid: generate file name for | terminal | ctermid(3S) |
| ptem: STREAMS Pseudo | Terminal Emulation module | ptem(7) |
| tcsetpgrp: set | terminal foreground process group id | tcsetpgrp(3C) |
| /tgetflag, tgetstr, tgoto, tputs: | terminal independent operation routines | termcap(3X) |
| /timeout, wtimeout, typeahead: curses | terminal input option control routines | curs_inopts(3X) |
| termio: general | terminal interface | termio(7) |

| | | |
|---|---|---|
| tcgetpgrp, tcsetpgrp, tcgetsid: general | terminal interface /cfsetospeed, | termios(3C) |
| termiox: extended general | terminal interface | termiox(7) |
| (or push back) characters from curses | terminal keyboard /ungetch: get | curs_getch(3X) |
| get character strings from curses | terminal keyboard /mvwgetnstr: | curs_getstr(3X) |
| back) wchar_t characters from curses | terminal keyboard /get (or push | curs_getwch(3X) |
| wchar_t character strings from curses | terminal keyboard /mvwgetnwstr: get | curs_getwstr(3X) |
| dial: establish an out-going | terminal line connection | dial(3C) |
| ldterm: standard STREAMS | terminal line discipline module | ldterm(7) |
| ttymon /ttydefs: | terminal line settings information for | ttydefs(4M) |
| last: indicate last user or | terminal logins | last(1) |
| tput: initialize a | terminal or query terminfo database | tput(1) |
| /wsetscrreg, scrollok, nl, nonl: curses | terminal output option control routines | curs_outopts(3X) |
| /admterminal: manage | terminal ports | admterminal(1M) |
| ttymon: monitor | terminal ports | ttymon(1M) |
| devtty: control | terminal pseudo-device | devtty(7) |
| clear: clear | terminal screen | clear(1) |
| start printer session with 40014A | Terminal Server /lptermprinter: | lptermprinter(1) |
| print a file using the 40014A | Terminal Server /termprinter: | termprinter(1) |
| script: make typescript of a | terminal session | script(1) |
| stty: set the options for a | terminal | stty(1) |
| tabs: set tabs on a | terminal | tabs(1) |
| tty: get the name of the | terminal | tty(1) |
| ttyname, isatty: find name of a | terminal | ttyname(3C) |
| discipline getty: set | terminal type, modes, speed, and line | getty(1M) |
| virtually hang up the current control | terminal /vhangup: | vhangup(2) |
| manage serving of X display | terminals /admxterminal: | admxterminal(1M) |
| term: conventional names for | terminals | term(5) |
| kill: | terminate a process by default | kill(1) |
| dg_kill: test for or | terminate a process | dg_kill(1) |
| trelease: | terminate access to a tape | trelease(1) |
| exit, _exit: | terminate process | exit(2) |
| wait3: wait for child process to stop or | terminate | wait3(2) |
| the specified child process to stop or | terminate /wait4: wait for | wait4(2) |
| atexit: add program | termination routine | atexit(3C) |
| abort: generate an abnormal | termination signal | abort(3C) |
| wait, waitpid: wait for process | termination | wait(2) |
| tic: | TERMINFO compiler | tic(1M) |
| tigetnum, tigetstr: curses interfaces to | terminfo database /mvcur, tigetflag, | curs_terminfo(3X) |
| tput: initialize a terminal or query | terminfo database | tput(1) |
| infocmp: compare or print out | TERMINFO descriptions | infocmp(1M) |
| convert a TERMCAP entry into a | TERMINFO entry /captoinfo: | captoinfo(1M) |
| create a temporary version of a | TERMINFO entry /chgtinfo: | chgtinfo(1) |
| capability database | terminfo: terminal and printer | terminfo(4) |
| | termio: general terminal interface | termio(7) |
| tcsendbreak, tcdrain, tcflush, tcflow,/ | termios: tcgetattr, tcsetattr, | termios(3C) |
| interface | termiox: extended general terminal | termiox(7) |
| /has_il, killchar, longname, termattrs, | termname: curses environment query/ | curs_termattrs(3X) |
| 40014A Terminal Server | termprinter: print a file using the | termprinter(1) |
| glossary: definitions of common | terms and symbols | glossary(1) |
| | t_error: produce error message | t_error(3N) |
| isastream: | test a file descriptor | isastream(3C) |
| | test: condition evaluation command | test(1) |
| dg_kill: | test for or terminate a process | dg_kill(1) |
| testlocale: | test locale definition | testlocale(1M) |
| | testlocale: test locale definition | testlocale(1M) |
| ed, red: | text editor | ed(1) |
| ex: | text editor | ex(1) |
| users) edit: | text editor (variant of ex for casual | edit(1) |
| newform: change the format of a | text file | newform(1) |
| fspec: format specification in | text files | fspec(4) |
| postprint: translate | text files into PostScript | postprint(1) |
| fmt: simple | text formatter | fmt(1) |
| plock: lock data, | text, or both into memory | plock(2) |
| gettxt: retrieve a | text string from a message data base | gettxt(1) |
| gettxt: retrieve a | text string | gettxt(3C) |
| /display contents of, or search for a | text string in, message data bases | srchtxt(1) |
| search trees tsearch, | tfind, tdelete, twalk: manage binary | tsearch(3C) |
| | t_free: free a library structure | t_free(3N) |
| tgoto, tputs: curses/ /curs_termcap: | tgetent, tgetflag, tgetnum, tgetstr, | curs_termcap(3X) |
| tgoto, tputs: terminal independent/ | tgetent, tgetnum, tgetflag, tgetstr, | termcap(3X) |
| tputs: curses/ /curs_termcap: tgetent, | tgetflag, tgetnum, tgetstr, tgoto, | curs_termcap(3X) |

