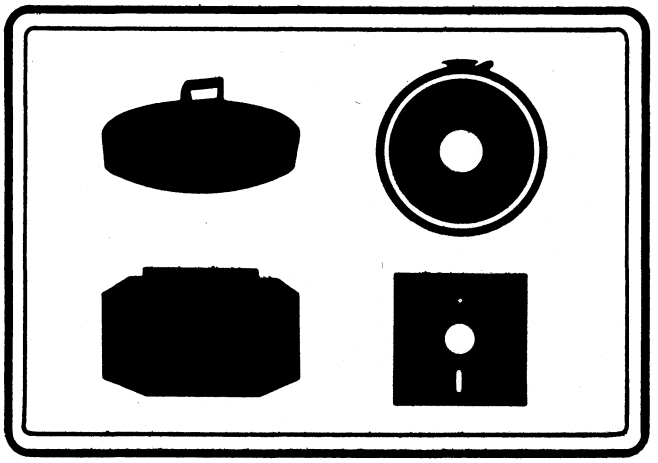
DataGeneral

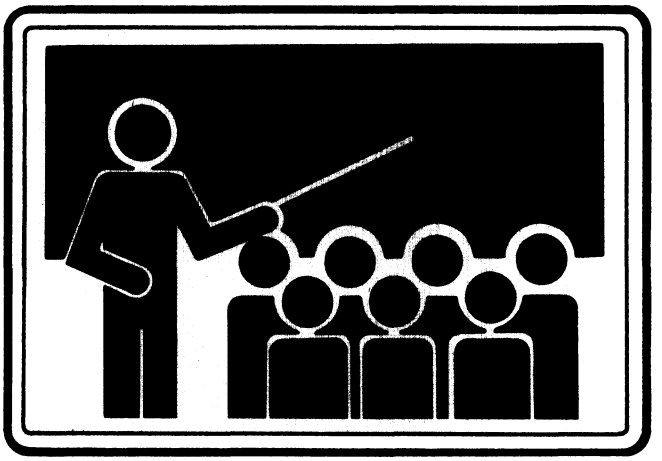
Educational Services

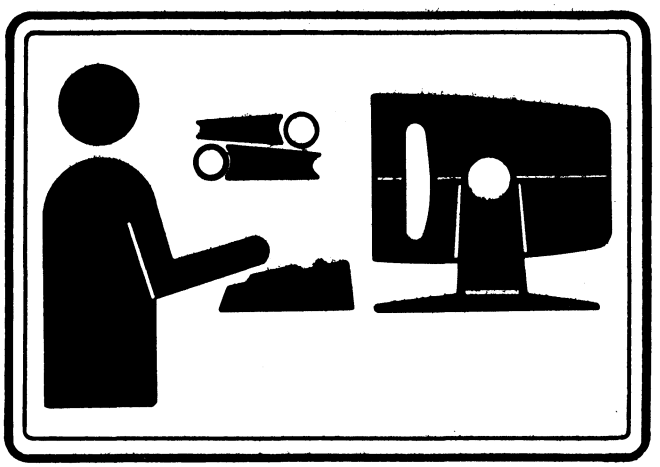
S309/VS
AOS/VS SYSTEM PROGRAMMING

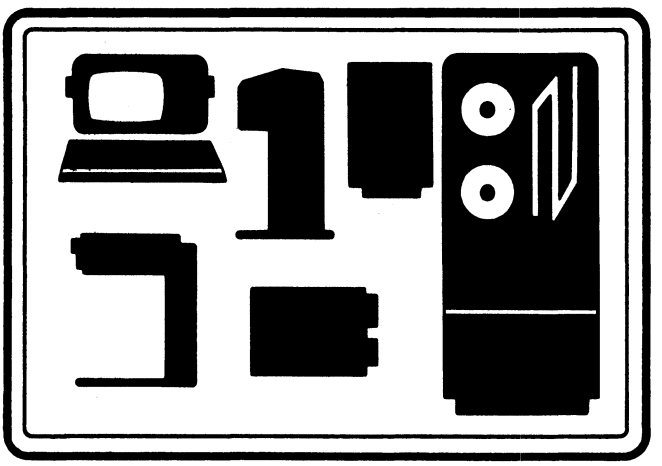
LABS

Appendix A – Assembler Language
Appendix B – FORTRAN 77
Appendix C – C Language









019-000162-01

NOTICE

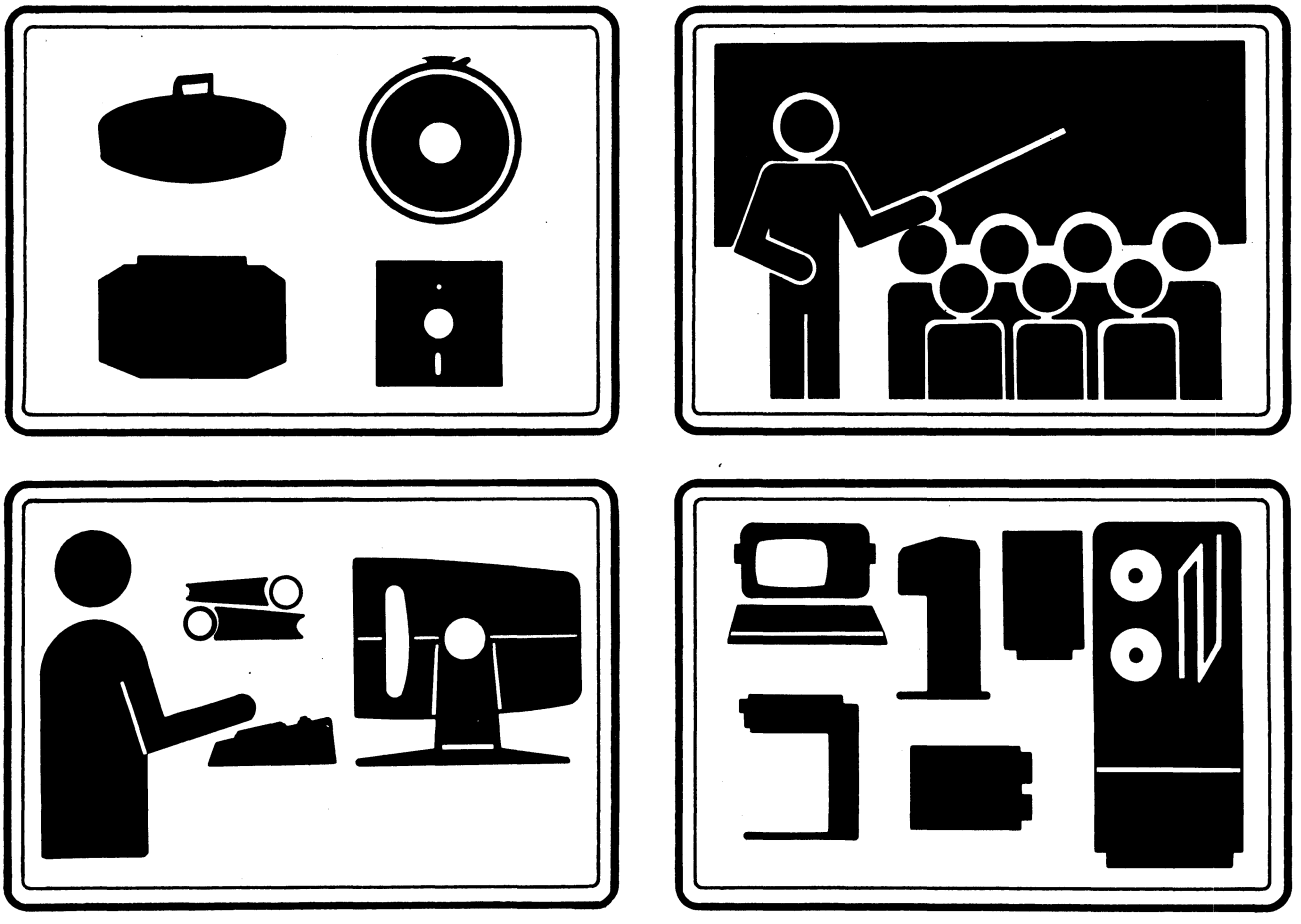
DataGeneral

Educational Services

S309/VS
AOS/VS SYSTEM PROGRAMMING

LABS

Appendix A - Assembler Language
Appendix B - FORTRAN 77
Appendix C - C Language



019-000162-01

ASSEMBLY LANGUAGE FILE SYSTEM LAB

* MOVE/R/NACL COPIES OF THE FILES 'CREATE.SR' AND 'ERROR.OB' FROM THE :S309VS:ASSY DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'CREATE' IS TO FIRST CREATE A DIRECTORY, THE NAME OF WHICH WILL BE "MYDIR". THE PROGRAM SHOULD NOT FAIL IF THE DIRECTORY ALREADY EXISTS. IT WILL THEN MAKE "MYDIR" THE WORKING DIRECTORY. (IF "MYDIR" IS NOT A DIRECTORY THEN THE PROGRAM WILL FAIL AND YOU WILL HAVE TO DELETE IT.) THE PROGRAM WILL THEN ATTEMPT TO DELETE AND CREATE A NEW FILE CALLED "MYFILE".

* THE FUNCTION OF 'ERROR' IS TO PROVIDE AN ERROR HANDLER FOR THE SYSTEM CALLS. IT WILL REPORT THE ERROR MESSAGE AND PROGRAM LOCATION WHEN A SYSTEM CALL FAILS TO EXECUTE PROPERLY.

* REFER TO THE PRINTOUT OF 'CREATE.SR' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM CALLS AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
DELETE/2=IGNORE,CREATE.LS
X,MASM/L=CREATE.LS,CREATE
X,LINK/L=CREATE.LS,CREATE,ERROR
```

* GET A HARDCOPY OF THE FILE 'CREATE.LS'.

* EXECUTE THE CREATE PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'FILE CREATION COMPLETED'.

* DO A FILESTATUS WITH /TYPE ON THE FILE 'MYDIR'. IT SHOULD BE TYPE "DIR".

* DO A DIRECTORY TO "MYDIR" THEN DO ANOTHER FILESTATUS/TYPE. YOU SHOULD FIND THAT "MYFILE" IS A TYPE "TXT".

* GO BACK TO YOUR PREVIOUS DIRECTORY AND EXECUTE THE PROGRAM AGAIN. THE PROGRAM SHOULD NOT FAIL AS IT WILL DELETE 'MYFILE' BEFORE TRYING TO CREATE IT AGAIN.

* YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE PROGRAM REPEATEDLY WITHOUT ERROR

```
        .TITL   CREATE  ; DELETES AND CREATES A FILE AND DIR
        .ENT    START,DIR,DELETE,CREATE
        .ENT    DLERR,CRERR,DIRPK,CREPK
        .ENT    FNAME,DNAME
        .NREL   1               ; SHARED CODE

START:
        LLEFB   0,DNAME*2       ; NAME OF DIR
        ??????? ?????          ; ATTEMPT TO CREATE THE DIR
        WBR     CRERR           ; CHECK THE ERROR

DIR:
        LLEFB   0,DNAME*2       ; NAME OF DIR
        WSUB    1,1
        WSUB    2,2
        ????                    ; CHANGE WORKING DIRECTORY
        DERR    1

DELETE:
        LLEFB   0,FNAME*2       ; NAME OF FILE
        ???????                 ; ATTEMPT TO DELETE THE FILE
        WBR     DLERR
CREATE:
        LLEFB   0,FNAME*2       ; NAME OF FILE
        ??????? ?????          ; ATTEMPT TO CREATE THE FILE
        DERR    2

RETURN:
        LLEFB   1,MSG*2
        NLDAI   MSGLN,2
        ?RETURN                 ; TERMINATE NORMALLY.
        DERR    3

MSG:    .TXT    /File creation completed./
MSGLN = .-MSG*2


;  ROUTINE TO CHECK ?DELETE ERROR

DLERR:
        WSNEI   ERFDE,0         ; WAS IT FILE DOES NOT EXIST ?
        WBR     CREATE          ; YES. IGNORE THE ERROR
        DERR    4               ; NO. TERMINATE WITH ERROR.

;  ROUTINE TO CHECK ?CREATE ERROR

CRERR:
        WSNEI   ERNAE,0  ; WAS IT FILE NAME ALREADY EXISTS ?
        WBR     DIR             ; YES. IGNORE THE ERROR
        DERR    5               ; NO. TERMINATE WITH ERROR.
```

```
        .NREL   0               ; UNSHARED

;  PARAMETER PACKET FOR FILE CREATION

CREPK:  .LOC    CREPK+?CFTYP    ; ENTRY TYPE AND RECORD FORMAT
        .WORD   ?????           ; FILE TYPE TEXT,NO RECORD FMT
        .LOC    CREPK+?CCPS     ; RECORD SIZE FOR FIXED
        .WORD   0               ; (IGNORED UNLESS FIXED RECORDS)
        .LOC    CREPK+?CTIM     ; ADDR OF TIME BLOCK
        .DWORD  ??              ; (USE CURRENT DATE AND TIME)
        .LOC    CREPK+?CACP     ; ACL BYTE PTR
        .DWORD  ??              ; (USE DEFAULT ACL)
        .LOC    CREPK+?CDEH     ; (RESERVED)
        .WORD   0               ;
        .LOC    CREPK+?CDEL     ; ELEMENT SIZE
        .WORD   ??              ; (USE SYSTEM DEFAULT)
        .LOC    CREPK+?CMIL     ; MAX INDEX LEVEL
        .WORD   ??              ; (USE DEFAULT SETTING)
        .LOC    CREPK+?CMRS     ; (RESERVED)
        .WORD   0               ;
        .LOC    CREPK+?CLTH     ; POSITION TO END OF PACKET

FNAME:   .TXT    /MYFILE/       ; FILENAME

;  PARAMETER PACKET FOR DIRECTORY CREATION

DIRPK:  .LOC    DIRPK+?CFTYP    ; ENTRY TYPE AND RECORD FORMAT
        .WORD   ?????           ; (USE FILE TYPE DIRECTORY)
        .LOC    DIRPK+?CHFS     ; DIRECTORY HASH FRAME SIZE
        .WORD   ??              ; (USE DEFAULT SIZE)
        .LOC    DIRPK+?CTIM     ; ADDR OF TIME BLOCK
        .DWORD  ??              ; (USE CURRENT DATE AND TIME)
        .LOC    DIRPK+?CACP     ; ACL BYTE PTR
        .DWORD  ??              ; (USE DEFAULT ACL)
        .LOC    DIRPK+?CMSH     ; MAX SIZE VALUE FOR CPD
        .DWORD  0               ; (MUST BE ZERO IF NON-CPD)
        .LOC    DIRPK+?CMIL     ; MAX INDEX LEVEL
        .WORD   ??              ; (SET TO DEFAULT)
        .LOC    DIRPK+?CMRS     ; (RESERVED)
        .WORD   0               ;
        .LOC    DIRPK+?CLTH     ; POSITION TO END OF PACKET

DNAME:   .TXT    /MYDIR/        ; DIRECTORY NAME

        .END    START
```

ASSEMBLY LANGUAGE FILE ACCESS LAB

* MOVE/R/NACL COPIES OF THE FILES 'CHANGEACL.SR' AND 'ERROR.OB' FROM THE :S309VS:ASSY DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'CHANGEACL' IS TO FIRST MAKE ITS WORKING DIRECTORY A SUB-DIRECTORY CALLED "MYDIR". THE PROGRAM WILL FAIL IF THE DIRECTORY DOES NOT EXIST. IT WILL THEN READ THE ACL OF A FILE CALLED "MYFILE" IN THAT DIRECTORY. THIS WILL ALSO FAIL IF THE FILE DOES NOT EXIST. NEXT IT WILL WRITE A NEW ACL FOR THE FILE, ADDING "+,RE" ACCESS TO THE PREVIOUS STRING.

NOTE: IF YOU HAVE NOT DONE THE "CREATE_LAB" BY THIS TIME YOU WILL HAVE TO CREATE THE NECESSARY FILES FOR THIS LAB AS FOLLOWS:

```
) CR/DIR,MYDIR
) CR,MYDIR:MYFILE
```

* REFER TO THE PRINTOUT OF 'CHANGEACL.SR' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUE. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS, AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
DELETE/2=IGNORE,CHANGEACL.LS
X,MASM/L=CHANGEACL.LS,CHANGEACL
X,LINK/L=CHANGEACL.LS,CHANGEACL,ERROR
```

* GET A HARDCOPY OF THE FILE 'CREATE.LS'.

* DO AN ACL WITH /V ON THE FILE 'MYDIR:MYFILE'.
IT SHOULD BE YOUR DEFAULT ACL (USERNAME,OWARE).

* EXECUTE THE CREATE PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'THE FILE'S ACL HAS BEEN CHANGED."

* DO ANOTHER ACL WITH /V ON THE FILE 'MYDIR:MYFILE'. IT SHOULD NOW BE YOUR DEFAULT ACL AND "+,RE".

* YOU HAVE COMPLETED THIS LAB WHEN MYFILE:MYDIR HAS THE ACCESS CONTROL LIST "USERNAME,OWARE +,RE"

```
        .TITL   CHANGEACL       ; CHANGES A FILE'S ACL
        .ENT    START,GETACL,SETACL,DELIM
        .ENT    BUFFER,COMBINE,RETURN
        .NREL   1               ; SHARED

START:
GETACL:
        LLEFB   0,?????*2       ; BPTR TO NAME OF FILE
        LLEFB   1,??????*2      ; BPTR TO BUFFER
        WSUB    2,2             ; (RESERVED - SET TO ZERO)
        ?????                   ; GET ACCESS CONTROL LIST
        DERR    1

COMBINE:
        LLEF    0,DELIM         ; GET DELIM TBL ADDR
        WLDAI   -?MXACL,1       ; GET MAX ACL LENGTH
        LLEFB   2,BUFEND*2      ; BP TO END OF BUFFER
        LLEFB   3,BUFEND*2      ; SAME BP AS AC2
        WCMT                    ; SCAN FOR NOT NULL

        NLDAI   ACLLN,0         ; GET # BYTES TO MOVE TO
        NLDAI   ACLLN,1         ; GET # BYTES TO MOVE FROM
        WINC    2,2             ; DESTINATION FOR BYTES
        LLEFB   3,ACLTXT*2      ; SOURCE OF BYTES
        WCMV                    ; ADD EXTENSION TO BUFFER

SETACL:
        LLEFB   0,?????*2       ; BPTR TO NAME OF FILE
        LLEFB   1,??????*2      ; BPTR TO BUFFER
        WSUB    2,2             ; (RESERVED - SET TO ZERO)
        ?????                   ; SET ACL TO FILE
        DERR    2

RETURN:
        LLEFB   1,MSG*2
        NLDAI   MSGLN,2
        ?RETURN                 ; TERMINATE NORMALLY.
        DERR    3
```

```
MSG:     .TXT    "The file's ACL has been changed."
MSGLN =  (.-MSG)*2

FNAME:   .TXT    "MYDIR:MYFILE"

ACLTXT:  .TXT    "+<0><???????????>"      ; TEXT FOR "+,RE" ACCESS
ACLLN=   (.-ACLTXT)*2

DELIM:   .WORD   ~1B0     ; ALL BITS EXCEPT BIT ZERO
         .DO     15.      ; ARE SET IN THIS 256. BIT TABLE
         .WORD   -1
         .ENDC

         .NREL   0                ; UNSHARED

BUFFER:  .BLK    ?MXACL/2
BUFEND:

         .END    START
```

ASSEMBLY LANGUAGE FILE STATUS LAB

* MOVE/R/NACL COPIES OF THE FILES 'FILES.SR', '$IO.OB', AND 'ERROR.OB' FROM THE :S309VS:ASSY DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'FILES' IS TO FIRST GO TO A DIRECTORY, SPECIFIED AS THE FIRST ARGUMENT ON THE COMMAND LINE. THE PROGRAM WILL FAIL IF THE DIRECTORY DOES NOT EXIST. IT WILL THEN GET A FILENAME WHICH MATCHES THE TEMPLATE SPECIFIED AS THE SECOND ARGUMENT ON THE COMMAND LINE. IT WILL PASS THE FILENAME AND BYTE LENGTH TO '$IO' FOR OUTPUT. THEN IT WILL GET THE NEXT FILENAME AND REPEAT THE PROCESS. WHEN ALL THE FILES FOR A TEMPLATE HAVE BEEN RECOVERED, THE PROGRAM WILL GET THE NEXT TEMPLATE FROM THE COMMAND LINE UNTIL ALL THE FILES HAVE BEEN DISPLAYED.

* THE FUNCTION OF '$IO' IS TO DISPLAY THE INFORMATION PASSED TO IT BY 'FILES'.

* THE FUNCTION OF 'ERROR' IS TO PROVIDE AN ERROR HANDLER FOR THE SYSTEM CALLS. IT WILL REPORT THE ERROR MESSAGE AND PROGRAM LOCATION WHEN A SYSTEM CALL FAILS TO EXECUTE PROPERLY.

* REFER TO THE PRINTOUT OF 'FILES.SR' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUE. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS, AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
DELETE/2=IGNORE,FILES.LS
X,MASM/L=FILES.LS,FILES
X,LINK/L=FILES.LS,FILES,$IO,ERROR
```

* GET A HARDCOPY OF THE FILE 'FILES.LS'.

* EXECUTE THE FILES PROGRAM WITH THE FOLLOWING COMMAND:

```
X,FILES,:UDD:yourusername,template...
```

* YOU SHOULD SOON SEE THE FILENAMES AND BYTE LENGTHS FOLLOWED BY THE MESSAGE 'INFORMATION PROCESSING COMPLETED'.

* DO A FILESTATUS WITH /LENGTH IN THE DIRECTORY IN YOUR COMMAND LINE AND CHECK THE RESULTS.

* YOU HAVE COMPLETED THIS LAB WHEN YOUR PROGRAM YIELDS THE SAME RESULTS AS THE FILESTATUS/LENGTH COMMAND.

```
        .TITL   FILES           ; RETURNS FILESTATUS INFORMATION
        .ENT    START,GOPEN,GTMES,GNFN,RETURN,FSTAT
        .ENT    GTMPK,BUFFER,FNAME
        .ENT    FSPK,END
        .NREL   1               ; SHARED

; Note: INITIO, WRITE, and WRITEAC are macros which are included
; in the training center's customized MASM.PS. They make calls to
; routines in $IO.OB, which must be linked with this program.

START:  INITIO                  ; PREPARES FOR INPUT AND OUTPUT
        ?GTMES  GTMPK           ; GET DIRECTORY NAME (FIRST ARG)
        DERR    1

GOPEN:  LLEFB   0,BUFFER*2      ; NAME OF DIR FROM GTMES PACKET
        WADC    1,1             ; -1, AOS/VS TO ASSIGN CHANNEL #
        ?GOPEN  GOPPK           ; OPEN DIRECTORY
        DERR    2
        LLEFB   0,BUFFER*2      ; NAME OF DIR FROM GTMES PACKET
        WSUB    1,1             ; (RESERVED)
        WSUB    2,2             ; (RESERVED)
        ????                    ; CHANGE WORKING DIRECTORY
        DERR    3

GTMES:  LNADI   1,GTMPK+?GNUM   ; INCREMENT ARGUMENT NUMBER
        WSUB    0,0             ; (RESERVED)
        LWSTA   0,??????????    ; RESET INTERNAL PNTR IN GNFN PKT
        WSUB    1,1             ; (RESERVED)
        ?GTMES  GTMPK           ; GET FILENAME TEMPLATE (NXT ARG)
        WBR     CKGTMES

GNFN:   WSUB    0,0             ; (RESERVED)
        LNLDA   1,GOPPK+?OPFL   ; LOAD CHANNEL NUMBER
        ?????   ?????           ; GET A FILENAME
        WBR     CKGNFN          ; CHECK ERROR

FSTAT:  LLEFB   0,???????       ; BPTR TO PATHNAME (GNFN PKT)
        WLDAI   1S1,??          ; IGNORE LINKS
        ??????  ????           ; GET FILE STATUS INFORMATION
        DERR    4
```

```
PRINT:  LLEF     0,DELIM          ; ADDRESS OF DELIMITER TABLE
        WLDAI    40.,1            ; MAX BYTE COUNT
        LLEFB    2,FNAME*2        ; DESTINATION BUFFER BPTR
        LLEFB    3,FNAME*2        ; SOURCE BUFFER BPTR
        WCMT                      ; SCAN FOR NULL BYTE
        WNEG     1,1              ; NEGATE BYTES REMAINING
        WLDAI    40.,0            ; DESTINATION # OF BYTES
        WADD     0,1              ; SOURCE # OF BYTES
        LLEFB    2,FNAME*2        ; DESTINATION BUFFER BPTR
        LLEFB    3,FNAME*2        ; SOURCE BUFFER BPTR
        WCMV                      ; PAD REMAINDER WITH SPACES

        WRITE    FNAME            ; OUTPUT FILENAME
        LWLDA    0,??????????    ; GET FILE LENGTH (FSTAT PKT)
        WRITEAC  0                ; OUTPUT FILE LENGTH
        WBR      GNFN             ; GET ANOTHER
```

```
RETURN: LLEFB   1,MSG*2          ; MESSAGE ADDRESS
        NLDAI   MSGLN,2          ; MESSAGE LENGTH
        ?RETURN                  ; TERMINATE NORMALLY.
        DERR    5

MSG:    .TXT    /Information processing completed./
MSGLN = .-MSG*2

;  ROUTINE TO CHECK ?GNFN ERROR

CKGNFN:
         WSNEI   EREOF,0         ; WAS IT END OF FILE?
         WBR     GTMES           ; YES. GET NEXT TEMPLATE.
         DERR    6               ; NO. TERMINATE WITH ERROR.

;  ROUTINE TO CHECK ?GTMES ERROR

CKGTMES:
         WSNEI   ERNAG,0         ; WAS IT NO SUCH ARGUMENT?
         WBR     RETURN          ; YES. WE ARE DONE.
         DERR    7               ; NO. TERMINATE WITH ERROR.

DELIM:  BITBL   16.,0            ; BIT ZERO IS THE ONLY BIT SET
                                 ; IN THIS 256. BIT TABLE
; BITBL is a macro defined in the training center's custom
MASM.PS
```

```
        .NREL   0                       ; UNSHARED

FNAME:  .BLK    40.                     ; BUFFER FOR FILENAMES
BUFFER: .BLK    40.                     ; BUFFER FOR TEMPLATES

;  PARAMETER PACKET FOR GTMES

GTMPK:  .LOC    GTMPK+?GREQ             ; REQUEST TYPE
        .WORD   ?GARG                   ; (GET ARGUMENT)
        .LOC    GTMPK+?GNUM             ; ARGUMENT IN THE COMMAND LINE
        .WORD   1                       ; (ZERO IS PROGRAM NAME)
        .LOC    GTMPK+?GSW              ; BPTR TO SWITCH TO TEST
        .DWORD  0                       ; (DO NOT TEST FOR SWITCHES)
        .LOC    GTMPK+?GRES             ; BPTR TO RESULT BUFFER
        .DWORD  BUFFER*2                ; (BUFFER BYTEPOINTER)
        .LOC    GTMPK+?GTLN             ; POSITION TO END OF PACKET

;  PARAMETER PACKET FOR GOPEN

GOPPK:  .LOC    GOPPK+?OPFL             ; OPEN FLAGS, RETURNS CH #
        .WORD   0                       ; (DO NOT SPECIFY ANY FLAGS)
        .LOC    GOPPK+?OPTY             ; RETURNS RECORD FMT & FILE TYPE
        .WORD   0                       ; (SET TO ZERO FOR OPEN)
        .LOC    GOPPK+?OPEH             ; RETURNS FILE SIZE IN BYTES
        .DWORD  0                       ; (SET TO ZERO FOR OPEN)
        .LOC    GOPPK+?OPFC             ; RETURNS FILE CONTROL PARAMETERS
        .WORD   0                       ; (SET TO ZERO FOR OPEN)
        .LOC    GOPPK+?OPLT             ; POSITION TO END OF PACKET

;  PARAMETER PACKET FOR GNFN

GNFPK:  .LOC    GNFPK+?NFKY             ; AOS/VS INTERNAL POINTER
        .WORD   0                       ; (ZERO FOR FIRST FILENAME)
        .LOC    GNFPK+?NFRS             ; (RESERVED)
        .WORD   0                       ;
        .LOC    GNFPK+?NFNM             ; BPTR TO BUFFER
        .DWORD  ???????                 ; (BPTR TO FILENAME BUFFER)
        .LOC    GNFPK+?NFTP             ; BPTR TO TEMPLATE
        .DWORD  ????????    ; (BPTR TO TEMPLATE BUFF FROM GTMES)
        .LOC    GNFPK+?NFLN             ; POSITION TO END OF PACKET
```

```
;  PARAMETER PACKET FOR ?FSTAT

FSPK:    .LOC    FSPK+?STYP      ; (RECORD FORMAT AND ENTRY TYPE)
         .WORD   0               ;
         .LOC    FSPK+?STIM      ; (NOT USED, RETURNS AS -1)
         .WORD   0               ;
         .LOC    FSPK+?SACP      ; (NOT USED, RETURNS AS -1)
         .WORD   0               ;
         .LOC    FSPK+?SCPS      ; (FILE CONTROL PARAMETERS)
         .WORD   0               ;
         .LOC    FSPK+?SLAU      ; (RESERVED)
         .WORD   0               ;
         .LOC    FSPK+?SMIL      ; (MAX NUMBER OF INDEX LEVELS)
         .WORD   0               ;
         .LOC    FSPK+?SDEH      ; (FILE ELEMENT SIZE)
         .DWORD  0               ;
         .LOC    FSPK+?STCH      ; (FILE CREATION DATE - SCALAR)
         .WORD   0               ;
         .LOC    FSPK+?STCL      ; (FILE CREATION TIME - SCALAR)
         .WORD   0               ;
         .LOC    FSPK+?STAH      ; (LAST ACCESS DATE - SCALAR)
         .WORD   0               ;
         .LOC    FSPK+?STAL      ; (LAST ACCESS TIME - SCALAR)
         .WORD   0               ;
         .LOC    FSPK+?STMH      ; (LAST MODIFICATION DATE
         .WORD   0               ;
         .LOC    FSPK+?STML      ; (LAST MODIFICATION TIME
         .WORD   0               ;
         .LOC    FSPK+?SSTS      ; (FILE CHARACTERISTICS)
         .WORD   0               ;
         .LOC    FSPK+?SEFW      ; (RESERVED)
         .WORD   0               ;
         .LOC    FSPK+?SEFM      ; (FILE SIZE IN BYTES)
         .DWORD  0               ;
         .LOC    FSPK+?SFAH      ; (STARTING LD ADDRESS)
         .DWORD  0               ;
         .LOC    FSPK+?SEFH      ; (RESERVED)
         .WORD   0               ;
         .LOC    FSPK+?SIDX      ; (CURRENT INDEX LEVEL)
         .WORD   0               ;
         .LOC    FSPK+?SCSH      ; (NOT USED)
         .DWORD  0               ;
         .LOC    FSPK+?SOPN      ; (OPEN COUNT)
         .WORD   0               ;
         .LOC    FSPK+?SLTH      ; (POSITION TO END OF PACKET)

END:     .END    START
```

ASSEMBLY LANGUAGE BLOCK I/O LAB

* MOVE/R/NACL COPIES OF THE FILES 'BLOCKIO.SR', 'ERROR.OB', AND 'BLKDATA' FROM THE :S309VS:ASSY DIRECTORY TO YOUR INITIAL WORKING DIRECTORY.

* TYPE THE BLKDATA FILE. IF YOU DON'T BELIEVE YOUR EYES QPRINT THE FILE. IS THERE STILL A PROBLEM READING THE FILE? USE THE DISPLAY UTILITY TO LOOK AT THE BLKDATA FILE.

* THE PROBLEM IS THAT CARRIAGE RETURNS RATHER THAN NEW LINE CHARACTERS WERE USED AS DELIMITERS WHEN THE FILE WAS CREATED. THE THRUST OF THIS EXERCISE IS TO WRITE A PROGRAM WHICH CONVERTS ALL CARRIAGE RETURNS TO NEW LINES.

* REFER TO THE PRINTOUT OF 'BLOCKIO.SR' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUE. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS, AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS.

```
X,MASM/L=BLOCKIO.LS,BLOCKIO
X,LINK/L=BLOCKIO.LS,BLOCKIO,ERROR
```

* GET A HARDCOPY OF THE FILE 'BLOCKIO.LS'.

* EXECUTE THE BLOCKIO PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'DELIMITER CONVERSION COMPLETED.'

* TYPE OUT THE OUTPUT FILE 'BLOCKOUT'. THE TEXT SHOULD NOW BE READABLE.

* DO A FILESTATUS WITH /ASSORTMENT ON BOTH THE BLKDATA AND BLOCKOUT FILES. THEY SHOULD HAVE IDENTICAL LENGTHS - EXACTLY ONE BLOCK.

* TO TEST THE PROGRAM WITH A DIFFERENT INPUT FILE:

  DELETE YOUR COPY OF 'BLKDATA'. COPY INTO A NEW 'BLKDATA' FILE IN YOUR OWN DIRECTORY FROM 'BLKDATA1' IN THE :S309VS:ASSY DIRECTORY. DO A FILESTATUS WITH ASSORTMENT ON THIS NEW 'BLKDATA' FILE IN YOUR OWN DIRECTORY. IT SHOULD BE LESS THAN ONE BLOCK LONG.

  CHANGING TO A DIFFERENT INPUT FILE WOULD BE EASIER IF A GENERIC FILENAME HAD BEEN USED IN THE PROGRAM ....HOWEVER THE SYSTEM DOES NOT PERMIT BLOCK IO TO GENERIC FILENAMES.

* EXECUTE THE PROGAM. NOTICE THAT YOUR OLD 'BLOCKOUT' FILE WILL BE DELETED BY THE PROGRAM AND A NEW ONE WILL BE CREATED. FIND THE SECTIONS OF CODE THAT ACCOMPLISH THIS. PROGRAMS THAT PERFORM BLOCK IO MUST HANDLE DETAILS SUCH AS THIS EXPLICITLY.

* YOU SHOULD AGAIN SEE THE COMPLETION MESSAGE. TYPE THE OUTPUT FILE 'BLOCKOUT'. IT SHOULD BE READABLE. FIND THE PARTS OF THE PROGRAM WHICH ENSURE THAT IT WILL WORK CORRECTLY IF THE LAST BLOCK IS NOT FULL. COMPARE THE LENGTHS OF YOUR NEW 'BLKDATA' AND 'BLOCKOUT'.... THEY SHOULD BE IDENTICAL.

```
        .TITL   BLOCKIO ; REPLACES CARRIAGE RETURNS WITH NEWLINES
        .ENT    START,CREATE,NXBLK,INCRB,NORM,RETN
        .ENT    DLERR,EOFCK,CHANG,LOOP,INCRE
        .ENT    COUNT,EOFLG,CREPK,PKTIN,PKTOU,RDPK,WRPK,BUFFER

CR      = 15                    ; SYMBOL FOR CARRIAGE RETURN
NL      = 12                    ; SYMBOL FOR NEWLINE

        .NREL   1

START:  LLEFB   0,IN*2          ; NAME OF INPUT FILE
        WADC    1,1             ; SET AC1 TO -1
        ??????  PKTIN           ; OPEN THE INPUT FILE
        DERR    1

        LLEFB   0,OUT*2         ; NAME OF OUTPUT FILE
        ?DELETE                 ; ATTEMPT TO DELETE THE FILE
        WBR     DLERR
CREATE: LLEFB   0,OUT*2         ; NAME OF OUTPUT FILE
        ?CREATE CREPK           ; THEN CREATE THE FILE
        DERR    2
        ??????  PKTOU           ; OPEN THE OUTPUT FILE
        DERR    3

;       READ NEXT BLOCK

NXBLK:  LNLDA   1,?????+?????   ; PUT CHANNEL NUMBER IN AC1
        ????    ????            ; READ A BLOCK
        WBR     EOFCK

INCRB:  XWISZ   ?????,2         ; INCR BLOCK NUMBER IN READ PKT
        NOP
        LNSTA   1,COUNT         ; NUMBER OF BYTES READ IS ALSO
        LNSTA   1,????+?????    ; NUMBER OF BYTES TO WRITE
        XJSR    CHANG           ; CHANGE CRS TO NLS

;       OUTPUT THE BLOCK

        LNLDA   1,?????+?????   ; LOAD CHANNEL NUMBER
        ????    ????            ; WRITE A BLOCK
        DERR    4
        XWISZ   ?????,2         ; INCR BLOCK NUMBER IN WRITE PKT
        NOP
        LNLDA   3,EOFLG         ; WAS THAT THE LAST BLOCK ?
        WSNEI   0,3
        WBR     NXBLK           ; NO. GET NEW BLOCK
```

```
NORM:   LNLDA   1,?????+?????    ; YES. CLOSE FILES
        ???????
        DERR    5
        LNLDA   1,?????+?????
        ???????
        DERR    6
        LLEFB   1,MSG*2
        NLDAI   MSGLN,2
RETN:   ?RETURN                  ; TERMINATE NORMALLY.
        DERR    7

MSG:    .TXT    /DELIMITER CONVERSION COMPLETED./
MSGLN = .-MSG*2                  ; LENGTH OF MESSAGE


;  ROUTINE TO CHECK ?DELETE ERROR

DLERR:                           ; GOT A ?DELETE ERROR
        WSNEI   ERFDE,0          ; WAS IT FILE DOES NOT EXIST ?
        WBR     CREATE           ; YES. IGNORE THE ERROR
        DERR    10               ; NO. TERMINATE WITH ERROR.


;  ROUTINE TO CHECK FOR END OF INPUT FILE

EOFCK:  WSEQI   EREOF,0  ; GOT A READ ERROR, WAS IT END-OF-FILE ?
        DERR    11               ; NO. TERMINATE WITH ERROR.

        WSNEI   0,1      ; YES, BUT WERE ANY CHARACTERS READ ?
        WBR     NORM             ; NO.
        LNISZ   EOFLG            ; YES. SET FLAG TO TERMINATE,
        WBR     INCRB            ; BUT FIRST PROCESS LAST BLOCK.


;  SUBROUTINE TO CHANGE CARRIAGE RETURNS TO NEWLINES

CHANG:  WSSVR   0
        LLEFB   2,BUFFER*2       ; GET BYTE POINTER.
                                 ; WE WILL BE LOOKING FOR CRS
        NLDAI   NL,3             ; AND REPLACING THEM WITH NLS.

LOOP:   WLDB    2,1              ; LOAD A BYTE INTO AC1
        WSNEI   CR,1             ; IS IT A CARRIAGE RETURN ?
        WSTB    2,3              ; YES. REPLACE IT WITH NEWLINE.

INCRE:  WINC    2,2              ; INCREMENT BYTE POINTER
        LNDSZ   COUNT            ; DECREMENT LOOP COUNT, DONE ?
        WBR     LOOP             ; NO. KEEP GOING.
        WRTN
```

```
        .NREL   0

COUNT:  .WORD   0
EOFLG:  .WORD   0

        ; PARAMETER PACKET FOR FILE CREATION

CREPK:  .LOC    CREPK+?CFTYP    ; ENTRY TYPE AND RECORD FORMAT
        .WORD   ?FTXT
        .LOC    CREPK+?CCPS     ; RECORD SIZE FOR FIXED
        .WORD   0               ; (IGNORED)
        .LOC    CREPK+?CTIM     ; ADDR OF TIME BLOCK
        .DWORD  -1              ; CURRENT TIME
        .LOC    CREPK+?CACP     ; ACL BYTE PTR
        .DWORD  -1              ; USE DEFACL
        .LOC    CREPK+?CDEH     ; (RESERVED)
        .WORD   0
        .LOC    CREPK+?CDEL     ; ELEMENT SIZE
        .WORD   -1              ; DEFAULT 1 BLK/ELEMENT
        .LOC    CREPK+?CMIL     ; MAX INDEX LEVEL
        .WORD   -1              ; USE SYSTEM DEFAULT
        .LOC    CREPK+?CMRS     ; (RESERVED)
        .WORD   0
        .LOC    CREPK+?CLTH     ; ?CLTH = PKT LENGTH


;  PACKETS FOR INFO RETURNED BY ?GOPEN

PKTIN:  .LOC    PKTIN+?OPCH     ; CHANNEL NUMBER
        .WORD   ??
        .LOC    PKTIN+?OPTY     ; FILE TYPE AND RECORD FORMAT
        .WORD   ??
        .LOC    PKTIN+?OPFC     ; RECORD LENGTH (IF FIXED FORMAT)
        .WORD   ??
        .LOC    PKTIN+?OPEH     ; FILE SIZE (BYTES)
        ??????  ??
        .LOC    PKTIN+?OPLT     ; PACKET LENGTH

IN:     .TXT    /BLKDATA/       ; INPUT FILENAME

PKTOU:  .LOC    PKTOU+?OPCH     ; CHANNEL NUMBER
        .WORD   ??
        .LOC    PKTOU+?OPTY     ; FILE TYPE AND RECORD FORMAT
        .WORD   ??
        .LOC    PKTOU+?OPFC     ; RECORD LENGTH (IF FIXED FORMAT)
        .WORD   ??
        .LOC    PKTOU+?OPEH     ; FILE SIZE (BYTES)
        ??????  ??
        .LOC    PKTOU+?OPLT     ; PACKET LENGTH

OUT:    .TXT    /BLOCKOUT/      ; OUTPUT FILENAME
```

```
;  TRANSFER PACKETS

RDPK:   .LOC    RDPK+?PSTI      ; NUMBER OF BLKS TO READ OR WRITE
        .WORD   ??              ; TRANSFER A SINGLE BLOCK
        .LOC    RDPK+?PSTO      ; RESERVED
        .WORD   ??
        .LOC    RDPK+?PCAD      ; *WORD* ADDRESS OF DATA BUFFER
        ??????  ??????
        .LOC    RDPK+?PRNH      ; BLOCK NUMBER
        ??????  ??
        .LOC    RDPK+?PRCL
        .WORD   ??
        .LOC    RDPK+?PRES      ; RESERVED
        .WORD   ??
        .LOC    RDPK+?PBLT      ; PACKET LENGTH

WRPK:   .LOC    WRPK+?PSTI      ; NUMBER OF BLKS TO READ OR WRITE
        .WORD   ??              ; TRANSFER A SINGLE BLOCK
        .LOC    WRPK+?PSTO      ; RESERVED
        .WORD   ??
        .LOC    WRPK+?PCAD      ; *WORD* ADDRESS OF DATA BUFFER
        ??????  ??????
        .LOC    WRPK+?PRNH      ; BLOCK NUMBER
        ??????  ??
        .LOC    WRPK+?PRCL
        .WORD   ??
        .LOC    WRPK+?PRES      ; RESERVED
        .WORD   ??
        .LOC    WRPK+?PBLT      ; PACKET LENGTH

BUFFER: .BLK    256.            ; TEXT BUFFER AREA

        .END    START
```

ASSEMBLY LANGUAGE SHARED DATA LAB

USING THE FOLLOWING CLI COMMANDS, MOVE THE NEEDED FILES INTO YOUR DIRECTORY:

```
DIRECTORY, :S309VS:ASSY
MOVE/R/NACL, :UDD:YOURUSERNAME, SHDATA.<ONE,TWO>.SR, ERROR.OB
DIRECTORY/I
```

REFER TO THE PRINTOUTS OF 'SHDATA.ONE.SR' AND 'SHDATA.TWO.SR' ON THE FOLLOWING PAGES.

DISCUSSION:

SHDATA.ONE AND SHDATA.TWO COMMUNICATE ACROSS A SINGLE PAGE OF MEMORY WHICH IS MAPPED TO BOTH PROCESSES CONCURRENTLY USING THE SHARED DATA FACILITY.

SHDATA.ONE FIRST CREATES A SHARED DATA FILE THEN READS IT ONCE TO MAP IN THE SHARED PAGE. A MESSAGE (FOR SHDATA.TWO) IS NOW MOVED TO A BUFFER LOCATED IN THE SHARED PAGE. AFTER SETTING A FLAG (ALSO LOCATED IN THE SHARED PAGE), SHDATA.ONE WAITS FOR SHDATA.TWO TO CLEAR THE FLAG. THEN SHDATA.ONE RETURNS CONTROL TO THE CLI WHICH DISPLAYS A RETURN MESSAGE.

SHDATA.TWO READS ONE PAGE FROM THE SHARED DATA FILE, WAITS FOR SHDATA.ONE TO SET THE FLAG, AND WRITES THE CONTENTS THE BUFFER IN THE SHARED PAGE TO @OUTPUT. IT NEXT CLEARS THE FLAG WORD SHDATA.ONE IS WAITING FOR AND PASSES CONTROL TO THE CLI.

STUDY THE SOURCES TO MAKE SURE YOU FOLLOW THE CODE THEN FILL IN THE ITEMS WHICH HAVE BEEN REPLACED WITH QUESTION MARKS. YOU WILL HAVE TO SUPPLY RELATED SYSTEM CALLS AND THEIR ARGUMENTS, ERROR CODE SYMBOLS, ADDRESSES OF VARIOUS ITEMS IN PACKETS, AND THE APPROPRIATE VALUES IN THOSE PACKETS. WHEN YOU HAVE DETERMINED THE PROPER ENTRIES FOR THE INDICATED ITEMS EDIT THE TWO SOURCE FILES AND MAKE THE CORRECTIONS. ASSEMBLE SHDATA.ONE AND SHDATA.TWO, THEN LINK EACH WITH THE ERROR MODULE.

PREDICT WHAT WILL OCCUR WHEN THE PROGRAMS ARE RUN. EXECUTE SHDATA.ONE AT ONE CONSOLE THEN LOG ON AT A SECOND CONSOLE AND EXECUTE SHDATA.TWO.

USE THE DISPLAY UTILITY TO EXAMINE THE SHARED DATA FILE:

1. YOU SHOULD FIND DATA IN THE FILE. WHY?
2. AT WHAT BITE LOCATION IN THE FILE IS THE FLAG WORD? WHAT IS ITS VALUE?

```
        .TITL   SDAT1
        .ENT START,CRSHF,CKDLER,OPNSF,RDSP,MOVTX,SETFLG,WAIT,EXIT
        .ENT    CHNUM,CREPK,SPPK
        .EXTL   ERRXT

        DELAY=1000.             ; DELAY TIME IN MS
        BUFSZ=100.              ; SHARED BUFFER SIZE

        .NREL   1

START:  LLEFB   0,SDF*2         ; GET BP TO SHARED DATA FILE
        ?DELETE                 ; ATTEMPT TO DELETE IT
        WBR     CKDLER
CRSHF:  LLEFB   0,SDF*2         ; GET BP TO SHARED DATA FILE
        ?CREATE CREPK           ; CREATE A FILE FOR SHARED DATA
        DERR    1

        ; OPEN SHARED PAGE FILE AND READ ONE PAGE

OPNSF:  WADC    1,1             ; LET SYSTEM ASSIGN CHANNEL #
        WADC    2,2             ; WILL PERFORM READ AND WRITE
        ??????                  ; OPEN FOR SHARED ACCESS
        DERR    2
        LWSTA   1,CHNUM         ; KEEP CHANNEL NUMBER
RDSP:   ??????  ????            ; READ ONCE TO GET PAGE IN MEMORY
        DERR    3

        ; MOVE TEXT TO SHARED PAGE BUFFER

MOVTX:  NLDAI   MSGLN,1         ; # OF WORDS TO MOVE = STR LNGTH
        LLEF    2,TSTMSG        ; SOURCE ADDRESS
        LLEF    3,SBUFF         ; DEST ADDRESS
        WBLM

SETFLG: WADC    0,0             ; SET FLAG WORD
        LWSTA   0,FLAG          ; FLAG IS FIRST WORD AFTER BUFFER

        ; WAIT UNTIL SHDATA.TWO RESETS FLAG WORD

WAIT:   LWLDA   0,FLAG          ; GET FLAG
        WSNEI   0.,0            ; IS FLAG RESET ?
        WBR     EXIT            ; YES, EXIT PROGRAM

        NLDAI   DELAY,0         ; NO, GET ?DELAY PERAMETER
        ?WDELAY                 ; WAIT DELAY MS
        DERR    4
        WBR     WAIT            ; WAIT UTIL FLAG IS RESET
```

```
        ; RELEASE SP,CLEAN UP, RETURN TO CLI

EXIT:   LWLDA   1,CHNUM         ; GET CHANNEL NUMBER
        WIORI   1S0,1           ; SET RELEASE BIT
        ???????                 ; CLOSE SHARED DATA FILE
        DERR    5

        LLEFB   1,EXTMSG*2      ; BP TO EXIT MSG
        NLDAI   EXMLN,2         ; MSG LENGTH
        ?RETURN
        DERR    6

CKDLER: WSEQI   ERFDE,0         ; WAS ERROR FILE DOES NOT EXIST ?
        DERR    7
        WBR     CRSHF           ; YES, IGNORE IT.


**      .NOLOC  1
SDF:    .TXT    /SDFILE/

TSTMSG: .TXT    /THIS IS A TEST MESSAGE/
**      .NOLOC  0

MSGLN = (.-TSTMSG)

**      .NOLOC  1
EXTMSG: .TXT    /SHDATA.TWO HAS RECEIVED THE TEST MESSAGE/
**      .NOLOC  0

EXMLN = (.-EXTMSG)*2

        .NREL   ??              ; SHARED DATA PARTITION

SBUFF:  .BLK    1024.           ; ONE PAGE
        .LOC    SBUFF+BUFSZ
FLAG:   .DWORD  0
```

```
        .NREL   0                       ; UNSHARED DATA

CHNUM:  0


        ; PARAMETER PACKET FOR FILE CREATION

CREPK:  .LOC    CREPK+?CFTYP            ; ENTRY TYPE AND RECORD FORMAT
        .WORD   ?FTXT                   ; TEXT
        .LOC    CREPK+?CCPS             ; RECORD SIZE FOR FIXED
        .WORD   0                       ; (IGNORED)
        .LOC    CREPK+?CTIM             ; ADDR OF TIME BLOCK
        .DWORD  -1                      ; CURRENT TIME
        .LOC    CREPK+?CACP             ; ACL BYTE PTR
        .DWORD  -1                      ; USE DEFACL
        .LOC    CREPK+?CDEH             ; (RESERVED)
        .WORD   0
        .LOC    CREPK+?CDEL             ; ELEMENT SIZE
        .WORD   ??                      ; MUST BE MULT OF 4 FOR SHARED IO
        .LOC    CREPK+?CMIL             ; MAX INDEX LEVEL
        .WORD   -1                      ; USE SYSTEM DEFAULT
        .LOC    CREPK+?CMRS             ; (RESERVED)
        .WORD   0
        .LOC    CREPK+?CLTH             ; ?CLTH = PKT LENGTH

        ; ?SPAGE PACKET

SPPK:   .LOC    SPPK+?PSTI              ; NUMBER OF BLKS TO READ OR WRITE
        .WORD   ??                      ; MUST BE MULT OF 4
        .LOC    SPPK+?PSTO              ; RESERVED
        .WORD   ??
        .LOC    SPPK+?PCAD              ; *WORD* ADDRESS OF DATA BUFFER
        .DWORD  ?????
        .LOC    SPPK+?PRNH              ; BLOCK NUMBER
        .DWORD  ??
        .LOC    SPPK+?PRCL
        .WORD   0
        .LOC    SPPK+?PRES              ; RESERVED
        .WORD   0
        .LOC    SPPK+?PBLT              ; PACKET LENGTH

        .END START
```

```
        .TITL   SDAT2
        .ENT START,OPNOK,CKOPER,CKFLG,CLRFLG
        .ENT WRIT,EXIT,SPPK,OUPKT,SBUFF
        .EXTL   ERRXT

        DELAY=1000.             ; DELAY TIME IN MS
        BUFSZ=100.              ; SHARED BUFFER SIZE

        .NREL   1

        ; OPEN FILE FOR SHARED DATA AND READ ONE PAGE

START:  LLEFB   0,SDF*2         ; BP TO SHARED DATA FILE
        WADC    1,1             ; LET SYSTEM ASSIGN CHANNEL #
        WADC    2,2             ; WILL PERFORM READ AND WRITE
        ??????                  ; OPEN FOR SHARED ACCESS
        WBR     CKOPER          ; CHECK OPEN ERROR

OPNOK:  ??????  ????            ; READ ONCE TO GET PAGE IN MEMORY
        DERR    1

        ?OPEN   OUPKT           ; OPEN @OUTPUT
        DERR    2

CKFLG:  LWLDA   0,FLAG          ; WAIT FOR SHDATA.ONE TO SET FLAG
        WSEQI   0.,0
        WBR     WRIT
        NLDAI   DELAY,0
        ?WDELAY
        DERR    3
        WBR     CKFLG

WRIT:   ?WRITE  OUPKT       ; WRITE TEXT FROM SHARED PG TO SCREEN
        DERR    4

CLRFLG: WSUB    0,0             ; RESET FLAG IN SHARED PAGE
        LWSTA   0,FLAG          ; FLAG IS FIRST WORD AFTER BUFFER

EXIT:   WSUB    2,2
        ?RETURN
        DERR    5

CKOPER: WSEQI   ERFDE,0         ; WAS IT FILE DOES NOT EXIST?
        DERR    6
        NLDAI   DELAY,0         ; YES. WAIT AND TRY AGAIN.
        ?WDELAY
        DERR    7
        WBR START
```

```
**      .NOLOC 1
SDF:    .TXT /SDFILE/
**      .NOLOC 0

        .NREL   ??              ; SHARED DATA PRATITION

SBUFF:  .BLK    1024.           ; ONE PAGE
        .LOC    SBUFF+BUFSZ
FLAG:   .DWORD  0

        .NREL   0

        ; ?SPAGE PACKET

SPPK:   .LOC    SPPK+?PSTI      ; NUMBER OF BLKS TO READ OR WRITE
        .WORD   ??              ; MUST BE MULT OF 4
        .LOC    SPPK+?PSTO      ; RESERVED
        .WORD   ??
        .LOC    SPPK+?PCAD      ; *WORD* ADDRESS OF DATA BUFFER
        .DWORD  ?????
        .LOC    SPPK+?PRNH      ; BLOCK NUMBER
        .DWORD  ??
        .LOC    SPPK+?PRCL
        .WORD   0
        .LOC    SPPK+?PRES      ; RESERVED
        .WORD   0
        .LOC    SPPK+?PBLT      ; PACKET LENGTH
```

```
        ; PARAMETER PACKET FOR FILE @OUTPUT

OUPKT:  .LOC    OUPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?ISTI     ; FILE SPECS
        .WORD   ?ICRF+?RTDS+?OFCE+?OFCR+?OFOT
                                ; CHG TO D-S, RECREATE FOR OUTPUT
        .LOC    OUPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IMRS     ; PHYSICAL BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    OUPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  SBUFF*2
        .LOC    OUPKT+?IRES     ; RESERVED
        .WORD   0
        .LOC    OUPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   BUFSZ
        .LOC    OUPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IRNW     ; RESERVED
        .WORD   0
        .LOC    OUPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    OUPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATOUTPUT*2
        .LOC    OUPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    OUPKT+?IOSZ     ; ?IOSZ=PACKET LENGTH

**      .NOLOC  1
ATOUTPUT:       .TXT    /@OUTPUT/
**      .NOLOC  0

        .END    START
```

S309VS RECORD IO LAB

PART ONE: DATASENSITIVE AND VARIABLE

ISSUE THE FOLLOWING COMMANDS TO MOVE NEEDED FILES TO YOUR OWN DIRECTORY:

```
DIRECTORY, :S309VS:ASSY
MOVE/R/NACL, :UDD:YOURUSERNAME, RECORDIO.SR,&
             ERROR.OB, DATASENS.TXT, VIEW.PR
DIRECTORY/I
```

REFER TO THE PRINTOUT OF 'RECORDIO.SR' ON THE FOLLOWING PAGES. THIS PROGRAM COPIES THE CONTENTS OF THE GENERIC FILE "@DATA" INTO THE FILE "RECORDIO.OUT" ONE RECORD AT A TIME.

IF AN ERROR OCCURS WHEN READING A RECORD, IT CHECKS TO SEE IF THE ERROR WAS "END OF FILE". IF SO, THE PROGRAM TERMINATES NORMALLY.

ON ALL OTHER ERRORS, THE PROGRAM WILL TERMINATE THRU THE SEPARATELY ASSEMBLED ERROR EXIT ROUTINE.

AT THE LABEL "ERRCK:" THE INTENT OF THE INSTRUCTION IS TO TEST AC0 FOR THE VALUE OF THE END-OF-FILE ERROR CODE. USING APPENDIX A OF THE PROGRAMMER'S REFERENCE MANUAL, FIND THE SYMBOL FOR THIS ERROR CODE. EDIT YOUR COPY OF THE SOURCE FILE AND FILL IT IN.

THE REMAINING PAGES OF THE SOURCE FILE ARE DEVOTED TO THE PARAMETER PACKETS FOR THE RECORD I/O SYSTEM CALLS.

WHAT IS THE SYMBOLIC ADDRESS OF THE PACKET FOR THE FILE @DATA ? _____

WHAT IS THE SYMBOLIC ADDRESS OF THE PACKET FOR THE FILE RECORDIO.OUT? _____

FILL IN THE PACKET ADDRESSES IN THE ?OPEN, ?READ, AND ?WRITE SYSTEM CALLS.

FILL IN THE FOLLOWING INFORMATION IN THE PACKETS:

I/O PACKET FOR FILE @DATA

OFFSET	OPTIONS OR VALUE
?ISTI	CHANGE RECORD FORMAT TO DATASENSITVE, INPUT ONLY
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	MAX RECORD LENGTH SAME AS NUMBER OF BYTES IN "BUFF"
?IFNP	BYTE POINTER TO FILENAME "@DATA"
?IDEL	DEFAULT DELIMITER TABLE

I/O PACKET FOR FILE RECORDIO.OUT

OFFSET	OPTIONS OR VALUE
?ISTI	CHANGE RECORD FORMAT TO DATASENSITVE, OUTPUT ONLY, DELETE OLD FILE IF IT EXISTS, CREATE NEW FILE
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	MAX RECORD LENGTH SAME AS NUMBER OF BYTES IN "BUFF"
?IFNP	BYTE POINTER TO FILENAME "RECORDIO.OUT"
?IDEL	DEFAULT DELIMITER TABLE

ASSEMBLE THE SOURCE FILE, OBTAINING A PRINTED LISTING. LINK THE MAIN PROGRAM AND ERROR ROUTINE TOGETHER.

EXECUTE THE PROGRAM. WHAT ERROR MESSAGE OCCURRED ?

USE THE INFORMATION RETURNED TO YOUR SCREEN BY THE ERROR EXIT ROUTINE TO DETERMINE WHERE IN YOUR PROGRAM THE ERROR OCCURRED.

BEAR IN MIND THAT THE LINKER DEFINED SYMBOL "?SBOT" -- BOTTOM OF THE SHARED CODE REGION -- HAS THE SAME ADDRESS AS THE LABEL "START" IN THIS PROGRAM AND THAT THERE WILL BE A FEW WORDS BETWEEN THE BEGINNING OF A SYSTEM CALL AND THE DERR INSTRUCTION WHICH CALLS THE ERROR EXIT ROUTINE. NOTE THAT THE DERR CODE REPORTED BY THE ERROR EXIT ROUTINE SHOULD MATCH THE DERR CODE IN THE SOURCE PROGRAM, IF YOU HAVE CORRECTLY IDENTIFIED THE LOCATION WHERE THE ERROR OCCURED. WHAT FILE IS CAUSING THE PROBLEM ?

USE A CLI COMMAND TO SET "DATASENS.TXT" AS THE GENERIC FILE YOUR PROGRAM WILL TRY TO READ. WHAT IS THE COMMAND ?

EXECUTE THE PROGRAM AGAIN. YOU SHOULD GET THE MESSAGE "IO COMPLETED"

COMPARE THE CONTENTS OF YOUR RECORDIO.OUT FILE AND DATASENS.TXT. THEY SHOULD BE IDENTICAL.

NOW MODIFY THE PROGRAM TO OUTPUT VARIABLE LENGTH RECORDS INSTEAD OF DATASENSITIVE:

NOTE THAT ONLY THE OUTPUT IS TO BE CHANGED - THE INPUT IS STILL TO BE DATASENSITIVE.

IT WILL BE NECESSARY TO SPECIFY THE LENGTH OF EACH RECORD AS IT IS WRITTEN OUT.

AFTER EACH RECORD HAS BEEN READ IN, WHERE IS ITS LENGTH STORED ?

_____

IN ORDER TO OUTPUT THE RECORD IN VARIABLE FORMAT, WHAT OFFSET IN THE OUTPUT PACKET MUST CONTAIN THIS INFORMATION ? _____

ADD SOME CODE AFTER THE ?READ AND BEFORE THE ?WRITE CALL TO GET THE ACTUAL RECORD LENGTH READ IN AND STORE IT IN THE OUTPUT PACKET:

TWO INSTRUCTIONS SHOULD BE ENOUGH

_____

_____

CHANGE THE RECORD FORMAT SPECIFICATION IN THE OUTPUT PACKET TO VARIABLE.

ASSEMBLE YOUR PROGRAM. LINK AS BEFORE.

EXECUTE THE PROGRAM.

USING THE CLI COMMAND "X VIEW RECORDIO.OUT", YOU CAN SEE THE RECORDS IN YOUR PROGRAM'S OUTPUT FILE. THESE VARIABLE LENGTH RECORDS EACH START WITH FOUR DIGITS THAT REVEAL THE RECORD LENGTH.

SINCE THE "VIEW" PROGRAM OPENS THE SPECIFIED FILE AS THOUGH IT HAD DATASENSITIVE RECORDS, THESE RECORD LENGTHS WILL BE DISPLAYED AS PART OF EACH LINE ON YOUR SCREEN. NOTICE THAT THE LENGTH OF EACH RECORD INCLUDES THE FOUR DIGITS.

```
        .TITLE   IO       ; RECORD I/O
        .ENT    START LOOP ERRCK EOF_ERR DATPK OUTPK BUFF

        BUFSZ=136.        ; BUFFER SIZE = 136. BYTES

        .NREL   1
START:  ?OPEN   ?????     ; OPEN @DATA FOR INPUT
        DERR    1

        ?OPEN   ?????     ; OPEN RECORDIO.OUT FOR OUTPUT
        DERR    2

LOOP:   ?READ   ?????     ; READ A RECORD
        WBR     ERRCK

        ?WRITE  ?????     ; WRITE A RECORD
        DERR    3

        WBR     LOOP      ; DO IT AGAIN.

ERRCK:  WSEQI   ?????,0 ; GOT AN ERROR, WAS IT END-OF-FILE ?
        DERR    4         ; NO.

EOF_ERR: NLDAI  ?RFCF+MESLEN,2 ; YES. GOOD EXIT.
        XLEFB   1,MSG*2 ; WITH MESSAGE
        ?RETURN
        DERR    5

MSG:    .TXT    /IO COMPLETED/
MESLEN  =       (.-MSG)*2
```

```
        .NREL   0

        ; I/O PACKET FOR FILE @DATA

DATPK:  .LOC    DATPK+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    DATPK+?ISTI     ; FILE SPECS
        .WORD   ?????!?????!?????      ; CHG TO DATASEN, INPUT
ONLY
        .LOC    DATPK+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    DATPK+?IMRS     ; PHYSICAL BLOCK SIZE.
        .WORD   -1              ; DEFAULT
        .LOC    DATPK+?IBAD     ; BUFFER BYTE POINTER
        ??????  ??????
        .LOC    DATPK+?IRES     ; RESERVED
        .WORD   0
        .LOC    DATPK+?IRCL     ; MAX RECORD LENGTH
        .WORD   ?????
        .LOC    DATPK+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    DATPK+?IRNW     ; RESERVED
        .WORD   0
        .LOC    DATPK+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    DATPK+?IFNP     ; FILENAME BYTE POINTER
        ??????  ????????
        .LOC    DATPK+?IDEL     ; DELIMITER TABLE
        ??????  ??              ; DEFAULT
        .LOC    DATPK+?IOSZ     ; ?IOSZ=PACKET LENGTH

BUFF:   .BLK    BUFSZ/2         ; BUFFER BUFSZ BYTES

ATDATA: .TXT    /@DATA/
```

```
        ; I/O PACKET FOR FILE RECORDIO.OUT

OUTPK:  .LOC    OUTPK+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUTPK+?ISTI     ; FILE SPECS
        .WORD   ?????!?????!?????!?????!?????
                                ; CHG TO D-S, RECREATE FOR OUTPUT
        .LOC    OUTPK+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUTPK+?IMRS     ; PHYSICAL BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    OUTPK+?IBAD     ; BUFFER BYTE POINTER
        ??????  ??????
        .LOC    OUTPK+?IRES     ; RESERVED
        .WORD   0
        .LOC    OUTPK+?IRCL     ; MAX RECORD LENGTH
        .WORD   ?????
        .LOC    OUTPK+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUTPK+?IRNW     ; RESERVED
        .WORD   0
        .LOC    OUTPK+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    OUTPK+?IFNP     ; FILENAME BYTE POINTER
        ??????  ????????
        .LOC    OUTPK+?IDEL     ; DELIMITER TABLE
        ??????  ??              ; DEFAULT
        .LOC    OUTPK+?IOSZ     ; ?IOSZ=PACKET LENGTH

OUFILE: .TXT    /RECORDIO.OUT/

        .END    START
```

S309VS RECORD IO LAB

PART TWO: DYNAMIC AND FIXED

ISSUE THE FOLLOWING COMMANDS TO MOVE NEEDED FILES TO YOUR OWN DIRECTORY:

```
DIRECTORY, :S309VS:ASSY
MOVE/R/NACL, :UDD:YOURUSERNAME, DYNIO.SR, SYSLOG
DIRECTORY/I
```

REFER TO THE PRINTOUT OF "DYNIO.SR" ON THE FOLLOWING PAGES. THIS PROGRAM READS A SYSTEM LOG FILE, OUTPUTTING ONLY RECORDS OF A CERTAIN TYPE.

THE FORMAT OF THE SYSTEM LOG FILE MUST BE STUDIED BEFORE ONE CAN WRITE A PROGRAM TO READ IT. DYNAMIC RECORD FORMAT MUST BE USED.

EACH RECORD IN THE FILE STARTS WITH A 16. BYTE RECORD HEADER. THE FIRST TWO **WORDS** OF EACH HEADER ARE A 32 BIT NUMBER (NOT FOUR DIGITS AS IN VARIABLE RECORD FORMAT) REVEALING THE LENGTH IN **WORDS** OF THAT RECORD.

WORD FIVE OF THE HEADER CONTAINS A CODE WHICH TELLS THE ENTRY TYPE OF THE RECORD. IN THIS PROGRAM WE WILL OUTPUT ONLY TYPE 3 RECORDS AND SKIP OVER ALL OTHERS.

IN THESE TYPE 3 RECORDS, THE 32 BYTES FOLLOWING THE HEADER CONTAIN A PROCESS NAME (PADDED WITH NULLS), WHICH WE WILL OUTPUT. THERE ARE ALSO 8 ADDITIONAL WORDS OF INFORMATION (16. BYTES) WHICH THE PROGRAM WILL NOT OUTPUT, BUT MUST SKIP OVER TO GET TO THE NEXT RECORD.

EDIT YOUR COPY OF THE SOURCE FILE "DYNIO.SR". FILL IN THE PACKET ADDRESSES IN THE I/O SYSTEM CALLS. FILL IN ALL THE VALUES IN THE PACKETS, USING THE COMMENTS, THE SYSTEM PROGRAMMER'S REFERENCE MANUAL, AND YOUR EXPERIENCE FROM PART ONE OF THIS LAB AS A GUIDE.

ASSEMBLE YOUR PROGRAM AND LINK IT WITH THE ERROR ROUTINE.

EXECUTE THE PROGRAM. YOU SHOULD SEE A BUNCH OF PROCESS NAMES ON YOUR SCREEN.

EACH PROCESS NAME WAS OUTPUT AS A FIXED LENGTH 33. BYTE RECORD. WHY DO YOU SUPPOSE THEY DON'T APPEAR TO BE THE SAME LENGTH ?

TO SEE EXACTLY WHAT IS OUTPUT, IT IS NECESSARY TO HAVE THE OUTPUT GO TO A DISK FILE INSTEAD OF A CONSOLE SCREEN.

CREATE AN EMPTY FILE CALLED DYNIO.OUT .

RUN YOUR PROGRAM AGAIN USING THE CLI COMMAND:

```
) PROC/BLOCK/DEF/OUT=DYNIO.OUT, DYNIO
```

NOW WE CAN EXAMINE THE CONTENTS OF THE OUTPUT FILE USING THE DISPLAY UTILITY:

```
) X, DISPLAY, DYNIO.OUT
```

THE FIRST COLUMN CONTAINS THE WORD ADDRESS OF THE FIRST WORD IN EACH LINE. EACH LINE CONTAINS 8. WORDS (2 CHARACTERS PER WORD) SHOWN IN OCTAL.

THE RIGHT COLUMNS SHOW THE TEXT CHARACTERS WITH NON-PRINTING CHARACTERS REPLACED BY PERIODS. WAS EACH RECORD 33. BYTES LONG AS EXPECTED ? WHAT NON-PRINTING CHARACTERS ARE CONTAINED IN EACH RECORD ? WHERE DID THEY COME FROM ?

****
CHALLENGE 1 -
****

WRITE A PROGRAM TO READ A FILE OF FIXED RECORDS LAST-LINE-FIRST AND WRITE THE RECORDS TO ANOTHER FILE IN REVERSE OF THEIR ORIGINAL ORDER.

THE FILE 'BOTTOM_UP' WHICH CONTAINS FIXED RECORDS OF 80. CHARACTERS, WRITTEN FIRST-LINE-LAST, IS PROVIDED IN THE :S309VS:ASSY DIRECTORY FOR TESTING YOUR PROGRAM.

****
CHALLENGE 2 -
****

WRITE A PROGRAM TO READ A FILE OF FIXED LENGTH 80 BYTE RECORDS, STRIPPING TRAILING BLANKS FROM EACH RECORD BY REPLACING THE FIRST BLANK AFTER THE LAST SIGNIFICANT CHARACTER WITH A NEW-LINE, AND WRITE THE RECORDS TO A DATA SENSITIVE FILE.

THE FILE 'PADDED.TXT' WHICH CONTAINS FIXED RECORDS OF 80. CHARACTERS, PADDED WITH TRAILING SPACES, IS PROVIDED IN THE :S309VS:ASSY DIRECTORY FOR TESTING YOUR PROGRAM.

```
        .TITLE  DYNIO    ; DYNAMIC RECORD I/O
        .ENT    START,NEXT,RDHDR,SKPRS,CIFC3
        .ENT    RDPNM,DORDP,WRPNM,ERRCK
        .ENT    LOGPK,BUFF,OUTPT
        .NREL   1

START:  ?OPEN   ?????    ; OPEN LOGFILE FOR INPUT
        DERR    1

        ?OPEN   ?????    ; OPEN @OUTPUT FOR OUTPUT
        DERR    2

        WSUB    1,1      ; SELECT FIRST RECORD

NEXT:   LWSTA   1,LOGPK+?IRNH    ; ADVANCE TO NEXT RECORD
        NLDAI   16.,0            ; READ 16. BYTE HEADER
        LNSTA   0,LOGPK+?IRCL

RDHDR:  ?READ   ?????            ; READ A RECORD HEADER
        WBR     ERRCK

        ; GET READY TO SKIP OVER THE REST OF THE RECORD

SKPRS:  LWLDA   1,BUFF           ; GET RECORD LENGTH
        NLDAI   8.,0             ; LENGTH OF HEADER
        WSUB    0,1              ; (SIZE-HEADER)
        WASHI   1,1              ;  *2

        ; SEE IF THIS IS A CODE 3 RECORD

CIFC3:  LNLDA   2,BUFF+5         ; GET RECORD CODE
        WSEQI   3,2              ; IS IT 3 ?
        WBR     NEXT             ; NO.

        ; YES. READ A 32. BYTE PROCNAME RECORD

RDPNM:  NLDAI   32.,0            ; 32. BYTES
        LNSTA   0,LOGPK+?IRCL
        WSUB    0,0              ; START AT NEXT BYTE
        LWSTA   0,LOGPK+?IRNH

DORDP:  ?READ   ?????            ; READ PROCNAME
        WBR     ERRCK

WRPNM:  ?WRITE  ?????            ; WRITE A RECORD
        DERR    3

        NLDAI   16.,1    ; SKIP NEXT 16. BYTES OF THIS RECORD
        WBR     NEXT
```

```
ERRCK:  WLDAI   EREOF,1 ; GOT AN ERROR
        WSEQ    0,1     ; WAS IT EOF ?
        DERR    4       ; NO. EXIT WITH ERROR.
        NLDAI   ?RFCF+MESLEN,2  ; YES. GOOD EXIT.
        LLEFB   1,MSG*2 ; WITH MESSAGE
        ?RETURN
        DERR    5


MSG:    .TXT    /IO COMPLETED/
MESLEN  =       (.-MSG)*2
```

```
        .NREL   0

        ; I/O PACKET FOR LOGFILE

LOGPK:  .LOC    LOGPK+?ICH          ; CHANNEL NUMBER
        .WORD   ??                  ; RETURNED BY SYS
        .LOC    LOGPK+?ISTI         ; FILE SPECS
        .WORD   ?????+?????+?????   ; CHANGE TO DYNAMIC, INPUT ONLY
        .LOC    LOGPK+?ISTO         ; FILE TYPE
        .WORD   ??                  ; RETURNED BY SYS
        .LOC    LOGPK+?IMRS         ; PHYS BLOCK SIZE
        .WORD   ??                  ; DEFAULT
        .LOC    LOGPK+?IBAD         ; BUFFER BYTE POINTER
        ??????  ??????
        .LOC    LOGPK+?IRES         ; RESERVED
        .WORD   ??
        .LOC    LOGPK+?IRCL         ; RECORD LENGTH
        .WORD   ??                  ; SET AT EACH ?READ
        .LOC    LOGPK+?IRLR         ; RECORD LENGTH
        .WORD   ??                  ; RETURNED BY SYS
        .LOC    LOGPK+?IRNW         ; RESERVED
        .WORD   0
        .LOC    LOGPK+?IRNH         ; RECORD NUMBER
        ??????  ??                  ; SET AT EACH ?READ
        .LOC    LOGPK+?IFNP         ; FILENAME BYTE POINTER
        ??????  ????????
        .LOC    LOGPK+?IDEL         ; DELIMITER TABLE
        ??????  ??                  ; DEFAULT
        .LOC    LOGPK+?IOSZ         ; ?IOSZ=PACKET LENGTH

BUFF:   .BLK    32./2               ; BUFFER 32. BYTES
        .TXT    /<12>/

SYSLOG: .TXT    /SYSLOG/
```

```
        ; I/O PACKET FOR OUTPUT FILE

OUTPT:  .LOC    OUTPT+?ICH      ; CHANNEL NUMBER
        .WORD   ??              ; RETURNED BY SYS
        .LOC    OUTPT+?ISTI     ; FILE SPECS
        .WORD   ?????+?????+?????+?????+?????
                                ; CHG TO FIXED,RECREATE,OUTPUT
        .LOC    OUTPT+?ISTO     ; FILE TYPE
        .WORD   ??              ; RETURNED BY SYS
        .LOC    OUTPT+?IMRS     ; PHYS BLOCK SIZE
        .WORD   ??              ; DEFAULT
        .LOC    OUTPT+?IBAD     ; BUFFER BYTE POINTER
        ??????  ??????
        .LOC    OUTPT+?IRES     ; RESERVED
        .WORD   ??
        .LOC    OUTPT+?IRCL     ; RECORD LENGTH
        .WORD   ???             ; 33. BYTE FIXED RECORDS
        .LOC    OUTPT+?IRLR     ; RECORD LENGTH
        .WORD   ??              ; RETURNED BY SYS
        .LOC    OUTPT+?IRNW     ; RESERVED
        .WORD   0
        .LOC    OUTPT+?IRNH     ; RECORD NUMBER
        ??????  ??
        .LOC    OUTPT+?IFNP     ; FILENAME BYTE POINTER
        ??????  ????????
        .LOC    OUTPT+?IDEL     ; DELIMITER TABLE
        ??????  ??              ; DEFAULT
        .LOC    OUTPT+?IOSZ     ; ?IOSZ=PACKET LENGTH

ATOUTP: .TXT    /@OUTPUT/

        .END    START
```

ASSEMBLY LANGUAGE SCREEN MANAGEMENT LAB

* MOVE/R/NACL COPIES OF THE FILES 'SCREEN.SR' AND 'ERROR.OB' FROM THE :S309VS:ASSY DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'SCREEN' IS TO PRESENT A DATA ENTRY FORM ON THE SCREEN AND ALLOW THE USER TO ENTER THE REQUIRED DATA ONE FIELD AT A TIME. WHEN THE USER COMPLETES ALL THE FIELDS, HE IS ASKED IF HE WANTS TO CHANGE ANY OF THEM. IF SO, HE IS ASKED FOR THE FIELD NUMBER AND ALLOWED TO EDIT THE FIELD. WHEN THE USER HAS NO MORE CHANGES, HE IS ASKED IF HE WISHES TO FILL IN ANOTHER DATA ENTRY SCREEN.

* THE FUNCTION OF 'ERROR' IS TO PROVIDE AN ERROR HANDLER FOR THE SYSTEM CALLS. IT WILL REPORT THE ERROR MESSAGE AND PROGRAM LOCATION WHEN A SYSTEM CALL FAILS TO EXECUTE PROPERLY.

* REFER TO THE PRINTOUT OF 'SCREEN.SR' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM CALLS AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
DELETE/2=IGNORE,SCREEN.LS
X,MASM/L=CREATE.LS,CREATE
X,LINK/STACK=200/L=SCREEN.LS,SCREEN,ERROR
```

* GET A HARDCOPY OF THE FILE 'SCREEN.LS'.

* EXECUTE THE SCREEN PROGRAM. YOU SHOULD SEE A BLANK DATA ENTRY FORM ON YOUR SCREEN.

* USING IMAGINARY NAMES AND ADDRESSES, EXPERIMENT WITH FILLING IN THE FIELDS. NOTICE THAT THE FIELDS ARE AUTO-TERMINATING.

* STUDY THE PROGRAM TO UNDERSTAND HOW IT MAKES USE OF THE SCREEN MANAGEMENT EXTENSION AND AUTO-TERMINATING READS.

* TRY ENTERING INVALID RESPONSES TO THE Y/N QUESTIONS AND/OR THE FIELD NUMBER QUESTION. NOTICE HOW THE PROGRAM ACCOMPLISHES THE SAVING OF THE INPUT CURSOR POSITION WHILE IT DISPLAYS THE ERROR MESSAGES. WHICH SCREEN MANAGEMENT FLAG IS REQUIRED FOR THIS PROGRAM FEATURE TO WORK?

* YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE PROGRAM REPEATEDLY WITHOUT ERROR AND UNDERSTAND ITS USE OF SCREEN MANAGEMENT AND AUTO-TERMINATING READS.

```
        .TITL   SCREEN  ; SCREEN MANAGEMENT LAB
    .ENT    START, NEWSC, FLOOP,MKCHG,BLANK_FORM,BLANK_FIELD
    .ENT    FILL_FIELD,UNSCR,RDFLD,CHANGE,DOCHG,RDFNM,RFN,FNBAD
    .ENT    AGAIN,YORN,YN,YES,NO,MESSAGE,FTBL,INPKT,SCPKT,OUPKT

        .NREL   1

START:  ?????   ?????          ; OPEN @INPUT
        DERR    1
        ?????   ?????          ; OPEN @OUTPUT
        DERR    2
NEWSC:  LCALL   BLANK_FORM      ; DISPLAY BLANK FORM
        WADC    0,0
        LNSTA   0,FLDNM
FLOOP:  NLDAI   MXFLD,2
        LNDO    2,MKCHG,FLDNM   ; FOR FLDNM = 0 TO MXFLD DO
        WSUB    0,0             ; CLEAR EDIT FLAG
        LCALL   FILL_FIELD      ; FILL THE FIELD WITH INPUT
        WBR     FLOOP
MKCHG:  LCALL   CHANGE          ; CHANGE A FIELD IF DESIRED
        WBR     MKCHG           ; A CHANGE WAS MADE - ANY MORE?
        LCALL   AGAIN           ; NO CHANGE.  ANOTHER FORM?
        WBR     NEWSC           ; YES, DO IT AGAIN.
        WSUB    2,2
        ?RETURN                 ; NO. EXIT PROGRAM.
        DERR    3
```

```
BLANK_FORM:

; ****************************************************
; *                                                  *
; *               CLEAR THE SCREEN                   *
; *            AND DISPLAY BLANK FORM                *
; *                                                  *
; *         INPUT                  OUTPUT            *
; *                                                  *
; *    AC0 - NONE              AC0 - UNCHANGED       *
; *    AC1 - NONE              AC1 - UNCHANGED       *
; *    AC2 - NONE              AC2 - UNCHANGED       *
; *                                                  *
; *                                                  *
; *                                                  *
; ****************************************************

        LBLNM = 2               ; STACK TEMP OFFSET FOR LABEL
NUMBER

        WSAVR   1
        WSUB    0,0
        LLEFB   1,CLR*2
        LCALL   MESSAGE         ; CLEAR THE SCREEN
        WADC    0,0
        XWSTA   0,LBLNM,3
BLOOP:  NLDAI   MXFLD,2
        LWDO    2,BDONE,LBLNM,3 ; FOR LBLNM = 0 TO MXFLD
        LCALL   BLANK_FIELD     ; DISPLAY THE LABEL
        WBR     BLOOP
BDONE:  WRTN
```

```
BLANK_FIELD:

; ************************************************
; *                                              *
; *           DISPLAY A FIELD LABEL              *
; *                                              *
; *                                              *
; *        INPUT                 OUTPUT          *
; *                                              *
; *   AC0 - NONE               AC0 - UNCHANGED   *
; *   AC1 - NONE               AC1 - UNCHANGED   *
; *   AC2 - FIELD NUMBER       AC2 - UNCHANGED   *
; *                                              *
; *                                              *
; *                                              *
; ************************************************

        WSAVR   0
        LNMUL   2,FENTSZ     ; AC2 <- FLDNM*FENTSZ = TABLE OFFSET
        LNLDA   0,FTBL+FLCR,2   ; GET LABEL CURSOR POSITION
        LWLDA   1,FTBL+FLBP,2   ; AND LABEL TEXT BYTEPOINTER
        LCALL   MESSAGE         ; DISPLAY THE FIELD LABEL
        WRTN
```

```
FILL_FIELD:

; ***************************************************
; *                                                 *
; *          FILL A FIELD WITH INPUT                *
; *                                                 *
; *                                                 *
; *         INPUT                   OUTPUT          *
; *                                                 *
; *    AC0 - EDIT FLAG*          AC0 - UNCHANGED    *
; *    AC1 - NONE                AC1 - UNCHANGED    *
; *    AC2 - FIELD NUMBER        AC2 - UNCHANGED    *
; *                                                 *
; *    * EDIT FLAG - NONZERO = EDIT CURRENT CONTENTS *
; *                                                 *
; ***************************************************

        WSAVR   0
        LNMUL   2,FENTSZ        ; AC2 <- FLDNM*FENTSZ=TABLE
OFFSET
        LNLDA   0,FTBL+FSCR,2   ; GET FIELD CURSOR POSITION
        LWLDA   1,FTBL+FBBP,2   ; AND FIELD BUFFER BYTEPOINTER
        LNLDA   2,FTBL+FLLN,2   ; AND FIELD BYTE LENGTH
        XWLDA   3,?OAC0,3
        WEQZ    3               ; ARE WE EDITING?
        WBR     DORD            ; YES, DON'T PUT IN UNDERSCORES
        LCALL   UNSCR           ; FILL BUFFER WITH UNDERSCORES
DORD:   LCALL   RDFLD           ; READ FIELD
        WRTN
```

```
UNSCR:

; ***************************************************
; *                                                 *
; *                                                 *
; *       FILL A BUFFER WITH UNDERSCORES            *
; *                                                 *
; *        INPUT                  OUTPUT            *
; *                                                 *
; *   AC0 - NONE              AC0 - UNCHANGED       *
; *   AC1 - BYTEPOINTER       AC1 - UNCHANGED       *
; *   AC2 - BYTE LENGTH       AC2 - UNCHANGED       *
; *                                                 *
; *                                                 *
; *                                                 *
; ***************************************************

        WSAVR   0
        NLDAI   "_,0
        WSTB    1,0             ; PUT UNDERSCORE IN FIRST BYTE
        XWLDA   0,?OAC2,3       ; GET BYTE LENGTH
        WNADI   -1,0            ; SUBTRACT ONE
        WMOV    0,1
        XWLDA   3,?OAC1,3       ; GET BYTEPOINTER FOR SOURCE
        WINC    3,2             ; PLUS ONE FOR DESTINATION
        WCMV                    ; PROPOGATE THE FIRST BYTE
        WRTN
```

```
RDFLD:

; ***************************************************
; *                                                 *
; *                READ A FIELD                     *
; *                                                 *
; *                                                 *
; *        INPUT                  OUTPUT            *
; *                                                 *
; *   AC0 - CURSOR POSITION    AC0 - UNCHANGED      *
; *   AC1 - BYTEPOINTER        AC1 - UNCHANGED      *
; *   AC2 - BYTE LENGTH        AC2 - UNCHANGED      *
; *                                                 *
; ***************************************************

        WSAVR   0
        LLEF    2,SCPKT
        XNSTA   0,?????,2       ; STORE CURSOR POS IN SCREEN PKT
        WMOV    2,0
        WIORI   1S0,0
        LLEF    2,INPKT         ; IN INPUT PACKET,
        XWSTA   0,?????,2       ; STORE SCREEN PKT ADDR AND FLAG
        XWSTA   1,?????,2       ; AND BUFFER BYTEPOINTER
        XWLDA   0,?OAC2,3
        WNADI   2,0             ; ( ADD 2 BYTES FOR AUTO-TERM )
        XNSTA   0,?????,2       ; AND FIELD BUFFER LENGTH.
        ?????                   ; FILL FIELD BUFFER
        DERR    6
        WRTN
```

```
CHANGE:

; ************************************************
; *                                              *
; *   CHANGE THE CONTENTS OF A FIELD IF DESIRED  *
; *                                              *
; *                                              *
; *        INPUT                 OUTPUT          *
; *                                              *
; *   AC0 - NONE               AC0 - UNCHANGED   *
; *   AC1 - NONE               AC1 - UNCHANGED   *
; *   AC2 - NONE               AC2 - UNCHANGED   *
; *                                              *
; *              LCALL CHANGE                    *
; *         < field changed return >             *
; *            < no change return >              *
; *                                              *
; ************************************************

        WSAVR   0
        LNLDA   0,CHGCP          ; GET QUESTION CURSOR POSITION
        LLEFB   1,CHMSG*2        ; AND BYTEPOINTER
        LCALL   MESSAGE          ; ASK THE QUESTION
        LNLDA   0,SCPKT+?????    ; GET RETURNED CURSOR POSITION
        LCALL   YORN             ; GET Y OR N ANSWER
        WBR     DOCHG            ; ANSWER WAS 'Y'
        XWISZ   ?ORTN,3          ; ANSWER WAS 'N'
        WRTN

DOCHG:  LNLDA   0,WFLCP          ; GET QUESTION CURSOR POSITION
        LLEFB   1,WFMSG*2        ; AND BYTEPOINTER
        LCALL   MESSAGE          ; ASK THE QUESTION
        LNLDA   0,SCPKT+?????    ; GET RETURNED CURSOR POSITION
        LCALL   RDFNM        ; GET FIELD NUMBER TO CHANGE (AC2)
        WADC    0,0              ; SET EDIT BUFFER FLAG
        LCALL   FILL_FIELD       ; EDIT THE FIELD
        WRTN
```

```
RDFNM:

; ***************************************************
; *                                                 *
; *             READ A FIELD NUMBER                 *
; *                                                 *
; *                                                 *
; *         INPUT                   OUTPUT          *
; *                                                 *
; *    AC0 - CURSOR POSITION    AC0 - UNCHANGED     *
; *    AC1 - NONE               AC1 - UNCHANGED     *
; *    AC2 - NONE               AC2 - FIELD NUMBER  *
; *                                                 *
; ***************************************************

         WSAVR   0
RFN:     XWLDA   0,?OAC0,3        ; GET CURSOR POSITION
         LLEFB   1,YNBUFF*2       ; AND BUFFER BYTEPOINTER
         NLDAI   " ,2
         WSTB    1,2              ; PUT A SPACE IN THE BUFFER
         NLDAI   YNLN,2           ; GET FIELD LENGTH
         LCALL   RDFLD            ; READ FIELD NUMBER
         LLDB    2,YNBUFF*2       ; GET ANSWER CHARACTER
         WCLM    2,2              ; IS IT A VALID FIELD NUMBER?
         .DWORD  "1
         .DWORD  "1+MXFLD
         WBR     FNBAD            ; NO.
         WNADI   -"1,2        ; YES, CONVERT CHR TO VALUE 0-MXFLD
         XWSTA   2,?OAC2,3        ; RETURN FIELD NUMBER TO CALLER
         WRTN

FNBAD:   LNLDA   0,YNCP           ; GET CURSOR POSITION,
         LLEFB   1,FBMSG*2        ; AND BYTEPOINTER
         LCALL   MESSAGE          ; DISPLAY ERROR MESSAGE
         WBR     RFN
```

```
AGAIN:

; ***************************************************
; *                                                 *
; *   ASKS WHETHER THE USER WANTS ANOTHER FORM      *
; *                                                 *
; *                                                 *
; *        INPUT                  OUTPUT            *
; *                                                 *
; *   AC0 - NONE               AC0 - UNCHANGED      *
; *   AC1 - NONE               AC1 - UNCHANGED      *
; *   AC2 - NONE               AC2 - UNCHANGED      *
; *                                                 *
; *               LCALL AGAIN                       *
; *              < yes return >                     *
; *               < no return >                     *
; *                                                 *
; ***************************************************

        WSAVR   0
        LNLDA   0,CHGCP          ; GET QUESTION CURSOR POSITION
        LLEFB   1,AGAIN?*2       ; AND QUESTION BYTEPOINTER
        LCALL   MESSAGE          ; DISPLAY QUESTION
        LNLDA   0,SCPKT+?????    ; GET RETURNED CURSOR POSITION
        LCALL   YORN             ; GET Y OR N ANSWER
        WRTN                     ; ANSWER WAS 'Y'
        XWISZ   ?ORTN,3          ; ANSWER WAS 'N'
        WRTN
```

```
YORN:

; ***************************************************
; *                                                 *
; *           GET 'Y' OR 'N' ANSWER                 *
; *                                                 *
; *                                                 *
; *        INPUT                  OUTPUT            *
; *                                                 *
; *   AC0 - CURSOR POSITION    AC0 - UNCHANGED      *
; *   AC1 - NONE               AC1 - UNCHANGED      *
; *   AC2 - NONE               AC2 - UNCHANGED      *
; *                                                 *
; *              LCALL YORN                         *
; *            < 'Y' return >                       *
; *            < 'N' return >                       *
; *                                                 *
; ***************************************************

        WSAVR   0
YN:     XWLDA   0,?OAC0,3       ; GET CURSOR POSITION
        LLEFB   1,YNBUFF*2      ; BYTEPOINTER
        NLDAI   " ,2
        WSTB    1,2             ; PUT SPACE IN BUFFER
        NLDAI   YNLN,2          ; GET BYTE LENGTH
        LCALL   RDFLD           ; READ ANSWER
        LLDB    0,YNBUFF*2      ; GET CHARACTER
        NLDAI   "Y,1
        WSNE    0,1             ; WAS IT 'Y'?
YES:    WRTN                    ; YES, IT WAS 'Y'.
        NLDAI   "N,1
        WSNE    0,1             ; WAS IT 'N'?
        WBR     NO              ; YES, IT WAS 'N'.
        LNLDA   0,YNCP
        LLEFB   1,YNMSG*2
        LCALL   MESSAGE         ; IT WAS NEITHER - DISPLAY ERROR
MSG
        WBR     YN              ; AND ASK AGAIN.
NO:     XWISZ   ?ORTN,3
        WRTN
```

```
MESSAGE:

; ************************************************
; *                                              *
; *             DISPLAY A MESSAGE                *
; *                                              *
; *                                              *
; *        INPUT                  OUTPUT         *
; *                                              *
; *   AC0 - CURSOR POSITION    AC0 - UNCHANGED   *
; *   AC1 - BYTEPOINTER        AC1 - UNCHANGED   *
; *   AC2 - NONE               AC2 - UNCHANGED   *
; *                                              *
; ************************************************

        WSAVR   0
        LLEF    2,SCPKT          ; IN SCREEN PACKET,
        XNSTA   0,?????,2        ; STORE CURSOR POSITION
        WMOV    2,0
        WIORI   1S0,0
        LLEF    2,OUPKT          ; IN OUTPUT PACKET,
        XWSTA   1,?????,2        ; STORE MSG BYTEPOINTER
        XWSTA   0,?????,2        ; AND SCREEN PKT ADDR AND FLAG
        ??????
        DERR    11
        WRTN
```

```
; ***************************************************
; *                                                 *
; *        SCREEN FORMAT DATA                       *
; *                                                 *
; *  The program is driven by the field table       *
; *  which describes the format of the screen       *
; *  using one entry for each field.                *
; *                                                 *
; ***************************************************

; STRUCTURE OF A FIELD ENTRY

        FSCR = 0          ; FIELD CURSOR POS (COL*400+ROW)
        FBBP = FSCR+1     ; FIELD BUFFER BYTEPOINTER
        FLLN = FBBP+2     ; FIELD BYTE LENGTH
        FLCR = FLLN+1     ; LABEL CURSOR POS (COL*400+ROW)
        FLBP = FLCR+1     ; LABEL TEXT BYTEPOINTER

        FELTH = FLCR+2+1          ; LENGTH OF A FIELD ENTRY
```

```
        .MACRO FIELD    ; COL,ROW,BUFFBP,LEN,LCOL,LROW,LBP
**      .NOMAC  1
; ************************************************
; *                                              *
; *           MACRO TO CREATE A FIELD ENTRY      *
; *                                              *
; *  ARGUMENTS: ^1 - FIELD COLUMN                *
; *             ^2 - FIELD ROW                   *
; *             ^3 - FIELD BUFFER WORD ADDRESS   *
; *             ^4 - FIELD BYTE LENGTH           *
; *             ^5 - LABEL COLUMN                *
; *             ^6 - LABEL ROW                   *
; *             ^7 - LABEL TEXT WORD ADDRESS     *
; *                                              *
; ************************************************

        $ENTRY = .
        .LOC    $ENTRY+FSCR
        .WORD   ^1*400+^2       ; FIELD CURSOR POSITION
        .LOC    $ENTRY+FBBP
        .DWORD  ^3*2            ; FIELD BUFFER BYTEPOINTER
        .LOC    $ENTRY+FLLN
        .WORD   ^4              ; FIELD BYTE LENGTH
        .LOC    $ENTRY+FLCR
        .WORD   ^5*400+^6       ; LABEL CURSOR POSITION
        .LOC    $ENTRY+FLBP
        .DWORD  ^7*2            ; LABEL TEXT BYTEPOINTER
        .LOC    $ENTRY+FELTH
%
FTBL:   ;******************** Field Table ******************
;     column  row   buffer          length  column  row     label

FIELD   20.     4   FIRSTNAME       FNMLN   20.     5       LBL0
FIELD   39.     4   INITIAL         MILN    37.     5       LBL1
FIELD   45.     4   LASTNAME        LNMLN   45.     5       LBL2
FIELD   25.     8.  STREET          STRLN   29.     9.      LBL3
FIELD   20.     12. CITY            CTYLN   25.     13.     LBL4
FIELD   46.     12. STATE           STALN   43.     13.     LBL5
FIELD   55.     12. ZIP             ZIPLN   54.     13.     LBL6
```

```
        FNMLN = 15.             ; FIRST NAME BYTE LENGTH
        MILN  = 1               ; MIDDLE INITIAL BYTE LENGTH
        LNMLN = 15.             ; LAST NAME BYTE LENGTH
        STRLN = 25.             ; STREET ADDRESS BYTE LENGTH
        CTYLN = 20.             ; CITY BYTE LENGTH
        STALN = 2               ; STATE BYTE LENGTH
        ZIPLN = 5               ; ZIP CODE BYTE LENGTH

FENTSZ: .WORD   FELTH           ; SIZE OF FIELD ENTRY
CHGCP:  .WORD   20.*400+18.     ; CHANGE QUESTION CURSOR POS
YNLN    =       1               ; Y/N ANSWER BYTE LENGTH
WFLCP:  .WORD   45.*400+18.     ; WHICH FIELD QUESTION CURSOR POS
YNCP:   .WORD   20.*400+20.     ; ERROR MESSAGE CURSOR POS


**      .NOLOC  1
LBL0:   .TXT    '1. First Name'
LBL1:   .TXT    '2. MI'
LBL2:   .TXT    '3. Last Name'
LBL3:   .TXT    '4. Street Address'
LBL4:   .TXT    '5. City'
LBL5:   .TXT    '6. State'
LBL6:   .TXT    '7. Zip'

CHMSG:  .TXT    'Any changes (Y or N)? '
WFMSG:  .TXT    'What number? '
AGAIN?: .TXT    'Do you wish to enter another form (Y or N)? '
YNMSG:  .TXT    'Please respond with Y or N<207>'
FBMSG:  .TXT    'Please respond with a digit from 1 to 7<207>'
CLR:    .TXT    '<14>'          ; CLEAR SCREEN
**      .NOLOC  0


        .NREL   0               ; UNSHARED DATA

FLDNM:          .WORD   0       ; CURRENT FIELD NUMBER
MXFLD           =       6       ; MAX FIELD NUMBER

; BUFFERS FOR AUTO-TERMINATING READS

FIRSTNAME:      .TXT    '               <177><177>'
INITIAL:        .TXT    ' <177><177>'
LASTNAME:       .TXT    '               <177><177>'
STREET:         .TXT    '                       <177><177>'
CITY:           .TXT    '                   <177><177>'
STATE:          .TXT    '  <177><177>'
ZIP:            .TXT    '     <177><177>'
YNBUFF:         .TXT    ' <177><177>'
```

```
        ; I/O PACKET FOR FILE @INPUT

INPKT:  .LOC    INPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?ISTI     ; FILE SPECS
        .WORD   ????????????????????
                                ; CHG TO DATATSEN,INPUT,EXT PKT
        .LOC    INPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?IMRS     ; PHYSICAL BLOCK SIZE.
        .WORD   -1              ; DEFAULT
        .LOC    INPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  -1              ; FILLED IN BEFORE EACH ?READ
        .LOC    INPKT+?IRES     ; RESERVED
        .WORD   0
        .LOC    INPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   -1              ; FILLED IN BEFORE EACH ?READ
        .LOC    INPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?IRNW     ; RESERVED
        .WORD   0
        .LOC    INPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    INPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATINPUT*2
        .LOC    INPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    INPKT+?ETSP
        .DWORD  ?????
        .LOC    INPKT+?ETFT
        .DWORD  0
        .LOC    INPKT+?ETLT
        .DWORD  0
        .LOC    INPKT+?ENET
        .DWORD  0
        .LOC    INPKT+?iblt     ; ?????=PACKET LENGTH

SCPKT:  .LOC    SCPKT+?????     ; FLAGS
        .WORD   ????????????????????????????
                                ; SCREEN EDIT,INITIAL CP,
                                ; NO ECHO DELIM, RETURN CP,
REDISPLAY
        .LOC    SCPKT+?????     ; RELATIVE CURSOR POSITION
        .WORD   0               ; NOT USED
        .LOC    SCPKT+?????     ; INITIAL CURSOR POSTITION
        .WORD   0               ; FILLED IN BEFORE EACH ?READ

ATINPUT: .TXT    /@INPUT/
```

```
        ; I/O PACKET FOR FILE @OUTPUT

OUPKT:  .LOC    OUPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?ISTI     ; FILE SPECS
        .WORD   ????????????????????
                                ; CHG TO DATASEN,OUTPUT,EXT PKT
        .LOC    OUPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IMRS     ; PHYSICAL BLOCK SIZE.
        .WORD   -1              ; DEFAULT
        .LOC    OUPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  -1              ; FILLED IN BEFORE EACH ?WRITE
        .LOC    OUPKT+?IRES     ; RESERVED
        .WORD   0
        .LOC    OUPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   80.
        .LOC    OUPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IRNW     ; RESERVED
        .WORD   0
        .LOC    OUPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    OUPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATOUTPUT*2
        .LOC    OUPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    OUPKT+?ETSP
        .DWORD  ?????
        .LOC    OUPKT+?ETFT
        .DWORD  0
        .LOC    OUPKT+?ETLT
        .DWORD  0
        .LOC    OUPKT+?ENET
        .DWORD  0
        .LOC    OUPKT+?IBLT     ; ?IBLT=PACKET LENGTH

ATOUTPUT:       .TXT    '@OUTPUT'

        .END    START
```

```
*       MOVE/R/NACL COPIES OF THE FILES 'PROC.SR' AND 'ERROR.OB'
        FROM THE :S309VS:ASSY DIRECTORY TO YOUR WORKING
        DIRECTORY.

*       THE FUNCTION OF 'PROC' IS TO CREATE A SUBORIDNATE PROCESS
        RUNNING THE CLI PROGRAM, AND TO PASS AN INNITIAL IPC
        MESSAGE CONTAINING A CLI COMMAND TO THE NEW PROCESS.  THE
        NEW PROCESS IS TO EXECUTE THE CLI COMMAND AND IMMEDIATELY
        TERMINATE.

*       REFER TO THE PRINTOUT OF 'PROC.SR' ON THE FOLLOWING
        PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?)
        WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM
        CALLS AND THE CONTENTS OF THE PACKETS.

*       AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING
        COMMANDS:

                DELETE/2=IGNORE,PROC.LS
                X,MASM/L=PROC.LS,PROC
                X,LINK/L=PROC.LS,PROC,ERROR

*       GET A HARDCOPY OF THE FILE 'PROC.LS'.

*       EXECUTE THE PROC PROGRAM. YOUR ORIGINAL CLI PROCESS
        BECOMES BLOCKED UNTIL THE 'PROC' PROGRAM TERMINATES.  THE
        'PROC' PROGRAM CREATES THE SUBORDINATE CLI AND THE 'PROC'
        PROCESS BECOMES BLOCKED WAITING FOR THE NEW CLI TO
        TERMINATE.  THE NEW CLI EXECUTES THE COMMAND IN THE IPC
        MESSAGE ASSOCIATED WITH THE ?PROC PACKET AND THEN
        TERMINATES.  THEN THE 'PROC' PROCESS BECOMES UNBLOCKED
        AND TERMINATES ITSELF, WHICH NOW UNBLOCKS YOUR ORIGINAL
        CLI PROCESS.

*       USE THE CLI TYPE COMMAND TO EXAMINE THE CONTENTS OF
        'OUTFILE'. IT SHOULD CONTAIN THE MESSAGE "HELLO FROM YOUR
        CREATOR".

*       YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE
        PROGRAM REPEATEDLY WITHOUT ERROR

    ************************************************************
    *                       CHALLENGE                          *
    ************************************************************

*       MODIFY THE PROGRAM SO THAT THE NEW CLI DOES NOT TERMINATE
        AFTER COMPLETING THE CLI COMMAND IN THE IPC MESSAGE, BUT
        GIVES THE CLI PROMPT INSTEAD.  YOU WILL NEED TO PASS
        ?RFCF (CLI FORMAT) IN THE USER FLAGS WORD, AND TO ADD
        'CLI,' TO THE BEGINNING OF THE IPC MESSAGE TEXT STRING.
```

```
        .TITL   PROC
        .ENT    START PPKT
        .NREL   1

START:  WSUB    0,0
        WSUB    1,1
        ?????   ????
        DERR    1

        WSUB    2,2
        ?RETURN
        DERR    2

        .NREL   0

        ; PROCESS CREATION PACKET

PPKT:   .LOC    PPKT+?PFLG      ; CREATION FLAGS
        .WORD   ?????           ; BLOCK CREATOR, SWAPPABLE
        .LOC    PPKT+?PPRI      ; PROCESS PRIORITY
        .WORD   ??              ; SAME AS CREATOR
        .LOC    PPKT+?PSNM      ; PROGRAM FILENAME BP
        .DWORD  PRFILE*2
        .LOC    PPKT+?PIPC      ; IPC HEADER
        .DWORD  ???
        .LOC    PPKT+?PNM       ; PROCESSNAME
        ??????  ??              ; DEFAULT
        .LOC    PPKT+?PMEM      ; MAXIMUM LOGICAL PAGES
        ??????  ??              ; SAME AS CREATOR
        .LOC    PPKT+?PDIR      ; INITIAL DIRECTORY
        .DWORD  0           ; SAME AS CREATOR'S CURRENT DIRECTORY
        .LOC    PPKT+?PCON      ; @CONSOLE
        ??????  ??              ; SAME AS CREATOR'S @CONSOLE
        .LOC    PPKT+?PCAL      ; MAX CONCURRENT SYSTEM CALLS
        ?????   ??              ; DEFAULT (2)
        .LOC    PPKT+?PWSS      ; MAX WORKING SET SIZE
        ?????   ??              ; SAME AS CREATOR'S
        .LOC    PPKT+?PUNM      ; USERNAME BP
        ??????  ??              ; SAME AS CREATOR'S
        .LOC    PPKT+?PPRV      ; PROCESS PRIVILEGES
        ?????   ??              ; SAME AS CREATOR
        .LOC    PPKT+?PPCR      ; SON'S QUOTA
        ?????   ??              ; REMAINDER OF CREATOR'S QUOTA
        .LOC    PPKT+?PWMI      ; MINIMUM WORKING SET SIZE
        ?????   ??              ; SAME AS CREATOR'S
        .LOC    PPKT+?PIFP      ; @INPUT
        ??????  ??              ; SAME AS CREATOR'S @INPUT
        .LOC    PPKT+?POFP      ; @OUTPUT
        ??????  ??              ; SAME AS CREATOR'S @OUTPUT
        .LOC    PPKT+?PLFP      ; @LIST
        ??????  ??              ; SAME AS CREATOR'S @LIST
        .LOC    PPKT+?PDFP      ; @DATA
        ??????  ??              ; SAME AS CREATOR'S @DATA
        .LOC    PPKT+?SMCH      ; CPU TIME LIMIT
        ??????  ??          ; REMAINDER OF CREATOR'S CPU LIMIT
        .LOC    PPKT+?PLTH
```

```
PRFILE: .TXT    "CLI.PR"

        ; IPC HEADER

HDR:    .LOC    HDR+?ISFL       ; SYSTEM FLAGS
        .WORD   0               ; NONE
        .LOC    HDR+?IUFL       ; USER FLAGS
        .WORD   ??              ; NONE - DO CMD AND TERMINATE
        .LOC    HDR+?IDPH       ; DESTINATION GLOBAL PORT
        ??????  ??
        .LOC    HDR+?IOPN       ; ORIGIN LOCAL PORT
        ?????   ??
        .LOC    HDR+?ILTH       ; BUFFER LENGTH (WORDS)
        .WORD   ?????
        .LOC    HDR+?IPTR       ; BUFFER ADDRESS
        .DWORD  ???
        .LOC    HDR+?IPLTH

MSG:    .TXT "WRITE/L=OUTFILE,HELLO,FROM,YOUR,CREATOR"
MSGLN   = (.-MSG)

        .END    START
```

ASSEMBLY LANGUAGE CONTROL IPC LAB

* MOVE/R/NACL A COPY OF THE FILES CONTROL.IPC.SR AND ERROR.OB FROM THE :S309VS:ASSY DIRECTORY TO YOUR INITIAL WORKING DIRECTORY.

* REFER TO THE PRINTOUT OF CONTROL.IPC.SR ON THE FOLLOWING PAGES.

* THIS PROGRAM ACCEPTS ANY IPC MESSAGE SENT TO THE FILE 'TRYIPC' FROM ANY CONSOLE. TO DO THIS THE CONSOLE DOES NOT HAVE TO BE ASSOCIATED WITH THE RECEIVING PROCESS.

* WHAT IS THE CLI COMMAND WHICH ALLOWS YOU TO INTERACTIVELY SEND AN IPC MESSAGE TO A PROCESS?

_____

* EACH MESSAGE RECEIVED IS DISPLAYED ON THE PROCESS' CONSOLE, THEN AN ACKNOWLEDGEMENT IS RETURNED TO THE SENDER.

* EDIT YOUR COPY OF THE SOURCE FILE AND FILL IN THE FOLLOWING:

1. THE 'LWSTA' INSTRUCTION AFTER THE LABEL 'INIT' TO SET THE GLOBAL PORT NUMBER IN THE ?IREC PACKET TO RECEIVE FROM ANY SENDER.

2. THE 'LNSTA' INSTRUCTION WHICH RESETS THE BUFFER LENGTH SPEC IN THE ?IREC HEADER.

3. AFTER THE MESSAGE HAS BEEN RECEIVED THE SENDER'S ACTUAL GLOBAL PORT NUMBER WILL BE PLACED IN THE ?IREC HEADER BY THE SYSTEM.

4. AT THE LABEL 'ACK', GET THIS GLOBAL PORT NUMBER FOR USE BY THE ?GPORT CALL.

5. CIPC, THE PACKET USED TO CREATE THE IPC FILE TRYIPC.

6. IPCPK, THE ?IREC (PACKET) HEADER.

* ASSEMBLE THE PROGRAM AND LINK IT WITH THE ERROR ROUTINE.

* EXECUTE YOUR PROGRAM.

* LOG ON AT ANOTHER TERMINAL AND SEND MESSAGES TO YOUR PROGRAM:

```
) CONTROL TRYIPC YOUR MESSAGE GOES HERE
```

EACH TIME YOU SHOULD GET THE RESPONSE:

```
FROM PID XX: I GOT THE MESSAGE
```

EACH MESSAGE WILL BE DISPLAYED ON YOUR RECEIVING PROCESS' CONSOLE. NOTE THAT THE CLI COMMAND USED TO SEND THESE MESSAGES EDITS THEM SOMEWHAT BEFORE THEY ARE SENT.

* WHEN YOU HAVE SENT AS MANY MESSAGES AS YOU WANT, ENTER CONTROL-C, CONTROL-B AT YOUR RECEIVING CONSOLE.

```
*       ----------------- CHALLENGE -----------------

        MODIFY THE PROGRAM SO THAT IT WILL TERMINATE NORMALLY BY
        ITSELF IF IT RECEIVES A MESSAGE WHICH IS LESS THAN TWO
        WORDS (4 BYTES) IN LENGTH.

        ( WE CHOSE TWO WORDS BECAUSE THE CLI WON'T LET US SEND A
        ZERO LENGTH MESSAGE. )

        YOU WILL FIND THE ?RETURN IS ALREADY PROVIDED IN THE
        SOURCE CODE. YOU MUST HAVE YOUR PROGRAM DECIDE WHEN TO
        EXECUTE IT.
```

```
        .TITLE  CIPC    ; CONTROL IPC LAB
        .ENT    START,INIT,ACK,EXIT
        .ENT    CIPC,IPCPK,OUTFL,BUFF
        .NREL   1

        BUFFLEN=136./2  ; BUFFER LENGTH SYMBOL
        NL   =  12

START:  LLEFB   0,IPCNM*2       ; GET BP TO IPC FILENAME
        ?CREATE CIPC            ; CREATE IPC FILE
        DERR    1

        ?OPEN   OUTFL   ; OPEN @OUTPUT
        DERR    2

INIT:   WSUB    3,3
        LWSTA   3,?????+?????   ; TO RECEIVE FROM ANY SENDER,
                                ; CLEAR GLOBAL PORT NUMBER
        NLDAI   BUFFLEN,3       ; RESET BUFFER LENGTH SPEC
        LNSTA   3,?????+?????   ; IN ?IREC HEADER
        ?IREC   IPCPK           ; WAIT FOR A MESSAGE TO COME IN
        DERR    10
        LNLDA   2,IPCPK+?ILTH   ; GET MSG LENGTH IN WORDS
        WASHI   1,2             ; TIMES TWO MAKES BYTE LENGTH
        LLEFB   3,BUFF*2        ; GET BYTE POINTER
        WADD    2,3             ; TO END OF MESSAGE
        WSBI    1,3
        NLDAI   NL,0            ; STORE A NEWLINE THERE.
        WSTB    3,0
        LNSTA   2,OUTFL+?IRCL   ; SET NUMBER OF BYTES TO WRITE
        ?WRITE  OUTFL           ; DISPLAY THE MESSAGE
        DERR    3

ACK:    LWLDA   1,?????+?????   ; GET SENDER'S GLOBAL PORT NUMBER
        ?GPORT                  ; GET THE SENDER'S PID
        DERR    4

        WMOV    1,0             ; INTO AC0
        LLEFB   1,MES*2         ; GET BP TO ACKNOWLEDGEMENT
        NLDAI   ACKLN,2         ; AND ITS LENGTH
        ?SEND               ; ACKNOWLEDGE RECEIPT OF THE MESSAGE
        DERR    5
        WBR     INIT            ; GET NEXT MESSAGE

;       THIS CODE IS HERE FOR YOU TO USE IF YOU ATTEMPT THE
;       CHALLENGE AT THE END OF THE LAB.

EXIT:   WSUB    2,2             ; NORMAL EXIT
        ?RETURN
        DERR    6
```

```
        ;PARAMETER PACKETS

        .NREL   0

        ; PACKET FOR IPC FILE CREATION

CIPC:   .LOC    CIPC+?CFTYP     ; FILE TYPE
        .WORD   ?????           ; TYPE=IPC
        .LOC    CIPC+?CPOR      ; LOCAL PORT
        .WORD   ??              ; PICK A PORT
        .LOC    CIPC+?CTIM      ; TIME BLOCK ADDR
        ??????  ??              ; DEFAULT TIME
        .LOC    CIPC+?CACP      ; ACL BYTE PTR
        ??????  ??              ; DEFAULT ACL
        .LOC    CIPC+?CLTH


IPCNM:  .TXT    "TRYIPC"


        ; ?IREC HEADER

IPCPK:  .LOC    IPCPK+?ISFL     ; SYSTEM FLAGS
        .WORD   ??              ; NONE. DO NOT SPOOL IF BUFFER
SHORT
        .LOC    IPCPK+?IUFL     ; USER FLAGS
        .WORD   ??              ; USER FLAGS FROM SENDER
        .LOC    IPCPK+?IOPH     ; ORIGIN GLOBAL PORT (HI)
        ??????  ??              ; ACCEPT FROM ANY SENDER PORT
        .LOC    IPCPK+?IDPN     ; DESTINATION LOCAL PORT
        .WORD   ??              ; SAME PORT YOU PICKED IN CIPC
        .LOC    IPCPK+?ILTH     ; BUFFER LENGTH
        .WORD   ???????         ; (WORDS)
        .LOC    IPCPK+?IPTR     ; BUFFER ADDRESS
        ??????  ????
        .LOC    IPCPK+?IPLTH    ; ?IPLTH=PACKET LENGTH


        ; DATA FOR ?SEND

MES:    .TXT    /I GOT THE MESSAGE<12>/
ACKLN = (.-MES)*2
```

```
        ; PARAMETER PACKET FOR FILE @OUTPUT

OUTFL:  .LOC    OUTFL+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUTFL+?ISTI     ; FILE SPECS
        .WORD   ?ICRF+?RTDY+?OFOT ; CHG TO DYNAMIC, OUTPUT ONLY
        .LOC    OUTFL+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUTFL+?IMRS     ; PHYS BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    OUTFL+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  BUFF*2
        .LOC    OUTFL+?IRES     ; RESERVED
        .WORD   0
        .LOC    OUTFL+?IRCL     ; MAX RECORD LENGTH
        .WORD   0               ; SET AT ?WRITE TIME
        .LOC    OUTFL+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUTFL+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    OUTFL+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATOUT*2
        .LOC    OUTFL+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    OUTFL+?IOSZ     ; ?IOSZ=PACKET LENGTH

ATOUT:  .TXT    /@OUTPUT/
BUFF:   .BLK    BUFFLEN

        .END    START
```

ASSEMBLY LANGUAGE TALK IPC LAB

USING THE FOLLOWING CLI COMMANDS, GET THE NEEDED FILES INTO YOUR DIRECTORY:

```
    DIRECTORY, :S309VS:ASSY
    MOVE/R/NACL, :UDD:YOURUSERNAME, TALK<ONE,TWO>.SR, ERROR.OB
DIRECTORY/I
```

EXAMINE THE PRINTOUTS OF TALKONE.SR AND TALKTWO.SR ON THE FOLLOWING PAGES AND FILL IN THE ITEMS WHICH HAVE BEEN REPLACED WITH QUESTION MARKS. YOU WILL HAVE TO SUPPLY VARIOUS SYSTEM CALLS AND THEIR ARGUMENTS, ERROR CODE SYMBOLS, ADDRESSES OF VARIOUS ITEMS IN PACKETS, AND THE APPROPRIATE VALUES IN THOSE PACKETS.

WHEN YOU HAVE DETERMINED THE CORRECT INFORMATION FOR EACH OF THESE ITEMS, EDIT YOUR TWO SOURCE FILES AND MAKE THE CORRECTIONS.

ASSEMBLE EACH SOURCE FILE AND LINK IT WITH THE ERROR ROUTINE.

LOG ON AT ANOTHER CONSOLE. EXECUTE TALKONE AT ONE CONSOLE AND TALKTWO AT THE SECOND. YOU SHOULD BE ABLE TO SEND MESSAGES BACK AND FORTH.

TO STOP EACH OF THESE PROGRAMS, ENTER CONTROL-D (END-OF-FILE).

********** CHALLENGE **********

WORKING WITH ANOTHER STUDENT USE THE PROGRAMS TO CONVERSE.

HINT:

WHEN YOU EXECUTED THE PROGRAMS BY YOURSELF, WHAT WAS IT THAT ENSURED YOU ONLY GOT YOUR OWN MESSAGES EVEN THOUGH OTHERS MAY HAVE BEEN EXECUTING THE SAME PROGRAMS ????

****

********** CHALLENGE **********

PREPARE A FILE FULL OF MESSAGES WHICH COMPRISE ONE HALF OF A CONVERSATION.

CAUSE TALKONE TO EXECUTE, SENDING THE MESSAGES FROM THIS FILE, AND STORING ANY MESSAGES IT MAY RECEIVE IN AN EMPTY FILE OF YOUR CHOICE.

EXECUTE TALKTWO AT YOUR CONSOLE, AND RESPOND TO THE MESSAGES COMING FROM YOUR TALKONE.

==========>> YOU MAY USE ONLY ONE CONSOLE !! <<==========

AFTER YOU HAVE RESPONDED TO THE LAST MESSAGE IN YOUR FILE, YOUR TALKTWO PROCESS SHOULD TERMINATE AUTOMATICALLY.
EXAMINE THE FILE IN WHICH TALKONE STORED THE MESSAGES IT RECEIVED.

****

PREPARE A SECOND FILE WITH THE RESPONSES TO THE MESSAGES IN THE FIRST FILE. CAUSE BOTH TALKONE AND TALKTWO TO EXECUTE, TAKING INPUT FROM THESE TWO MESSAGE FILES, AND STORING ANY RECEIVED MESSAGES IN TWO NEW EMPTY FILES OF YOUR CHOICE.