| | | |
|---|---|---|
| terminal independent/ tgetent, tgetnum, | tgetflag, tgetstr, tgoto, tputs: | termcap(3X) |
| information | t_getinfo: get protocol-specific service | t_getinfo(3N) |
| tputs: terminal independent/ tgetent, | tgetnum, tgetflag, tgetstr, tgoto, | termcap(3X) |
| /curs_termcap: tgetent, tgetflag, | tgetnum, tgetstr, tgoto, tputs: curses/ | curs_termcap(3X) |
| | t_getstate: get the current state | t_getstate(3N) |
| (emulated)/ /tgetent, tgetflag, tgetnum, | tgetstr, tgoto, tputs: curses interfaces | curs_termcap(3X) |
| independent/ tgetent, tgetnum, tgetflag, | tgetstr, tgoto, tputs: terminal | termcap(3X) |
| /tgetent, tgetflag, tgetnum, tgetstr, | tgoto, tputs: curses interfaces/ | curs_termcap(3X) |
| tgetent, tgetnum, tgetflag, tgetstr, | tgoto, tputs: terminal independent/ | termcap(3X) |
| merge: | three-way file merge | merge(1) |
| | tic: TERMINFO compiler | tic(1M) |
| /tputs, putp, vidputs, vidattr, mvcur, | tigetflag, tigetnum, tigetstr: curses/ | curs_terminfo(3X) |
| /vidputs, vidattr, mvcur, tigetflag, | tigetnum, tigetstr: curses interfaces to/ | curs_terminfo(3X) |
| /vidattr, mvcur, tigetflag, tigetnum, | tigetstr: curses interfaces to terminfo/ | curs_terminfo(3X) |
| system activity /timex: | time a command; report process data and | timex(1) |
| time: | time a command | time(1) |
| admdate: manipulate the system date, | time and time zone | admdate(1M) |
| at, batch: execute commands at a later | time | at(1) |
| ftime: get date and | time | ftime(3C) |
| | time: get system time | time(2) |
| convert user format date and | time /getdate, getdate_err: | getdate(3C) |
| /gettimeofday: get date and | time | gettimeofday(2) |
| page: display file one screenful at a | time /more, | more(1) |
| display a prompt; verify and return a | time of day /cktime: | cktime(1) |
| forward or backward one screenful at a | time /pg: display file | pg(1) |
| setting up an environment at login | time /profile: | profile(4) |
| profil: set up execution | time profiling for a process | profil(2) |
| rtime: get remote | time | rtime(3N) |
| /settimeofday: set date and | time | settimeofday(2) |
| stime: set | time | stime(2) |
| | time: time a command | time(1) |
| time: get system | time | time(2) |
| system clock adjtime: correct the | time to allow synchronization of the | adjtime(2) |
| gmtime, asctime, tzset: convert date and | time to string /ctime, localtime, | ctime(3C) |
| cftime, ascftime: convert date and | time to string /strftime, | strftime(3C) |
| clock: report CPU | time used | clock(3C) |
| manipulate the system date, time and | time zone /admdate: | admdate(1M) |
| timezone: set default system | time zone and locale | timezone(4) |
| zic: | time zone compiler | zic(1M) |
| zdump: | time zone dumper | zdump(1M) |
| /raw, noraw, noqiflush, qiflush, | timeout, wtimeout, typeahead: curses/ | curs_inopts(3X) |
| setitimer: get or set value of interval | timer /getitimer, | getitimer(2) |
| the jobs queued to run at specified | times /atq: display | atq(1) |
| times | times: get process and child process | times(2) |
| touch: update access and modification | times of a file | touch(1) |
| times: get process and child process | times | times(2) |
| utime: set file access and modification | times | utime(2) |
| utimes: set file access and modification | times | utimes(2) |
| data and system activity | timex: time a command; report process | timex(1) |
| and locale | timezone: set default system time zone | timezone(4) |
| STREAMS module | timod: Transport Interface cooperating | timod(7) |
| interface STREAMS module | tirdwr: Transport Interface read/write | tirdwr(7) |
| parameters | tkey: set label and data translation | tkey(1) |
| label | tlabel: initialize a tape with a volume | tlabel(1) |
| | t_listen: listen for a connect request | t_listen(3N) |
| transport endpoint | t_look: look at the current event on a | t_look(3N) |
| | tmpfile: create a temporary file | tmpfile(3S) |
| temporary file | tmpnam, tempnam: create a name for a | tmpnam(3S) |
| read (write) a curses screen from | (to) a file /scr_init, scr_set: | curs_scr_dump(3X) |
| toupper, tolower, _toupper, _tolower, | toascii: translate characters /conv: | conv(3C) |
| popen, pclose: initiate pipe | to/from a process | popen(3S) |
| vitr: Vilya | TokenRing Controller interface | vitr(7) |
| /conv: toupper, tolower, _toupper, | _tolower, toascii: translate characters | conv(3C) |
| translate characters conv: toupper, | tolower, _toupper, _tolower, toascii: | conv(3C) |
| execute environment-sensitive | tool /sde-chooser: | sde-chooser(4) |
| description interpreter idi_tools: | tools for use with the interface | idi_tools(1) |
| valtools: introduction to validation | tools | valtools(1) |
| | t_open: establish a transport endpoint | t_open(3N) |
| display a one-line summary about a | topic /whatis: | whatis(1) |
| tsort: | topological sort | tsort(1) |
| manipulation routines panel_top: | top_panel, bottom_panel: panels deck | panel_top(3X) |