==========>> YOU MAY NOT USE ANY CONSOLE !! <<==========
(EXCEPT TO START AND STOP THE PROCESSES)

****

```
        .TITLE  TALK1   ; THIS PROCESS COOPERATES WITH TALK2
        .ENT    START,LKAGN,ERRCK,BEGIN,RDIT,SDIT,RCIT
        .ENT    WRIT,EOFCK,HDR,INPKT,OUPKT,BUFF

        .NREL   1
        BUFFLEN=80./2           ; BUFFER LENGTH SYMBOL

START:  ?OPEN   INPKT           ; OPEN @INPUT
        DERR    1

        ?OPEN   OUPKT           ; OPEN @OUTPUT
        DERR    2

LKAGN:  LLEFB   0,SFILE*2       ; GET BP TO OTHER PROC'S IPC FILE
        ??????                  ; GET ITS GLOBAL PORT IN AC1
        WBR     ERRCK
        WBR     BEGIN

ERRCK:  NLDAI   ?????,1         ; GOT AN ERROR,
        WSNE    0,1             ; WAS IT FILE DOES NOT EXIST ?
        WBR     LKAGN           ; YES. LOOK AGAIN.
        DERR    3               ; NO. TAKE ERROR EXIT.

BEGIN:  LWSTA   1,???+?????     ; STORE GLOBAL PORT NUMBER
                                ; IN IPC HDR
        ?WRITE  OUPKT           ; DISPLAY INITIAL PROMPT
        DERR    4

RDIT:   ?READ   INPKT           ; READ A LINE FROM @INPUT
        WBR     EOFCK

        LNLDA   0,?IRLR,2       ; GET ITS BYTE LENGTH
        WSKBZ   31.             ; IF IT'S ODD,
        WINC    0,0             ; MAKE IT EVEN
        WASHI   -1,0            ; AND THEN MAKE A WORD COUNT
        LNSTA   0,???+?????     ; AND STORE IT IN THE IPC HEADER

SDIT:   ??????  ???             ; SEND IT AS AN IPC MESSAGE
        DERR    5

RCIT:   NLDAI   BUFFLEN,0       ; RESET THE BUFFER LENGTH
        LNSTA   0,???+?????     ; IN THE IPC HEADER
        ?????   ???             ; RECEIVE AN IPC MESSAGE
        DERR    6

        XNLDA   0,?????,?       ; GET ITS LENGTH.
        WSNEI   0,0             ; WAS IT ZERO ?
        WBR     EXIT            ; YES. TERMINATE.

WRIT:   ?WRITE  OUPKT           ; DISPLAY IT ON @OUTPUT
        DERR    7
        WBR     RDIT            ; DO IT ALL AGAIN
```

```
EOFCK:  NLDAI   EREOF,1         ; GOT AN ERROR
        WSEQ    0,1             ; WAS IT END-OF-FILE ??
        DERR    10              ; NO. TAKE ERROR EXIT.
        WSUB    0,0             ; YES. SEND ZERO LENGTH MSG.
        LNSTA   0,???+?????
        ??????  ???
        DERR    11

EXIT:   WXOR    2,2             ; NORMAL EXIT.
        ?RETURN
        DERR    12
```

```
        .NREL   0

        ; IPC HEADER

HDR:    .LOC    HDR+?ISFL       ; SYSTEM FLAGS
        .WORD   ??              ; NONE
        .LOC    HDR+?IUFL       ; USER FLAGS
        .WORD   ??
        .LOC    HDR+?IOPH       ; ORIGIN GLOBAL PORT
        ??????  ??              ; FILLED IN AFTER ?ILKUP
        .LOC    HDR+?IDPN       ; DESTINATION LOCAL PORT
        .WORD   ??              ; CHOOSE ANY LOCAL PORT
        .LOC    HDR+?ILTH       ; BUFFER LENGTH (WORDS)
        .WORD   ??              ; SET BEFORE EACH ?IREC
        .LOC    HDR+?IPTR       ; BUFFER ADDRESS
        ??????  ????
        .LOC    HDR+?IPLTH

SFILE:  .TXT    /FILEB/
```

```
        ; PARAMETER PACKET FOR FILE @INPUT

INPKT:  .LOC    INPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?ISTI     ; FILE SPECS
        .WORD   ?ICRF+?RTDS+?OFIN       ; CHG TO DATASEN, INPUT
ONLY
        .LOC    INPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?IMRS     ; PHYS BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    INPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  BUFF*2
        .LOC    INPKT+?IRES     ; RESERVED (TAPE DENSITY)
        .WORD   0
        .LOC    INPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   BUFFLEN*2       ; (BYTES)
        .LOC    INPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    INPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATINPUT*2
        .LOC    INPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    INPKT+?IOSZ     ; ?IOSZ=PACKET LENGTH

ATINPUT:        .TXT    /@INPUT/
```

```
        ; PARAMETER PACKET FOR FILE @OUTPUT

OUPKT:  .LOC    OUPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?ISTI     ; FILE SPECS
        .WORD   ?ICRF+?RTDS+?OFOT  ; CHG TO DATASEN, OUTPUT ONLY
        .LOC    OUPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IMRS     ; PHYS BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    OUPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  BUFF*2
        .LOC    OUPKT+?IRES     ; RESERVED
        .WORD   0
        .LOC    OUPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   BUFFLEN*2       ; (BYTES)
        .LOC    OUPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    OUPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATOUTPUT*2
        .LOC    OUPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    OUPKT+?IOSZ     ; ?IOSZ=PACKET LENGTH

ATOUTPUT:       .TXT    /@OUTPUT/

BUFF:   .BLK    BUFFLEN
        .LOC    BUFF            ; INITIAL PROMPT MESSAGE
        .TXT    /TYPE A MESSAGE AND WAIT FOR A RESPONSE...<12>/

        .END    START
```

```
        .TITL   TALK2    ; THIS PROCESS COOPERATES WITH TALK1
        .ENT    START,BEGIN,RDIT,SDIT,RCIT
        .ENT    WRIT,EOFCK,IPCPK,HDR,INPKT,OUPKT,BUFF

        .NREL   1
        BUFFLEN=80./2           ; BUFFER LENGTH SYMBOL

START:  LLEFB   0,RFILE*2       ; GET IPC FILNAME BP
        ??????? ?????           ; CREATE IPC FILE
        DERR    1

        ?OPEN   INPKT           ; OPEN @INPUT
        DERR    2

        ?OPEN   OUPKT           ; OPEN @OUTPUT
        DERR    3

BEGIN:  ?WRITE  OUPKT           ; DISPLAY INITIAL PROMPT
        DERR    4

RCIT:   NLDAI   BUFFLEN,0       ; RESET THE BUFFER LENGTH
        LNSTA   0,???+?????     ; IN THE IPC HEADER
        ?????   ???             ; RECEIVE AN IPC MESSAGE
        DERR    5

        XNLDA   0,?????,?       ; GET ITS LENGTH.
        WSNEI   0,0             ; WAS IT ZERO ?
        WBR     EXIT            ; YES. TERMINATE.

WRIT:   ?WRITE  OUPKT           ; DISPLAY IT ON @OUTPUT
        DERR    6

RDIT:   ?READ   INPKT           ; READ A LINE FROM @INPUT
        WBR     EOFCK

        LNLDA   0,?IRLR,2       ; GET ITS BYTE LENGTH
        WSKBZ   31.             ; IF IT'S ODD,
        WINC    0,0             ; MAKE IT EVEN
        WASHI   -1,0            ; AND THEN MAKE A WORD COUNT
        LNSTA   0,???+?????     ; AND STORE IT IN THE IPC HEADER

SDIT:   ??????  ???             ; SEND IT AS AN IPC MESSAGE
        DERR    7

        WBR     RCIT            ; DO IT ALL AGAIN
```

```
EOFCK:  NLDAI   EREOF,1         ; GOT AN ERROR
        WSEQ    0,1             ; WAS IT END-OF-FILE ?
        DERR    10              ; NO. TAKE ERROR EXIT.

        WSUB    0,0             ; YES. SEND ZERO LENGTH MSG.
        LNSTA   0,???+?????
        ??????  ???
        DERR    11

EXIT:   WXOR    2,2             ; NORMAL EXIT.
        ?RETURN
        DERR    12
```

```
        .NREL   0

        ; IPC FILE ?CREATE PACKET

IPCPK:  .LOC    IPCPK+?CFTYP    ; FILE TYPE
        .WORD   ?????           ; IPC
        .LOC    IPCPK+?CPOR     ; LOCAL PORT NUMBER
        .WORD   ??              ; OF YOUR CHOICE
        .LOC    IPCPK+?CTIM     ; TIME BLOCK ADDR
        ??????  ??              ; DEFAULT TO CURRENT TIME
        .LOC    IPCPK+?CACP     ; ACL BYTE POINTER
        ??????  ??              ; DEFAULT ACL
        .LOC    IPCPK+?CLTH

        ; IPC HEADER

HDR:    .LOC    HDR+?ISFL       ; SYSTEM FLAGS
        .WORD   ??              ; NONE
        .LOC    HDR+?IUFL       ; USER FLAGS
        .WORD   ??
        .LOC    HDR+?IOPH       ; ORIGIN GLOBAL PORT
        ??????  ??              ; RECEIVE FROM ANY SENDER
        .LOC    HDR+?IDPN       ; DESTINATION LOCAL PORT
        .WORD   ??              ; SAME AS IN IPC FILE ?CREATE PKT
        .LOC    HDR+?ILTH       ; BUFFER LENGTH (WORDS)
        .WORD   ??              ; SET BEFORE EACH ?IREC
        .LOC    HDR+?IPTR       ; BUFFER ADDRESS
        ??????  ????
        .LOC    HDR+?IPLTH

RFILE:  .TXT    /FILEB/
```

```
        ; PARAMETER PACKET FOR FILE @INPUT

INPKT:  .LOC    INPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?ISTI     ; FILE SPECS
        .WORD   ?ICRF+?RTDS+?OFIN  ; CHG TO DATASEN, INPUT ONLY
        .LOC    INPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?IMRS     ; PHYS BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    INPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  BUFF*2
        .LOC    INPKT+?IRES     ; RESERVED
        .WORD   0
        .LOC    INPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   BUFFLEN*2       ; (BYTES)
        .LOC    INPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    INPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    INPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATINPUT*2
        .LOC    INPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    INPKT+?IOSZ     ; ?IOSZ=PACKET LENGTH

ATINPUT:        .TXT    /@INPUT/
```

```
        ; PARAMETER PACKET FOR FILE @OUTPUT

OUPKT:  .LOC    OUPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?ISTI     ; FILE SPECS
        .WORD   ?ICRF+?RTDS+?OFOT  ; CHG TO DATASEN, OUTPUT ONLY
        .LOC    OUPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IMRS     ; PHYS BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    OUPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  BUFF*2
        .LOC    OUPKT+?IRES     ; RESERVED (TAPE DENSITY)
        .WORD   0
        .LOC    OUPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   BUFFLEN*2       ; (BYTES)
        .LOC    OUPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    OUPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATOUTPUT*2
        .LOC    OUPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    OUPKT+?IOSZ     ; ?IOSZ=PACKET LENGTH

ATOUTPUT:       .TXT    /@OUTPUT/

BUFF:   .BLK    BUFFLEN
        .LOC    BUFF            ; INITIAL PROMPT MESSAGE
        .TXT    /WAIT FOR A MESSAGE AND TYPE A RESPONSE...<12>/

        .END    START
```

TCB LAB

MOVE/R/NACL TRCON.SR AND ERROR.OB FROM THE DIRECTORY :S309VS:ASSY INTO YOUR DIRECTORY.

REFER TO THE PRINTOUT OF TRCON.SR ON THE FOLLOWING PAGES. FILL IN THE SYSTEM CALLS, PACKET VALUES, AND THE STACK DECLARATION WHICH HAVE ALL BEEN REPLACED WITH QUESTION MARKS. EDIT YOUR CHANGES INTO YOUR COPY OF THE SOURCE FILE AND ASSEMBLE IT.

LINK IT WITH THE EXTERNAL ERROR EXIT ROUTINE, REMEMBERING TO SPECIFY THE CORRECT NUMBER OF TCB'S TO RESERVE.

EXAMINE THE PROGRAM LOGIC AND TRY TO PREDICT THE PROGRAM'S OPERATION.

EXECUTE THE PROGRAM. WHEN YOU SEE THE FIRST ">" PROMPT, TYPE IN A TASK ID NUMBER, A COMMA, AND A MESSAGE TO THAT TASK. IF THE TASK YOU SPECIFIED IS ONE WHICH HAS DONE A ?TRCON, YOU SHOULD GET THREE LINES OF RESPONSE FROM IT. TRY SOME OTHER TASK ID'S AND MESSAGES. YOU WILL NOTICE THAT THE NEXT ?TRCON PROMPT APPEARS AT THE BEGINNING OF THE RESPONSE, BUT JUST TYPE IN YOUR NEXT MESSAGE ANYWAY.

WHAT HAPPENS IF YOU SPECIFY AN INVALID TASK ID ?

STUDY THE FLOW OF THE CODE (ALL THREE OF THE ?TRCON TASKS EXECUTE THE SAME CODE PATH) AND DETERMINE HOW EACH TASK IS ABLE TO KEEP TRACK OF ITS OWN MESSAGES AND BUFFER.

ENTER CNTRL-C, CNTRL-A. WHY DID THE PROGRAM TERMINATE ?

USING THE DEBUGGER, USE THE $K COMMAND TO OBTAIN TCB INFORMATION (NOTE THAT "$" IS THE ESCAPE CHARACTER). HOW MANY TCB'S ARE RESERVED IN THIS PROGRAM ? _____

HOW MANY TCB'S ARE ON THE ACTIVE SCHEDULING CHAIN? HOW MANY ARE FREE? ACTIVE: _____ FREE: _____.

THE NUMBER ON THE ACTIVE CHAIN PLUS THE NUMBER ON THE FREE CHAIN SHOULD EQUAL THE TOTAL NUMBER OF TCB'S RESERVED.

WHAT IS THE TCB NUMBER OF THE CURRENTLY ACTIVE TASK? _____

WHAT IS THE TASK ID AND PRIORITY IN THE ACTIVE TASK'S TCB ?

_____

THIS IS THE DEFAULT TASK. PUT A BREAKPOINT AT 'INTWT' AND RUN ($R). WHAT ARE THE ACTIVE TCB'S NOW ? _____

ARE THERE ANY FREE TCB'S ? _____

SCAN THE ACTIVE CHAIN LOOKING AT THE ID/PRIORITY OF EACH TCB.

TCB NUMBER _____ _____ _____ _____

ID/PRI _____ _____ _____ _____ _____ _____ _____ _____

PUT A BREAKPOINT AT 'ACK'. FOR THE THREE TASKS INITIATED BY THE ?TASK CALL, FIND THE STACK POINTER VALUE OF THE TASK, AND THE VALUE STORED ON THE TOP OF ITS STACK.

TO FIND THESE VALUES, PERFORM THE FOLLOWING STEPS FOR EACH TASK:

PROCEED ($P).
ENTER A ?TRCON MESSAGE TO THE TASK.

THE PROGRAM SHOULD ENCOUNTER THE BREAKPOINT AT 'ACK'. DETERMINE WHICH TASK REACHED THE BREAKPOINT ($K - ACTIVE TASK). FIND THE TASK'S STACK POINTER ($E). FIND THE VALUE ON THE TASK'S STACK BY EXAMINING THE LOCATION POINTED TO BY THE STACK POINTER.

TASK ID _____ _____ _____

STACK PTR _____ _____ _____

VALUE ON STACK _____ _____ _____

THESE VALUES ARE POINTERS TO THE BYTE POINTER TABLE OF EACH TASK. WHAT ARE THE CONTENTS OF EACH OF TASK'S BYTE POINTER TABLE ?

EXAMPLE: SUPPOSE THE VALUE ON TOP OF TASK 3'S STACK IS 535.....

ENTER THE DEBUGGER COMMAND '535+ACKBP\' TO SEE THE 'ACK' BYTE POINTER VALUE IN TASK 3'S BYTE POINTER TABLE. THEN, TO SEE THIS BYTE POINTER EXPRESSED AS A WORD ADDRESS PLUS A BYTE INDICATOR BIT, PRESS FUNCTION-KEY-6.

TASK ID _____ _____ _____

ACKBP (AS WORD ADDR)_____ _____ _____

BUFFBP(AS WORD ADDR)_____ _____ _____

TNXBP (AS WORD ADDR)_____ _____ _____

USE THE DEBUGGER TO SEE THE MESSAGE POINTED TO BY EACH BUFFBP, ACKBP, AND TNXBP:

ENTER ASCII DISPLAY MODE WITH CNTRL-FUNCTION-KEY-7.
TO EXAMINE A BUFFER, USE ITS WORD ADDR FOLLOWED BY A BACKSLASH.
PRESS CARRIAGE RETURN TO SEE ADDITIONAL CHARACTERS IN THE BUFFER.
PRESS NEWLINE WHEN DONE WITH THE CURRENT BUFFER.

TO TERMINATE THE PROGRAM WHILE THE DEBUGGER HAS CONTROL, ENTER ESCAPE-Z. TO TERMINATE THE PROGRAM WHILE THE DEBUGGER DOES NOT HAVE CONTROL, USE CNTRL-C, CNTRL-A.

```
        .TITL   TRCON   ; MULTITASK USING CONSOLE MESSAGE MANAGER
        .ENT    START,INTWT,ALLST,LOOP,ACK,REPEAT,THANX
        .ENT MSTBL,TBL2,TBL3,TBL4,ACK2,ACK3,ACK4,TNX2,TNX3,TNX4
        .ENT    BUFF2,BUFF3,BUFF4,TDPKT,OUPKT,SBASE
        .ENT    ACKBP,BUFFBP,TNXBP

        .NREL   1
        BUFFSZ=80.      ; 80. BYTE BUFFERS
        STKSZ=60.       ; 30. DOUBLE WORD STACK/TASK

START:  ?OPEN   OUPKT   ; OPEN @OUTPUT
        DERR    1

        ?????   ?????   ; CREATE THREE NEW TASKS
        DERR    2

INTWT:  ??????          ; WAIT FOR CNTRL-C, CNTRL-A
        DERR    3

        WXOR    2,2     ; NORMAL TERMINATION
        ?RETURN
        DERR    4

; ***************************************************************
; *              ALL THE NEW TASKS START HERE                   *
; ***************************************************************

ALLST:  WPSH    2,2  ; SAVE THE MSG RCVD IN AC2

LOOP:   LDATS   2    ; GET THE ORIGINAL AC2 MSG (IT'S TBL ADDR)
        LWLDA   0,BUFFBP,2
                     ; GET THE BUFFER BP FROM THIS TASK'S TBL
        ?TRCON       ; READ A CONSOLE MESSAGE INTO THAT BUFFER
        DERR    5

        LWLDA   1,ACKBP,2  ; GET THE ACK BP FROM THIS TASK'S TBL
        LWSTA   1,OUPKT+?IBAD   ; STORE IT IN THE OUTPUT PACKET.
ACK:    ?WRITE  OUPKT  ; ACKNOWLEDGE RECEIPT OF THE CONSOLE MSG
        DERR    6

        LWSTA   0,OUPKT+?IBAD   ; STORE THE BUFFER BP IN THE PKT
REPEAT: ?WRITE  OUPKT   ; AND REPEAT THE MESSAGE THIS TASK RCVD.
        DERR    7

        LLEFB   3,NL*2
        LWSTA   3,OUPKT+?IBAD
        ?WRITE  OUPKT       ; PUT OUT A NEWLINE FOR READABILITY
        DERR    10

        LDATS   2               ; GET THE TBL ADDR AGAIN
        LWLDA   0,TNXBP,2  ; GET THE BP TO THIS TASK'S THANK YOU
        LWSTA   0,OUPKT+?IBAD   ; STORE IT IN THE OUTPUT PACKET
THANX:  ?WRITE  OUPKT           ; AND THANK THE USER FOR HIS MSG.
        DERR    11

        WBR     LOOP            ; DO IT ALL AGAIN.
```

```
        .NREL   0

MSTBL:  TBL2            ; TABLE OF AC2 MESSAGES FOR THE NEW TASKS
        TBL3            ; EACH MSG IS THE ADDRESS OF A SPECIFIC
        TBL4            ; TABLE OF BYTE POINTERS FOR EACH TASK

        ACKBP=0         ; SYMBOLIC OFFSETS INTO THE BP TABLES
        BUFFBP=ACKBP+2  ; DEFINE THE STRUCTURE OF THESE TABLES.
        TNXBP=BUFFBP+2

TBL2:   ACK2*2                  ; BP TABLE FOR TASK 2
        BUFF2*2
        TNX2*2

TBL3:   ACK3*2                  ; BP TABLE FOR TASK 3
        BUFF3*2
        TNX3*2

TBL4:   ACK4*2                  ; BP TABLE FOR TASK 4
        BUFF4*2
        TNX4*2

**      .NOLOC  1
ACK2:   .TXT    /This is TASK 2 - I have received your message
which said:<12>/
ACK3:   .TXT    /<7><7><7>HELLO...TASK 3 HERE...I THOUGHT I HEARD
YOU SAY:<12>/
ACK4:   .TXT    /TEN-FOUR GOOD BUDDY!! TASK 4 COPIED YOUR LAST
MESSAGE AS:<12>/

TNX2:   .TXT    /TASK 2 thanks you for your interest and awaits
your next
 message.<12>/
TNX3:   .TXT    /...THANK YOU VERY MUCH...SINCERELY, TASK
3<7><7><7><12>/
TNX4:   .TXT    /TASK 4 IS TEN-EIGHT, TEN-TEN ON THE CHANNEL FOR
YOU !<12>/
**      .NOLOC  0

NL:     .TXT    /<012>/
BUFF2:  .BLK    BUFFSZ/2
BUFF3:  .BLK    BUFFSZ/2
BUFF4:  .BLK    BUFFSZ/2
```

```
        ; TASK DEFINITION PACKET

TDPKT:  .LOC    TDPKT+?DLNK     ; EXTENDED OR NORMAL PACKET
        .DWORD  ??              ; NORMAL
        .LOC    TDPKT+?DLNKB    ; RESERVED
        .DWORD  ??
        .LOC    TDPKT+?DPRI     ; PRIORITY OF NEW TASK(S)
        .WORD   ??
        .LOC    TDPKT+?DID      ; ID OF FIRST NEW TASK
        .WORD   ??
        .LOC    TDPKT+?DPC      ; START ADDRESS
        ??????  ?????           ; ALL NEW TASKS EXECUTE SAME CODE
        .LOC    TDPKT+?DAC2     ; MESSAGE TO NEW TASK'S AC2
        ??????  ?????           ; TABLE OF MSG TO EACH NEW TASK
        .LOC    TDPKT+?DSTB     ; STACK BASE
        .DWORD  SBASE
        .LOC    TDPKT+?DSSZ     ; STACK SIZE FOR EACH TASK
        ??????  ?????
        .LOC    TDPKT+?DSFLT    ; STACK FAULT ROUTINE
        .WORD   ??              ; DEFAULT
        .LOC    TDPKT+?DFLGS    ; FLAGS
        .WORD   0               ; NONE
        .LOC    TDPKT+?DRES     ; RESERVED
        .WORD   0
        .LOC    TDPKT+?DNUM     ; NUMBER OF TASKS TO START
        .WORD   ??
        .LOC    TDPKT+?DSLTH    ; ?DSLTH=PACKET LENGTH
```

```
        ; PARAMETER PACKET FOR FILE @OUTPUT

OUPKT:  .LOC    OUPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?ISTI     ; FILE SPECS
        .WORD   ?ICRF+?RTDS+?OFOT       ; CHG TO DATASEN, OUTPUT
        .LOC    OUPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IMRS     ; PHYS BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    OUPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  -1              ; SET AT ?WRITE TIME
        .LOC    OUPKT+?IRES     ; RESERVED
        .WORD   0
        .LOC    OUPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   BUFFSZ
        .LOC    OUPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    OUPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  ATOUT*2
        .LOC    OUPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    OUPKT+?IOSZ     ; ?IOSZ=PACKET LENGTH

ATOUT:  .TXT    /@OUTPUT/

SBASE:  .BLK    ???????

        .END    START
```

ASSEMBLY LANGUAGE MULTITASKING LAB

MOVE/R/NACL INTO YOUR OWN DIRECTORY A COPY OF THE FOLLOWING FILES FROM THE :S309VS:ASSY DIRECTORY:

MULTITASK.SR
COPY.SR
ERROR.OB

THE PROGRAM ACCEPTS TWO TO (ARGMX) ARGUMENTS IN THE COMMAND LINE. THERE MUST BE AN EVEN NUMBER OF ARGUMENTS. IN EACH PAIR OF ARGUMENTS, THE FIRST IS CONSIDERED TO BE AN INPUT FILE, WHILE THE SECOND IS AN OUTPUT FILE. FOR EACH PAIR OF FILES, A TASK IS CREATED WHICH COPIES, LINE BY LINE, FROM THE INPUT FILE TO THE OUTPUT FILE. EACH TASK, WHEN THE COPY IS COMPLETE, SENDS AN INTERTASK MSG TO THE MAIN TASK AND THEN KILLS ITSELF. WHEN THE MAIN TASK RECEIVES THE MSG, IT NOTIFIES THE USER THAT THIS PARTICULAR COPY OPERATION IS COMPLETED. WHEN ALL THE COPIES ARE COMPLETE, THE PROGRAM TERMINATES.

REFER TO THE PRINTOUTS OF MULTITASK.SR AND COPY.SR ON THE FOLLOWING PAGES AND DETERMINE THE CORRECT ENTRIES FOR THE FOLLOWING ITEMS:

PAGE	LINE	MULTITASK.SR
1	23	INSERT THE ASSEMBLER PSUEDO-OP TO RESERVE THE CORRECT NUMBER OF TCB'S.
4	26	STORE THE NUMBER OF TASKS TO BE CREATED INTO THE PACKET.
4	31	STORE THE ADDRESS OF THE TASK INITIAL MESSAGE OR TABLE INTO THE PACKET.
4	33	INSERT THE SYSTEM CALL TO CREATE THE TASKS.
5	4-5	INSERT THE CODE NEEDED TO WAIT FOR A MESSAGE FROM ANOTHER TASK. (WHERE WILL THE MESSAGE BE FOUND AFTER RECEIPT?)
5	15	GET THE TABLE ADDRESS FROM THE INTERTASK MESSAGE.
5	26	GET THE TABLE ADDRESS FROM THE INTERTASK MESSAGE.
5	37	INSERT CODE TO DECREMENT THE NUMBER OF TASKS REMAINING
7	4-28	USING THE COMMENTS AS A GUIDE, INSERT THE CORRECT VALUES IN PLACE OF THE QUESTION MARKS IN THE TASK DEFINITION PACKET. FOR THE STACK ITEMS, REFER TO THE SYMBOLS DEFINED ON LINE 30 AND ON PAGE 1 LINE 25.
7	13	ALL THE NEW TASKS ARE TO EXECUTE THE SAME CODE. THIS CODE IS FOUND IN THE SEPERATELY ASSEMBLED MODULE 'COPY.SR'. INSERT THE ADDRESS OF THE ENTRY POINT OF THIS ROUTINE.
7	30	DEFINE AN AREA LARGE ENOUGH TO HOLD ALL THE STACKS REQUIRED BY THE TASKS. USE A SYMBOLIC EXPRESSION TO SPECIFY THE NUMBER OF WORDS TO RESERVE.

PAGE	LINE	COPY.SR
1	23,24	INSERT THE CODE NEEDED TO GET THE BYTEPOINTERS TO THE FILENAMES TO BE ASSOCIATED WITH THIS PARTICULAR TASK. (HOW DOES EACH TASK MAINTAIN A SEPERATE IDENTITY, KNOWING WHICH FILES TO WORK WITH, EVEN THOUGH ALL THE TASKS EXECUTE THE IDENTICAL CODE ?) KEEP IN MIND THAT EACH BYTEPOINTER ENTRY IS TWO WORDS LONG.
3	32-35	INSERT THE CODE NEEDED FOR THE CURRENT TASK TO TRANSMIT A MESSAGE TO THE MAILBOX, AND THEN KILL ITSELF. KEEP IN MIND THAT THE MAILBOX MIGHT ALREADY BE IN USE.
3	37	INSERT THE CODE NEEDED TO CHECK FOR THE EXPECTED BOX-IN-USE ERROR.

USE AN EDITOR TO MAKE THE CHANGES IN BOTH SOURCE FILES. WHEN YOU HAVE SUCCESSFULLY ASSEMBLED BOTH MODULES, LINK THEM TOGETHER WITH THE 'ERROR' MODULE.

NOW YOU ARE READY TO TEST YOUR PROGRAM. THE PROGRAM IS DESIGNED TO BE INVOKED WITH A COMMAND LINE OF THE FOLLOWING FORMAT:

X, MULTITASK, infile1, outfile1 [, infile2, outfile2...]

WHERE THE infile'S ALREADY EXIST AND THE outfile'S ARE TO BE CREATED BY THE PROGRAM. THERE MUST BE AN EVEN NUMBER OF ARGUMENTS. WHAT IS THE MAXIMUM NUMBER OF ARGUMENTS PERMITTED?

_____

TRY YOUR PROGRAM WITH FOUR ARGUMENTS. YOU MIGHT USE THE NAMES OF YOUR SOURCE FILES FOR THE infile'S AND PERHAPS THE SAME FILENAMES WITH '.CPY' APPENDED FOR THE outfile'S.

YOU SHOULD GET CONSOLE MESSAGES STATING THE COMPLETION OF EACH COPY OPERATION. WHICH COPY FINISHES FIRST ?

_____

DOES THE ORDER OF THE COPY OPERATIONS SPECIFIED IN THE COMMAND LINE MAKE ANY DIFFERENCE IN THE ORDER OF COMPLETION ? (TRY IT.)_____

YOU CAN ALSO TEST YOUR PROGRAM USING "@NULL" AS ANY (OR ALL) OF THE infile'S AND/OR outfile'S.

TRY YOUR PROGRAM WITH VARIOUS NUMBERS OF ARGUMENTS. IT SHOULD ABORT IF THE NUMBER IS ODD, LESS THAN TWO, OR MORE THAN (ARGMX). DOES IT WORK CORRECTLY (ABORT PROPERLY OR COPY PROPERLY) FOR:

NO ARGUMENTS _____
ONE ARGUMENT _____
TWO ARGUMENTS _____
THREE ARGUMENTS _____
FOUR ARGUMENTS _____
FIVE ARGUMENTS _____
SIX ARGUMENTS _____
SEVEN ARGUMENTS _____
EIGHT ARGUMENTS _____

IF YOU HAVE CORRECTLY EDITED, ASSEMBLED, AND LINKED YOUR PROGRAM, ALL OF THE CASES ABOVE SHOULD FUNCTION CORRECTLY.

```
        .TITL   MULTI

; PROGRAM ACCEPTS TWO TO (ARGMX) ARGUMENTS IN THE COMMAND LINE.
; THERE MUST BE AN EVEN NUMBER OF ARGUMENTS.
; IN EACH PAIR OF ARGUMENTS, THE FIRST IS CONSIDERED TO BE AN
; INPUT FILE, WHILE THE SECOND IS AN OUTPUT FILE.
; FOR EACH PAIR OF FILES, A TASK IS CREATED WHICH COPIES,
; LINE BY LINE, FROM THE INPUT FILE TO THE OUTPUT FILE.
; EACH TASK, WHEN THE COPY IS COMPLETE, SENDS AN INTERTASK MSG
; TO THE MAIN TASK AND THEN KILLS ITSELF.
; WHEN THE MAIN TASK RECEIVES THE MSG, IT NOTIFIES THE USER THAT
; THIS PARTICULAR COPY OPERATION IS COMPLETED.
; WHEN ALL THE COPIES ARE COMPLETE, THE PROGRAM TERMINATES.

        .ENT    BOX,ARGCNT,TATBL,T2TBL,T3TBL,T4TBL
        .ENT    START,ARGLUP,GETNM,CTSKS,TSKS
        .ENT    WAIT,NOTIFY,NFIN,NM2,NFOU,NM3,WTMORE,DONE,BADARG
        .ENT    GCPKT,GAPKT,TDPKT,OUPKT
        .EXTL   COPY

        ARGMX=6                 ; MAX NUMBER OF ARGUMENTS
        ????    ARGMX/2+1       ; NUMBER OF TCB'S
        BUFFLN=136.             ; BUFFER LENGTH (BYTES)
        STKSZ=(?IOSZ*2)+BUFFLN+60.    ; STACK SIZE FOR EACH TASK
```

```
        .NREL   0

ARGCNT: 0                       ; NUMBER OF ARGUMENTS FOUND
BOX:    0                       ; INTERTASK MAILBOX

; TABLE OF MESSAGES TO EACH NEW TASK'S AC2
; EACH MESSAGE IS THE ADDRESS OF
; A TABLE OF TWO FILENAME BYTEPOINTERS.

TATBL:  T2TBL                   ; MESSAGE FOR TASK ID 2
        T3TBL                   ; MESSAGE FOR TASK ID 3
        T4TBL                   ; MESSAGE FOR TASK ID 4

; TABLE OF FILENAME BYTEPOINTERS
; THE TABLE IS ACCESSED TO FILL IN THE FILENAMES
; FROM THE COMMAND LINE ARGUMENTS.
; THEN THE TABLE IS ACCESSED BY EACH TASK
; THRU ITS AC2 MESSAGE AS IT BEGINS EXECUTION.

FNTBL:
T2TBL:  IN2NM*2                 ; FILENAME IS ARGUMENT 1
        OU2NM*2                 ; FILENAME IS ARGUMENT 2
T3TBL:  IN3NM*2                 ; FILENAME IS ARGUMENT 3
        OU3NM*2                 ; FILENAME IS ARGUMENT 4
T4TBL:  IN4NM*2                 ; FILENAME IS ARGUMENT 5
        OU4NM*2                 ; FILENAME IS ARGUMENT 6

IN2NM:  .BLK    ?MXPL/2
OU2NM:  .BLK    ?MXPL/2
IN3NM:  .BLK    ?MXPL/2
OU3NM:  .BLK    ?MXPL/2
IN4NM:  .BLK    ?MXPL/2
OU4NM:  .BLK    ?MXPL/2
```

```
        .NREL   1

BADARG: LLEFB   1,BADMSG*2      ; GET BP TO MSG
        XWLDA   2,FLAGS
        ?RETURN                 ; TERMINATE WITH MSG
        DERR    1

FLAGS:  ?RFER+?RFCF+BADLN
BADMSG: .TXT    "WRONG NUMBER OF ARGUMENTS"
BADLN=.-BADMSG*2                ; LENGTH OF BADMSG (BYTES)


START:  ?OPEN   OUPKT           ; OPEN CONSOLE FOR OUTPUT
        DERR    2
        ?GTMES  GCPKT           ; GET ARGUMENT COUNT
        DERR    3
        LWSTA   0,ARGCNT        ; STORE IT.
        WCLM    0,0             ; 2 < ARGCNT < ARGMX ??
        2
        ARGMX
        WBR     BADARG          ; NO.
        WSKBZ   31.             ; YES. IS ARGCNT EVEN ?
        WBR     BADARG          ; NO.
```

```
; STARTING WITH THE FIRST ARGUMENT, GET EACH ONE INTO THE
; THE ASSOCIATED FILENAME BUFFER.
; AC1 IS ARGUMENT NUMBER,  AC2 -2 IS OFFSET
; IN FNTBL (2 WD ENTRIES)

        NLDAI   1.,1

ARGLUP: WMOV    1,2
        WADD    1,2          ; TBL OFFSET IS TWICE ARG NUM (-2)
        LWLDA   0,FNTBL-2,2  ; GET BP TO CURRENT FILENAME BUFFER
        LWSTA   0,GAPKT+?GRES   ; STORE IT IN GET-ARG PKT.
        LNSTA   1,GAPKT+?GNUM   ; UPDATE ARG NUMBER.

GETNM:  ?GTMES  GAPKT           ; GET FILENAME ARGUMENT
        DERR    4               ; INTO FNAME BUFFER.
        LNLDA   1,GAPKT+?GNUM   ; GET ARG NUMBER
        WINC    1,1             ; POINT TO NEXT ARG
        LWLDA   0,ARGCNT        ; ANY MORE ARGS ??
        WSGT    1,0
        WBR     ARGLUP          ; YES.

; NOW CREATE ONE TASK FOR EACH PAIR OF ARGUMENTS

CTSKS:  LWLDA   0,ARGCNT        ; GET NUMBER OF ARGUMENTS
        WASHI   -1,0            ; DIVIDE BY TWO
        LNSTA   0,?????+?????   ; THIS IS NUMBER OF TASKS.
        LLEF    1,T2TBL         ; GET MSG ITSELF FOR ONE TASK.
        WSGTI   1,0          ; IS THERE MORE THAN ONE NEW TASK?
        WBR     STMSG           ; NO.
        LLEF    1,TATBL      ; YES, GET ADDR OF TBL OF TASK MSGS
STMSG:  LWSTA   1,?????+?????   ; STORE IN TASK DEFINITION PKT.

TSKS:   ?????   ?????           ; CREATE THE TASKS.
        DERR    5
```

```
; WAIT FOR A TASK TO FINISH

WAIT:   LLEF    ??,BOX          ; GET MAILBOX ADDR
        ????                    ; WAIT FOR A MESSAGE
        DERR    6

; NOTIFY THE USER THAT THE TASK HAS FINISHED

NOTIFY: LLEFB   3,MSG1*2        ; GET BP TO FIRST PART OF MSG
        LWSTA   3,OUPKT+?IBAD
        ?WRITE  OUPKT           ; WRITE MSG SEGMENT
        DERR    7

        WMOV    ??,3            ; GET TBL ADDR FROM TASK MSG
        XWLDA   0,0,3           ; GET BP TO INPUT FILENAME
        LWSTA   0,OUPKT+?IBAD   ; STORE IN @OUTPUT PKT.
NFIN:   ?WRITE  OUPKT           ; WRITE MSG SEGMENT
        DERR    10

        LLEFB   3,MSG2*2        ; GET BP TO SECOND PART OF MSG
        LWSTA   3,OUPKT+?IBAD
NM2:    ?WRITE  OUPKT           ; WRITE MSG SEGMENT
        DERR    11

        WMOV    ??,3            ; GET TBL ADDR FROM TASK MSG
        XWLDA   0,2,3           ; GET BP TO OUTPUT FILENAME
        LWSTA   0,OUPKT+?IBAD   ; STORE IN @OUTPUT PKT.
NFOU:   ?WRITE  OUPKT           ; WRITE MSG SEGMENT
        DERR    12

        LLEFB   3,MSG3*2        ; GET BP TO LAST PART OF MSG
        LWSTA   3,OUPKT+?IBAD
NM3:    ?WRITE  OUPKT           ; WRITE MSG SEGMENT
        DERR    13

WTMORE: LNDSZ   ?????+?????     ; DECR NUMBER TASKS REMAINING
        WBR     WAIT            ; IF THERE ARE MORE TASKS, LOOP

DONE:   WSUB    2,2             ; ALL TASKS DONE,
        ?RETURN                 ; TERMINATE THE PROGRAM
        DERR    14



**      .NOLOC  1
MSG1:   .TXT    /COPY COMPLETED: FILE "/
MSG2:   .TXT    /" TO FILE "/
MSG3:   .TXT    /"<12>/
**      .NOLOC  0
```

```
        .NREL   0

        ; PACKET TO GET ARGUMENT COUNT

GCPKT:  .LOC    GCPKT+?GREQ     ; REQUEST TYPE
        .WORD   ?GCNT           ; ARGUMENT COUNT
        .LOC    GCPKT+?GNUM     ; ARGUMENT NUMBER
        .WORD   0               ; (NOT USED)
        .LOC    GCPKT+?GSW      ; SWITCH BYTE PTR
        .DWORD  0               ; (NOT USED)
        .LOC    GCPKT+?GRES     ; BUFFER BYTE PTR
        .DWORD  -1              ; (NOT USED)
        .LOC    GCPKT+?GTLN     ; (?GTLN=PKT LENGTH)

        ; PACKET TO GET AN ARGUMENT

GAPKT:  .LOC    GAPKT+?GREQ     ; REQUEST TYPE
        .WORD   ?GARG           ; GET ARGUMENT
        .LOC    GAPKT+?GNUM     ; ARGUMENT NUMBER
        .WORD   -1              ; SPECIFY AT ?GTMES TIME
        .LOC    GAPKT+?GSW      ; SWITCH BYTE PTR
        .DWORD  0               ; (NOT USED)
        .LOC    GAPKT+?GRES     ; BUFFER BYTE PTR
        .DWORD  -1              ; SPECIFY AT ?GTMES TIME
        .LOC    GAPKT+?GTLN     ; (?GTLN=PKT LENGTH)
```

```
        ; TASK DEFINITION PACKET

TDPKT:  .LOC    TDPKT+?DLNK     ; EXTENDED OR NORMAL PACKET
        .DWORD  ??              ; NORMAL
        .LOC    TDPKT+?DLNKB    ; RESERVED
        .DWORD  ??
        .LOC    TDPKT+?DPRI     ; PRIORITY OF NEW TASK(S)
        .WORD   ??
        .LOC    TDPKT+?DID      ; ID OF FIRST NEW TASK
        .WORD   ??
        .LOC    TDPKT+?DPC      ; START ADDRESS
        ??????  ????            ; ALL TASKS EXECUTE SAME CODE
        .LOC    TDPKT+?DAC2     ; MESSAGE TO NEW TASK'S AC2
        ??????  ??              ; SPECIFY AT ?TASK TIME
        .LOC    TDPKT+?DSTB     ; STACK BASE
        ??????  ?????
        .LOC    TDPKT+?DSFLT    ; STACK FAULT ROUTINE
        ?????   ??              ; DEFAULT
        .LOC    TDPKT+?DSSZ     ; STACK SIZE FOR EACH TASK
        ??????  ?????
        .LOC    TDPKT+?DFLGS    ; FLAGS
        .WORD   ??              ; NONE
        .LOC    TDPKT+?DRES     ; RESERVED
        .WORD   ??
        .LOC    TDPKT+?DNUM     ; NUMBER OF TASKS TO START
        .WORD   ??              ; DEPENDS ON NUMBER OF ARGS
        .LOC    TDPKT+?DSLTH    ; ?DSLTH=PACKET LENGTH

STACK:  .BLK    ?????*?????/?
```

```
OUPKT:  .LOC    OUPKT+?ICH      ; CHANNEL NUMBER
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?ISTI     ; FILE SPECS
        .WORD   ?ICRF+?RTDS+?OFOT       ; CHG TO DATASEN, OUTPUT
        .LOC    OUPKT+?ISTO     ; FILE TYPE
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IMRS     ; PHYS BLOCK SIZE
        .WORD   -1              ; DEFAULT
        .LOC    OUPKT+?IBAD     ; BUFFER BYTE POINTER
        .DWORD  -1              ; SPECIFY AT ?WRITE TIME
        .LOC    OUPKT+?IRES     ; RESERVED
        .WORD   0
        .LOC    OUPKT+?IRCL     ; MAX RECORD LENGTH
        .WORD   BUFFLN
        .LOC    OUPKT+?IRLR     ; RECORD LENGTH
        .WORD   0               ; RETURNED BY SYS
        .LOC    OUPKT+?IRNH     ; RECORD NUMBER
        .DWORD  0
        .LOC    OUPKT+?IFNP     ; FILENAME BYTE POINTER
        .DWORD  CONNM*2
        .LOC    OUPKT+?IDEL     ; DELIMITER TABLE
        .DWORD  -1              ; DEFAULT
        .LOC    OUPKT+?IOSZ     ; ?IOSZ=PACKET LENGTH

CONNM:  .TXT    /@OUTPUT/

        .END    START
```

```
        .TITL   COPY

; MODULE FOR A TASK TO COPY FROM INPUT FILE TO OUTPUT FILE
; AND SEND BACK A MESSAGE WHEN DONE.

        .ENT    COPY,OPENF,CLOOP,RDERR,TDONE,DIE,XMERR
        .EXTL   BOX

        .NREL   1

        BUFSZ = 136.            ; BUFFER SIZE (BYTES)
        INFPK = 2               ; INPUT PKT STACK OFFSET
        OUFPK = INFPK+?IOSZ     ; OUTPUT PKT STACK OFFSET
        BUFF  = OUFPK+?IOSZ     ; BUFFER STACK OFFSET

; AC2 IS ADDR OF TWO ENTRY TABLE.
; EACH ENTRY IS TWO WORDS.
; FIRST ENTRY IS BP TO INPUT FILENAME.
; SECOND ENTRY IS BP TO OUTPUT FILENAME.

COPY:   WSSVR   ((?IOSZ*2)+(BUFSZ/2))/2
                                ; STACK SPACE FOR 2 PKTS + BUFFER
        XWLDA   1,0,??                  ; GET INPUT FILENAME BP
        XWLDA   2,2,??                  ; GET OUTPUT FILENAME BP
```

```
; SET UP INPUT PKT ON STACK

        NLDAI   0.,0            ; RETURNED BY SYSTEM
        XNSTA   0,INFPK+?ICH,3  ; CHANNEL NUMBER
        WLDAI   ?ICRF+?RTDS+?OFIN,0     ; CHG TO DATASENS, INPUT
        XNSTA   0,INFPK+?ISTI,3 ; FILE SPECS
        NLDAI   0.,0            ; RETURNED BY SYSTEM
        XNSTA   0,INFPK+?ISTO,3 ; FILE TYPE
        NLDAI   -1.,0           ; DEFAULT
        XNSTA   0,INFPK+?IMRS,3 ; PHYSICAL BLOCK SIZE
        XLEFB   0,BUFF*2,3      ; BP TO BUFFER ON STACK
        XWSTA   0,INFPK+?IBAD,3 ; BUFFER BYTE POINTER
        NLDAI   0.,0
        XNSTA   0,INFPK+?IRES,3 ; RESERVED
        NLDAI   BUFSZ,0         ; BUFFER LENGTH
        XNSTA   0,INFPK+?IRCL,3 ; MAX RECORD LENGTH
        NLDAI   0.,0            ; RETURNED BY SYSTEM
        XNSTA   0,INFPK+?IRLR,3 ; RECORD LENGTH RETURNED
        XNSTA   0,INFPK+?IRNW,3 ; RESERVED
        NLDAI   0.,0            ; NEXT SEQUENTIAL RECORD
        XWSTA   0,INFPK+?IRNH,3 ; RECORD NUMBER
        XWSTA   1,INFPK+?IFNP,3 ; STORE INPUT FILENAME BP IN PKT
        NLDAI   -1.,0           ; DEFAULT
        XWSTA   0,INFPK+?IDEL,3 ; DELIMITER TABLE

; SET UP OUTPUT PKT ON STACK

        NLDAI   0.,0            ; RETURNED BY SYSTEM
        XNSTA   0,OUFPK+?ICH,3  ; CHANNEL NUMBER
        WLDAI   ?ICRF+?RTDS+?OFCE+?OFCR+?OFOT,0
                ; CHG TO DATASENSITIVE, RECREATE FOR OUTPUT
        XNSTA   0,OUFPK+?ISTI,3
        NLDAI   0.,0            ; DEFAULT
        XNSTA   0,OUFPK+?ISTO,3 ; FILE TYPE
        NLDAI   -1.,0           ; DEFAULT
        XNSTA   0,OUFPK+?IMRS,3 ; PHYSICAL BLOCK SIZE
        XLEFB   0,BUFF*2,3      ; BP TO BUFFER ON STACK
        XWSTA   0,OUFPK+?IBAD,3 ; BUFFER BYTE POINTER
        NLDAI   0.,0
        XNSTA   0,OUFPK+?IRES,3 ; RESERVED
        NLDAI   BUFSZ,0         ; BUFFER LENGTH
        XNSTA   0,OUFPK+?IRCL,3 ; MAX RECORD LENGTH
        NLDAI   0.,0            ; RETURNED BY SYSTEM
        XNSTA   0,INFPK+?IRLR,3 ; RECORD LENGTH RETURNED
        XNSTA   0,INFPK+?IRNW,3 ; RESERVED
        NLDAI   0.,0            ; NEXT SEQUENTIAL RECORD
        XWSTA   0,OUFPK+?IRNH,3 ; RECORD NUMBER
        XWSTA   2,OUFPK+?IFNP,3 ; STORE OUTPUT FILENAME BP IN PKT
        NLDAI   -1.,0           ; DEFAULT
        XWSTA   0,OUFPK+?IDEL,3 ; DELIMITER TABLE
```

```
OPENF:  XLEF    2,INFPK,3       ; GET ADDR OF INPUT PKT
        ?OPEN                   ; OPEN INPUT FILE
        DERR    21

        XLEF    2,OUFPK,3       ; GET ADDR OF OUTPUT PKT
        ?OPEN                   ; OPEN OUTPUT FILE
        DERR    22

CLOOP:  XLEF    2,INFPK,3       ; GET ADDR OF INPUT PKT
        ?READ                   ; READ A LINE
        WBR     RDERR

        XLEF    2,OUFPK,3       ; GET ADDR OF OUTPUT PKT
        ?WRITE                  ; WRITE THE LINE
        DERR    23
        WBR     CLOOP

RDERR:  WSEQI   EREOF,0         ; GOT A ?READ ERROR
        DERR    24              ; WAS IT END-OF-FILE ?

TDONE:  XLEF    2,INFPK,3       ; GET ADDR OF INPUT PKT
        ?CLOSE                  ; CLOSE INPUT FILE
        DERR    25

        XLEF    2,OUFPK,3       ; GET ADDR OF OUTPUT PKT
        ?CLOSE                  ; CLOSE OUTPUT FILE
        DERR    26

        XWLDA   1,?OAC2,3       ; YES. RETRIEVE TABLE ADDR
TBOX:   LLEF    ??,BOX          ; GET MAILBOX ADDR
        ????                    ; SEND TBL ADDR AS MSG
        WBR     XMERR
DIE:    ?????                   ; SUICIDE

XMERR:  WSEQI   ?????,0         ; GOT A ?XMT ERROR
        DERR    27              ; WAS IT BOX-IN-USE ?
        WBR     TBOX            ; YES. TRY AGAIN.

        .END
```

FORTRAN 77 FILE SYSTEM LAB

* MOVE/R/NACL COPIES OF THE FILES 'CREATE.F77', 'FILES_QSYM.F77.IN', AND 'BLOCKIO_QSYM.F77.IN' FROM THE :S309VS:F77 DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'CREATE' IS TO FIRST CREATE A DIRECTORY, THE NAME OF WHICH WILL BE "MYDIR". THE PROGRAM SHOULD NOT FAIL IF THE DIRECTORY ALREADY EXISTS. IT WILL THEN MAKE "MYDIR" THE WORKING DIRECTORY. (IF "MYDIR" IS NOT A DIRECTORY THEN THE PROGRAM WILL FAIL AND YOU WILL HAVE TO DELETE "MYDIR"). THE PROGRAM WILL THEN ATTEMPT TO DELETE AND CREATE A NEW FILE CALLED "MYFILE".