| | | |
|---|---|---|
| menus items /current_item, set_top_row, | top_row, item_index: set and get current | menu_item_current(3X) |
| transport endpoint | t_optmgmt: manage options for a | t_optmgmt(3N) |
| acctmerg: merge or add | total accounting files | acctmerg(1M) |
| times of a file | touch: update access and modification | touch(1) |
| is_linetouched,/ curs_touch: touchwin, | touchline, untouchwin, wtouchln, | curs_touch(3X) |
| wtouchln, is_linetouched,/ curs_touch: | touchwin, touchline, untouchwin, | curs_touch(3X) |
| characters /conv: toupper, tolower, | _toupper, _tolower, toascii: translate | conv(3C) |
| toascii: translate characters conv: | toupper, tolower, _toupper, _tolower, | conv(3C) |
| wconv: towupper, | towlower: translate characters | wconv(3W) |
| /wconv: | towupper, towlower: translate characters | wconv(3W) |
| /set_curterm, del_curterm, restartterm, | tparm, tputs, putp, vidputs, vidattr,/ | curs_terminfo(3X) |
| | tposn: position tape to specified file | tposn(1) |
| terminfo database | tput: initialize a terminal or query | tput(1) |
| the/ /tgetflag, tgetnum, tgetstr, tgoto, | tputs: curses interfaces (emulated) to | curs_termcap(3X) |
| /del_curterm, restartterm, tparm, | tputs, putp, vidputs, vidattr, mvcur,/ | curs_terminfo(3X) |
| /tgetnum, tgetflag, tgetstr, tgoto, | tputs: terminal independent operation/ | termcap(3X) |
| | tr: translate characters | tr(1) |
| ctrace: | trace a C program to debug it | ctrace(1) |
| dg_xtrace: extended process | trace | dg_xtrace(2) |
| strace: print STREAMS | trace messages | strace(1M) |
| ptrace: process | trace | ptrace(2) |
| to STREAMS error logging and event | tracing /log: interface | log(7) |
| /pkgadd: | transfer software package to the system | pkgadd(1M) |
| strxfrm: string | transformation | strxfrm(3C) |
| wchar_t string operations and type | transformation /wstok, wstostr, strtows: | wstring(3W) |
| tolower, _toupper, _tolower, toascii: | translate characters /conv: toupper, | conv(3C) |
| tr: | translate characters | tr(1) |
| wconv: towupper, towlower: | translate characters | wconv(3W) |
| mailalias: | translate mail alias names | mailalias(1) |
| /cof2elf: | translate object file from COFF to ELF | cof2elf(1) |
| pkgtrans: | translate package format | pkgtrans(1) |
| postprint: | translate text files into PostScript | postprint(1) |
| elf32_xlatetom: class-dependent data | translation /elf_xlate: elf32_xlatetof, | elf_xlate(3E) |
| att_kbd: generalized string | translation module | att_kbd(7) |
| generic transport name-to-address | translation /netdir_sperror: | netdir(3N) |
| tkey: set label and data | translation parameters | tkey(1) |
| tdisplay: display label and record | translation settings | tdisplay(1) |
| ctl: COFF-to-legend | translator | ctl(1) |
| postdaisy: PostScript | translator for Diablo 630 files | postdaisy(1) |
| postdmd: PostScript | translator for DMD bitmap files | postdmd(1) |
| postplot: PostScript | translator for plot(4) graphics files | postplot(1) |
| posttek: PostScript | translator for tektronix 4014 files | posttek(1) |
| tcdrain, tcflush, tcflow: control data | transmission /tcsendbreak, | tcflush(3C) |
| encode/decode a binary file for | transmission via mail /uudecode: | uuencode(1) |
| t_bind: bind an address to a | transport endpoint | t_bind(3N) |
| t_close: close a | transport endpoint | t_close(3N) |
| t_look: look at the current event on a | transport endpoint | t_look(3N) |
| t_open: establish a | transport endpoint | t_open(3N) |
| t_optmgmt: manage options for a | transport endpoint | t_optmgmt(3N) |
| t_unbind: disable a | transport endpoint | t_unbind(3N) |
| module /timod: | Transport Interface cooperating STREAMS | timod(7) |
| STREAMS module /tirdwr: | Transport Interface read/write interface | tirdwr(7) |
| t_sync: synchronize | transport library | t_sync(3N) |
| /netdir_perror, netdir_sperror: generic | transport name-to-address translation | netdir(3N) |
| surrogate commands for routing and | transport of mail /mailsurr: | mailsurr(4M) |
| uucico: file | transport program for the uucp system | uucico(1M) |
| uusched: the scheduler for the uucp file | transport program | uusched(1M) |
| /nlsprovider: get name of | transport provider | nlsprovider(3N) |
| establish a connection with another | transport user /t_connect: | t_connect(3N) |
| /admsnmptrap: manage the SNMP | traps database | admsnmptrap(1M) |
| panel_above, panel_below: panels deck | traversal primitives /panel_above: | panel_above(3X) |
| sent over a connection | t_rcv: receive data or expedited data | t_rcv(3N) |
| from a connect request | t_rcvconnect: receive the confirmation | t_rcvconnect(3N) |
| disconnect | t_rcvdis: retrieve information from | t_rcvdis(3N) |
| orderly release indication | t_rcvrel: acknowledge receipt of an | t_rcvrel(3N) |
| | t_rcvudata: receive a data unit | t_rcvudata(3N) |
| indication | t_rcvuderr: receive a unit data error | t_rcvuderr(3N) |
| | tread: read file(s) from tape | tread(1) |
| ftw, nftw: walk a file | tree | ftw(3C) |
| tdelete, twalk: manage binary search | trees /tsearch, tfind, | tsearch(3C) |
| | trelease: terminate access to a tape | trelease(1) |

| | | |
|---|---|---|
| | umask: set file-creation mode mask | umask(1) |
| /mount, | umount: mount and dismount filesystems | mount(1M) |
| | umount: remove a file system device | umount(2) |
| system | uname, nuname: get name of current UNIX | uname(2) |
| | uname: print name of current system | uname(1) |
| display expanded files compress, | uncompress, zcat: compress, expand or | compress(1) |
| putwin, getwin,/ curs_util: | unctrl, keyname, filter, use_env, | curs_util(3X) |
| /mvdelch, mvwdelch: delete character | under cursor in a curses window. | curs_delch(3X) |
| /insert a character before the character | under the cursor in a curses window | curs_insch(3X) |
| /insert string before character | under the cursor in a curses window | curs_insstr(3X) |
| /a wchar_t character before the character | under the cursor in a curses window | curs_inswch(3X) |
| /insert wchar_t string before character | under the cursor in a curses window | curs_inswstr(3X) |
| ul: do | underlining | ul(1) |
| unget: | undo a previous get of an SCCS file | unget(1) |
| file | unget: undo a previous get of an SCCS | unget(1) |
| stream | ungetc: push character back onto input | ungetc(3S) |
| from/ /getch, wgetch, mvgetch, mvwgetch, | ungetch: get (or push back) characters | curs_getch(3X) |
| into input stream | ungetwc: push wchar_t character back | ungetwc(3W) |
| /getwch, wgetwch, mvgetwch, mvwgetwch, | ungetwch: get (or push back) wchar_t/ | curs_getwch(3X) |
| /srand48, seed48, lcong48: generate | uniformly distributed pseudo-random/ | drand48(3C) |
| /elf_rawfile: retrieve | uninterpreted file contents | elf_rawfile(3E) |
| | uniq: report repeated lines in a file | uniq(1) |
| mkstemp: make a | unique file name | mkstemp(3C) |
| mktemp: make a | unique file name | mktemp(3C) |
| gethostid: get | unique identifier of current host | gethostid(2) |
| sethostid: set | unique identifier of current host | sethostid(2) |
| connld: line discipline for | unique stream connections | connld(7) |
| systemid: display the | unique system identifier | systemid(1M) |
| t_rcvuderr: receive a | unit data error indication | t_rcvuderr(3N) |
| t_rcvudata: receive a data | unit | t_rcvudata(3N) |
| t_sndudata: send a data | unit | t_sndudata(3N) |
| | units: conversion program | units(1) |
| cu: call another | UNIX system | cu(1) |
| uname, nuname: get name of current | UNIX system | uname(2) |
| host | unix_ipc: piping communications within a | unix_ipc(6F) |
| /uux: | UNIX-to-UNIX system command execution | uux(1) |
| uucp, uulog, uuname: | UNIX-to-UNIX system copy | uucp(1) |
| uuto, uupick: public | UNIX-to-UNIX system file copy | uuto(1) |
| calls /link, | unlink: exercise link and unlink system | link(1M) |
| | unlink: remove a directory entry | unlink(2) |
| link, unlink: exercise link and | unlink system calls | link(1M) |
| pair unlockpt: | unlock a pseudo-terminal master/slave | unlockpt(3C) |
| mlockall, munlockall: lock or | unlock address space | mlockall(3C) |
| mlock, munlock: lock (or | unlock) pages in memory | mlock(3C) |
| master/slave pair | unlockpt: unlock a pseudo-terminal | unlockpt(3C) |
| munmap: | unmap pages of memory | munmap(2) |
| routines /finite, | unordered, copysign: IEEE floating-point | ieeefp(3C) |
| isnan, isnand, isnanf, finite, fpclass, | unordered: determine type of/ | isnan(3C) |
| pack, pcat, | unpack: compress and expand files | pack(1) |
| associated/ /form_post: post_form, | unpost_form: write or erase forms from | form_post(3X) |
| associated/ /menu_post: post_menu, | unpost_menu: write or erase menus from | menu_post(3X) |
| curs_touch: touchwin, touchline, | untouchwin, wtouchln, is_linetouched,/ | curs_touch(3X) |
| profile: setting | up an environment at login time | profile(4) |
| process profil: set | up execution time profiling for a | profil(2) |
| vhangup: virtually hang | up the current control terminal | vhangup(2) |
| bgets: read stream | up to next delimiter | bgets(3G) |
| a file /touch: | update access and modification times of | touch(1) |
| elf_update: | update an ELF descriptor | elf_update(3E) |
| programs make: maintain, | update, and regenerate groups of | make(1) |
| lsearch, lfind: linear search and | update | lsearch(3C) |
| sync: | update the super-block | sync(1M) |
| refresh routine /panel_update: | update_panels: panels virtual screen | panel_update(3X) |
| signal: specify what to do | upon presentation of a signal | signal(2) |
| sigset: specify what to do | upon presentation of a signal | sigset(2) |
| sigvec: specify what to do | upon presentation of a signal | sigvec(2) |
| du: summarize disk | usage | du(1) |
| retrieve a command description and | usage examples /usage: | usage(1) |
| by process key /dg_file_info: get file | usage information for process identified | dg_file_info(2) |
| and usage examples | usage: retrieve a command description | usage(1) |
| vtimes: get information about resource | usage | vtimes(3C) |
| mkmsgs: create message files for | use by gettxt | mkmsgs(1) |