* REFER TO THE PRINTOUT OF 'CREATE.F77' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM CALLS AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
F77,CREATE
F77LINK,CREATE
```

* GET A HARDCOPY OF THE FILE 'CREATE.LS'.

* EXECUTE THE CREATE PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'FILE CREATION COMPLETED'.

* DO A FILESTATUS WITH /TYPE ON THE FILE 'MYDIR'. IT SHOULD BE TYPE "DIR".

* DO A DIRECTORY TO "MYDIR" THEN DO ANOTHER FILESTATUS/TYPE. YOU SHOULD FIND THAT "MYFILE" IS A TYPE "TXT".

* GO BACK TO YOUR PREVIOUS DIRECTORY AND EXECUTE THE PROGRAM AGAIN. THE PROGRAM SHOULD NOT FAIL AS IT WILL DELETE 'MYFILE' BEFORE TRYING TO CREATE IT AGAIN.

* YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE PROGRAM REPEATEDLY WITHOUT ERROR.

```
        PROGRAM CREATE

%LIST(OFF)
%INCLUDE 'BLOCKIO_QSYM.F77.IN'
%INCLUDE 'FILES_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 CREPK(0:ISYS_CLTH-1)   ! FILE CREATION PKT
        INTEGER*2 DIRPK(0:ISYS_CLTH-1)   ! DIRECTORY CREATION PKT

C   DOUBLE WORDS

        INTEGER*4 CREPK_CTIM,CREPK_CACP ! IN FILE CREATION PKT
        EQUIVALENCE (CREPK(ISYS_CTIM),CREPK_CTIM)
        EQUIVALENCE (CREPK(ISYS_CACP),CREPK_CACP)

        INTEGER*4 DIRPK_CTIM,DIRPK_CACP,DIRPK_CMSH !IN DIRECTORY
                                                   ! CREATION PKT
        EQUIVALENCE (DIRPK(ISYS_CTIM),DIRPK_CTIM)
        EQUIVALENCE (DIRPK(ISYS_CACP),DIRPK_CACP)
        EQUIVALENCE (DIRPK(ISYS_CMSH),DIRPK_CMSH)

        CHARACTER*10 FNAME
        CHARACTER*10 DNAME
        CHARACTER*80 BUFFER

        FNAME = 'MYFILE<0>'
        DNAME = 'MYDIR<0>'

C   PKT INITIALIZATION

C   FILE CREATION PKT
        CREPK(ISYS_CFTYP) = ISYS_????    ! TYPE TEXT, NO RCD FMT
        CREPK(ISYS_CCPS) = 0             ! FIXED REC SIZE - IGNORE
        CREPK_CTIM = ??                  ! CURRENT DATE AND TIME
        CREPK_CACP = ??                  ! DEFAULT ACL
        CREPK(ISYS_CDEH) = 0             ! RESERVED
        CREPK(ISYS_CDEL) = ??            ! DEFAULT ELEMENT SIZE
        CREPK(ISYS_CMIL) = ??            ! DEFAULT MAX INDEX LEVEL
        CREPK(ISYS_CMRS) = 0             ! RESERVED

C   DIRECTORY CREATION PKT
        DIRPK(ISYS_CFTYP) = ISYS_????    ! TYPE = DIR
        DIRPK(ISYS_CHFS) = ??            ! DEFAULT HASH FRAME SIZE
        DIRPK_CTIM = ??                  ! CURRENT DATE AND TIME
        DIRPK_CACP = ??                  ! DEFAULT ACL
        DIRPK_CMSH = 0                   ! MAX SIZE IF CPD, ELSE 0
        DIRPK(ISYS_CMIL) = ??            ! DEFAULT MAX INDEX LEVEL
        DIRPK(ISYS_CMRS) = 0             ! RESERVED
```

```
C    PROGRAM

C    CREATE DIRECTORY "MYDIR"
         IAC0 = BYTEADDR(DNAME)          ! NAME OF DIRECTORY
         IAC1 = 0                        ! RESERVED
         IAC2 = ????????(?????)
         IER = ISYS(ISYS_??????,IAC0,IAC1,IAC2)
         IF (IER.EQ.ISYS_ERNAE) GOTO 100 ! IF FILE NAME EXISTS,
                                         ! IGNORE ERROR
         IF (IER.NE.0) GOTO 999

C    CHANGE WORKING DIRECTORY
100      IAC0 = BYTEADDR(DNAME)          ! NAME OF DIRECTORY
         IAC1 = 0                        ! RESERVED
         IAC2 = 0                        ! RESERVED
         IER = ISYS(ISYS_???,IAC0,IAC1,IAC2)
         IF (IER.NE.0) GOTO 999

C    TRY TO DELETE THE FILE
         IAC0 = BYTEADDR(FNAME)          ! NAME OF FILE
         IER = ISYS(ISYS_??????,IAC0,IAC1,IAC2)
         IF (IER.EQ.ISYS_ERFDE) GOTO 200
                                  ! IGNORE FILE DOESN'T EXIST
         IF (IER.NE.0) GOTO 999

C    CREATE THE FILE
200      IAC0 = BYTEADDR(FNAME)          ! NAME OF FILE
         IAC1 = 0                        ! RESERVED
         IAC2 = ????????(?????)          ! FILE CREATION PKT
         IER = ISYS(ISYS_??????,IAC0,IAC1,IAC2)
         IF (IER.NE.0) GOTO 999

C    GOOD END
         BUFFER = 'File creation completed.<NL>'
         IAC1 = BYTEADDR(BUFFER)
         IAC2 = ISYS_RFCF+LEN('FILE CREATION COMPLETED.<NL>')
         IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
         IF (IER.NE.0) GOTO 999

C    BAD END
999      IAC2 = ISYS_RFEC+ISYS_RFCF+ISYS_RFAB
         IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
         IF (IER.NE.0) GOTO 999

         END
```

FORTRAN 77 FILE ACCESS LAB

* MOVE/R/NACL A COPY OF THE FILE 'CHANGEACL.F77' FROM THE :S309VS:F77 DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'CHANGEACL' IS TO READ THE ACL OF YOUR FILE CALLED 'MYFILE' IN YOUR SUBDIRECTORY 'MYDIR' AND THEN TO WRITE A NEW ACL FOR THE FILE, ADDING '+,RE' ACCESS TO THE PREVIOUS STRING.

NOTE: IF YOU HAVE NOT DONE THE 'CREATE_LAB' BY THIS TIME YOU WILL HAVE TO CREATE THE NECESSARY FILES FOR THIS LAB AS FOLLOWS:

```
) CR/DIR,MYDIR
) CR,MYDIR:MYFILE
```

* REFER TO THE PRINTOUT OF 'CHANGEACL.F77' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUE. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS, THE NEW ACL STRING, AND THE VALUES YOU WILL NEED FOR SETTING UP YOUR ACCUMULATORS.

* AFTER UPDATING THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
F77,CHANGEACL
F77LINK,CHANGEACL
```

* GET A HARDCOPY OF THE FILE 'CHANGEACL.LS'.

* DO AN ACL WITH /V ON THE FILE 'MYDIR:MYFILE'.
IT SHOULD BE YOUR DEFAULT ACL (USERNAME,OWARE).

* EXECUTE THE CHANGEACL PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'THE FILE'S ACL HAS BEEN CHANGED.'

* DO ANOTHER ACL WITH /V ON THE FILE 'MYDIR:MYFILE'. IT SHOULD NOW BE YOUR DEFAULT ACL AND '+,RE'.

* YOU HAVE COMPLETED THIS LAB WHEN MYFILE:MYDIR HAS THE ACCESS CONTROL LIST 'USERNAME,OWARE +,RE'.

```
        PROGRAM CHANGEACL

%LIST(OFF)
%INCLUDE 'BLOCKIO_QSYM.F77.IN'
%INCLUDE 'FILES_QSYM.F77.IN'
%LIST(ON)

        CHARACTER*13 FNAME
        CHARACTER*(ISYS_MXACL) BUFFER
        CHARACTER*4 ACLTXT

        INTEGER*2 ACLBITS
        CHARACTER*2 ACL_BITS
        EQUIVALENCE (ACLBITS,ACL_BITS)

        FNAME = 'MYDIR:MYFILE<0>'
        ACLTXT = '+<0><0><0>'        ! ACL STRING, TO WHICH ADD
        ACLBITS = ??????????+??????????   ! READ, EXECUTE ACCESS
        ACLTXT(3:3) = ACL_BITS(2:2)
        ACLLEN = LEN(ACLTXT)

C   INITIALIZE EMPTY BUFFER
        DO I = 1, ISYS_MXACL
                 BUFFER(I:I) = '<0>'
        END DO

C   GET ACL
        IAC0 = ????????(?????)           ! POINTER TO NAME OF FILE
        IAC1 = ????????(??????)          ! POINTER TO BUFFER
        IAC2 = 0                         ! RESERVED
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)  ! GET ACL
        IF (IER.NE.0) GOTO 999

C   SCAN BUFFER FOR END OF ACL STRING
        I = ISYS_MXACL                   ! START WITH MAX LENGTH
        DO WHILE (BUFFER(I:I) .EQ. '<0>')
                I = I-1
                IF (I .LE. 0) GOTO 999   ! IF ALL NULLS, THERE'S A
                                         ! PROBLEM
        END DO

C   APPEND NEW ACL TO OLD
        DO J = 1, ACLLEN
                I = I+1
                BUFFER(I:I) = ACLTXT(J:J)
        END DO
```

```
C    SET THE NEW ACL
         IAC0 = ????????(?????)           ! POINTER TO NAME OF FILE
         IAC1 = ????????(??????)          ! POINTER TO BUFFER
         IAC2 = 0                         ! RESERVED
         IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)    ! SET ACL TO FILE
         IF (IER.NE.0) GOTO 999

C    GOOD END
         BUFFER = "<NL>The file's ACL has been changed.<NL><0>"
         IAC1 = BYTEADDR(BUFFER)
         IAC2 = ISYS_RFCF+LEN("<NL>The file's ACL has been
changed.<NL><0>")
         IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

C    BAD END
999      IAC2 = ISYS_RFEC+ISYS_RFCF+ISYS_RFAB
         IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
         IF (IER.NE.0) GOTO 999

         END
```

FORTRAN 77 FILE STATUS LAB

* MOVE/R/NACL COPIES OF THE FILES 'FILES.F77', 'FILES_QSYM.F77.IN', 'BLOCKIO_QSYM.F77.IN', AND 'GTMES_QSYM.F77.IN' FROM THE :S309VS:F77 DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'FILES' IS TO FIRST GO TO A DIRECTORY, SPECIFIED AS THE FIRST ARGUMENT ON THE COMMAND LINE. THE PROGRAM WILL FAIL IF THE DIRECTORY DOES NOT EXIST. IT WILL THEN GET A FILENAME WHICH MATCHES THE TEMPLATE SPECIFIED AS THE SECOND ARGUMENT ON THE COMMAND LINE. IT WILL TYPE THE FILENAME AND BYTE LENGTH ON THE CONSOLE. THEN IT WILL GET THE NEXT FILENAME AND REPEAT THE PROCESS. WHEN ALL THE FILES FOR A TEMPLATE HAVE BEEN RECOVERED, THE PROGRAM WILL GET THE NEXT TEMPLATE FROM THE COMMAND LINE UNTIL ALL THE FILES HAVE BEEN DISPLAYED.

* REFER TO THE PRINTOUT OF 'FILES.F77' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUES. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
F77,FILES
F77LINK,FILES
```

* GET A HARDCOPY OF THE FILE 'FILES.LS'.

* EXECUTE THE FILES PROGRAM WITH THE FOLLOWING COMMAND:

```
X,FILES,:UDD:yourusername,template...
```

* YOU SHOULD SOON SEE THE FILENAMES AND BYTE LENGTHS FOLLOWED BY THE MESSAGE 'INFORMATION PROCESSING COMPLETED'.

* DO A FILESTATUS WITH /LENGTH IN THE DIRECTORY IN YOUR COMMAND LINE AND CHECK THE RESULTS.

* YOU HAVE COMPLETED THIS LAB WHEN YOUR PROGRAM YIELDS THE SAME RESULTS AS THE FILESTATUS/LENGTH COMMAND.

```
        PROGRAM FILES

%LIST(OFF)
%INCLUDE 'FILES_QSYM.F77.IN'
%INCLUDE 'BLOCKIO_QSYM.F77.IN'
%INCLUDE 'GTMES_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 GTMPK(0:ISYS_GTLN-1)   ! ?GTMES PKT
        INTEGER*2 GOPPK(0:ISYS_OPLT-1)   ! ?GOPEN PKT FOR DIR
        INTEGER*2 GNFPK(0:ISYS_NFLN-1)   ! ?GNFN PKT
        INTEGER*2 FSPK(0:ISYS_SLTH-1)    ! ?FSTAT PKT

C DOUBLE WORDS

        INTEGER*4 GTMPK_GSW,GTMPK_GRES           ! IN ?GTMES PKT
        EQUIVALENCE (GTMPK(ISYS_GSW),GTMPK_GSW)
        EQUIVALENCE (GTMPK(ISYS_GRES),GTMPK_GRES)

        INTEGER*4 GOPPK_OPEH                     ! IN ?GOPEN PKT
        EQUIVALENCE (GOPPK(ISYS_OPEH),GOPPK_OPEH)

        INTEGER*4 GNFPK_NFNM,GNFPK_NFTP          ! IN ?GNFN PKT
        EQUIVALENCE (GNFPK(ISYS_NFNM),GNFPK_NFNM)
        EQUIVALENCE (GNFPK(ISYS_NFTP),GNFPK_NFTP)

        INTEGER*4 FSPK_SDEH,FSPK_SEFM,FSPK_SFAH,FSPK_SCSH! ?FSTAT
        EQUIVALENCE (FSPK(ISYS_SDEH),FSPK_SDEH)
        EQUIVALENCE (FSPK(ISYS_SEFM),FSPK_SEFM)
        EQUIVALENCE (FSPK(ISYS_SFAH),FSPK_SFAH)
        EQUIVALENCE (FSPK(ISYS_SCSH),FSPK_SCSH)

        CHARACTER*80  BUFFER             ! DIRECTORY NAME BUFFER
        CHARACTER*80  FNAME              ! FILE NAME BUFFER
```

```
C   PKT INITIALIZATION

C   ?GTMES PKT
        GTMPK(ISYS_GREQ) = ISYS_GARG            ! GET ARGUMENT
        GTMPK(ISYS_GNUM) = 1                  ! 1ST ARG (0=PROG NAME)
        GTMPK_GSW = 0                         ! NO SWITCH TO TEST
        GTMPK_GRES = BYTEADDR(BUFFER)           ! BUFFER BYTE PTR

C   ?GOPEN PKT
        GOPPK(ISYS_OPCH) = 0          ! NO FLAGS, RETURN CHAN. NUM.
        GOPPK(ISYS_OPTY) = 0          ! RETURN REC FMT, FILE TYPE
        GOPPK_OPEH = 0                ! RETURN FILE SIZE IN BYTES
        GOPPK(ISYS_OPFC) = 0          ! NO FILE CONTROL PARAMETERS

C   ?GNFN PKT
        GNFPK(ISYS_NFKY) = ?          ! 0 FOR FIRST FILENAME
        GNFPK(ISYS_NFRS) = 0          ! RESERVED
        GNFPK_NFNM = ????????(?????)! BYTE PTR TO FILENAME BUFFER
        GNFPK_NFTP = ????????(??????)! BYTE PTR TO TEMPLATE BUFF

C   ?FSTAT PKT
        FSPK(ISYS_STYP) = 0           ! RCD FMT AND ENTRY TYPE
        FSPK(ISYS_STIM) = 0           ! NOT USED, RETURNS -1
        FSPK(ISYS_SACP) = 0           ! NOT USED, RETURNS -1
        FSPK(ISYS_SCPS) = 0           ! FILE CONTROL PARAMETERS
        FSPK(ISYS_SLAU) = 0           ! RESERVED
        FSPK(ISYS_SMIL) = 0           ! MAXIMUM NUMBER INDEX LEVELS
        FSPK_SDEH = 0                 ! FILE ELEMENT SIZE
        FSPK(ISYS_STCH) = 0           ! FILE CREATION DATE - SCALAR
        FSPK(ISYS_STCL) = 0           ! FILE CREATION TIME - SCALAR
        FSPK(ISYS_STAH) = 0           ! LAST ACCESS DATE - SCALAR
        FSPK(ISYS_STAL) = 0           ! LAST ACCESS TIME - SCALAR
        FSPK(ISYS_STMH) = 0           ! LAST MOD. DATE - SCALAR
        FSPK(ISYS_STML) = 0           ! LAST MOD. TIME - SCALAR
        FSPK(ISYS_SSTS) = 0           ! FILE CHARACTERISTICS
        FSPK(ISYS_SEFW) = 0           ! RESERVED
        FSPK_SEFM = 0                 ! FILE SIZE IN BYTES
        FSPK_SFAH = 0                 ! STARTING LD ADDRESS
        FSPK(ISYS_SEFH) = 0           ! RESERVED
        FSPK(ISYS_SIDX) = 0           ! CURRENT INDEX LEVEL
        FSPK_SCSH = 0                 ! NOT USED
        FSPK(ISYS_SOPN) = 0           ! OPEN COUNT
```

```
C       PROGRAM

C   GTMES - GET DIRECTORY NAME FOR FILE LOOKUP
        IAC0 = 0
        IAC1 = 0
        IAC2 = WORDADDR(GTMPK)
        IER = ISYS(ISYS_GTMES,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C   GOPEN DIRECTORY
        IAC0 = BYTEADDR(BUFFER)
        IAC1 = -1                          ! ASSIGN CHANNEL NUMBER
        IAC2 = WORDADDR(GOPPK)
        IER = ISYS(ISYS_GOPEN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C   CHANGE WORKING DIRECTORY
        IAC0 = BYTEADDR(BUFFER)
        IAC1 = 0
        IAC2 = 0
        IER = ISYS(ISYS_???,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C   GTMES 2 - GET TEMPLATE OR PATHNAME
        GTMPK(ISYS_GNUM) = GTMPK(ISYS_GNUM)+1   ! NEXT ARGUMENT
        IAC0 = 0                                ! RESERVED
        ?????(ISYS_????) = 0    ! RESET INT'L PTR IN GNFN PKT
        IAC1 = 0                                ! RESERVED
        IAC2 = WORDADDR(GTMPK)
        IER = ISYS(ISYS_GTMES,IAC0,IAC1,IAC2)
        IF (IER.EQ.ISYS_ERNAG) GOTO 900
                               ! NORMAL RETURN IF NO SUCH ARG
        IF (IER.NE.0) GOTO 999                  ! ERROR

C   GET NEXT FILE NAME
100     IAC0 = 0                                ! RESERVED
        IAC1 = GOPPK(ISYS_OPCH)                 ! CHANNEL NUMBER
        IAC2 = ???????? (?????)
        FNAME = ' '                             ! INIT FNAME TO BLANK
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.EQ.ISYS_EREOF) GOTO 900         ! DONE IF EOF
        IF (IER.NE.0) GOTO 999             ! ELSE, CHECK FOR ERRORS

C    GET FILE STATUS
        IAC0 = ????????(?????)
        IAC1 = 0
        IAC1 = IBSET4(IAC1,30)                  ! TURN ON BIT 1
        IAC2 = ????????(????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C   PRINT INFO
        WRITE (6,200) FNAME,?????????           ! WRITE FILE NAME AND
                                                ! BYTE COUNT
200     FORMAT (" ",A40,I10)
        GOTO 100
```

```
C   GOOD END
900     BUFFER = '<NL>Information processing completed.<NL>'
        IAC1 = BYTEADDR(BUFFER)
        IAC2 = ISYS_RFCF+LEN('<NL>Information processing
completed.<NL>')
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

C   BAD END
999     IAC2 = ISYS_RFEC+ISYS_RFCF+ISYS_RFAB
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

FORTRAN 77 BLOCK I/O LAB

* MOVE COPIES OF THE FILES 'BLOCKIO.F77', BLOCKIO_QSYM.F77.IN, AND 'BLKDATA' FROM THE :S309VS:F77 DIRECTORY TO YOUR INITIAL WORKING DIRECTORY.

* TYPE THE DATA FILE. IF YOU DON'T BELIEVE YOUR EYES QPRINT THE FILE. IS THERE STILL A PROBLEM READING THE FILE? USE THE DISPLAY UTILITY TO LOOK AT THE INPUT FILE.

* THE PROBLEM IS THAT CARRIAGE RETURNS RATHER THAN NEW LINE CHARACTERS WERE USED AS DELIMITERS WHEN THE FILE WAS CREATED. THE THRUST OF THIS EXERCISE IS TO WRITE A PROGRAM WHICH CONVERTS ALL CARRIAGE RETURNS TO NEW LINES.

* QPRINT A COPY OF THE SOURCE FILE.

* YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUE. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS, AND THE CONTENTS OF THE PACKETS.

* COMPILE AND LINK THE PROGRAM.

* EXECUTE THE BLOCKIO PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'DELIMITER CONVERSION COMPLETED.'

* TYPE OUT THE OUTPUT FILE 'BLOCKOUT'.
THE TEXT SHOULD NOW BE READABLE.

* DO A FILESTATUS WITH /ASSORTMENT ON BOTH THE INPUT AND OUTPUT FILES. THEY SHOULD HAVE IDENTICAL LENGTHS - EXACTLY ONE BLOCK.

* TO TEST THE PROGRAM WITH A DIFFERENT INPUT FILE:

    DELETE YOUR COPY OF 'BLKDATA'. COPY INTO A NEW 'BLKDATA' FILE IN YOUR OWN DIRECTORY FROM 'BLKDATA1' IN THE :S309VS:F77 DIRECTORY. DO A FILESTATUS WITH ASSORTMENT ON THIS NEW 'BLKDATA' FILE IN YOUR OWN DIRECTORY. IT SHOULD BE LESS THAN ONE BLOCK LONG.

    CHANGING TO A DIFFERENT INPUT FILE WOULD BE EASIER IF A GENERIC FILENAME HAD BEEN USED IN THE PROGRAM ... HOWEVER THE SYSTEM DOES NOT PERMIT BLOCK IO TO GENERIC FILENAMES.

* EXECUTE THE PROGAM.

    NOTICE THAT YOUR OLD 'BLOCKOUT' FILE WILL BE DELETED BY THE PROGRAM AND A NEW ONE WILL BE CREATED. FIND THE SECTIONS OF CODE THAT ACCOMPLISH THIS. PROGRAMS THAT PERFORM BLOCK IO MUST HANDLE DETAILS SUCH AS THIS EXPLICITLY.

* YOU SHOULD AGAIN SEE THE COMPLETION MESSAGE. TYPE THE OUTPUT FILE 'BLOCKOUT'. IT SHOULD BE READABLE. FIND THE PARTS OF THE PROGRAM WHICH ENSURE THAT IT WILL WORK CORRECTLY IF THE LAST BLOCK IS NOT FULL. COMPARE THE LENGTHS OF YOUR NEW 'BLKDATA' AND 'BLOCKOUT'.... THEY SHOULD BE IDENTICAL.

```
        PROGRAM BLOCKIO

%LIST(OFF)
%INCLUDE 'BLOCKIO_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 CREPK(0:ISYS_CLTH-1)
        INTEGER*2 PKTIN(0:ISYS_????-1)   ! ?GOPEN PKT FOR INPUT
        INTEGER*2 PKTOU(0:ISYS_????-1)   ! ?GOPEN PKT FOR OUTPUT
        INTEGER*2 RDPK(0:ISYS_????-1)    ! READ BLOCK PKT
        INTEGER*2 WRPK(0:ISYS_????-1)    ! WRITE BLOCK PKT

        INTEGER*4 CREPK_CTIM,CREPK_CACP
        EQUIVALENCE (CREPK(ISYS_CTIM),CREPK_CTIM)
        EQUIVALENCE (CREPK(ISYS_CACP),CREPK_CACP)

        INTEGER*4 RDPK_PCAD,RDPK_PRNH
        EQUIVALENCE (RDPK(ISYS_PCAD),RDPK_PCAD)
        EQUIVALENCE (RDPK(ISYS_PRNH),RDPK_PRNH)

        INTEGER*4 WRPK_PCAD,WRPK_PRNH
        EQUIVALENCE (WRPK(ISYS_PCAD),WRPK_PCAD)
        EQUIVALENCE (WRPK(ISYS_PRNH),WRPK_PRNH)

        CHARACTER*512 BUFF

        LOGICAL EOFLG /.FALSE./
```

```
C       FILE CREATION PKT

        CREPK(ISYS_CFTYP) = ISYS_FTXT   ! FILE TYPE TEXT
        CREPK(ISYS_CCPS)  = 0           ! FILE CONTROL PARAMS IGNORED
        CREPK_CTIM = -1                 ! DEFAULT TIME
        CREPK_CACP = -1                 ! DEFAULT ACL
        CREPK(ISYS_CDEH)  = 0           ! RESERVED
        CREPK(ISYS_CDEL)  = -1          ! DEFAULT ELEMENTSIZE
        CREPK(ISYS_CMIL)  = -1          ! DEFAULT MAX INDEX LEVELS
        CREPK(ISYS_CMRS)  = 0           ! RESERVED

        PKTIN(ISYS_OPFL)  = 0           ! NO ?GOPEN FLAGS INPUT FILE
        PKTOU(ISYS_OPFL)  = 0           ! NO ?GOPEN FLAGS OUTPUT FILE


C       READ BLOCK PKT

        RDPK(ISYS_PSTI) = ?             ! READ ONE BLOCK
        RDPK(ISYS_PSTO) = ?             ! RESERVED
        RDPK_PCAD = ????????(????)      ! BUFFER ADDRESS
        RDPK_PRNH = ?                   ! STARTING BLOCK NUMBER
        RDPK(ISYS_PRCL) = ?             ! BYTE COUNT
        RDPK(ISYS_PRES) = ?             ! RESERVED

C       WRITE BLOCK PKT

        WRPK(ISYS_PSTI) = ?             ! WRITE ONE BLOCK
        WRPK(ISYS_PSTO) = ?             ! RESERVED
        WRPK_PCAD = ????????(????)      ! BUFFER ADDRESS
        WRPK_PRNH = ?                   ! STARTING BLOCK NUMBER
        WRPK(ISYS_PRCL) = ?     ! BYTE COUNT (SET EACH ?WRITE)
        WRPK(ISYS_PRES) = ?             ! RESERVED
```

```
???? = ????????('BLKDATA<0>')    ! FILENAME BYTEPOINTER
IAC1 = ??                        ! SYSTEM ASSIGNS CHANNEL
IAC2 = ????????(?????)           ! PKT ADDRESS
                              ! OPEN 'BLKDATA' FOR BLOCK IO
IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
IF (IER.NE.0) GOTO 999

                           ! DELETE 'BLOCKOUT' IF IT EXISTS
IAC0 = BYTEADDR('BLOCKOUT<0>')
IAC1 = 0
IAC2 = 0
IER = ISYS(ISYS_DELETE,IAC0,IAC1,IAC2)
IF (IER.NE.0.AND.IER.NE.ISYS_ERFDE) GOTO 999

IAC0 = BYTEADDR('BLOCKOUT<0>')   ! CREATE 'BLOCKOUT'
IAC1 = 0
IAC2 = WORDADDR(CREPK)
IER = ISYS(ISYS_CREATE,IAC0,IAC1,IAC2)
IF (IER.NE.0) GOTO 999

IAC2 = ????????(?????)           ! ?GOPEN PKT ADDRESS
IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)   ! OPEN 'BLOCKOUT'
IF (IER.NE.0) GOTO 999
```

```
500     ???? = ?????(ISYS_????) ! GET INPUT FILE CHANNEL NUMBER
        IAC0 = 0
        ???? = ????????(????)   ! ADDRESS OF READ BLOCK PKT
        IER = ISYS(ISYS_???,IAC0,IAC1,IAC2)     ! READ A BLOCK
        IF (IER.NE.ISYS_EREOF.AND.IER.NE.0) GOTO 999
        IF (IER.EQ.ISYS_EREOF) EOFLG = .TRUE.
        IF (EOFLG.AND.IAC1.EQ.0) GOTO 900

        RDPK_???? = RDPK_????+1 ! INCR BLOCK NUMBER IN RDB PKT
        WRPK(ISYS_????) = ????  ! NUMBER OF BYTES TO WRITE EQUALS
                                ! NUMBER OF BYTES ACTUALLY READ
        DO 600, I = 1,IAC1      ! CONVERT CR'S TO NL'S
        IF (BUFF(I:I).EQ.'<CR>') BUFF(I:I) = '<NL>'
600     CONTINUE

        ???? = ?????(ISYS_????) ! GET CH NUMBER OF OUTPUT FILE
        IAC0 = 0
        ???? = ????????(????)   ! ADDRESS OF WRITE BLOCK PKT
        IER = ISYS(ISYS_???,IAC0,IAC1,IAC2)     ! WRITE A BLOCK
        IF (IER.NE.0) GOTO 999

        WRPK_???? = WRPK_????+1 ! INCR BLOCK NUMBER IN WRB PKT

        IF (.NOT.EOFLG) GOTO 500

900     IAC0 = 0
        ???? = ?????(ISYS_????) ! GET CH NUMBER OF INPUT FILE
        IAC2 = 0
        IER = ISYS(ISYS_??????,IAC0,IAC1,IAC2)  ! CLOSE
        IF (IER.NE.0) GOTO 999

        IAC0 = 0
        ???? = ?????(ISYS_????)  ! GET CH NUMBER OF OUTPUT FILE
        IAC2 = 0
        IER = ISYS(ISYS_??????,IAC0,IAC1,IAC2)  ! CLOSE
        IF (IER.NE.0) GOTO 999

        BUFF = 'DELIMITER CONVERSION COMPLETED<NL>'
        IAC1 = BYTEADDR(BUFF)
        IAC2 = ISYS_RFCF+LEN('DELIMITER CONVERSION
COMPLETED<NL>')
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

999     IAC2 = ISYS_RFEC+ISYS_RFCF+ISYS_RFAB
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

FORTRAN 77 SHARED DATA LAB

USING THE FOLLOWING CLI COMMANDS, MOVE THE NEEDED FILES INTO YOUR DIRECTORY:

```
DIRECTORY, :S309VS:F77
MOVE/NACL, :UDD:YOURUSERNAME, SHDATA.<ONE,TWO>.F77,&
                                SHAREDIO_QSYM.F77.IN
DIRECTORY/I
```

QPRINT A COPY OF EACH OF THE SOURCE FILES.

DISCUSSION:

SHDATA.ONE AND SHDATA.TWO COMMUNICATE ACROSS A SINGLE PAGE OF MEMORY WHICH IS MAPPED TO BOTH PROCESSES CONCURRENTLY USING THE SHARED DATA FACILITY.

SHDATA.ONE FIRST CREATES A SHARED DATA FILE THEN READS IT ONCE TO MAP IN THE SHARED PAGE. A MESSAGE (FOR SHDATA.TWO) IS NOW MOVED TO A BUFFER LOCATED IN THE SHARED PAGE. AFTER SETTING A FLAG (ALSO LOCATED IN THE SHARED PAGE), SHDATA.ONE WAITS FOR SHDATA.TWO TO CLEAR THE FLAG. THEN SHDATA.ONE RETURNS CONTROL TO THE CLI WHICH DISPLAYS A RETURN MESSAGE.

SHDATA.TWO READS ONE PAGE FROM THE SHARED DATA FILE, WAITS FOR SHDATA.ONE TO SET THE FLAG, AND WRITES THE CONTENTS THE BUFFER IN THE SHARED PAGE TO @OUTPUT. IT NEXT CLEARS THE FLAG WORD SHDATA.ONE IS WAITING FOR AND PASSES CONTROL TO THE CLI.

STUDY THE SOURCES TO MAKE SURE YOU FOLLOW THE CODE THEN FILL IN THE ITEMS WHICH HAVE BEEN REPLACED WITH QUESTION MARKS. YOU WILL HAVE TO SUPPLY RELATED SYSTEM CALLS AND THEIR ARGUMENTS, ERROR CODE SYMBOLS, ADDRESSES OF VARIOUS ITEMS IN PACKETS, AND THE APPROPRIATE VALUES IN THOSE PACKETS. WHEN YOU HAVE DETERMINED THE PROPER ENTRIES FOR THE INDICATED ITEMS EDIT THE TWO SOURCE FILES AND MAKE THE CORRECTIONS. COMPILE SHDATA.ONE AND SHDATA.TWO SEPARATELY, THEN LINK EACH USING THE COMMAND LINE SHOWN IN THE SOURCE FILES.

PREDICT WHAT WILL OCCUR WHEN THE PROGRAMS ARE RUN. EXECUTE SHDATA.ONE AT ONE CONSOLE THEN LOG ON AT A SECOND CONSOLE AND EXECUTE SHDATA.TWO.

USE THE DISPLAY UTILITY TO EXAMINE THE SHARED DATA FILE.

1. YOU SHOULD FIND DATA IN THE FILE. WHY?
2. AT WHAT BITE LOCATION IN THE FILE IS THE FLAG WORD? WHAT IS ITS VALUE?

```
        PROGRAM SHDATA1

%LIST(OFF)
%INCLUDE 'SHAREDIO_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 CREPK(0:ISYS_CLTH-1)  ! CREATE PACKET
        INTEGER*2 SPPK(0:ISYS_PBLT-1)   ! SPAGE PACKET

        INTEGER*4 CREPK_CTIM,CREPK_CACP
        EQUIVALENCE (CREPK(ISYS_CTIM),CREPK_CTIM)
        EQUIVALENCE (CREPK(ISYS_CACP),CREPK_CACP)

        INTEGER*4 SPPK_PCAD,SPPK_PRNH
        EQUIVALENCE (SPPK(ISYS_PCAD),SPPK_PCAD)
        EQUIVALENCE (SPPK(ISYS_PRNH),SPPK_PRNH)

        INTEGER CHNUM

C       SHARED BUFFER IS PLACED IN NAMED COMMON SO IT CAN BE
C       MADE PAGE ALIGNED AND SHARED AT LINK TIME ...
C       F77LINK, SHDATA.ONE, SHPG/SHARED/ALIGN=10

        CHARACTER*80 SBUFF
        CHARACTER FLAG              ! TOTAL SIZE OF SHARED BLK
        CHARACTER*1967 WASTE        ! MUST BE MULT 0F 2048 BYTES
        COMMON /SHPG/ SBUFF, FLAG, WASTE

C       FILE CREATION PACKET

        CREPK(ISYS_CFTYP) = ISYS_FTXT   ! FILE TYPE TEXT
        CREPK(ISYS_CCPS)  = 0           ! FILE CONTROL PARAMS IGNORED
        CREPK_CTIM = -1                 ! DEFAULT TIME
        CREPK_CACP = -1                 ! DEFAULT ACL
        CREPK(ISYS_CDEH)  = 0           ! RESERVED
        CREPK(ISYS_CDEL)  = 4    ! ELEMENTSIZE MUST BE MULT OF 4
        CREPK(ISYS_CMIL)  = -1   ! DEFAULT MAXIMUM INDEX LEVELS
        CREPK(ISYS_CMRS)  = 0           ! RESERVED

C       SPAGE PACKET

        SPPK(ISYS_PSTI) = ?         ! # BLOCKS MUST BE MULT OF 4
        SPPK(ISYS_PSTO) = 0         ! RESERVED
        SPPK_PCAD = ????????(?????)
                            ! BUFFER ADDRESS (MUST BE PG ALIGNED)
        SPPK_PRNH = ?               ! STARTING BLOCK # (MULT OF 4)
        SPPK(ISYS_PRCL) = 0
        SPPK(ISYS_PRES) = 0         ! RESERVED
```

```
        IAC0 = BYTEADDR('SDFILE<0>') ! DELETE FILE IF IT EXISTS
        IER = ISYS(ISYS_DELETE,IAC0,IAC1,IAC2)
        IF (IER.NE.0.AND.IER.NE.ISYS_ERFDE) CALL ERRCODE(IER)

        IAC0 = BYTEADDR('SDFILE<0>')
        IAC2 = WORDADDR(CREPK)          ! CREATE SHARED FILE
        IER = ISYS(ISYS_CREATE,IAC0,IAC1,IAC2)
        IF (IER.NE.0) CALL ERRCODE(IER)

        IAC0 = BYTEADDR('SDFILE<0>')
        IAC1 = -1                       ! LET SYSTEM ASSIGN CH
        IAC2 = -1                       ! OPEN FOR READ/WRITE
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)! OPEN FOR SHARED IO
        IF (IER.NE.0) CALL ERRCODE(IER)

        IAC0 = 0
        CHNUM = IAC1                    ! SAVE CHANNEL NUMBER
        IAC2 = ????????(????)           ! ADDRESS OF SPAGE PKT
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)   ! READ FIRST PG
        IF (IER.NE.0) CALL ERRCODE(IER)

        FLAG = 'N'                      ! INITIALIZE FLAG

C       MOVE TEXT TO SHARED BUFFER

        SBUFF = 'THIS IS A TEST MESSAGE'

C       SET FLAG IN BUFFER THEN WAIT UNTIL SHDATA2 RESETS IT.