| | | |
|---|---|---|
| devfree: release devices from exclusive | use | devfree(1M) |
| devreserv: reserve devices for exclusive | use | devreserv(1M) |
| kbdpipe: | use the KBD module in a pipeline | kbdpipe(1) |
| interpreter idi_tools: tools for | use with the interface description | idi_tools(1) |
| clock: report CPU time | used | clock(3C) |
| days /holidays: accounting information | used to distinguish prime and non-prime | holidays(4) |
| of severity levels for application to be | used with fmtmsg /build list | addseverity(3C) |
| lpfilter: administer filters | used with the LP print service | lpfilter(1M) |
| lpforms: administer forms | used with the LP print service | lpforms(1M) |
| /curs_util: unctrl, keyname, filter, | use_env, putwin, getwin, delay_output,/ | curs_util(3X) |
| logins: list | user and system login information | logins(1M) |
| setcontext: get and set current | user context /getcontext, | getcontext(2) |
| ucontext: | user context | ucontext(5) |
| /swapcontext: manipulate | user contexts | swapcontext(3C) |
| crontab: | user crontab file | crontab(1) |
| environ: | user environment | environ(5) |
| getdate, getdate_err: convert | user format date and time | getdate(3C) |
| chown, lchown: change | user id and group id of a file | chown(2) |
| fchown: change | user id and group id of a file | fchown(2) |
| ckuid: prompt for and validate a | user ID | ckuid(1) |
| generate disk accounting data by | user id /diskusg: | diskusg(1M) |
| seteuid: set the effective | user id of the current process | seteuid(2) |
| database admuser: manage | user information in the password | admuser(1M) |
| fingerd, in.fingerd: remote | user information server | fingerd(1M) |
| ulimit: get and set | user limits | ulimit(2) |
| listusers: list | user login information | listusers(1) |
| useradd: administer a new | user login on the system | useradd(1M) |
| logname: return login name of | user | logname(3X) |
| /id: print the | user name and ID, and group name and ID | id(1) |
| /cuserid: get character login name or | user name associated with effective UID | cuserid(3S) |
| dispuid: display a list of all valid | user names | dispuid(1) |
| notify: notify | user of the arrival of new mail | notify(1) |
| last: indicate last | user or terminal logins | last(1) |
| su: become super-user or another | user | su(1) |
| get character login name of the | user /sysv3_cuserid: | sysv3_cuserid(3S) |
| a connection with another transport | user /t_connect: establish | t_connect(3N) |
| the slot in the utmp file of the current | user /ttyslot: find | ttyslot(3C) |
| write: write to another | user | write(1) |
| /svcfd_create, svcudp_create, | user2netname, xdr_accepted_reply,/ | rpc(3N) |
| the system | useradd: administer a new user login on | useradd(1M) |
| system | userdel: delete a user's login from the | userdel(1M) |
| t_snddis: send | user-initiated disconnect request | t_snddis(3N) |
| information on the system | usermod: modify a user's login | usermod(1M) |
| text editor (variant of ex for casual | users) /edit: | edit(1) |
| information about local and remote | users /finger: display | finger(1) |
| userdel: delete a | user's login from the system | userdel(1M) |
| /usermod: modify a | user's login information on the system | usermod(1M) |
| mail, rmail: read mail or send mail to | users | mail(1) |
| starter: information for beginning | users | starter(1) |
| wall: write to all | users | wall(1M) |
| which: locate a program file for csh(1) | users | which(1) |
| call | ustat: data returned by the ustat system | ustat(5) |
| | ustat: get file system device statistics | ustat(2) |
| ustat: data returned by the | ustat system call | ustat(5) |
| syacdb: syac debugger | utility program | syacdb(1M) |
| flushinp: miscellaneous curses | utility routines /getwin, delay_output, | curs_util(3X) |
| get information about resource | utilization /getrusage: | getrusage(2) |
| times | utime: set file access and modification | utime(2) |
| times | utimes: set file access and modification | utimes(2) |
| utmp, wtmp: | utmp and wtmp entry formats | utmp(4) |
| setutent, endutent, utmpname: access | utmp file entry /getutline, pututline, | getut(3C) |
| ttyslot: find the slot in the | utmp file of the current user | ttyslot(3C) |
| | utmp, wtmp: utmp and wtmp entry formats | utmp(4) |
| /pututline, setutent, endutent, | utmpname: access utmp file entry | getut(3C) |
| permissions file | uucheck: check the uucp directories and | uucheck(1M) |
| uucp system | uucico: file transport program for the | uucico(1M) |
| | uucleanup: uucp spool directory clean-up | uucleanup(1M) |
| uucheck: check the | uucp directories and permissions file | uucheck(1M) |
| uusched: the scheduler for the | uucp file transport program | uusched(1M) |
| uucleanup: | uucp spool directory clean-up | uucleanup(1M) |
| uustat: | uucp status inquiry and job control | uustat(1) |

| | | |
|---|---|---|
| compver: compatible | versions file | compver(4) |
| sccsdiff: compare two | versions of an SCCS file | sccsdiff(1) |
| create curses borders, horizontal and | vertical lines /box, whline, wvline: | curs_border(3X) |
| memory efficient way | vfork: spawn new process in a virtual | vfork(2) |
| output of a variable argument/ vprintf, | vfprintf, vsprintf: print formatted | vprintf(3S) |
| output of a variable argument/ vprintf, | vfprintf, vsprintf: print formatted | vprintf(3W) |
| input using varargs argument/ vscanf, | vfscanf, vsscanf: convert formatted | vscanf(3S) |
| control terminal | vhangup: virtually hang up the current | vhangup(2) |
| (visual) display editor based on ex | vi, vedit, view: screen-oriented | vi(1) |
| a binary file for transmission | via mail /uudecode: encode/decode | uuencode(1) |
| make a directory available for mounting | via NFS /exportfs: | exportfs(2) |
| nlsgetcall: get client's data passed | via the listener | nlsgetcall(3N) |
| tigetstr:/ /tparm, tputs, putp, vidputs, | vidattr, mvcur, tigetflag, tigetnum, | curs_terminfo(3X) |
| /restartterm, tparm, tputs, putp, | vidputs, vidattr, mvcur, tigetflag,/ | curs_terminfo(3X) |
| editor based on ex /vi, vedit, | view: screen-oriented (visual) display | vi(1) |
| point directory /cpd: change or | view the allocation limits for a control | cpd(1) |
| vitr: | Vilya TokenRing Controller interface | vitr(7) |
| | vipw: edit the system password file | vipw(1M) |
| vfork: spawn new process in a | virtual memory efficient way | vfork(2) |
| move_panel: move a panels window on the | virtual screen /panel_move: | panel_move(3X) |
| /panel_update: update_panels: panels | virtual screen refresh routine | panel_update(3X) |
| terminal vhangup: | virtually hang up the current control | vhangup(2) |
| item_visible: tell if menus item is | visible /menu_item_visible: | menu_item_visible(3X) |
| vi, vedit, view: screen-oriented | (visual) display editor based on ex | vi(1) |
| interface | vitr: Vilya TokenRing Controller | vitr(7) |
| consumption | vlimit: control maximum system resource | vlimit(3C) |
| label checking | volcopy, labelit: copy file systems with | volcopy(1M) |
| tlabel: initialize a tape with a | volume label | tlabel(1) |
| formatted output of a variable argument/ | vprintf, vfprintf, vsprintf: print | vprintf(3S) |
| formatted output of a variable argument/ | vprintf, vfprintf, vsprintf: print | vprintf(3W) |
| /vsccheck: verify that the | VSC synchronous controller is operable | vsccheck(1M) |
| download board resident software onto | VSC synchronous controller /vscload: | vscload(1M) |
| formatted input using varargs argument/ | vscanf, vfscanf, vsscanf: convert | vscanf(3S) |
| synchronous controller is operable | vsccheck: verify that the VSC | vsccheck(1M) |
| software onto VSC synchronous/ | vscload: download board resident | vscload(1M) |
| variable argument/ vprintf, vfprintf, | vsprintf: print formatted output of a | vprintf(3S) |
| variable argument/ vprintf, vfprintf, | vsprintf: print formatted output of a | vprintf(3W) |
| varargs argument list /vscanf, vfscanf, | vsscanf: convert formatted input using | vscanf(3S) |
| usage | vtimes: get information about resource | vtimes(3C) |
| /printw, wprintw, mvprintw, mvwprintw, | vwprintw: print formatted output in/ | curs_printw(3X) |
| /scanw, wscanw, mvscanw, mvwscanw, | vwscanw: convert formatted input from a/ | curs_scanw(3X) |
| wechochar: add a/ curs_addch: addch, | waddch, mvaddch, mvwaddch, echochar, | curs_addch(3X) |
| /addchstr, addchnstr, waddchstr, | waddchnstr, mvaddchstr, mvaddchnstr,/ | curs_addchst(3X) |
| /addchstr, addchnstr, waddchstr, | waddchnstr, mvaddchstr, mvaddchnstr,/ | curs_addchstr(3X) |
| /curs_addchstr: addchstr, addchnstr, | waddchstr, waddchnstr, mvaddchstr,/ | curs_addchst(3X) |
| /curs_addchstr: addchstr, addchnstr, | waddchstr, waddchnstr, mvaddchstr,/ | curs_addchstr(3X) |
| /curs_addstr: addstr, addnstr, waddstr, | waddnstr, mvaddstr, mvaddnstr,/ | curs_addstr(3X) |
| /addwstr, addnwstr, waddwstr, | waddnwstr, mvaddwstr, mvaddnwstr,/ | curs_addwstr(3X) |
| /curs_addstr: addstr, addnstr, | waddstr, waddnstr, mvaddstr, mvaddnstr,/ | curs_addstr(3X) |
| wechowchar: add a/ /curs_addwch: addwch, | waddwch, mvaddwch, mvwaddwch, echowchar, | curs_addwch(3X) |
| /addwchstr, addwchnstr, waddwchstr, | waddwchnstr, mvaddwchstr, mvaddwchnstr,/ | curs_addwchstr(3X) |
| /curs_addwchstr: addwchstr, addwchnstr, | waddwchstr, waddwchnstr, mvaddwchstr,/ | curs_addwchstr(3X) |
| /curs_addwstr: addwstr, addnwstr, | waddwstr, waddnwstr, mvaddwstr,/ | curs_addwstr(3X) |
| | wait: await completion of process | wait(1) |
| sigsuspend: | wait for a signal | sigsuspend(2) |
| /waitid: | wait for child process to change state | waitid(2) |
| terminate wait3: | wait for child process to stop or | wait3(2) |
| select: | wait for I/O conditions | select(2) |
| requests to complete /dg_lock_wait: | wait for previously delayed lock | dg_lock_wait(2) |
| wait, waitpid: | wait for process termination | wait(2) |
| stop or terminate /wait4: | wait for the specified child process to | wait4(2) |
| wstat: | wait status | wstat(5) |
| termination | wait, waitpid: wait for process | wait(2) |
| terminate | wait3: wait for child process to stop or | wait3(2) |
| process to stop or terminate | wait4: wait for the specified child | wait4(2) |
| state | waitid: wait for child process to change | waitid(2) |
| wait, | waitpid: wait for process termination | wait(2) |
| ftw, nftw: | walk a file tree | ftw(3C) |
| | wall: write to all users | wall(1M) |
| wattrset,/ curs_attr: attroff, | wattroff, attron, wattron, attrset, | curs_attr(3X) |
| /curs_attr: attroff, wattroff, attron, | wattron, attrset, wattrset, standend,/ | curs_attr(3X) |