        FLAG = 'Y'

        DO WHILE (FLAG.EQ.'Y')
                IAC0 = 1000             ! DELAY = 1000 MSEC
                IER = ISYS(ISYS_WDELAY,IAC0,IAC1,IAC2)
                IF (IER.NE.0) CALL ERRCODE(IER)
        END DO

        IAC1 = ISHFT(1,31)+CHNUM
        IER = ISYS(ISYS_??????,IAC0,IAC1,IAC2)  ! CLOSE
        IF (IER.NE.0) CALL ERRCODE(IER)

        IAC1 = BYTEADDR('SHDATA.TWO HAS RECEIVED THE TEST
MESSAGE.<NL>')
        IAC2 = ISYS_RFCF+LEN('SHDATA.TWO HAS RECEIVED THE TEST
MESSAGE.<NL>')
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

        END
```

```
        PROGRAM SHDATA2

%LIST(OFF)
%INCLUDE 'SHAREDIO_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 SPPK(0:ISYS_PBLT-1)    ! SPAGE PACKET

        INTEGER*4 SPPK_PCAD,SPPK_PRNH
        EQUIVALENCE (SPPK(ISYS_PCAD),SPPK_PCAD)
        EQUIVALENCE (SPPK(ISYS_PRNH),SPPK_PRNH)

        INTEGER CHNUM

C       SHARED BUFFER IS PLACED IN NAMED COMMON SO IT CAN BE
C       MADE PAGE ALIGNED AND SHARED AT LINK TIME ...
C       F77LINK, SHDATA.TWO, SHPG/SHARED/ALIGN=10

        CHARACTER*80 SBUFF
        CHARACTER FLAG           ! TOTAL SIZE OF SHARED BLK
        CHARACTER*1967 WASTE     ! MUST BE MULT OF 2048 BYTES
        COMMON /SHPG/ SBUFF, FLAG, WASTE

C       SPAGE PACKET

        SPPK(ISYS_PSTI) = ?      ! # BLOCKS MUST BE MULT OF 4
        SPPK(ISYS_PSTO) = 0      ! RESERVED
        SPPK_PCAD = ????????(?????)
                                 ! BUFFER ADDRESS (MUST BE PG ALIGNED)
        SPPK_PRNH = ?            ! STARTING BLOCK # (MULT OF 4)
        SPPK(ISYS_PRCL) = 0
        SPPK(ISYS_PRES) = 0      ! RESERVED
```

```
100     IAC0 = BYTEADDR('SDFILE<0>')
        IAC1 = -1                ! LET SYSTEM ASSIGN CHANNEL
        IAC2 = -1                ! OPEN FOR READ/WRITE
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)! OPEN FOR SHARED IO
        IF (IER.EQ.ISYS_ERFDE) THEN
                IAC0 = 1000      ! DELAY = 1000 MSEC
                IER = ISYS(ISYS_WDELAY,IAC0,IAC1,IAC2)
                IF (IER.NE.0) CALL ERRCODE(IER)
                GOTO 100
        END IF
        IF (IER.NE.0) CALL ERRCODE(IER)

        CHNUM = IAC1             ! SAVE CHANNEL NUMBER
        IAC2 = ????????(????)    ! ADDRESS OF SPAGE PKT
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)    ! READ FIRST PG
        IF (IER.NE.0) CALL ERRCODE(IER)

        DO WHILE (FLAG.NE.'Y')
                IAC0 = 1000              ! DELAY = 1000 MSEC
                IER = ISYS(ISYS_WDELAY,IAC0,IAC1,IAC2)
                IF (IER.NE.0) CALL ERRCODE(IER)
        END DO

        PRINT '(1X,A80)',SBUFF

        FLAG = 'N'

        IAC1 = ISHFT(1,31)+CHNUM
        IER = ISYS(ISYS_SCLOSE,IAC0,IAC1,IAC2)   ! CLOSE
        IF (IER.NE.0) CALL ERRCODE(IER)

        END
```

S309VS RECORD IO LAB

PART ONE: DATASENSITIVE AND VARIABLE

ISSUE THE FOLLOWING COMMANDS TO MOVE NEEDED FILES TO YOUR OWN DIRECTORY:

```
DIRECTORY, :S309VS:F77
MOVE/NACL, :UDD:YOURUSERNAME, RECORDIO.F77,&
                IO_QSYM.F77.IN, DATASENS.TXT, VIEW.PR
DIRECTORY/I
```

QPRINT A COPY OF RECORDIO.F77. THIS PROGRAM COPIES THE CONTENTS OF THE GENERIC FILE "@DATA" INTO THE FILE "RECORDIO.OUT" ONE RECORD AT A TIME.

IF AN ERROR OCCURS WHEN READING A RECORD, IT CHECKS TO SEE IF THE ERROR WAS "END OF FILE". IF SO, THE PROGRAM TERMINATES NORMALLY.

ON ALL OTHER ERRORS, THE PROGRAM WILL ABORT, RETURNING THE ERROR CODE TO THE PARENT PROCESS.

AT THE LABEL "900" THE INTENT OF THE INSTRUCTION IS TO TEST IAC0 FOR THE VALUE OF THE END-OF-FILE ERROR CODE. FIND THE SYMBOL FOR THIS ERROR CODE. EDIT YOUR COPY OF THE SOURCE FILE AND FILL IT IN.

THE FIRST PAGE OF THE SOURCE FILE IS DEVOTED TO THE PARAMETER PACKETS FOR THE RECORD I/O SYSTEM CALLS.

FILL IN THE FOLLOWING INFORMATION IN THE PACKETS:

I/O PACKET FOR FILE @DATA

OFFSET	OPTIONS OR VALUE
?ISTI	CHANGE RECORD FORMAT TO DATASENSITVE, INPUT ONLY
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	MAX RECORD LENGTH SAME AS NUMBER OF BYTES IN "BUFF"
?IRNH	NEXT SEQUENTIAL RECORD
?IFNP	BYTE POINTER TO FILENAME "@DATA"
?IDEL	DEFAULT DELIMITER TABLE

I/O PACKET FOR FILE RECORDIO.OUT

OFFSET	OPTIONS OR VALUE
?ISTI	CHANGE RECORD FORMAT TO DATASENSITVE, OUTPUT ONLY, DELETE OLD FILE IF IT EXISTS, CREATE NEW FILE
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	MAX RECORD LENGTH SAME AS NUMBER OF BYTES IN "BUFF"
?IRNH	NEXT SEQUENTIAL RECORD
?IFNP	BYTE POINTER TO FILENAME "RECORDIO.OUT<0>"
?IDEL	DEFAULT DELIMITER TABLE

WHAT IS THE SYMBOLIC NAME OF THE PACKET FOR THE FILE @DATA ?
_____ WHAT IS THE SYMBOLIC NAME OF THE PACKET FOR THE FILE RECORDIO.OUT?_____

ON PAGE TWO, FILL IN THE PACKET NAMES INTO IAC2 FOR THE SYSTEM CALLS. FILL IN THE NAMES OF THE SYSTEM CALLS.

COMPILE THE SOURCE FILE, OBTAINING A PRINTED LISTING. LINK THE PROGRAM.

EXECUTE THE PROGRAM. WHAT ERROR MESSAGE OCCURRED ?

_____

THE ERROR HAS OCCURRED AT THE FIRST SYSTEM CALL IN THE PROGRAM. WHAT FILE IS CAUSING THE PROBLEM ?

_____

USE A CLI COMMAND TO SET "DATASENS.TXT" AS THE GENERIC FILE YOUR PROGRAM WILL TRY TO READ. WHAT IS THE COMMAND ?

_____

EXECUTE THE PROGRAM. YOU SHOULD GET THE MESSAGE "IO COMPLETED"

COMPARE THE CONTENTS OF YOUR RECORDIO.OUT FILE AND DATASENS.TXT. THEY SHOULD BE IDENTICAL.

NOW MODIFY THE PROGRAM TO OUTPUT VARIABLE LENGTH RECORDS INSTEAD OF DATASENSITIVE:

NOTE THAT ONLY THE OUTPUT IS TO BE CHANGED - THE INPUT IS STILL TO BE DATASENSITIVE.

IT WILL BE NECESSARY TO SPECIFY THE LENGTH OF EACH RECORD AS IT IS WRITTEN OUT.

AFTER EACH RECORD HAS BEEN READ IN, WHERE IS ITS LENGTH STORED ?

_____

IN ORDER TO OUTPUT THE RECORD IN VARIABLE FORMAT, WHAT OFFSET IN THE OUTPUT PACKET MUST CONTAIN THIS INFORMATION ? _____

ADD SOME CODE AFTER THE ?READ AND BEFORE THE ?WRITE CALL TO GET THE ACTUAL RECORD LENGTH READ IN AND STORE IT IN THE OUTPUT PACKET:

ONE STATEMENT SHOULD BE ENOUGH

_____

CHANGE THE RECORD FORMAT SPECIFICATION IN THE @OUTPUT PACKET TO VARIABLE.

COMPILE AND LINK YOUR PROGRAM.

USING THE CLI COMMAND "X, VIEW, RECORDIO.OUT", YOU CAN SEE THE RECORDS IN YOUR PROGRAM'S OUTPUT FILE. THESE VARIABLE LENGTH RECORDS EACH START WITH FOUR DIGITS THAT REVEAL THE RECORD LENGTH.

SINCE THE "VIEW" PROGRAM OPENS THE SPECIFIED FILE AS THOUGH IT HAD DATASENSITIVE RECORDS, THESE RECORD LENGTHS WILL BE DISPLAYED AS PART OF EACH LINE ON YOUR SCREEN. NOTICE THAT THE LENGTH OF EACH RECORD INCLUDES THE FOUR DIGITS.

```
      PROGRAM RECORDIO

%LIST(OFF)
%INCLUDE 'IO_QSYM.F77.IN'
%LIST(ON)

      INTEGER*2 DATAF(0:ISYS_IBLT-1)
      INTEGER*2 OUTPK(0:ISYS_IBLT-1)

      INTEGER*4 DATAF_IBAD,DATAF_IRNH,DATAF_IFNP,DATAF_IDEL
      EQUIVALENCE (DATAF(ISYS_IBAD),DATAF_IBAD)
      EQUIVALENCE (DATAF(ISYS_IRNH),DATAF_IRNH)
      EQUIVALENCE (DATAF(ISYS_IFNP),DATAF_IFNP)
      EQUIVALENCE (DATAF(ISYS_IDEL),DATAF_IDEL)

      INTEGER*4 OUTPK_IBAD,OUTPK_IRNH,OUTPK_IFNP,OUTPK_IDEL
      EQUIVALENCE (OUTPK(ISYS_IBAD),OUTPK_IBAD)
      EQUIVALENCE (OUTPK(ISYS_IRNH),OUTPK_IRNH)
      EQUIVALENCE (OUTPK(ISYS_IFNP),OUTPK_IFNP)
      EQUIVALENCE (OUTPK(ISYS_IDEL),OUTPK_IDEL)

      CHARACTER*136 BUFF

C     PARAMETER PACKET FOR FILE @DATA

      DATAF(ISYS_ICH) = 0
      DATAF(ISYS_ISTI) = ISYS_????+ISYS_????+ISYS_????
      DATAF(ISYS_ISTO) = 0
      DATAF(ISYS_IMRS) = -1
      DATAF_IBAD = ????????(????)
      DATAF(ISYS_IRES) = 0
      DATAF(ISYS_IRCL) = ???
      DATAF(ISYS_IRLR) = 0
      DATAF(ISYS_IRNW) = 0
      DATAF_IRNH = ?
      DATAF_IFNP = ????????('@DATA')
      DATAF_IDEL = ??

C     PARAMETER PACKET FOR OUTPUT FILE

      OUTPK(ISYS_ICH) = 0
      OUTPK(ISYS_ISTI) =
     +          ISYS_????+ISYS_????+ISYS_????+ISYS_????+ISYS_????
      OUTPK(ISYS_ISTO) = 0
      OUTPK(ISYS_IMRS) = -1
      OUTPK_IBAD = ????????(????)
      OUTPK(ISYS_IRES) = 0
      OUTPK(ISYS_IRCL) = ???
      OUTPK(ISYS_IRLR) = 0
      OUTPK(ISYS_IRNW) = 0
      OUTPK_IRNH = 0
      OUTPK_IFNP = ????????('RECORDIO.OUT<0>')
      OUTPK_IDEL = ??
```

```
C       OPEN @DATA

        IAC2 = WORDADDR(?????)
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       OPEN OUTPUT FILE

        IAC2 = WORDADDR(?????)
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999




C       READ A LINE FROM @DATA

500     IAC2 = WORDADDR(?????)
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 900

C       WRITE THE LINE TO OUTPUT FILE

        IAC2 = WORDADDR(?????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       DO IT AGAIN

        GOTO 500




900     IF (IAC0.NE.ISYS_?????) GOTO 999

        BUFF = 'IO COMPLETED'
        IAC1 = BYTEADDR(BUFF)
        IAC2 = ISYS_RFCF+LEN('IO COMPLETED')

        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)


999     IAC2 = ISYS_RFEC+ISYS_RFCF+ISYS_RFAB

        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

S309VS RECORD IO LAB

PART TWO: DYNAMIC AND FIXED

ISSUE THE FOLLOWING COMMANDS TO MOVE NEEDED FILES TO YOUR OWN DIRECTORY:

```
DIRECTORY, :S309VS:F77
MOVE/NACL, :UDD:YOURUSERNAME, DYNIO.F77,&
             IVAL.F77, SYSLOG
DIRECTORY/I
```

QPRINT THE SOURCE FILE "DYNIO.F77".
THIS PROGRAM READS A SYSTEM LOG FILE, OUTPUTTING ONLY RECORDS OF A CERTAIN TYPE.

THE FORMAT OF THE SYSTEM LOG FILE MUST BE STUDIED BEFORE ONE CAN WRITE A PROGRAM TO READ IT. DYNAMIC RECORD FORMAT MUST BE USED.

EACH RECORD IN THE FILE STARTS WITH A 16. BYTE RECORD HEADER. THE FIRST TWO **WORDS** OF EACH HEADER ARE A 32 BIT NUMBER (NOT FOUR DIGITS AS IN VARIABLE RECORD FORMAT) REVEALING THE LENGTH IN **WORDS** OF THAT RECORD.

WORD FIVE OF THE HEADER CONTAINS A CODE WHICH TELLS THE ENTRY TYPE OF THE RECORD. IN THIS PROGRAM WE WILL OUTPUT ONLY TYPE 3 RECORDS AND SKIP OVER ALL OTHERS.

IN THESE TYPE 3 RECORDS, THE 32 BYTES FOLLOWING THE HEADER CONTAIN A PROCESS NAME (PADDED WITH NULLS), WHICH WE WILL OUTPUT. THERE ARE ALSO 8 ADDITIONAL WORDS OF INFORMATION (16. BYTES) WHICH THE PROGRAM WILL NOT OUTPUT, BUT MUST SKIP OVER TO GET TO THE NEXT RECORD.

EDIT YOUR COPY OF THE SOURCE FILE "DYNIO.F77". FILL IN THE PACKET NAMES INTO IAC2 AND THE I/O SYSTEM CALLS.

FILL IN THE FOLLOWING INFORMATION IN THE PACKETS:

I/O PACKET FOR FILE 'SYSLOG'

OFFSET	OPTIONS OR VALUE
?ICH	(CHANNEL NUMBER WILL BE RETURNED BY SYSTEM)
?ISTI	CHANGE RECORD FORMAT TO DYNAMIC, INPUT ONLY
?ISTO	(FILE TYPE WILL BE RETURNED BY SYSTEM)
?IMRS	DEFAULT PHYSICAL BLOCK SIZE
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	MAX RECORD LENGTH SAME AS NUMBER OF BYTES IN "BUFF"
?IRLR	(RECORD LENGTH WILL BE RETURNED BY SYSTEM)
?IRNH	(WILL BE SET BY PROGRAM AT EACH ?READ)
?IFNP	BYTE POINTER TO FILENAME "@DATA"
?IDEL	DEFAULT DELIMITER TABLE

I/O PACKET FOR FILE '@OUTPUT'

OFFSET	OPTIONS OR VALUE
?ICH	(CHANNEL NUMBER WILL BE RETURNED BY SYSTEM)
?ISTI	CHANGE RECORD FORMAT TO FIXED, RECREATE FOR OUTPUT
?ISTO	(FILE TYPE WILL BE RETURNED BY SYSTEM)
?IMRS	DEFAULT PHYSICAL BLOCK SIZE
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	33 BYTE FIXED RECORDS
?IRLR	(RECORD LENGTH WILL BE RETURNED BY SYSTEM)
?IRNH	NEXT SEQUENTIAL RECORD
?IFNP	BYTE POINTER TO FILENAME "@OUTPUT"
?IDEL	DEFAULT DELIMITER TABLE

COMPILE THE MAIN PROGRAM AND THE SUBPROGRAM 'IVAL' TOGETHER. THIS PRODUCES SEPARATE OBJECT FILES, BUT A SINGLE LISTING. LINK THE MAIN PROGRAM TOGETHER WITH THE SUBPROGRAM 'IVAL'.

EXECUTE THE PROGRAM. YOU SHOULD SEE A BUNCH OF PROCESS NAMES ON YOUR SCREEN.

EACH PROCESS NAME WAS OUTPUT AS A FIXED LENGTH 33. BYTE RECORD. WHY DO YOU SUPPOSE THEY DON'T APPEAR TO BE THE SAME LENGTH ?

TO SEE EXACTLY WHAT IS OUTPUT, IT IS NECESSARY TO HAVE THE OUTPUT GO TO A DISK FILE INSTEAD OF A CONSOLE SCREEN.

CREATE AN EMPTY FILE CALLED DYNIO.OUT .

RUN YOUR PROGRAM AGAIN USING THE CLI COMMAND:

```
) PROC/BLOCK/DEF/OUT=DYNIO.OUT, DYNIO
```

NOW WE CAN EXAMINE THE CONTENTS OF THE OUTPUT FILE USING THE DISPLAY UTILITY:

```
) X, DISPLAY, DYNIO.OUT
```

THE FIRST COLUMN CONTAINS THE WORD ADDRESS OF THE FIRST WORD IN EACH LINE. EACH LINE CONTAINS 8. WORDS (2 CHARACTERS PER WORD) SHOWN IN OCTAL.

THE RIGHT COLUMNS SHOW THE TEXT CHARACTERS WITH NON-PRINTING CHARACTERS REPLACED BY PERIODS. WAS EACH RECORD 33. BYTES LONG AS EXPECTED ? WHAT NON-PRINTING CHARACTERS ARE CONTAINED IN EACH RECORD ? WHERE DID THEY COME FROM ?

****
CHALLENGE 1 -
****

WRITE A PROGRAM TO READ A FILE OF FIXED RECORDS LAST-LINE-FIRST AND WRITE THE RECORDS TO ANOTHER FILE IN REVERSE OF THEIR ORIGINAL ORDER.

THE FILE 'BOTTOM_UP' WHICH CONTAINS FIXED RECORDS OF 80. CHARACTERS, WRITTEN FIRST-LINE-LAST, IS PROVIDED IN THE :S309VS:F77 DIRECTORY FOR TESTING YOUR PROGRAM.

****
CHALLENGE 2 -
****

WRITE A PROGRAM TO READ A FILE OF FIXED LENGTH 80 BYTE RECORDS, STRIPPING TRAILING BLANKS FROM EACH RECORD BY REPLACING THE FIRST BLANK AFTER THE LAST SIGNIFICANT CHARACTER WITH A NEW-LINE, AND WRITE THE RECORDS TO A DATA SENSITIVE FILE.

THE FILE 'PADDED.TXT' WHICH CONTAINS FIXED RECORDS OF 80. CHARACTERS, PADDED WITH TRAILING SPACES, IS PROVIDED IN THE :S309VS:F77 DIRECTORY FOR TESTING YOUR PROGRAM.

```
      PROGRAM DYNIO

%LIST(OFF)
%INCLUDE 'IO_QSYM.F77.IN'
%LIST(ON)

      INTEGER*2 LOGPK(0:ISYS_IBLT-1)
      INTEGER*2 OUPKT(0:ISYS_IBLT-1)

      INTEGER*4 LOGPK_IBAD,LOGPK_IRNH,LOGPK_IFNP,LOGPK_IDEL
      EQUIVALENCE (LOGPK(ISYS_IBAD),LOGPK_IBAD)
      EQUIVALENCE (LOGPK(ISYS_IRNH),LOGPK_IRNH)
      EQUIVALENCE (LOGPK(ISYS_IFNP),LOGPK_IFNP)
      EQUIVALENCE (LOGPK(ISYS_IDEL),LOGPK_IDEL)

      INTEGER*4 OUPKT_IBAD,OUPKT_IRNH,OUPKT_IFNP,OUPKT_IDEL
      EQUIVALENCE (OUPKT(ISYS_IBAD),OUPKT_IBAD)
      EQUIVALENCE (OUPKT(ISYS_IRNH),OUPKT_IRNH)
      EQUIVALENCE (OUPKT(ISYS_IFNP),OUPKT_IFNP)
      EQUIVALENCE (OUPKT(ISYS_IDEL),OUPKT_IDEL)

      INTEGER*4 IWRDLN
      INTEGER*2 ICODE
      CHARACTER*33 BUFF
      BUFF(33:33) = '<NL>'

C     PARAMETER PACKET FOR FILE SYSLOG

      LOGPK(ISYS_ICH)  = ?
      LOGPK(ISYS_ISTI) = ISYS_????+ISYS_????+ISYS_????
      LOGPK(ISYS_ISTO) = ?
      LOGPK(ISYS_IMRS) = -1
      LOGPK_IBAD = ????????(????)
      LOGPK(ISYS_IRES) = ?
      LOGPK(ISYS_IRCL) = ?
      LOGPK(ISYS_IRLR) = ?
      LOGPK(ISYS_IRNW) = ?
      LOGPK_IRNH = ?
      LOGPK_IFNP = ????????('SYSLOG')
      LOGPK_IDEL = ??

C     PARAMETER PACKET FOR FILE @OUTPUT

      OUPKT(ISYS_ICH) = ?
      OUPKT(ISYS_ISTI) =
     +          ISYS_????+ISYS_????+ISYS_????+ISYS_????+ISYS_????
      OUPKT(ISYS_ISTO) = ?
      OUPKT(ISYS_IMRS) = -1
      OUPKT_IBAD = ????????(????)
      OUPKT(ISYS_IRES) = ?
      OUPKT(ISYS_IRCL) = ??
      OUPKT(ISYS_IRLR) = ?
      OUPKT(ISYS_IRNW) = ?
      OUPKT_IRNH = ?
      OUPKT_IFNP = ????????('@OUTPUT')
      OUPKT_IDEL = ??
```

```
C       OPEN SYSLOG

        IAC2 = WORDADDR(?????)
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       OPEN @OUTPUT

        IAC2 = WORDADDR(?????)
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999


C       READ A 16 BYTE RECORD HEADER FROM SYSLOG

500     LOGPK(ISYS_IRCL) = 16
        IAC2 = WORDADDR(?????)
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 900

C       GET THE RECORD'S WORD LENGTH AND TYPE CODE

        IWRDLN = IVAL(BUFF,1,4)
        ICODE =  IVAL(BUFF,11,12)

        IF (ICODE.NE.3) THEN    ! IF THIS IS NOT A CODE 3 RECORD,

           LOGPK_IRNH = 2*(IWRDLN-8) ! SKIP THIS ENTIRE RECORD.

        ELSE                    ! ELSE IF THIS IS A CODE 3 RECORD,

           LOGPK(ISYS_IRCL) = 32 ! READ A 32 BYTE PROCNAME ...
           LOGPK_IRNH = 0        ! STARTING AT THE NEXT BYTE ...
           IAC2 = WORDADDR(?????)
           IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
           IF (IER.NE.0) GOTO 900

           IAC2 = WORDADDR(?????) ! WRITE IT TO @OUTPUT ...
           IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
           IF (IER.NE.0) GOTO 999

           LOGPK_IRNH = 16       ! AND SKIP THE REMAINING 16
                                 ! BYTES OF THIS RECORD.
        ENDIF

C       GO READ THE NEXT RECORD HEADER

        GOTO 500
```

```
900     IF (IAC0.NE.ISYS_EREOF) GOTO 999

        BUFF = 'IO COMPLETED'
        IAC1 = BYTEADDR(BUFF)
        IAC2 = ISYS_RFCF+LEN('IO COMPLETED')

        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

999     IAC2 = ISYS_RFEC+ISYS_RFCF+ISYS_RFAB

        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

FORTRAN 77 SCREEN MANAGEMENT LAB

* MOVE/R/NACL A COPY OF THE FILES 'SCREEN.F77' AND 'IO_QSYM.F77.IN' FROM THE :S309VS:F77 DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'SCREEN' IS TO PRESENT A DATA ENTRY FORM ON THE SCREEN AND ALLOW THE USER TO ENTER THE REQUIRED DATA ONE FIELD AT A TIME. WHEN THE USER COMPLETES ALL THE FIELDS, HE IS ASKED IF HE WANTS TO CHANGE ANY OF THEM. IF SO, HE IS ASKED FOR THE FIELD NUMBER AND ALLOWED TO EDIT THE FIELD. WHEN THE USER HAS NO MORE CHANGES, HE IS ASKED IF HE WISHES TO FILL IN ANOTHER DATA ENTRY SCREEN.

* REFER TO THE PRINTOUT OF 'SCREEN.F77' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM CALLS AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
F77 SCREEN
F77LINK SCREEN
```

* GET A HARDCOPY OF THE FILE 'SCREEN.LS'.

* EXECUTE THE SCREEN PROGRAM. YOU SHOULD SEE A BLANK DATA ENTRY FORM ON YOUR SCREEN.

* USING IMAGINARY NAMES AND ADDRESSES, EXPERIMENT WITH FILLING IN THE FIELDS. NOTICE THAT THE FIELDS ARE AUTO-TERMINATING.

* STUDY THE PROGRAM TO UNDERSTAND HOW IT MAKES USE OF THE SCREEN MANAGEMENT EXTENSION AND AUTO-TERMINATING READS.

* TRY ENTERING INVALID RESPONSES TO THE Y/N QUESTIONS AND/OR THE FIELD NUMBER QUESTION. NOTICE HOW THE PROGRAM ACCOMPLISHES THE SAVING OF THE INPUT CURSOR POSITION WHILE IT DISPLAYS THE ERROR MESSAGES.

* YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE PROGRAM REPEATEDLY WITHOUT ERROR AND UNDERSTAND ITS USE OF SCREEN MANAGEMENT AND AUTO-TERMINATING READS.

```
        PROGRAM SCREEN
%LIST(OFF)
%INCLUDE 'IO_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 INPKT(0:ISYS_IBLT-1)
        INTEGER*2 OUPKT(0:ISYS_IBLT-1)
        INTEGER*2 SCPKT(0:2)

        INTEGER*4 INPKT_IBAD,INPKT_IRNH,INPKT_IFNP,INPKT_IDEL
        INTEGER*4 INPKT_ETSP,INPKT_ETFT,INPKT_ETLT,INPKT_ENET
        EQUIVALENCE (INPKT(ISYS_IBAD),INPKT_IBAD)
        EQUIVALENCE (INPKT(ISYS_IRNH),INPKT_IRNH)
        EQUIVALENCE (INPKT(ISYS_IFNP),INPKT_IFNP)
        EQUIVALENCE (INPKT(ISYS_IDEL),INPKT_IDEL)
        EQUIVALENCE (INPKT(ISYS_ETSP),INPKT_ETSP)
        EQUIVALENCE (INPKT(ISYS_ETFT),INPKT_ETFT)
        EQUIVALENCE (INPKT(ISYS_ETLT),INPKT_ETLT)
        EQUIVALENCE (INPKT(ISYS_ENET),INPKT_ENET)

        INTEGER*4 OUPKT_IBAD,OUPKT_IRNH,OUPKT_IFNP,OUPKT_IDEL
        INTEGER*4 OUPKT_ETSP,OUPKT_ETFT,OUPKT_ETLT,OUPKT_ENET
        EQUIVALENCE (OUPKT(ISYS_IBAD),OUPKT_IBAD)
        EQUIVALENCE (OUPKT(ISYS_IRNH),OUPKT_IRNH)
        EQUIVALENCE (OUPKT(ISYS_IFNP),OUPKT_IFNP)
        EQUIVALENCE (OUPKT(ISYS_IDEL),OUPKT_IDEL)
        EQUIVALENCE (OUPKT(ISYS_ETSP),OUPKT_ETSP)
        EQUIVALENCE (OUPKT(ISYS_ETFT),OUPKT_ETFT)
        EQUIVALENCE (OUPKT(ISYS_ETLT),OUPKT_ETLT)
        EQUIVALENCE (OUPKT(ISYS_ENET),OUPKT_ENET)

        INTEGER*2 MXFLD
        PARAMETER (MXFLD = 6)    ! LAST FIELD ON PAGE

        CHARACTER*23 CHMSG       ! CHANGE MESSAGE
        INTEGER*2 CHGCP          ! CURSOR POS. FOR CHANGE MESS.

        CHARACTER*14 WFMSG       ! WHICH FIELD? MESSAGE
        INTEGER*2 WFLCP          ! WHICH FIELD? CURSOR

        CHARACTER*3 YNBUFF       ! Y OR N BUFFER
        INTEGER*2 YNLN           ! BUFFER LENGTH
        PARAMETER (YNLN = 1)     ! 1 BYTE
        INTEGER*2 YNCP           ! CURSOR MSG POSITION
        CHARACTER*28 YNMSG       ! Y/N ERROR MSG
        INTEGER*2 YNCURS         ! HOLDS Y/N CURSOR POS

        CHARACTER*45 AGAINMSG
```

```
      CHARACTER*41 FBMSG       ! INVALID FIELD MESSAGE
      INTEGER*2 FNCURS         ! FIELD NUMBER CURSOR POSITION
      INTEGER*2 FLDNUM         ! EDIT FIELD NUMBER

C     FIELD TABLE OFFSETS
      INTEGER*2 FSCR,FBBP,FLLN,FLCR,FLBP,FELTH
      PARAMETER (FSCR = 0)     ! FIELD CURSOR POS (COL*400+ROW)
      PARAMETER (FBBP = FSCR+1) ! FIELD BUFFER BYTEPOINTER
      PARAMETER (FLLN = FBBP+2) ! FIELD BYTE LENGTH
      PARAMETER (FLCR = FLLN+1) ! LBL CURSOR POS (COL*400+ROW)
      PARAMETER (FLBP = FLCR+1) ! LABEL TEXT BYTEPOINTER
      PARAMETER (FELTH = FLCR+2+1) ! LENGTH OF A FIELD ENTRY

C     FIELD TABLE
      INTEGER*4 FTBL(0:6,0:FELTH-1)    ! 7 FIELD TABLE ENTRIES

C     FIELD LENGTHS
      INTEGER*2 FNMLN,MILN,LNMLN,STRLN,CTYLN,STALN,ZIPLN
      PARAMETER (FNMLN = 15)           ! FIRST NAME
      PARAMETER (MILN = 1)             ! MIDDLE INITIAL
      PARAMETER (LNMLN = 15)           ! LAST NAME
      PARAMETER (STRLN = 25)           ! STREET
      PARAMETER (CTYLN = 20)           ! CITY
      PARAMETER (STALN = 2)            ! STATE
      PARAMETER (ZIPLN = 5)            ! ZIP CODE
C     FIELD BUFFERS (FOR AUTO-TERMINATING READS)
      CHARACTER*27 FIRSTNAME
      CHARACTER*27 INITIAL
      CHARACTER*27 LASTNAME
      CHARACTER*27 STREET
      CHARACTER*27 CITY
      CHARACTER*27 STATE
      CHARACTER*27 ZIP
C     DEFINE WHOLE BUFFER
      CHARACTER*(27*7) FLDBUF
      EQUIVALENCE (FIRSTNAME(1:27),FLDBUF(1:27))
      EQUIVALENCE (INITIAL(1:27),FLDBUF(28:54))
      EQUIVALENCE (LASTNAME(1:27),FLDBUF(55:81))
      EQUIVALENCE (STREET(1:27),FLDBUF(82:108))
      EQUIVALENCE (CITY(1:27),FLDBUF(109:135))
      EQUIVALENCE (STATE(1:27),FLDBUF(136:162))
      EQUIVALENCE (ZIP(1:27),FLDBUF(163:189))
```

```
C   NOW BUILD THE FIELD TABLE
C   FIRST ENTRY - FIRST NAME
        FTBL(0,FSCR) = ISHFT4(20,8)+4    ! STARTING COL. AND ROW
        FTBL(0,FBBP) = BYTEADDR(FIRSTNAME)      ! BUFFER ADDRESS
        FTBL(0,FLLN) = FNMLN             ! BUFFER LENGTH
        FTBL(0,FLCR) = ISHFT4(20,8)+5    ! LABEL COL. AND ROW
        FTBL(0,FLBP) = BYTEADDR('1. First Name<0>')! LBL TXT PTR

C   SECOND ENTRY - MIDDLE INITIAL
        FTBL(1,FSCR) = ISHFT4(39,8)+4     ! STARTING COL. AND ROW
        FTBL(1,FBBP) = BYTEADDR(INITIAL) ! BUFFER ADDRESS
        FTBL(1,FLLN) = MILN               ! BUFFER LENGTH
        FTBL(1,FLCR) = ISHFT4(37,8)+5     ! LABEL COL. AND ROW
        FTBL(1,FLBP) = BYTEADDR('2. MI<0>') ! LABEL TEXT POINTER

C   THIRD ENTRY - LAST NAME
        FTBL(2,FSCR) = ISHFT4(45,8)+4      ! STARTING COL. AND ROW
        FTBL(2,FBBP) = BYTEADDR(LASTNAME) ! BUFFER ADDRESS
        FTBL(2,FLLN) = LNMLN               ! BUFFER LENGTH
        FTBL(2,FLCR) = ISHFT4(45,8)+5      ! LABEL COL. AND ROW
        FTBL(2,FLBP) = BYTEADDR('3. Last Name<0>')! LBL TXT PTR

C   FOURTH ENTRY - STREET
        FTBL(3,FSCR) = ISHFT4(25,8)+8      ! STARTING COL. AND ROW
        FTBL(3,FBBP) = BYTEADDR(STREET)    ! BUFFER ADDRESS
        FTBL(3,FLLN) = STRLN               ! BUFFER LENGTH
        FTBL(3,FLCR) = ISHFT4(29,8)+9      ! LABEL COL. AND ROW
        FTBL(3,FLBP) = BYTEADDR('4. Street Address<0>')! LBL PTR

C   FIFTH ENTRY - CITY
        FTBL(4,FSCR) = ISHFT4(20,8)+12   ! STARTING COL. AND ROW
        FTBL(4,FBBP) = BYTEADDR(CITY)    ! BUFFER ADDRESS
        FTBL(4,FLLN) = CTYLN             ! BUFFER LENGTH
        FTBL(4,FLCR) = ISHFT4(25,8)+13   ! LABEL COL. AND ROW
        FTBL(4,FLBP) = BYTEADDR('5. City<0>') ! LABEL PTR

C   SIXTH ENTRY - STATE
        FTBL(5,FSCR) = ISHFT4(46,8)+12   ! STARTING COL. AND ROW
        FTBL(5,FBBP) = BYTEADDR(STATE)          ! BUFFER ADDRESS
        FTBL(5,FLLN) = STALN             ! BUFFER LENGTH
        FTBL(5,FLCR) = ISHFT4(43,8)+13   ! LABEL COL. AND ROW
        FTBL(5,FLBP) = BYTEADDR('6. State<0>')   ! LABEL POINTER

C   SEVENTH ENTRY - ZIP
        FTBL(6,FSCR) = ISHFT4(55,8)+12   ! STARTING COL. AND ROW
        FTBL(6,FBBP) = BYTEADDR(ZIP)            ! BUFFER ADDRESS
        FTBL(6,FLLN) = ZIPLN             ! BUFFER LENGTH
        FTBL(6,FLCR) = ISHFT4(54,8)+13   ! LABEL COL. AND ROW
        FTBL(6,FLBP) = BYTEADDR('7. Zip<0>')     ! LABEL POINTER
```

```
C   INITIALIZE THE BUFFERS (END WITH 2 DELETE'S)
        FIRSTNAME = '                <177><177>'
        INITIAL = ' <177><177>'
        LASTNAME = '                <177><177>'
        STREET = '                          <177><177>'
        CITY = '                     <177><177>'
        STATE = '  <177><177>'
        ZIP = '     <177><177>'
        YNBUFF = ' <177><177>'

C       PARAMETER PACKET FOR FILE @INPUT
        INPKT(ISYS_ICH) = 0              ! CHANNEL NUM, RETURNED
        INPKT(ISYS_ISTI) = ISYS_????+ISYS_????+ISYS_????+ISYS_????
C       CHANGE TO DATASEN, INPUT, EXT PKT
        INPKT(ISYS_ISTO) = 0             ! FILE TYPE, RETURNED
        INPKT(ISYS_IMRS) = -1            ! DEFAULT PHYS. BLK SIZE
        INPKT_IBAD = 0              ! BYTE PTR, FILLED IN LATER
        INPKT(ISYS_IRES) = 0             ! RESERVED
        INPKT(ISYS_IRCL) = -1            ! MAX REC LENGTH
        INPKT(ISYS_IRLR) = 0             ! REAL REC LENGTH
        INPKT(ISYS_IRNW) = 0             ! RESERVED
        INPKT_IRNH = 0                   ! RECORD NUMBER
        INPKT_IFNP = BYTEADDR('@INPUT') ! FILENAME BYTE PTR
        INPKT_IDEL = -1                  ! DEFAULT DELIM. TABLE
        INPKT_ETSP = ????????(?????)
        INPKT_ETFT = 0
        INPKT_ETLT = 0
        INPKT_ENET = 0

        SCPKT(ISYS_????) =
      +  ISYS_????+ISYS_????+ISYS_????+ISYS_????+ISYS_????
C FLAGS: SCREEN EDIT, INITIAL CP, NO ECHO DELIM, RETURN CP, REDISPLAY
        SCPKT(ISYS_????) = 0     !  REL. CURSOR POS, NOT USED
        SCPKT(ISYS_????) = 0     ! INIT CURSOR POS, FILLED IN EACH READ

C       PARAMETER PACKET FOR FILE @OUTPUT

        OUPKT(ISYS_ICH) = 0      ! CHANNEL NUMBER, RETURNED
        OUPKT(ISYS_ISTI) = ISYS_????+ISYS_????+ISYS_????+ISYS_????
C FLAGS: CHG TO DATASEN, OUTPUT, EXT PKT
        OUPKT(ISYS_ISTO) = 0     ! FILE TYPE, RETURNED
        OUPKT(ISYS_IMRS) = -1    ! DEFAULT PHYS. BLOCK SIZE
        OUPKT_IBAD = -1          ! BUFFER BYTE POINTER
        OUPKT(ISYS_IRES) = 0     ! RESERVED
        OUPKT(ISYS_IRCL) = 80    ! MAX. RECORD LENGTH
        OUPKT(ISYS_IRLR) = 0     ! RECORD LENGTH
        OUPKT(ISYS_IRNW) = 0     ! RESERVED
        OUPKT_IRNH = 0           ! RECORD NUMBER
        OUPKT_IFNP = BYTEADDR('@OUTPUT')! FILENAME BYTE POINTER
        OUPKT_IDEL = -1          ! DEFAULT DELIMITER TABLE
        OUPKT_ETSP = ????????(?????)
        OUPKT_ETFT = 0
        OUPKT_ETLT = 0
        OUPKT_ENET = 0
```

```
C   CHANGE MESSAGE SETUP
        CHGCP = ISHFT4(20,8)+18          ! CURSOR POSITION
        CHMSG = 'Any changes (Y or N)? <0>'

C  WHICH FIELD? MESSAGE SETUP
        WFLCP = ISHFT4(45,8)+18          ! CURSOR POSITION
        WFMSG = 'What number? <0>'

C   Y/N BUFFER
        YNCP = ISHFT4(20,8)+20           ! CURSOR POSITION
        YNMSG = 'Please respond with Y or N<207><0>'

C   ANOTHER FORM?
        AGAINMSG = 'Do you wish to enter another form (Y or N)? <0>'

C   INVALID FIELD
        FBMSG = 'Please respond with a digit from 1 to 7<207><0>'
```

```
C    THE PROGRAM
C    OPEN @INPUT
         IAC0 = 0                              ! RESERVED
         IAC1 = 0                              ! RESERVED
         IAC2 = ????????(?????)
         IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
         IF (IER.NE.0) GOTO 999

C    OPEN @OUTPUT
         IAC0 = 0                              ! RESERVED
         IAC1 = 0                              ! RESERVED
         IAC2 = ????????(?????)
         IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
         IF (IER.NE.0) GOTO 999

C    CLEAR SCREEN
50       SCPKT(ISYS_????) = 0                  ! TOP OF SCREEN
         OUPKT_???? = WORDADDR(SCPKT)          ! SCREEN PACKET ADDRESS
         OUPKT_???? = IBSET4(OUPKT_????,31)       ! + 1 IN BIT 0
                                               ! IN WRITE PACKET
         OUPKT_???? = BYTEADDR('<14>')         ! FORM FEED
C    DO THE WRITE
         IAC0 = 0
         IAC1 = 0
         IAC2 = ????????(?????)
         IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
         IF (IER.NE.0) GOTO 999

C    DISPLAY BLANK FORM
         DO 75 FLDNUM = 0, MXFLD
            SCPKT(ISYS_????) = FTBL(FLDNUM,FLCR)  ! CURSOR POS FOR FLD
            OUPKT_???? = WORDADDR(SCPKT) ! SCREEN PACKET ADDRESS
            OUPKT_???? = IBSET4(OUPKT_????,31)    ! + 1 IN BIT 0
                                               ! IN WRITE PACKET
            OUPKT_???? = FTBL(FLDNUM,FLBP)  ! BYTE POINTER TO LABEL
            IAC0 = 0
            IAC1 = 0
            IAC2 = ????????(?????)
            IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
            IF (IER.NE.0) GOTO 999
75       CONTINUE
```

```
C   FILL FIELDS WITH INPUT
        DO 100 FLDNUM = 0, MXFLD

C   START WITH UNDERSCORES
            IPOS = FLDNUM*27+1
            DO 85 I = IPOS, IPOS+FTBL(FLDNUM,FLLN)-1
                 FLDBUF(I:I) = '_'
85          CONTINUE

C GET READY TO READ A FIELD
            SCPKT(ISYS_????) = FTBL(FLDNUM,FSCR) ! STORE POS IN PKT
            INPKT_???? = WORDADDR(SCPKT) ! SCPKT ADDR IN INPKT
            INPKT_???? = IBSET4(INPKT_????,31)! TURN ON BIT 0
            INPKT_???? = FTBL(FLDNUM,FBBP)       ! STORE BYTE POINTER
            INPKT(ISYS_????) = FTBL(FLDNUM,FLLN)+2! AND LENGTH,
                                               ! WITH 2 BYTES FOR AUTO-TERM

C   THE READ
            IAC0 = 0
            IAC1 = 0
            IAC2 = ????????(?????)
            IER = ISYS(ISYS_????, IAC0, IAC1, IAC2)
            IF (IER.NE.0) GOTO 999
100     CONTINUE

C   CHANGE A FIELD IF DESIRED
150     SCPKT(ISYS_ESCR) = CHGCP          ! CURS POS FOR CHG-FLD MESS
        OUPKT_???? = WORDADDR(SCPKT)      ! SCREEN PACKET ADDRESS
        OUPKT_???? = IBSET4(OUPKT_????,31)      ! + 1 IN BIT 0
                                          ! IN WRITE PACKET
        OUPKT_???? = BYTEADDR(CHMSG)      ! MESSAGE BYTE POINTER
C   DO THE WRITE
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999
        YNCURS = SCPKT(ISYS_????)         ! SAVE THE CURSOR POSITION
                                          ! IN CASE OF Y/N ERROR

C   WANT ANOTHER?  Y OR N
160     YNBUFF(1:1) = ' '                 ! PUT SPACE IN Y-N BUFFER
        INPKT_???? = WORDADDR(SCPKT)      ! SCPKT ADDR IN INPKT
        INPKT_???? = IBSET4(INPKT_????,31)! TURN ON BIT 0
        INPKT_???? = BYTEADDR(YNBUFF)     ! STORE BYTE POINTER
        INPKT(ISYS_????) = YNLEN+2        ! AND LENGTH,
                                          ! WITH 2 BYTES FOR AUTO-TERM
```

```
C   READ THE Y/N BUFFER
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_????, IAC0, IAC1, IAC2)
        IF (IER.NE.0) GOTO 999

        IF (YNBUFF(1:1).EQ.'N') GOTO 200        ! IF NO CHANGES, NEXT
                                                ! FORM, IF ANY
        IF (YNBUFF(1:1).NE.'Y') GOTO 175        ! 175 IF INVALID RESP.

C   FIND OUT WHICH FIELD
        SCPKT(ISYS_????) = WFLCP          ! OUTPUT FIELD MESSAGE
        OUPKT_???? = WORDADDR(SCPKT)      ! SCREEN PACKET ADDRESS
        OUPKT_???? = IBSET4(OUPKT_????,31)      ! + 1 IN BIT 0
                                          ! IN WRITE PACKET
        OUPKT_???? = BYTEADDR(WFMSG)      ! WHICH FIELD? MESSAGE
C   DO THE WRITE
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999
        FNCURS = SCPKT(ISYS_ESCR)         ! SAVE THE CURSOR POSITION
                                          ! IN CASE OF INVALID FIELD #

C   READ A FIELD NUMBER
163     YNBUFF(1:1) = ' '                 ! INIT BUFFER TO BLANK
        INPKT_???? = WORDADDR(SCPKT)      ! SCPKT ADDR IN
INPKT
        INPKT_???? = IBSET4(INPKT_????,31)! TURN ON BIT 0
        INPKT_???? = BYTEADDR(YNBUFF)     ! STORE BYTE POINTER
        INPKT(ISYS_????) = YNLN+2           ! AND LENGTH,
                                          ! WITH 2 BYTES FOR AUTO-TERM

C   THE READ
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_????, IAC0, IAC1, IAC2)
        IF (IER.NE.0) GOTO 999
        FLDNUM = ICHAR2(YNBUFF(1:1))-48-1    ! GET THE FIELD NUMBER
                                             ! (48 = ASCII-NUM CONV.)
                                             ! (-1 = START WITH FLD 0)
        IF ((FLDNUM.LT.0).OR.(FLDNUM.GT.MXFLD)) GOTO 165
                                             ! TAKE INV-FIELD PATH

C GET READY TO READ A FIELD
        SCPKT(ISYS_????) = FTBL(FLDNUM,FSCR) ! STORE  POS IN SCREEN PKT
        INPKT_???? = WORDADDR(SCPKT)      ! SCPKT ADDR IN
INPKT
        INPKT_???? = IBSET4(INPKT_????,31)! TURN ON BIT 0
        INPKT_???? = FTBL(FLDNUM,FBBP)    ! STORE BYTE POINTER
        INPKT(ISYS_????) = FTBL(FLDNUM,FLLN)+2! AND LENGTH,
                                          ! WITH 2 BYTES FOR AUTO-TERM
```

```
C   THE READ
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_????, IAC0, IAC1, IAC2)
        IF (IER.NE.0) GOTO 999
        GOTO 150

C   INVALID FORM NUMBER
165     SCPKT(ISYS_????) = YNCP          ! POSITION OF ERROR MSG
        OUPKT_???? = WORDADDR(SCPKT)     ! SCREEN PACKET ADDRESS
        OUPKT_???? = IBSET4(OUPKT_ETSP,31)      ! + 1 IN BIT 0
                                         ! IN WRITE PACKET
        OUPKT_???? = BYTEADDR(FBMSG)     ! MESSAGE BYTE POINTER
C   DO THE WRITE
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999
        SCPKT(ISYS_????) = FNCURS        ! RESTORE THE CURSOR POSITION
        GOTO 163                         ! TRY AGAIN

C  NEITHER Y NOR N
C  DISPLAY TRY-AGAIN MSG
175     SCPKT(ISYS_????) = YNCP          ! POSITION OF ERROR MSG
        OUPKT_???? = WORDADDR(SCPKT)     ! SCREEN PACKET ADDRESS
        OUPKT_???? = IBSET4(OUPKT_????,31)      ! + 1 IN BIT 0
                                         ! IN WRITE PACKET
        OUPKT_???? = BYTEADDR(YNMSG)     ! MESSAGE BYTE POINTER
C   DO THE WRITE
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999
        SCPKT(ISYS_????) = YNCURS        ! RESTORE THE CURSOR POS
        GOTO 160                         ! TRY AGAIN

C  ANOTHER FORM?
200     SCPKT(ISYS_????) = CHGCP         ! GET QUESTION CURS. POS.
        OUPKT_???? = WORDADDR(SCPKT)     ! SCREEN PACKET ADDRESS
        OUPKT_???? = IBSET4(OUPKT_????,31)      ! + 1 IN BIT 0
                                         ! IN WRITE PACKET
        OUPKT_???? = BYTEADDR(AGAINMSG)  ! AGAIN? MSG
C   DO THE WRITE
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999
        YNCURS = SCPKT(ISYS_????)        ! SAVE THE CURSOR POSITION
                                         ! IN CASE OF Y/N ERROR
```

```
C   WANT ANOTHER?  Y OR N
260     YNBUFF(1:1) = ' '                   ! PUT SPACE IN Y-N BUFFER
        INPKT_???? = WORDADDR(SCPKT)        ! SCPKT ADDR IN
INPKT
        INPKT_???? = IBSET4(INPKT_????,31)! TURN ON BIT 0
        INPKT_???? = BYTEADDR(YNBUFF)       ! STORE BYTE POINTER
        INPKT(ISYS_????) = YNLEN+2          ! AND LENGTH,
                                            ! WITH 2 BYTES FOR AUTO-TERM

C   READ THE Y/N BUFFER
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_????, IAC0, IAC1, IAC2)
        IF (IER.NE.0) GOTO 999

        IF (YNBUFF(1:1).EQ.'Y') GOTO 50          ! IF MORE, BACK TO BEG.
        IF (YNBUFF(1:1).EQ.'N') GOTO 900         ! IF NOT, THEN DONE
C  NEITHER Y NOR N
C  DISPLAY TRY-AGAIN MSG
        SCPKT(ISYS_????) = YNCP             ! POSITION OF ERROR MSG
        OUPKT_???? = WORDADDR(SCPKT)        ! SCREEN PACKET ADDRESS
        OUPKT_???? = IBSET4(OUPKT_????,31)       ! + 1 IN BIT 0
                                            ! IN WRITE PACKET
        OUPKT_???? = BYTEADDR(YNMSG)        ! ERROR MSG
C   DO THE WRITE
        IAC0 = 0
        IAC1 = 0
        IAC2 = ????????(?????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999
        SCPKT(ISYS_????) = YNCURS           ! RESTORE THE CURSOR POS
        GOTO 260                            ! TRY AGAIN

C   GOOD END
900     IAC0 = 0
        IAC1 = 0
        IAC2 = 0
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

C   BAD END
999     IAC2 = ISYS_RFEC+ISYS_RFCF+ISYS_RFAB
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