| | | |
|---|---|---|
| /xdrrec_endofrecord, xdrrec_eof, | xdrrec_skiprecord, xdr_reference,/ | xdr(3N) |
| /xdrrec_eof, xdrrec_skiprecord, | xdr_reference, xdr_setpos, xdr_short,/ | xdr(3N) |
| /xdr_opaque_auth, xdr_pmap, xdr_pmaplist, | xdr_rejected_reply, xdr_replymsg,/ | rpc(3N) |
| xdr_pmaplist, xdr_rejected_reply, | xdr_replymsg, xprt_register,/ /xdr_pmap, | rpc(3N) |
| /xdrrec_skiprecord, xdr_reference, | xdr_setpos, xdr_short, xdrstdio_create,/ | xdr(3N) |
| xdr_u_char,/ /xdr_reference, xdr_setpos, | xdr_short, xdrstdio_create, xdr_string, | xdr(3N) |
| /xdr_reference, xdr_setpos, xdr_short, | xdrstdio_create, xdr_string, xdr_u_char,/ | xdr(3N) |
| /xdr_setpos, xdr_short, xdrstdio_create, | xdr_string, xdr_u_char, xdr_u_int,/ | xdr(3N) |
| /xdr_short, xdrstdio_create, xdr_string, | xdr_u_char, xdr_u_int, xdr_u_long,/ | xdr(3N) |
| /xdrstdio_create, xdr_string, xdr_u_char, | xdr_u_int, xdr_u_long, xdr_u_short,/ | xdr(3N) |
| /xdr_string, xdr_u_char, xdr_u_int, | xdr_u_long, xdr_u_short, xdr_union,/ | xdr(3N) |
| /xdr_u_int, xdr_u_long, xdr_u_short, | xdr_union, xdr_vector, xdr_void,/ | xdr(3N) |
| /xdr_u_char, xdr_u_int, xdr_u_long, | xdr_u_short, xdr_union, xdr_vector,/ | xdr(3N) |
| /xdr_u_long, xdr_u_short, xdr_union, | xdr_vector, xdr_void, xdr_wrapstring:/ | xdr(3N) |
| /xdr_u_short, xdr_union, xdr_vector, | xdr_void, xdr_wrapstring: library/ | xdr(3N) |
| /xdr_union, xdr_vector, xdr_void, | xdr_wrapstring: library routines for/ | xdr(3N) |
| ttcompat: V7, 4BSD and | XENIX STREAMS compatibility module | ttcompat(7) |
| lpprint, | xlpprint: menu-driven lp interface | lpprint(1M) |
| /xdr_rejected_reply, xdr_replymsg, | xprt_register, xprt_unregister: library/ | rpc(3N) |
| remote/ /xdr_replymsg, xprt_register, | xprt_unregister: library routines for | rpc(3N) |
| implement shared strings | xstr: extract strings from C programs to | xstr(1) |
| administration interface sysadm, | xsysadm: menu-driven system | sysadm(1M) |
| bessel: j0, j1, jn, | y0, y1, yn: Bessel functions | bessel(3M) |
| bessel: j0, j1, jn, y0, | y1, yn: Bessel functions | bessel(3M) |
| | yacc: yet another compiler-compiler | yacc(1) |
| ckyorn: prompt for and validate | yes/no | ckyorn(1) |
| bessel: j0, j1, jn, y0, y1, | yn: Bessel functions | bessel(3M) |
| yp_unbind, yp_match, yp_first, yp_next, | yp_all, yp_order, yp_master,/ /yp_bind, | ypclnt(3N) |
| yp_next,/ /ypclnt, yp_get_default_domain, | yp_bind, yp_unbind, yp_match, yp_first, | ypclnt(3N) |
| yp_unbind, yp_match, yp_first, yp_next,/ | ypclnt, yp_get_default_domain, yp_bind, | ypclnt(3N) |
| /yp_next, yp_all, yp_order, yp_master, | yperr_string, ypprot_err: Network/ | ypclnt(3N) |
| /yp_bind, yp_unbind, yp_match, | yp_first, yp_next, yp_all, yp_order,/ | ypclnt(3N) |
| yp_unbind, yp_match, yp_first,/ ypclnt, | yp_get_default_domain, yp_bind, | ypclnt(3N) |
| /yp_first, yp_next, yp_all, yp_order, | yp_master, yperr_string, ypprot_err:/ | ypclnt(3N) |
| yp_order,/ /yp_bind, yp_unbind, | yp_match, yp_first, yp_next, yp_all, | ypclnt(3N) |
| /yp_bind, yp_unbind, yp_match, yp_first, | yp_next, yp_all, yp_order, yp_master,/ | ypclnt(3N) |
| /yp_match, yp_first, yp_next, yp_all, | yp_order, yp_master, yperr_string,/ | ypclnt(3N) |
| /yp_order, yp_master, yperr_string, | ypprot_err: Network Information Service/ | ypclnt(3N) |
| /ypclnt, yp_get_default_domain, yp_bind, | yp_unbind, yp_match, yp_first, yp_next,/ | ypclnt(3N) |
| expanded files compress, uncompress, | zcat: compress, expand or display | compress(1) |
| | zdump: time zone dumper | zdump(1M) |
| bzero: | zero a portion of memory | bzero(3C) |
| | zero: source of zeroes | zero(7) |
| zero: source of | zeroes | zero(7) |
| | zic: time zone compiler | zic(1M) |
| the system date, time and time | zone /admdate: manipulate | admdate(1M) |
| timezone: set default system time | zone and locale | timezone(4) |
| zic: time | zone compiler | zic(1M) |
| zdump: time | zone dumper | zdump(1M) |

End of Chapter

# Index

# Related Documents

The following list of related manuals gives titles of Data General manuals followed by nine-digit numbers used for ordering. You can order any of these manuals via mail or telephone (see the TIPS Order Form in the back of this manual).

For a complete list of AViiON® and DG/UX™ manuals, see the *Guide to AViiON® and DG/UX™ Documentation* (069-701085). The on-line version of this manual found in **/usr/release/doc_guide** contains the most current list.

# Data General Software Manuals

## User's Manuals

*User's Reference for the DG/UX™ System*
Contains an alphabetical listing of manual pages for commands relating to general system operation. Ordering Number — 093-701054

*Using the DG/UX™ Editors*
Describes the text editors **vi** and **ed**, the batch editor **sed**, and the command line editor **editread**. Ordering Number — 069-701036

*Using the DG/UX™ System*
Describes the DG/UX system and its major features, including the C and Bourne shells, typical user commands, the file system, and communications facilities such as **mailx**. Ordering Number — 069-701035

## Installation and Administration Manuals

*System Manager's Reference for the DG/UX™ System*
Contains an alphabetical listing of manual pages for commands relating to system administration or operation. Ordering Number — 093-701050

## Programming Manuals

*Porting and Developing Applications on the DG/UX™ System*
A compendium of useful information for experienced programmers developing or porting applications to the DG/UX™ system. It includes information on how to: set up your environment, use the software development tools, compile and link programs, port to the windowing environment, and build BCS applications. It also describes available debuggers and the various industry standards the DG/UX system supports. Ordering Number — 069-701059

*Programmer's Guide: ANSI C and Programming Support Tools (UNIX System V Release 4)*
Describes the standard tools of the UNIX program development environment including compiling, linking, debugging, and analysis and revision control. An accompanying supplement, *Supplement for Programmer's Guide: ANSI C and Programming Support Tools* (086-000180) describes the DG/UX system enhancements and differences. Ordering Number — 093-701104

*Programmer's Guide: Systems Services and Application Packaging Tools (UNIX System V Release 4)*
Describes standard programming procedures and interfaces available to the C application developer in the UNIX environment. Topics include interprocess communications, memory management, file and record locking and application packaging. Note: Chapters 5 and 9 of this Prentice Hall manual discuss topics that do not apply to the DG/UX system. Ordering Number — 093-701103

*Programmer's Reference for the DG/UX™ System, (Volume 1)*
Alphabetical listing of manual pages for DG/UX programming commands and system calls. This is part of a three-volume set. Ordering Number — 093-701055

*Programmer's Reference for the DG/UX™ System, (Volume 2)*
Alphabetical listing of manual pages for DG/UX subroutines and libraries. This is part of a three-volume set. Ordering Number — 093-701056

End of Related Documents

# TIPS ORDERING PROCEDURES

## TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a) MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

   Send your order form with payment to:

   Data General Corporation
   ATTN: Educational Services/TIPS G155
   4400 Computer Drive
   Westboro, MA 01581–9973

   b) TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

## METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a) Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.
   b) Check or Money Order – Make payable to Data General Corporation.
   c) Credit Card – A minimum order of $20 is required for Mastercard or Visa orders.

## SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

| Total Quantity | Shipping & Handling Charge |
|---|---|
| 1–4 Units | $5.00 |
| 5–10 Units | $8.00 |
| 11–40 Units | $10.00 |
| 41–200 Units | $30.00 |
| Over 200 Units | $100.00 |

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

## VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

| Order Amount | Discount |
|---|---|
| $1–$149.99 | 0% |
| $150–$499.99 | 10% |
| Over $500 | 20% |

## TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

## DELIVERY

6. Allow at least two weeks for delivery.

## RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

## INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

# TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

| BILL TO: | SHIP TO: (No P.O. Boxes - Complete Only If Different Address) |
|---|---|
| COMPANY NAME________ | COMPANY NAME________ |
| ATTN:________ | ATTN:________ |
| ADDRESS________ | ADDRESS (NO PO BOXES)________ |
| CITY________ | CITY________ |
| STATE________ ZIP________ | STATE________ ZIP________ |

Priority Code ________ (See label on back of catalog)

| Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext. |
|---|---|---|---|---|
| | | | | |

| ORDER # | QTY | DESCRIPTION | UNIT PRICE | TOTAL PRICE |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

| | |
|---|---|
| ORDER TOTAL | |
| Less Discount See B | – |
| SUB TOTAL | |
| Your local* sales tax | + |
| Shipping and handling – See A | + |
| TOTAL – See C | |

Tax Exempt # or Sales Tax (if applicable) ________

**A**

| ☐ UPS | ADD |
|---|---|
| 1-4 Items | $ 5.00 |
| 5-10 Items | $ 8.00 |
| 11-40 Items | $ 10.00 |
| 41-200 Items | $ 30.00 |
| 200+ Items | $100.00 |

**Check for faster delivery**

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)

☐ Red Label (overnight shipping)

**B VOLUME DISCOUNTS**

| Order Amount | Save |
|---|---|
| $0 – $149.99 | 0% |
| $150 – $499.99 | 10% |
| Over $500.00 | 20% |

**C PAYMENT METHOD**

☐ Purchase Order Attached ($50 minimum)
P.O. number is________. (Include hardcopy P.O.)

☐ Check or Money Order Enclosed

☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiration Date ☐☐☐☐

________

Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508-870-1600.

Form 702
Rev. 8/87

# DATA GENERAL CORPORATION TECHNICAL INFORMATION AND PUBLICATIONS SERVICE TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

**1. CUSTOMER CERTIFICATION**
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

**2. TAXES**
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

**3. DATA AND PROPRIETARY RIGHTS**
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

**4. LIMITED MEDIA WARRANTY**
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

**5. DISCLAIMER OF WARRANTY**
**EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.**

**6. LIMITATION OF LIABILITY**
**A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.**

**B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.**

**7. GENERAL**
A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

**8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)**
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

Cut here and insert in binder spine pocket

Data General

Data General Corporation, Westboro, Massachusetts 01580