```
*       MOVE/R/NACL COPIES OF THE FILES 'PROC.F77' AND
        'PROC_QSYM.F77.IN' FROM THE :S309VS:ASSY DIRECTORY TO
        YOUR WORKING DIRECTORY.

*       THE FUNCTION OF 'PROC' IS TO CREATE A SUBORIDNATE PROCESS
        RUNNING THE CLI PROGRAM, AND TO PASS AN INNITIAL IPC
        MESSAGE CONTAINING A CLI COMMAND TO THE NEW PROCESS.  THE
        NEW PROCESS IS TO EXECUTE THE CLI COMMAND AND IMMEDIATELY
        TERMINATE.

*       REFER TO THE PRINTOUT OF 'PROC.F77' ON THE FOLLOWING
        PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?)
        WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM
        CALLS AND THE CONTENTS OF THE PACKETS.

*       EDIT YOUR SOURCE FILE AND MAKE THE CORRECTIONS.

*       COMPILE AND LINK YOUR PROGRAM.

*       GET A HARDCOPY OF THE FILE 'PROC.LS'.

*       EXECUTE THE PROC PROGRAM. YOUR ORIGINAL CLI PROCESS
        BECOMES BLOCKED UNTIL THE 'PROC' PROGRAM TERMINATES.  THE
        'PROC' PROGRAM CREATES THE SUBORDINATE CLI AND THE 'PROC'
        PROCESS BECOMES BLOCKED WAITING FOR THE NEW CLI TO
        TERMINATE.  THE NEW CLI EXECUTES THE COMMAND IN THE IPC
        MESSAGE ASSOCIATED WITH THE ?PROC PACKET AND THEN
        TERMINATES.  THEN THE 'PROC' PROCESS BECOMES UNBLOCKED
        AND TERMINATES ITSELF, WHICH NOW UNBLOCKS YOUR ORIGINAL
        CLI PROCESS.

*       USE THE CLI TYPE COMMAND TO EXAMINE THE CONTENTS OF
        'OUTFILE'. IT SHOULD CONTAIN THE MESSAGE "HELLO FROM YOUR
        CREATOR".

*       YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE
        PROGRAM REPEATEDLY WITHOUT ERROR

************************************************************
*                      CHALLENGE                           *
************************************************************

*       MODIFY THE PROGRAM SO THAT THE NEW CLI DOES NOT TERMINATE
        AFTER COMPLETING THE CLI COMMAND IN THE IPC MESSAGE, BUT
        GIVES THE CLI PROMPT INSTEAD.  YOU WILL NEED TO PASS
        ?RFCF (CLI FORMAT) IN THE USER FLAGS WORD, AND TO ADD
        'CLI,' TO THE BEGINNING OF THE IPC MESSAGE TEXT STRING.
```

```
        PROGRAM PROC

%LIST(OFF)
%INCLUDE 'PROC_QSYM.F77.IN'
%INCLUDE 'IPC_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 PPKT(0:ISYS_PLTH-1)    ! PROCESS CREATION PACKET
        INTEGER*4 PPKT_PSNM
        EQUIVALENCE(PPKT_PSNM,PPKT(ISYS_PSNM))
        INTEGER*4 PPKT_PIPC
        EQUIVALENCE(PPKT_PIPC,PPKT(ISYS_PIPC))
        INTEGER*4 PPKT_PNM
        EQUIVALENCE(PPKT_PNM,PPKT(ISYS_PNM))
        INTEGER*4 PPKT_PMEM
        EQUIVALENCE(PPKT_PMEM,PPKT(ISYS_PMEM))
        INTEGER*4 PPKT_PDIR
        EQUIVALENCE(PPKT_PDIR,PPKT(ISYS_PDIR))
        INTEGER*4 PPKT_PCON
        EQUIVALENCE(PPKT_PCON,PPKT(ISYS_PCON))
        INTEGER*4 PPKT_PUNM
        EQUIVALENCE(PPKT_PUNM,PPKT(ISYS_PUNM))
        INTEGER*4 PPKT_PIFP
        EQUIVALENCE(PPKT_PIFP,PPKT(ISYS_PIFP))
        INTEGER*4 PPKT_POFP
        EQUIVALENCE(PPKT_POFP,PPKT(ISYS_POFP))
        INTEGER*4 PPKT_PLFP
        EQUIVALENCE(PPKT_PLFP,PPKT(ISYS_PLFP))
        INTEGER*4 PPKT_PDFP
        EQUIVALENCE(PPKT_PDFP,PPKT(ISYS_PDFP))
        INTEGER*4 PPKT_SMCH
        EQUIVALENCE(PPKT_SMCH,PPKT(ISYS_SMCH))

        INTEGER*2 HDR(0:ISYS_IPLTH-1)    ! IPC HEADER
        INTEGER*4 HDR_IDPH
        EQUIVALENCE(HDR_IDPH,HDR(ISYS_IDPH))
        INTEGER*4 HDR_IPTR
        EQUIVALENCE(HDR_IPTR,HDR(ISYS_IPTR))
```

```
      PPKT(ISYS_PFLG) = ISYS_???? ! BLOCK CREATOR, SWAPPABLE
      PPKT(ISYS_PPRI) = ??        ! SAME PRIORITY AS CREATOR
      PPKT_PSNM = BYTEADDR('CLI.PR<0>')
      PPKT_PIPC = ????????(HDR)   ! IPC HEADER
      PPKT_PNM  = ??              ! DEFAULT PROCESSNAME
      PPKT_PMEM = ??              ! DEFAULT MAX MEMORY
      PPKT_PDIR = 0           ! SAME DIR AS CREATOR'S CURRENT
      PPKT_PCON = ??          ! SAME AS CREATOR'S @CONSOLE
      PPKT(ISYS_PCAL) = ??    ! DEFAULT CONCURRENT CALLS
      PPKT(ISYS_PWSS) = ??    ! DEFAULT MAX WSS
      PPKT_PUNM = ??          ! SAME AS CREATOR'S USERNAME
      PPKT(ISYS_PPRV) = ??    ! SAME PRIVILEGES AS CREATOR
      PPKT(ISYS_PPCR) = ??            ! REMAINDER OF SONS QUOTA
      PPKT(ISYS_PWMI) = ??            ! DEFAULT MIN WSS
      PPKT_PIFP = ??                  ! SAME @INPUT AS CREATOR
      PPKT_POFP = ??                  ! SAME @OUTPUT AS CREATOR
      PPKT_PLFP = ??                  ! SAME @LIST AS CREATOR
      PPKT_PDFP = ??                  ! SAME @DATA AS CREATOR
      PPKT_SMCH = ??                  ! REMAINDER OF CPU LIMIT

      HDR(ISYS_ISFL) = 0              ! NO SYSTEM FLAGS
      HDR(ISYS_IUFL) = ??     ! NO USER FLAGS = DO CMD, TERM
      HDR_IDPH = ??
      HDR(ISYS_IOPN) = ??
      HDR(ISYS_????) =
     +  LEN('WRITE/L=OUTFILE,HELLO,FROM,YOUR,CREATOR')
      HDR_???? =
     +  ???????('WRITE/L=OUTFILE,HELLO,FROM,YOUR,CREATOR')

      IAC0 = 0
      IAC1 = 0
      IAC2 = ????????(PPKT)
      IER = ISYS(ISYS_????,????,????,????)
      IF (IER.NE.0) GOTO 999

      IAC2 = 0
      IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

999   IAC0 = IER
      IAC2 = ISYS_RFCF+ISYS_RFEC+ISYS_RFAB
      IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
      IF (IER.NE.0) GOTO 999

      END
```

* MOVE/NACL A COPY OF THE FILE CONTROL.IPC.F77 AND IPC_QSYM.F77.IN FROM THE :S309VS:F77 DIRECTORY TO YOUR INITIAL WORKING DIRECTORY.

YOU SHOULD ALREADY HAVE MOVED A COPY OF IO_QSYM.F77.IN DURING A PREVIOUS LAB.

* REFER TO THE PRINTOUT OF CONTROL.IPC.SR ON THE FOLLOWING PAGES.

* THIS PROGRAM ACCEPTS ANY IPC MESSAGE SENT TO THE FILE 'TRYIPC' FROM ANY CONSOLE. TO DO THIS THE CONSOLE DOES NOT HAVE TO BE ASSOCIATED WITH THE RECEIVING PROCESS.

* WHAT IS THE CLI COMMAND WHICH ALLOWS YOU TO INTERACTIVELY SEND AN IPC MESSAGE TO A PROCESS?

---

* EACH MESSAGE RECEIVED IS DISPLAYED ON THE PROCESS' CONSOLE, THEN AN ACKNOWLEDGEMENT IS RETURNED TO THE SENDER.

* THIS PROGRAM ACCEPTS ANY IPC MESSAGE SENT TO THE FILE 'TRYIPC' FROM ANY CONSOLE. TO DO THIS THE CONSOLE DOES NOT HAVE TO BE ASSOCIATED WITH THE RECEIVING PROCESS.

* EACH MESSAGE RECEIVED IS DISPLAYED ON THE PROCESS' CONSOLE, THEN AN ACKNOWLEDGEMENT IS RETURNED TO THE SENDER.

* EDIT YOUR COPY OF THE SOURCE FILE AND FILL IN THE FOLLOWING:

  -- THE STATEMENT AT LABEL '500' TO SET THE GLOBAL PORT NUMBER IN THE ?IREC PACKET TO RECEIVE FROM ANY SENDER.

  -- THE NEXT STATEMENT WHICH RESETS THE BUFFER LENGTH SPEC IN THE ?IREC HEADER.

  -- AFTER THE MESSAGE HAS BEEN RECEIVED THE SENDER'S ACTUAL GLOBAL PORT NUMBER WILL BE PLACED IN THE ?IREC HEADER BY THE SYSTEM.

  -- AT THE COMMENT 'ACKNOWLEDGE THE MESSAGE', GET THIS GLOBAL PORT NUMBER FOR USE BY THE ?GPORT CALL.

PACKET NAME	
CIPC	- THE PACKET IS USED TO CREATE THE IPC FILE TRYIPC.
IPCPK	- THE ?IREC (PACKET) HEADER.

* COMPILE AND LINK THE PROGRAM.

* EXECUTE YOUR PROGRAM.

* LOG ON AT ANOTHER TERMINAL AND SEND MESSAGES TO YOUR PROGRAM:

) CONTROL TRYIPC YOUR MESSAGE GOES HERE

  EACH TIME YOU SHOULD GET THE RESPONSE:

  FROM PID XX: I GOT THE MESSAGE

  EACH MESSAGE WILL BE DISPLAYED ON YOUR RECEIVING PROCESS' CONSOLE. NOTE THAT THE CLI COMMAND USED TO SEND THESE MESSAGES EDITS THEM SOMEWHAT BEFORE THEY ARE SENT.

* WHEN YOU HAVE SENT AS MANY MESSAGES AS YOU WANT, ENTER CONTROL-C, CONTROL-B AT YOUR RECEIVING CONSOLE.

* ---------- CHALLENGE ----------

  MODIFY THE PROGRAM SO THAT IT WILL TERMINATE NORMALLY BY ITSELF IF IT RECEIVES A MESSAGE WHICH IS LESS THAN TWO WORDS (4 BYTES) IN LENGTH.

  (WE CHOSE TWO WORDS BECAUSE THE CLI WON'T LET US SEND A ZERO LENGTH MESSAGE.)

```
        PROGRAM CONTROL_IPC

%LIST(OFF)
%INCLUDE 'IPC_QSYM.F77.IN'
%INCLUDE 'IO_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 CREPK(0:ISYS_CLTH-1)

        INTEGER*4 CREPK_CTIM,CREPK_CACP
        EQUIVALENCE (CREPK(ISYS_CTIM),CREPK_CTIM)
        EQUIVALENCE (CREPK(ISYS_CACP),CREPK_CACP)

        INTEGER*2 IPCHDR(0:ISYS_IPLTH-1)

        INTEGER*4 IPCHDR_IOPH,IPCHDR_IDPH,IPCHDR_IPTR
        EQUIVALENCE (IPCHDR(ISYS_IOPH),IPCHDR_IOPH)
        EQUIVALENCE (IPCHDR(ISYS_IDPH),IPCHDR_IDPH)
        EQUIVALENCE (IPCHDR(ISYS_IPTR),IPCHDR_IPTR)

        INTEGER*2 OUPKT(0:ISYS_IBLT-1)

        INTEGER*4 OUPKT_IBAD,OUPKT_IRNH,OUPKT_IFNP,OUPKT_IDEL
        EQUIVALENCE (OUPKT(ISYS_IBAD),OUPKT_IBAD)
        EQUIVALENCE (OUPKT(ISYS_IRNH),OUPKT_IRNH)
        EQUIVALENCE (OUPKT(ISYS_IFNP),OUPKT_IFNP)
        EQUIVALENCE (OUPKT(ISYS_IDEL),OUPKT_IDEL)

        CHARACTER*136 BUFF
```

```
C       PACKET FOR IPC FILE CREATION

        CREPK(ISYS_CFTYP) = ISYS_????           ! FILE TYPE IPC
        CREPK(ISYS_CPOR)  = ??          ! ANY LOCAL PORT 1-2047
        CREPK_CTIM = ??                         ! DEFAULT TIME
        CREPK_CACP = ??                         ! DEFAULT ACL

C       IREC HEADER

        IPCHDR(ISYS_ISFL) = ? ! NONE. DO NOT SPOOL IF BUFF SHORT
        IPCHDR(ISYS_IUFL) = ?                   ! NO USER FLAGS
        IPCHDR_IOPH = ?              ! RCV FROM ANY SENDER
        IPCHDR(ISYS_IDPN) = ??       ! SAME PORT AS IN CREPK
        IPCHDR(ISYS_ILTH) = ?        ! BUFFLEN SET AT EA ?IREC
        IPCHDR_IPTR = ????????(????)! BUFFER ADDRESS

C       PARAMETER PACKET FOR FILE @OUTPUT

        OUPKT(ISYS_ICH) = 0
        OUPKT(ISYS_ISTI) = ISYS_ICRF+ISYS_RTDY+ISYS_OFOT
        OUPKT(ISYS_ISTO) = 0
        OUPKT_IBAD = BYTEADDR(BUFF)
        OUPKT(ISYS_IRES) = 0
        OUPKT(ISYS_IRCL) = 136
        OUPKT(ISYS_IRLR) = 0
        OUPKT(ISYS_IRNW) = 0
        OUPKT_IRNH = 0
        OUPKT_IFNP = BYTEADDR('@OUTPUT')
        OUPKT_IDEL = -1
```

```
C       CREATE IPC FILE FOR OTHER PROCESSES TO LOOK UP

        IAC0 = BYTEADDR('TRYIPC')
        IAC2 = WORDADDR(CREPK)
        IER = ISYS(ISYS_CREATE,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       OPEN @OUPUT

        IAC2 = WORDADDR(OUPKT)
        IER = ISYS(ISYS_OPEN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       WAIT FOR IPC MESSAGE FROM ANY SENDER

500     IPCHDR_IOPH = ?                 ! RECEIVE FROM ANY SENDER
        IPCHDR(ISYS_ILTH) = ???         ! SET BUFFER LENGTH
        ???? = ????????(??????)         ! IPC HDR ADDRESS
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0.AND.IER.NE.ISYS_?????) GOTO 999! IGNORE OVF

C       GET IPC MSG LEN, INSERT NL, AND WRITE TO @OUTPUT

        I = IPCHDR(ISYS_ILTH)*2         ! MESSAGE LENGTH IN BYTES
        BUFF(I:I) = '<NL>'              ! INSERT NEWLINE

        OUPKT(ISYS_IRCL) = I            ! NUMBER OF BYTES TO WRITE
        IAC2 = WORDADDR(OUPKT)
        IER = ISYS(ISYS_WRITE,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       ACKNOWLEDGE THE MESSAGE

        IAC1 = ??????_????              ! FROM SENDER'S GLOBAL PORT,
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)! GET SENDER'S PID.
        IF (IER.NE.0) GOTO 999

        IAC0 = IAC1                     ! SENDER'S PID

        ???? = BYTEADDR('I GOT THE MESSAGE<NL>')
        ???? = LEN('I GOT THE MESSAGE<NL>') ! SEND ACK TO
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)! SENDER'S CONSOLE
        IF (IER.NE.0) GOTO 999

C       GO WAIT FOR THE NEXT IPC MESSAGE

        GOTO 500

999     IAC2 = ISYS_RFCF+ISYS_RFEC+ISYS_RFAB
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

FORTRAN 77 TALK IPC LAB

USING THE FOLLOWING CLI COMMANDS, GET THE NEEDED FILES INTO YOUR DIRECTORY:

```
DIRECTORY, :S309VS:F77
MOVE/NACL, :UDD:YOURUSERNAME, TALK<ONE,TWO>.F77
DIRECTORY/I
```

YOU SHOULD ALREADY HAVE MOVED A COPY OF IO_QSYM.F77.IN AND IPC_QSYM.F77.IN DURING PREVIOUS LABS.
QPRINT A COPY OF EACH OF THE SOURCE FILES.

EXAMINE YOUR PRINTOUTS AND FILL IN THE ITEMS WHICH HAVE BEEN REPLACED WITH QUESTION MARKS. YOU WILL HAVE TO SUPPLY VARIOUS SYSTEM CALLS AND THEIR ARGUMENTS, ERROR CODE SYMBOLS, ADDRESSES OF VARIOUS ITEMS IN PACKETS, AND THE APPROPRIATE VALUES IN THOSE PACKETS.

WHEN YOU HAVE DETERMINED THE CORRECT INFORMATION FOR EACH OF THESE ITEMS, EDIT YOUR TWO SOURCE FILES AND MAKE THE CORRECTIONS.

COMPILE AND LINK EACH PROGRAM SEPERATELY.

LOG ON AT ANOTHER CONSOLE. EXECUTE TALKONE AT ONE CONSOLE AND TALKTWO AT THE SECOND. YOU SHOULD BE ABLE TO SEND MESSAGES BACK AND FORTH.

TO STOP EACH OF THESE PROGRAMS, ENTER CONTROL-D (END-OF-FILE).

********** CHALLENGE **********

WORKING WITH ANOTHER STUDENT USE THE PROGRAMS TO CONVERSE.

HINT:

WHEN YOU EXECUTED THE PROGRAMS BY YOURSELF, WHAT WAS IT THAT ENSURED YOU ONLY GOT YOUR OWN MESSAGES EVEN THOUGH OTHERS MAY HAVE BEEN EXECUTING THE SAME PROGRAMS ????

****

********** CHALLENGE **********

PREPARE A FILE FULL OF MESSAGES WHICH COMPRISE ONE HALF OF A CONVERSATION.

CAUSE TALKONE TO EXECUTE, SENDING THE MESSAGES FROM THIS FILE, AND STORING ANY MESSAGES IT MAY RECEIVE IN AN EMPTY FILE OF YOUR CHOICE.

EXECUTE TALKTWO AT YOUR CONSOLE, AND RESPOND TO THE MESSAGES COMING FROM YOUR TALKONE.

==========>> YOU MAY USE ONLY ONE CONSOLE !! <<=========

AFTER YOU HAVE RESPONDED TO THE LAST MESSAGE IN YOUR FILE, YOUR TALKTWO PROCESS SHOULD TERMINATE AUTOMATICALLY.
EXAMINE THE FILE IN WHICH TALKONE STORED THE MESSAGES IT RECEIVED.

****

PREPARE A SECOND FILE WITH THE RESPONSES TO THE MESSAGES IN THE FIRST FILE. CAUSE BOTH TALKONE AND TALKTWO TO EXECUTE, TAKING INPUT FROM THESE TWO MESSAGE FILES, AND STORING ANY RECEIVED MESSAGES IN TWO NEW EMPTY FILES OF YOUR CHOICE.

==========>> YOU MAY NOT USE ANY CONSOLE !! <<=========
(EXCEPT TO START AND STOP THE PROCESSES)

****

```
        PROGRAM TALKONE

%LIST(OFF)
%INCLUDE 'IO_QSYM.F77.IN'
%INCLUDE 'IPC_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 IPCHDR(0:ISYS_IPLTH-1)

        INTEGER*4 IPCHDR_IOPH,IPCHDR_IDPH,IPCHDR_IPTR
        EQUIVALENCE (IPCHDR(ISYS_IOPH),IPCHDR_IOPH)
        EQUIVALENCE (IPCHDR(ISYS_IDPH),IPCHDR_IDPH)
        EQUIVALENCE (IPCHDR(ISYS_IPTR),IPCHDR_IPTR)

        INTEGER*2 INPKT(0:ISYS_IBLT-1)
        INTEGER*2 OUPKT(0:ISYS_IBLT-1)

        INTEGER*4 INPKT_IBAD,INPKT_IRNH,INPKT_IFNP,INPKT_IDEL
        EQUIVALENCE (INPKT(ISYS_IBAD),INPKT_IBAD)
        EQUIVALENCE (INPKT(ISYS_IRNH),INPKT_IRNH)
        EQUIVALENCE (INPKT(ISYS_IFNP),INPKT_IFNP)
        EQUIVALENCE (INPKT(ISYS_IDEL),INPKT_IDEL)

        INTEGER*4 OUPKT_IBAD,OUPKT_IRNH,OUPKT_IFNP,OUPKT_IDEL
        EQUIVALENCE (OUPKT(ISYS_IBAD),OUPKT_IBAD)
        EQUIVALENCE (OUPKT(ISYS_IRNH),OUPKT_IRNH)
        EQUIVALENCE (OUPKT(ISYS_IFNP),OUPKT_IFNP)
        EQUIVALENCE (OUPKT(ISYS_IDEL),OUPKT_IDEL)

        CHARACTER*80 BUFF
```

```
C       IPC HEADER

        IPCHDR(ISYS_ISFL) = ?    ! NONE. DO NOT SPOOL IF BUFFER
SHORT
        IPCHDR(ISYS_IUFL) = 0    ! NO USER FLAGS
        IPCHDR_IOPH = ?      ! GLOBAL PORT FILLED IN AFTER ?ILKUP
        IPCHDR(ISYS_IDPN) = ??   ! ANY LOCAL PORT 1-2047
        IPCHDR(ISYS_ILTH) = ?    ! BUFFLEN FILLED IN AT EACH ?IREC
        IPCHDR_IPTR = ????????(????)     ! BUFFER ADDRESS


C       PARAMETER PACKET FOR FILE @INPUT

        INPKT(ISYS_ICH) = 0
        INPKT(ISYS_ISTI) = ISYS_ICRF+ISYS_RTDS+ISYS_OFIN
        INPKT(ISYS_ISTO) = 0
        INPKT_IBAD = BYTEADDR(BUFF)
        INPKT(ISYS_IRES) = 0
        INPKT(ISYS_IRCL) = 80
        INPKT(ISYS_IRLR) = 0
        INPKT(ISYS_IRNW) = 0
        INPKT_IRNH = 0
        INPKT_IFNP = BYTEADDR('@INPUT')
        INPKT_IDEL = -1

C       PARAMETER PACKET FOR FILE @OUTPUT

        OUPKT(ISYS_ICH) = 0
        OUPKT(ISYS_ISTI) = ISYS_ICRF+ISYS_RTDS+ISYS_OFOT
        OUPKT(ISYS_ISTO) = 0
        OUPKT_IBAD = BYTEADDR(BUFF)
        OUPKT(ISYS_IRES) = 0
        OUPKT(ISYS_IRCL) = 80
        OUPKT(ISYS_IRLR) = 0
        OUPKT(ISYS_IRNW) = 0
        OUPKT_IRNH = 0
        OUPKT_IFNP = BYTEADDR('@OUTPUT')
        OUPKT_IDEL = -1
```

```
C       OPEN @INPUT

        IAC2 = WORDADDR(INPKT)
        IER = ISYS(ISYS_OPEN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       OPEN @OUTPUT

        IAC2 = WORDADDR(OUPKT)
        IER = ISYS(ISYS_OPEN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999


C   LOOK UP OTHER PROCESS' IPC FILE, STORE GLOBAL PORT IN IPC HDR

100     ???? = ????????('FILEB')
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.EQ.ISYS_?????) GOTO 100
C   IF FILE DOES NOT EXIST, TRY AGAIN
        IF (IER.NE.0) GOTO 999
        IPCHDR_???? = IAC1      ! STORE PORT NUMBER IN IPC HDR

C       SET UP TO DISPLAY INITIAL PROMPT ON @OUTPUT

        BUFF = 'TYPE A MESSAGE AND WAIT FOR A RESPONSE...<NL>'
```

```
C       DISPLAY BUFFER CONTENTS ON @OUTPUT

200     IAC2 = WORDADDR(OUPKT)
        IER = ISYS(ISYS_WRITE,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       READ A LINE FROM @INPUT

        IAC2 = WORDADDR(INPKT)
        IER = ISYS(ISYS_READ,IAC0,IAC1,IAC2)
        IF (IER.EQ.ISYS_EREOF) GOTO 900
        IF (IER.NE.0) GOTO 999

        I = ?????(ISYS_????)        ! GET ITS BYTE LENGTH
        IF (MOD(I,2).NE.0) I = I+1 ! IF ITS ODD, MAKE IT EVEN
        I = I/2                     ! THEN MAKE IT A WORD LENGTH.

C       SEND THE LINE TO THE OTHER PROCESS AS AN IPC MESSAGE

        ??????(ISYS_????) = I           ! MESSAGE LENGTH
        ???? = ????????(??????)         ! IPC HDR ADDRESS
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       WAIT FOR AN IPC MESSAGE FROM THE OTHER PROCESS

        ??????(ISYS_????) = 40          ! SET BUFFER LENGTH
        ???? = ????????(??????)         ! IPC HDR ADDRESS
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       IF MESSAGE LENGTH WAS ZERO, EXIT

        IF (??????(ISYS_????).EQ.0) GOTO 950

        GOTO 200        ! OTHERWISE, WRITE THE MESSAGE ON @OUTPUT
```

```
C       GOT END-OF-FILE ... SEND ZERO LENGTH IPC MESSAGE

900     ??????(ISYS_????) = 0              ! ZERO LENGTH MESSAGE
        ???? = ????????(??????)            ! IPC HDR ADDRESS
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       NORMAL EXIT

950     IAC2 = 0
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

C       ERROR EXIT

999     IAC2 = ISYS_RFCF+ISYS_RFEC+ISYS_RFAB
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

```
        PROGRAM TALKTWO

%LIST(OFF)
%INCLUDE 'IO_QSYM.F77.IN'
%INCLUDE 'IPC_QSYM.F77.IN'
%LIST(ON)

        INTEGER*2 CREPK(0:ISYS_CLTH-1)
        INTEGER*4 CREPK_CTIM,CREPK_CACP
        EQUIVALENCE (CREPK(ISYS_CTIM),CREPK_CTIM)
        EQUIVALENCE (CREPK(ISYS_CACP),CREPK_CACP)

        INTEGER*2 IPCHDR(0:ISYS_IPLTH-1)

        INTEGER*4 IPCHDR_IOPH,IPCHDR_IDPH,IPCHDR_IPTR
        EQUIVALENCE (IPCHDR(ISYS_IOPH),IPCHDR_IOPH)
        EQUIVALENCE (IPCHDR(ISYS_IDPH),IPCHDR_IDPH)
        EQUIVALENCE (IPCHDR(ISYS_IPTR),IPCHDR_IPTR)

        INTEGER*2 INPKT(0:ISYS_IBLT-1)
        INTEGER*2 OUPKT(0:ISYS_IBLT-1)

        INTEGER*4 INPKT_IBAD,INPKT_IRNH,INPKT_IFNP,INPKT_IDEL
        EQUIVALENCE (INPKT(ISYS_IBAD),INPKT_IBAD)
        EQUIVALENCE (INPKT(ISYS_IRNH),INPKT_IRNH)
        EQUIVALENCE (INPKT(ISYS_IFNP),INPKT_IFNP)
        EQUIVALENCE (INPKT(ISYS_IDEL),INPKT_IDEL)

        INTEGER*4 OUPKT_IBAD,OUPKT_IRNH,OUPKT_IFNP,OUPKT_IDEL
        EQUIVALENCE (OUPKT(ISYS_IBAD),OUPKT_IBAD)
        EQUIVALENCE (OUPKT(ISYS_IRNH),OUPKT_IRNH)
        EQUIVALENCE (OUPKT(ISYS_IFNP),OUPKT_IFNP)
        EQUIVALENCE (OUPKT(ISYS_IDEL),OUPKT_IDEL)

        CHARACTER*80 BUFF
```

```
C       PACKET FOR CREATING IPC FILE

        CREPK(ISYS_CFTYP) = ISYS_????    ! FILE TYPE IPC
        CREPK(ISYS_CPOR)  = ??           ! ANY LOCAL PORT 1-2047
        CREPK_CTIM = ??                  ! DEFAULT TIME
        CREPK_CACP = ??                  ! DEFAULT ACL


C       IPC HEADER

        IPCHDR(ISYS_ISFL) = ? ! NONE. DO NOT SPOOL IF BUFF SHORT
        IPCHDR(ISYS_IUFL) = ?    ! NO USER FLAGS
        IPCHDR_IOPH = ?          ! RCV FROM ANY SENDER
        IPCHDR(ISYS_IDPN) = ??   ! SAME PORT AS IN CREPK
        IPCHDR(ISYS_ILTH) = ?    ! BUFFLEN FILLED IN AT EACH ?IREC
        IPCHDR_IPTR = ????????(????)     ! BUFFER ADDRESS


C       PARAMETER PACKET FOR FILE @INPUT

        INPKT(ISYS_ICH) = 0
        INPKT(ISYS_ISTI) = ISYS_ICRF+ISYS_RTDS+ISYS_OFIN
        INPKT(ISYS_ISTO) = 0
        INPKT_IBAD = BYTEADDR(BUFF)
        INPKT(ISYS_IRES) = 0
        INPKT(ISYS_IRCL) = 80
        INPKT(ISYS_IRLR) = 0
        INPKT(ISYS_IRNW) = 0
        INPKT_IRNH = 0
        INPKT_IFNP = BYTEADDR('@INPUT')
        INPKT_IDEL = -1

C       PARAMETER PACKET FOR FILE @OUTPUT

        OUPKT(ISYS_ICH) = 0
        OUPKT(ISYS_ISTI) = ISYS_ICRF+ISYS_RTDS+ISYS_OFOT
        OUPKT(ISYS_ISTO) = 0
        OUPKT_IBAD = BYTEADDR(BUFF)
        OUPKT(ISYS_IRES) = 0
        OUPKT(ISYS_IRCL) = 80
        OUPKT(ISYS_IRLR) = 0
        OUPKT(ISYS_IRNW) = 0
        OUPKT_IRNH = 0
        OUPKT_IFNP = BYTEADDR('@OUTPUT')
        OUPKT_IDEL = -1
```

```
C       CREATE IPC FILE FOR OTHER PROCESS TO LOOK UP

        ???? = ????????('FILEB')
        ???? = ????????(CREPK)
        IER = ISYS(ISYS_??????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       OPEN @INPUT

        IAC2 = WORDADDR(INPKT)
        IER = ISYS(ISYS_OPEN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       OPEN @OUTPUT

        IAC2 = WORDADDR(OUPKT)
        IER = ISYS(ISYS_OPEN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999


C       SET UP TO DISPLAY INITIAL PROMPT ON @OUTPUT

        BUFF = 'WAIT FOR A MESSAGE AND TYPE A RESPONSE...<NL>'

C       DISPLAY BUFFER CONTENTS ON @OUTPUT

        IAC2 = WORDADDR(OUPKT)
        IER = ISYS(ISYS_WRITE,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999
```

```
C       WAIT FOR AN IPC MESSAGE FROM THE OTHER PROCESS

200     ??????(ISYS_????) = 40          ! SET BUFFER LENGTH
        ???? = ????????(??????)         ! IPC HDR ADDRESS
        IER = ISYS(ISYS_????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       IF MESSAGE LENGTH WAS ZERO, EXIT

        IF (??????(ISYS_????).EQ.0) GOTO 950

C       OTHERWISE, DISPLAY BUFFER CONTENTS ON @OUTPUT

        IAC2 = WORDADDR(OUPKT)
        IER = ISYS(ISYS_WRITE,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       READ A LINE FROM @INPUT

        IAC2 = WORDADDR(INPKT)
        IER = ISYS(ISYS_READ,IAC0,IAC1,IAC2)
        IF (IER.EQ.ISYS_EREOF) GOTO 900
        IF (IER.NE.0) GOTO 999

        I = ?????(ISYS_????)        ! GET ITS BYTE LENGTH
        IF (MOD(I,2).NE.0) I = I+1 ! IF ITS ODD, MAKE IT EVEN
        I = I/2                     ! THEN MAKE IT A WORD LENGTH.

C       SEND THE LINE TO THE OTHER PROCESS AS AN IPC MESSAGE

        ??????(ISYS_????) = I           ! MESSAGE LENGTH
        ???? = ????????(??????)
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        GOTO 200                ! WAIT FOR ANOTHER MESSAGE
```

```
C       GOT END-OF-FILE ... SEND ZERO LENGTH IPC MESSAGE

900     ??????(ISYS_????) = 0            ! ZERO LENGTH MESSAGE
        ???? = ????????(??????)          ! IPC HDR ADDRESS
        IER = ISYS(ISYS_?????,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

C       NORMAL EXIT

950     IAC2 = 0
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

C       ERROR EXIT

999     IAC2 = ISYS_RFCF+ISYS_RFEC+ISYS_RFAB
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

S309VS MULTITASKING LAB

MOVE/NACL/R INTO YOUR OWN DIRECTORY A COPY OF THE FOLLOWING FILES FROM THE :S309VS:F77 DIRECTORY:

MULTITASK.F77 COPY.F77 GTMES_QSYM.F77.IN IO_QSYM.F77.IN

REFER TO THE PRINTOUTS OF THE SOURCE FILES ON THE FOLLOWING PAGES.

THE PROGRAM ACCEPTS TWO TO (ARGMX) ARGUMENTS IN THE COMMAND LINE.
THERE MUST BE AN EVEN NUMBER OF ARGUMENTS.
IN EACH PAIR OF ARGUMENTS, THE FIRST IS CONSIDERED TO BE AN INPUT FILE, WHILE THE SECOND IS AN OUTPUT FILE.
FOR EACH PAIR OF FILES, A TASK IS CREATED WHICH COPIES, LINE BY LINE, FROM THE INPUT FILE TO THE OUTPUT FILE. EACH TASK, WHEN THE COPY IS COMPLETE, SENDS AN INTERTASK MSG TO THE MAIN TASK AND THEN KILLS ITSELF.
WHEN THE MAIN TASK RECEIVES THE MSG, IT NOTIFIES THE USER THAT THIS PARTICULAR COPY OPERATION IS COMPLETED.
WHEN ALL THE COPIES ARE COMPLETE, THE PROGRAM TERMINATES.

IN FORTRAN77, THE FOLLOWING LIBRARY ROUTINES MAY BE CALLED INSTEAD OF USING THE MULTITASKING SYSTEM CALLS:

ROUTINE	SYSTEM CALL
TQSTASK	?TASK (ROUTINE NEEDS NO PKT)
TQXMT	?XMT
TQXMTW	?XMTW
TQREC	?REC
TQRECNW	?RECNW
TQKILL	?KILL

EDIT YOUR COPIES OF THE MAIN PROGRAM AND SUBROUTINE, REPLACING THE QUESTION MARKS WITH THE APPROPRIATE VALUES AND SYMBOLS.

AS YOU READ THE SOURCE CODE, STUDY HOW THE PROGRAM FUNCTIONS. DISCUSS THE OPERATION OF THE PROGRAM WITH NEIGHBORING F77 STUDENTS.

VISUALIZE THE MAIN PROGRAM TASK AND ONE OR MORE SUBPROGRAM TASKS FUNCTIONING SIMULTANEOUSLY. THINK ABOUT THE OPERATING SYSTEM MECHANISMS YOU HAVE LEARNED ABOUT WHICH MAKE THIS POSSIBLE.

HOW MANY TCB'S WILL BE REQUIRED FOR THE PROGRAM TO EXECUTE SUCCESSFULLY WITH THE MAXIMUM NUMBER OF FILENAME ARGUMENTS ?

COMPILE THE MAIN PROGRAM AND THE SUBPROGRAM ON ONE COMMAND LINE. THIS WILL PRODUCE SEPERATE OBJECT FILES BUT A COMBINED LISTING. USE THE 'LINEID' SWITCH TO ENABLE LINE NUMBER TRACEBACK:

```
F77/LINEID MULTITASK COPY
```

WHEN YOU HAVE BOTH FILES COMPILED WITH NO ERRORS, LINK THEM TOGETHER, REMEMBERING TO RESERVE THE REQUIRED NUMBER OF TCB'S !!

```
F77LINK/TASKS=?  MULTITASK COPY
```

NOW YOU ARE READY TO TEST YOUR PROGRAM. THE PROGRAM IS DESIGNED TO BE INVOKED WITH A COMMAND LINE OF THE FOLLOWING FORMAT:

```
X, MULTITASK, infile1, outfile1 [ , infile2, outfile2...]
```

WHERE THE infile'S ALREADY EXIST AND THE outfile'S ARE TO BE CREATED BY THE PROGRAM. THERE MUST BE AN EVEN NUMBER OF ARGUMENTS. WHAT IS THE MAXIMUM NUMBER OF ARGUMENTS PERMITTED?

_____

TRY YOUR PROGRAM WITH FOUR ARGUMENTS. YOU MIGHT USE THE NAMES OF YOUR SOURCE FILES FOR THE infile'S AND PERHAPS THE SAME FILENAMES WITH '.CPY' APPENDED FOR THE outfile'S.

YOU SHOULD GET CONSOLE MESSAGES STATING THE COMPLETION OF EACH COPY OPERATION. WHICH COPY FINISHES FIRST ?

_____

DOES THE ORDER OF THE COPY OPERATIONS SPECIFIED IN THE COMMAND LINE MAKE ANY DIFFERENCE IN THE ORDER OF COMPLETION ? (TRY IT.)_____

YOU CAN ALSO TEST YOUR PROGRAM USING "@NULL" AS ANY (OR ALL) OF THE infile'S AND/OR outfile'S.

TRY YOUR PROGRAM WITH VARIOUS NUMBERS OF ARGUMENTS. IT SHOULD ABORT IF THE NUMBER IS ODD, LESS THAN TWO, OR MORE THAN (ARGMX). DOES IT WORK CORRECTLY (ABORT PROPERLY OR COPY PROPERLY) FOR:

NO ARGUMENTS _____
ONE ARGUMENT _____
TWO ARGUMENTS _____
THREE ARGUMENTS _____
FOUR ARGUMENTS _____
FIVE ARGUMENTS _____
SIX ARGUMENTS _____
SEVEN ARGUMENTS _____
EIGHT ARGUMENTS _____

IF YOU HAVE CORRECTLY EDITED, COMPILED, AND LINKED YOUR PROGRAM, ALL OF THE CASES ABOVE SHOULD FUNCTION CORRECTLY. ENSURE THAT THIS IS TRUE.

```
        PROGRAM MULTITASK
C
C  PROGRAM ACCEPTS TWO TO (ARGMX) ARGUMENTS IN THE COMMAND LINE.
C  THERE MUST BE AN EVEN NUMBER OF ARGUMENTS.
C  IN EACH PAIR OF ARGUMENTS, THE FIRST IS CONSIDERED TO BE AN
C  INPUT FILE, AND THE SECOND IS AN OUTPUT FILE.
C  FOR EACH PAIR OF FILES, A TASK IS CREATED WHICH COPIES,
C  LINE BY LINE, FROM THE INPUT FILE TO THE OUTPUT FILE.
C  EACH TASK, WHEN THE COPY IS COMPLETED, SENDS AN INTERTASK MSG
C  TO THE MAIN TASK AND THEN KILLS ITSELF.
C  WHEN THE MAIN TASK RECEIVES THE MSG, IT NOTIFIES THE USER THAT
C  THIS PARTICULAR COPY OPERATION IS COMPLETED.
C  WHEN ALL THE COPIES ARE COMPLETE, THE PROGRAM TERMINATES.

%LIST(OFF)
%INCLUDE 'IO_QSYM.F77.IN'
%INCLUDE 'GTMES_QSYM.F77.IN'
%LIST(ON)

        INTEGER ARGMX
        PARAMETER (ARGMX = 6)

        INTEGER SBOX,FBOX       ! INTERTASK MESSAGE MAILBOXES
        COMMON  /MAIL/ SBOX,FBOX

        CHARACTER*128 FNAME(ARGMX)  ! ARRAY OF FILENAMES
        COMMON /NAMES/ FNAME
        INTEGER LFNAME(ARGMX)       ! AND FILENAME LENGTHS

        INTEGER ARGCNT,ARG,MSG  ! ARG COUNT, CURRENT ARG, TSK MSG

        INTEGER*2 GCPKT(0:ISYS_GTLN-1)          ! GTMES PKT
        INTEGER*4 GCPKT_GSW,GCPKT_GRES
        EQUIVALENCE (GCPKT(ISYS_GSW),GCPKT_GSW)
        EQUIVALENCE (GCPKT(ISYS_GRES),GCPKT_GRES)


        EXTERNAL COPY
```

```
C...GET ARGUMENT COUNT PKT

                GCPKT(ISYS_GREQ) = ISYS_GCNT
                GCPKT(ISYS_GNUM) = 0
                GCPKT_GSW  = 0
                GCPKT_GRES = -1

        IAC2 = WORDADDR(GCPKT)
        IER = ISYS(ISYS_GTMES,ARGCNT,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        IF (ARGCNT.GE.2.AND.ARGCNT.LE.ARGMX.AND.MOD(ARGCNT,2).EQ.0)
     +    GO TO 5

        IAC1 = BYTEADDR('WRONG NUMBER OF ARGUMENTS')
        IAC2 = ISYS_RFCF+ISYS_RFER+LEN('WRONG NUMBER OF ARGUMENTS')
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

5       CONTINUE


C       FOR EACH PAIR OF FILENAME ARGUMENTS,
C       CREATE A TASK TO PERFORM THE COPY OPERATION.
C       ODD NUMBERED ARGUMENTS ARE FOR INPUT FILES,
C       AND EVEN NUMBERS ARE FOR OUTPUT FILES.

        GCPKT(ISYS_GREQ) = ISYS_GARG

        DO 50 I=1,ARGCNT/2

                GCPKT(ISYS_GNUM) = I*2-1
                GCPKT_GRES = BYTEADDR(FNAME(I*2-1))
                IAC2 = WORDADDR(GCPKT)
                IER = ISYS(ISYS_GTMES,IAC0,IAC1,IAC2)
                IF (IER.NE.0) GOTO 999
                LFNAME(I*2-1) = IAC0

                GCPKT(ISYS_GNUM) = I*2
                GCPKT_GRES = BYTEADDR(FNAME(I*2))
                IAC2 = WORDADDR(GCPKT)
                IER = ISYS(ISYS_GTMES,IAC0,IAC1,IAC2)
                IF (IER.NE.0) GOTO 999
                LFNAME(I*2) = IAC0
C
C       THE LOOP COUNTER IS USED AS A MESSAGE TO EACH NEW TASK.

                CALL ???????(COPY,I+1,2,0,IER)  ! CREATE A TASK
                IF (IER.NE.0) GOTO 990

                CALL ??????(SBOX,I,0,IER)    ! SEND IT A MSG AND WAIT
                IF (IER.NE.0) GOTO 990

50      CONTINUE
```

```
C       WHEN ANY TASK COMPLETES, IT SENDS BACK A MESSAGE
C       TO THE MAILBOX WHICH IS THE SAME AS THE MESSAGE
C       ORIGINALLY SENT TO IT.
C
C       USING THE MESSAGE TO ACCESS THE APPROPRIATE FILENAMES,
C       (SEE CALCULATIONS ABOVE), NOTIFY THE USER THAT
C       THIS PARTICULAR COPY OPERATION IS COMPLETE.

        DO 70 I=1,ARGCNT/2

                CALL ?????(FBOX,MSG,IER)
                IF (IER.NE.0) GOTO 990
                PRINT *,'COPY COMPLETED: FILE '
     +                  ,FNAME(MSG*2-1)(1:LFNAME(MSG*2-1)),
     +                  ' TO FILE ',FNAME(MSG*2)(1:LFNAME(MSG*2))

70      CONTINUE

C       WHEN ALL THE TASKS HAVE COMPLETED, TERMINATE THE PROGRAM.

        IAC2 = 0
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

C       ERROR EXIT

990     IAC0 = IER
999     IAC2 = ISYS_RFCF+ISYS_RFEC+ISYS_RFAB
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        END
```

```
        SUBROUTINE COPY

%LIST(OFF)
%INCLUDE 'IO_QSYM.F77.IN'
%LIST(ON)

        INTEGER ARGMX
        PARAMETER (ARGMX = 6)

        INTEGER SBOX,FBOX           ! INTERTASK MESSAGE MAILBOXES
        COMMON  /MAIL/ SBOX,FBOX

        CHARACTER*128 FNAME(ARGMX)
        COMMON /NAMES/ FNAME

        INTEGER*2 PKTIN(0:ISYS_IBLT-1)
        INTEGER*4 PKTIN_IBAD, PKTIN_IRNH, PKTIN_IFNP, PKTIN_IDEL
        EQUIVALENCE (PKTIN_IBAD, PKTIN(ISYS_IBAD))
        EQUIVALENCE (PKTIN_IRNH, PKTIN(ISYS_IRNH))
        EQUIVALENCE (PKTIN_IFNP, PKTIN(ISYS_IFNP))
        EQUIVALENCE (PKTIN_IDEL, PKTIN(ISYS_IDEL))

        INTEGER*2 PKTOU(0:ISYS_IBLT-1)
        INTEGER*4 PKTOU_IBAD, PKTOU_IRNH, PKTOU_IFNP, PKTOU_IDEL
        EQUIVALENCE (PKTOU_IBAD, PKTOU(ISYS_IBAD))
        EQUIVALENCE (PKTOU_IRNH, PKTOU(ISYS_IRNH))
        EQUIVALENCE (PKTOU_IFNP, PKTOU(ISYS_IFNP))
        EQUIVALENCE (PKTOU_IDEL, PKTOU(ISYS_IDEL))

        INTEGER MSG
        CHARACTER*136 BUFF

        CALL ?????(SBOX,MSG,IER)        ! RECEIVE INTERTASK MSG
        IF (IER.NE.0) GOTO 990
```

```
C...I/O PACKET FOR INPUT FILE

                PKTIN(ISYS_ICH)  = 0     ! CHANNEL NUMBER
                PKTIN(ISYS_ISTI) = ISYS_ICRF+ISYS_RTDS+ISYS_OFIN
                PKTIN(ISYS_ISTO) = 0     ! FILE TYPE RETURNED
                PKTIN(ISYS_IMRS) = -1
                PKTIN_IBAD = BYTEADDR(BUFF)
                PKTIN(ISYS_IRES) = 0     ! (RESERVED)
                PKTIN(ISYS_IRCL) = 136   ! MAX RECORD LENGTH
                PKTIN(ISYS_IRLR) = 0     ! RECORD LENGTH RETURNED
                PKTIN(ISYS_IRNW) = 0     ! (RESERVED)
                PKTIN_IRNH = 0           ! RECORD NUMBER
                PKTIN_IFNP = BYTEADDR(FNAME(MSG*2-1))
                PKTIN_IDEL = -1          ! DELIMITER TABLE, DEFAULT

C...I/O PACKET FOR OUTPUT FILE

                PKTOU(ISYS_ICH)  = 0     ! CHANNEL NUMBER
                PKTOU(ISYS_ISTI) = ISYS_ICRF+ISYS_RTDS
     +                           +ISYS_OFCE+ISYS_OFCR+ISYS_OFOT
                PKTOU(ISYS_ISTO) = 0     ! FILE TYPE RETURNED
                PKTOU(ISYS_IMRS) = -1
                PKTOU_IBAD = BYTEADDR(BUFF)
                PKTOU(ISYS_IRES) = 0     ! (RESERVED)
                PKTOU(ISYS_IRCL) = 136   ! MAX RECORD LENGTH
                PKTOU(ISYS_IRLR) = 0     ! RECORD LENGTH RETURNED
                PKTOU(ISYS_IRNW) = 0     ! (RESERVED)
                PKTOU_IRNH = 0           ! RECORD NUMBER
                PKTOU_IFNP = BYTEADDR(FNAME(MSG*2))
                PKTOU_IDEL = -1          ! DELIMITER TABLE, DEFAULT
```

```
        IAC2 = WORDADDR(PKTIN)
        IER = ISYS(ISYS_OPEN,IAC0,IAC1,IAC2)
        IF (IER.EQ.ISYS_EREOF) GO TO 999
        IF (IER.NE.0) GOTO 999

        IAC2 = WORDADDR(PKTOU)
        IER = ISYS(ISYS_OPEN,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999


10      IAC2 = WORDADDR(PKTIN)
        IER = ISYS(ISYS_READ,IAC0,IAC1,IAC2)
        IF (IER.EQ.ISYS_EREOF) GO TO 20
        IF (IER.NE.0) GOTO 999

        IAC2 = WORDADDR(PKTOU)
        IER = ISYS(ISYS_WRITE,IAC0,IAC1,IAC2)
        IF (IER.NE.0) GOTO 999

        GO TO 10

C       WHEN THE COPY OPERATION IS COMPLETE, THE TASK SENDS
C       ITS ORIGINAL INTERTASK MESSAGE TO A COMMON INTERTASK MAILBOX,
C       AND THEN KILLS ITSELF.

20      CALL ?????(FBOX,MSG,0,IER)          ! SEND MESSAGE
                                            ! TO INTERTASK MAILBOX....
        IF (IER.EQ.ISYS_?????) GOTO 20      ! IF BOX IN USE, TRY AGAIN
        IF (IER.NE.0) GOTO 990

        CALL ??????(IER)                    ! KILL THIS TASK.
        IF (IER.NE.0) GOTO 990

C       ERROR EXIT

990     IAC0 = IER
999     IAC2 = ISYS_RFCF+ISYS_RFEC+ISYS_RFAB
        IER = ISYS(ISYS_RETURN,IAC0,IAC1,IAC2)

        END
```

C LANGUAGE FILE SYSTEM LAB

* MOVE/R/NACL COPIES OF THE FILES 'CREATE.C' AND 'ERROR.OB' FROM THE :S309VS:C DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF CREATE' IS TO FIRST CREATE A DIRECTORY, THE NAME OF WHICH WILL BE "MYDIR". THE PROGRAM SHOULD NOT FAIL IF THE DIRECTORY ALREADY EXISTS. IT WILL THEN MAKE "MYDIR" THE WORKING DIRECTORY. (IF "MYDIR" IS NOT A DIRECTORY THEN THE PROGRAM WILL FAIL AND YOU WILL HAVE TO DELETE IT.) THE PROGRAM WILL THEN ATTEMPT TO DELETE AND CREATE A NEW FILE CALLED "MYFILE".

* THE FUNCTION OF 'ERROR' IS TO PROVIDE AN ERROR HANDLER FOR THE SYSTEM CALLS. IT WILL REPORT THE ERROR MESSAGE AND PROGRAM LOCATION WHEN A SYSTEM CALL FAILS TO EXECUTE PROPERLY.

* REFER TO THE PRINTOUT OF 'CREATE.C' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM CALLS AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
CC/LINEID, CREATE
CCL, CREATE, ERROR
```

* GET A HARDCOPY OF THE FILE 'CREATE.LS'.

* EXECUTE THE CREATE PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'FILE CREATION COMPLETED'.

* DO A FILESTATUS WITH /TYPE ON THE FILE 'MYDIR'. IT SHOULD BE TYPE "DIR".

* DO A DIRECTORY TO "MYDIR" THEN DO ANOTHER FILESTATUS/TYPE. YOU SHOULD FIND THAT "MYFILE" IS A TYPE "TXT".

* GO BACK TO YOUR PREVIOUS DIRECTORY AND EXECUTE THE PROGRAM AGAIN. THE PROGRAM SHOULD NOT FAIL AS IT WILL DELETE 'MYFILE' BEFORE TRYING TO CREATE IT AGAIN.

* YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE PROGRAM REPEATEDLY WITHOUT ERROR

```
                                                    /* CREATE.C */
#nolist
#include <stdio.h>
#include <packets:create.h>
#list

main()
{
P_??????_??? dirpkt;     /* packet for dir. creation   */
P_??????     filepkt;    /* packet for file creation   */

int ac0, ac1, ac2, error;
char *fname;              /* filename byte pointer */

/* DIRECTORY CREATION PACKET */

  dirpkt.cftyp_format = 0;       /* reserved */
  dirpkt.cftyp_entry  = ?????;   /* file type DIR  */
  dirpkt.chfs         = ??;      /* default hashframe size */
  dirpkt.ctim         = ??;      /* default time */
  dirpkt.cacp         = ??;      /* default ACL */
  dirpkt.cmsh         = 0;       /* must be zero if not CPD */
  dirpkt.cmil         = ??;      /* default max index levels */
  dirpkt.cmrs         = 0;       /* reserved */


/* FILE CREATION PACKET */

  filepkt.cftyp_format = 0;      /* no record format */
  filepkt.cftyp_entry  = ?????;  /* text type file */
  filepkt.ccps         = 0;      /* ignored unless fixed fmt */
  filepkt.ctim         = ??;     /* default time */
  filepkt.cacp         = ??;     /* default ACL *
  filepkt.cdeh         = 0;      /* reserved */
  filepkt.cdel         = ??;     /* default element size */
  filepkt.cmil         = ??;     /* default max index levels */
  filepkt.cmrs         = 0;      /* reserved */
```

```
/* ATTEMPT TO CREATE THE DIRECTORY,
 IGNORE ERROR IF IT ALREADY EXISTS */

  fname = "MYDIR";       /* B.P. to directory name */
  ac0   = ?????;         /* ac0 <- B.P. */
  ac1   = 0;             /* reserved */
  ac2   = ???????;       /* ac2 <- packet address */
  if ((error = sys(???????,&ac0,&ac1,&ac2)) &&
               (error != ?????)) errxt(error);

/* CHANGE WORKING DIRECTORY */

  fname = "MYDIR";       /* B.P. to directory name */
  ac0   = ?????;         /* ac0 <- B.P. */
  ac1   = 0;             /* reserved */
  ac2   = 0;             /* reserved */
  if (error = sys(????,&ac0,&ac1,&ac2)) errxt(error);

/* ATTEMPT TO DELETE THE FILE,
 IGNORE THE ERROR IF IT DOESN'T EXIST */

  fname   = "MYFILE";    /* B.P. to filename */
  ac0     = ?????;       /* ac0 <- B.P. */
  ac1     = 0;           /* reserved */
  ac2     = 0;           /* reserved */
  if ((error = sys (???????,&ac0,&ac1,&ac2)) &&
            (error != ?????)) errxt(error);

/* CREATE THE FILE */

  fname   = "MYFILE";    /* B.P. to filename */
  ac0   = ?????;         /* ac0 <- B.P. */
  ac1     = 0;           /* reserved */
  ac2   = ????????;      /* ac2 <- packet address */
  if (error = sys(???????,&ac0,&ac1,&ac2)) errxt(error);

} /* end main */
```

C LANGUAGE FILE ACCESS LAB

* MOVE/R/NACL COPIES OF THE FILES 'CHANGEACL.C' AND 'ERROR.OB' FROM THE :S309VS:C DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'CHANGEACL' IS TO FIRST MAKE ITS WORKING DIRECTORY A SUB-DIRECTORY CALLED "MYDIR". THE PROGRAM WILL FAIL IF THE DIRECTORY DOES NOT EXIST. IT WILL THEN READ THE ACL OF A FILE CALLED "MYFILE" IN THAT DIRECTORY. THIS WILL ALSO FAIL IF THE FILE DOES NOT EXIST. NEXT IT WILL WRITE A NEW ACL FOR THE FILE, ADDING "+,RE" ACCESS TO THE PREVIOUS STRING.

NOTE: IF YOU HAVE NOT DONE THE "CREATE_LAB" BY THIS TIME YOU WILL HAVE TO CREATE THE NECESSARY FILES FOR THIS LAB AS FOLLOWS:

```
) CR/DIR,MYDIR
) CR,MYDIR:MYFILE
```

* REFER TO THE PRINTOUT OF 'CHANGEACL.C' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUE. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS, AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
CC/LINEID ,CHANGEACL
CCL, CHANGEACL, ERROR
```

* GET A HARDCOPY OF THE FILE 'CREATE.LS'.

* DO AN ACL WITH /V ON THE FILE 'MYDIR:MYFILE'.
IT SHOULD BE YOUR DEFAULT ACL (USERNAME,OWARE).

* EXECUTE THE CREATE PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'THE FILE'S ACL HAS BEEN CHANGED."

* DO ANOTHER ACL WITH /V ON THE FILE 'MYDIR:MYFILE'. IT SHOULD NOW BE YOUR DEFAULT ACL AND "+,RE".

* YOU HAVE COMPLETED THIS LAB WHEN MYFILE:MYDIR HAS THE ACCESS CONTROL LIST "USERNAME,OWARE +,RE"

```
        /* CHANGEACL.C */

#nolist
#include <sysid.h>
#include <paru.h>
#list

main()
{
int  ac0, ac1, ac2, i, error;
char  *fname;
char  buff[$MXACL];
char   *buffbp = buff;
char   *msg  = "The file's ACL has been changed.";

        /* FILL ACL BUFFER WITH NULLS */

  for (i=0;i<$MXACL;i++) buff[i] = '\0';

        /* GET FILE'S ACL */

  fname  = "MYDIR:MYFILE";       /* filename BP */
  ac0  = ?????;                  /* ac0 <- BP */
  ac1  = ??????;                 /* ac1 <- buffer BP */
  ac2  = 0;                      /* ac2 reserved */
  if (error = sys(?????,&ac0,&ac1,&ac2)) errxt(error);

        /* APPEND "+,RE" TO ACL */

  for (i=$MXACL; buff[i] == '\0'; i--); /* find last non-null */
  buff[i+1] = '+';               /* append "+" */
  buff[i+3] = (????? | ?????);   /* append access byte */

        /* SET FILE'S ACL */

  fname  = "MYDIR:MYFILE";       /* filename BP */
  ac0  = ?????;                  /* ac0 <- BP */
  ac1  = ??????;                 /* ac1 <- buffer BP */
  ac2  = 0;                      /* ac2 reserved */
  if (error = sys(?????,&ac0,&ac1,&ac2)) errxt(error);


        /* RETURN TO CLI WITH MSG */

  ac0  = 0;
  ac1  = msg;
  ac2  = $RFCF + strlen(msg);
  if (sys ($RETURN, &ac0, &ac1, &ac2)) errxt(error);

}        /* end main */
```

C LANGUAGE FILE STATUS LAB

* MOVE/R/NACL COPIES OF THE FILES 'FILES.C' AND 'ERROR.OB' FROM THE :S309VS:C DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'FILES' IS TO FIRST GO TO A DIRECTORY, SPECIFIED AS THE FIRST ARGUMENT ON THE COMMAND LINE. THE PROGRAM WILL FAIL IF THE DIRECTORY DOES NOT EXIST. IT WILL THEN GET A FILENAME WHICH MATCHES THE TEMPLATE SPECIFIED AS THE SECOND ARGUMENT ON THE COMMAND LINE. IT WILL PASS THE FILENAME AND BYTE LENGTH TO '$IO' FOR OUTPUT. THEN IT WILL GET THE NEXT FILENAME AND REPEAT THE PROCESS. WHEN ALL THE FILES FOR A TEMPLATE HAVE BEEN RECOVERED, THE PROGRAM WILL GET THE NEXT TEMPLATE FROM THE COMMAND LINE UNTIL ALL THE FILES HAVE BEEN DISPLAYED.

* THE FUNCTION OF 'ERROR' IS TO PROVIDE AN ERROR HANDLER FOR THE SYSTEM CALLS. IT WILL REPORT THE ERROR MESSAGE AND PROGRAM LOCATION WHEN A SYSTEM CALL FAILS TO EXECUTE PROPERLY.

* REFER TO THE PRINTOUT OF 'FILES.C' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUE. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS, AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
CC/LINEID, FILES
CCL, FILES, ERROR
```

* GET A HARDCOPY OF THE FILE 'FILES.LS'.

* EXECUTE THE FILES PROGRAM WITH THE FOLLOWING COMMAND:

```
X,FILES,:UDD:yourusername,template...
```

* YOU SHOULD SOON SEE THE FILENAMES AND BYTE LENGTHS FOLLOWED BY THE MESSAGE 'INFORMATION PROCESSING COMPLETED'.

* DO A FILESTATUS WITH /LENGTH IN THE DIRECTORY IN YOUR COMMAND LINE AND CHECK THE RESULTS.

* YOU HAVE COMPLETED THIS LAB WHEN YOUR PROGRAM YIELDS THE SAME RESULTS AS THE FILESTATUS/LENGTH COMMAND.

```
#nolist         /* FILES.C    FILE STATUS LAB */
#include <packets:block_io.h>
#include <packets:filestatus.h>
#list

main(argc,argv)
int     argc;           /* argument count */
char    *argv[];        /* command line arguments */
{
int     ac0, ac1, ac2, error, argnum;
char    fbuff[$MXFN], *fname = fbuff;

P_GOPEN gopkt;          /* ?GOPEN packet */
P_????  gnpkt;          /* ?GNFN packet */
P_????? fspkt;          /* ?FSTAT packet */

/* GOPEN PACKET FOR DIRECTORY */

gopkt.opch = 0;         /* channel number (returned) */
gopkt.opty_format = 0;  /* record format (returned) */
gopkt.opty_type = 0;    /* file type (returned) */
gopkt.opfc = 0;         /* file control parms (returned)*/
gopkt.opew = 0;         /* reserved */
gopkt.opeh = 0;         /* file length (returned) */

ac0 = argv[1];  /* OPEN DIRECTORY */
ac1 = -1;       /* system selects channel */
ac2 = &gopkt;   /* ac2 <- packet address */
if (error = sys($GOPEN,&ac0,&ac1,&ac2)) errxt(error);

ac0 = argv[1];  /* CHANGE WORKING DIRECTORY */
ac1 = 0;
ac2 = 0;
if (error = sys(????,&ac0,&ac1,&ac2)) errxt(error);

for (argnum=2; argnum<argc; argnum++)
  {                     /* for each template (arg) */
  gnpkt.nfky = ?;       /* reset GNFN counter to zero */
  gnpkt.nfrs = 0;       /* reserved */
  gnpkt.nfnm = ?????;   /* filename buffer BP */
  gnpkt.nftp = argv[argnum];    /* set template BP */
  do{
    ac0 = ?????;        /* get next filename */
    ac1 = ?????.????;
    ac2 = ??????;
    if (!(error = sys(?????,&ac0,&ac1,&ac2)))
      {
      ac0 = ?????;      /* get filestatus info */
      ac1 = ????;
      ac2 = ??????;
      if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);
      printf("%s%40t%9d\n",fname,fspkt.????);
      }                 /* print filename & byte len  */
    }while (!error);
  if (error != EREOF) errxt(error);
  }             /* end for */

}                               /* end main */
```

C LANGUAGE BLOCK I/O LAB

* MOVE/R/NACL COPIES OF THE FILES 'BLOCKIO.C', 'ERROR.OB', AND 'BLKDATA' FROM THE :S309VS:C DIRECTORY TO YOUR INITIAL WORKING DIRECTORY.

* TYPE THE BLKDATA FILE. IF YOU DON'T BELIEVE YOUR EYES QPRINT THE FILE. IS THERE STILL A PROBLEM READING THE FILE? USE THE DISPLAY UTILITY TO LOOK AT THE BLKDATA FILE.

* THE PROBLEM IS THAT CARRIAGE RETURNS RATHER THAN NEW LINE CHARACTERS WERE USED AS DELIMITERS WHEN THE FILE WAS CREATED. THE THRUST OF THIS EXERCISE IS TO WRITE A PROGRAM WHICH CONVERTS ALL CARRIAGE RETURNS TO NEW LINES.

* REFER TO THE PRINTOUT OF 'BLOCKIO.C' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUE. THESE VALUES INCLUDE THE ACTUAL SYSTEM CALLS, AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS.

```
CC/LINEID, BLOCKIO
CCL, BLOCKIO, ERROR
```

* GET A HARDCOPY OF THE FILE 'BLOCKIO.LS'.

* EXECUTE THE BLOCKIO PROGRAM. YOU SHOULD SOON SEE THE MESSAGE: 'DELIMITER CONVERSION COMPLETED.'

* TYPE OUT THE OUTPUT FILE 'BLOCKOUT'. THE TEXT SHOULD NOW BE READABLE.

* DO A FILESTATUS WITH /ASSORTMENT ON BOTH THE BLKDATA AND BLOCKOUT FILES. THEY SHOULD HAVE IDENTICAL LENGTHS - EXACTLY ONE BLOCK.

* TO TEST THE PROGRAM WITH A DIFFERENT INPUT FILE:

  DELETE YOUR COPY OF 'BLKDATA'. COPY INTO A NEW 'BLKDATA' FILE IN YOUR OWN DIRECTORY FROM 'BLKDATA1' IN THE :S309VS:C DIRECTORY. DO A FILESTATUS WITH ASSORTMENT ON THIS NEW 'BLKDATA' FILE IN YOUR OWN DIRECTORY. IT SHOULD BE LESS THAN ONE BLOCK LONG.

  CHANGING TO A DIFFERENT INPUT FILE WOULD BE EASIER IF A GENERIC FILENAME HAD BEEN USED IN THE PROGRAM ....HOWEVER THE SYSTEM DOES NOT PERMIT BLOCK IO TO GENERIC FILENAMES.

* EXECUTE THE PROGAM. NOTICE THAT YOUR OLD 'BLOCKOUT' FILE WILL BE DELETED BY THE PROGRAM AND A NEW ONE WILL BE CREATED. FIND THE SECTIONS OF CODE THAT ACCOMPLISH THIS. PROGRAMS THAT PERFORM BLOCK IO MUST HANDLE DETAILS SUCH AS THIS EXPLICITLY.

* YOU SHOULD AGAIN SEE THE COMPLETION MESSAGE. TYPE THE OUTPUT FILE 'BLOCKOUT'. IT SHOULD BE READABLE. FIND THE PARTS OF THE PROGRAM WHICH ENSURE THAT IT WILL WORK CORRECTLY IF THE LAST BLOCK IS NOT FULL. COMPARE THE LENGTHS OF YOUR NEW 'BLKDATA' AND 'BLOCKOUT'.... THEY SHOULD BE IDENTICAL.

```
#include <packets:create.h>
#include <packets:block_io.h>

main( )                     /* BLOCK I/O PROGRAM */
{
P_?????? crepk;             /* declare parm packet for ?CREATE */
P_??? rdpk, wrpk;           /* declare parm packet for ?RDB, ?WRB */
P_????? gopnin, gopnout;/* declare parm packets for ?GOPEN */

char buff[512];             /* block I/O buffer */
int ac0, ac1, ac2;          /* accumulator arguments for sys( ) */
int error;                  /* error return code from system calls*/
char *byptr;                /* byte pointer */
int eoflg;                  /* end of file flag */

/* GOPEN PACKET FOR INPUT FILE */

gopnin.opch = ?;            /* channel number (returned)
gopnin.opty_format = ?; /* record format (returned) */
gopnin.opty_type = ?;   /* file type (returned) */
gopnin.opfc = ?;            /* file control parms (returned)*/
gopnin.opew = ?;            /* reserved */
gopnin.opeh = ?;            /* file length (returned) */


/* GOPEN PACKET FOR OUTPUT FILE */

gopnout.opch = ?????;   /* exclusive open */
gopnout.opty_format = ?;/* record format (returned) */
gopnout.opty_type = ?;  /* file type (returned) */
gopnout.opfc = ?;           /* file control parms (returned) */
gopnout.opew = ?;           /* reserved */
gopnout.opeh = ?;           /* file length (returned) */

/* FILE CREATION PACKET */

crepk.cftyp_format = 0; /* no record format */
crepk.cftyp_entry = $FTXT;/* file type text */
crepk.ccps = 0;             /* file cntl parms (ignored)  */
crepk.ctim = -1;            /* default time */
crepk.cacp = -1;            /* default acl */
crepk.cdeh = 0;             /* reserved */
crepk.cdel = -1;            /* default element size */
crepk.cmil = -1;            /* default max index level */
crepk.cmrs = 0;             /* reserved */
```

```
/* READ BLOCK PACKET */

rdpk.psti = ?;          /* read one block */
rdpk.psto = ?;          /* reserved */
rdpk.pcad = ?????;      /* address of data buffer */
rdpk.prnh = ?;          /* starting block number */
rdpk.prcl = ?;
rdpk.pres = ?;          /* reserved */

/* WRITE BLOCK PACKET */

wrpk.psti = ?;          /* write one block */
wrpk.psto = ?;          /* reserved */
wrpk.pcad = ?????;      /* address of data buffer */
wrpk.prnh = ?;          /* starting block number */
wrpk.prcl = ?;          /* byte count (set at each ?WRB) */
wrpk.pres = ?;          /* reserved */

/* OPEN 'BLKDATA' FOR BLOCK IO */

byptr = "blkdata";
ac0 = ?????;            /* filename byte pointer */
ac1 = ??;               /* system assigns channel # */
ac2 = ???????;          /* packet address */

if (error = sys (??????,&ac0,&ac1,&ac2)) errxt(error);

/* DELETE 'BLOCKOUT' IF IT EXISTS */

byptr = "blockout";
ac0 = ?????;            /* filename byte pointer */
ac1 = 0;                /* reserved */
ac2 = 0;                /* reserved */

error = sys (???????,&ac0,&ac1,&ac2);
if (0 != error && ????? != error) errxt(error);

/* CREATE 'BLOCKOUT' */

byptr = "blockout";
ac0 = ?????;            /* ptr to file to be created */
ac1 = 0;                /* reserved */
ac2 = ??????;           /* create packet */

if (error = sys (???????,&ac0,&ac1,&ac2)) errxt(error);

/* ?GOPEN 'BLOCKOUT' */

byptr = "blockout";
ac0 = ?????;            /* byte ptr to filename */
ac1 = ??;               /* let system assign channel # */
ac2 = ????????;         /* addr of ?GOPEN packet */

if (error = sys (??????,&ac0,&ac1,&ac2)) errxt(error);
```

```
/* START RDB, WRB, LOOP */

eoflg = 0;

do
  {

/* READ A BLOCK FROM INPUT FILE */

  ac0 = 0;                /* reserved */
  ac1 = ??????.????;      /* input file channel number */
  ac2 = ?????;            /* ?RDB packet */

  if ((error = sys (????,&ac0,&ac1,&ac2)) && (error != ?????))
     errxt(error);
  if (error == ?????) eoflg = 1;
  rdpk.???? = rdpk.???? + 1;/* incr blk num in RDB pak */

/* SUBSTITUTE <NL> FOR <CR> IN THE BUFFER */

  delimconv(buff,ac1);

/* WRITE THE BLOCK TO THE OUTPUT FILE */

  if ( ac1 > 0 )
    {
    wrpk.???? = ???;      /* number of bytes to write */
    ac0 = 0;              /* reserved */
    ac1 = ???????.????; /* output file channel number */
    ac2 = ?????;          /* ?WRB packet */

    if (error = sys(????,&ac0,&ac1,&ac2)) errxt(error);

    wrpk.???? = wrpk.???? + 1;
        /* incr block number in WRB pak */
    }
  }
while (eoflg == 0);
```

```
/* CLOSE INPUT FILE */

  ac0 = 0;                  /* reserved */
  ac1 = ??????.????;        /* input file channel number */
  ac2 = 0;                  /* reserved */

  if (error = sys (???????,&ac0,&ac1,&ac2)) errxt(error);

/* CLOSE OUTPUT FILE */

  ac0 = 0;                  /* reserved */
  ac1 = ???????.????;       /* output file channel number */
  ac2 = 0;                  /* reserved */

  if (error = sys (???????,&ac0,&ac1,&ac2)) errxt(error);

/* USE A ?RETURN TO GO BACK TO CLI */

  byptr = "DELIMITER CONVERSION COMPLETED\n";

  ac0 = 0;       /* will be ignored */
  ac1 = byptr;   /* byte pointer to message */
  ac2 = $RFCF + strlen(byptr);

  if (error = sys($RETURN,&ac0,&ac1,&ac2)) errxt(error);

}       /* END OF MAIN */


delimconv(bffr,count)   /* delimiter conversion routine */
char bffr[];
int count;
  {
  int i;

  for ( i=0; i<count; ++i )
    if (bffr[i] == '\015') bffr[i] = '\012';
  }
```

C LANGUAGE SHARED DATA LAB

USING THE FOLLOWING CLI COMMANDS, MOVE THE NEEDED FILES INTO YOUR DIRECTORY:

```
DIRECTORY, :S309VS:C
MOVE/R/NACL, :UDD:YOURUSERNAME, SHDATA.<ONE,TWO>.C, ERROR.OB
DIRECTORY/I
```

REFER TO THE PRINTOUTS OF 'SHDATA.ONE.C' AND 'SHDATA.TWO.C' ON THE FOLLOWING PAGES.

DISCUSSION:

SHDATA.ONE AND SHDATA.TWO COMMUNICATE ACROSS A SINGLE PAGE OF MEMORY WHICH IS MAPPED TO BOTH PROCESSES CONCURRENTLY USING THE SHARED DATA FACILITY.

SHDATA.ONE FIRST CREATES A SHARED DATA FILE THEN READS IT ONCE TO MAP IN THE SHARED PAGE. A MESSAGE (FOR SHDATA.TWO) IS NOW MOVED TO A BUFFER LOCATED IN THE SHARED PAGE. AFTER SETTING A FLAG (ALSO LOCATED IN THE SHARED PAGE), SHDATA.ONE WAITS FOR SHDATA.TWO TO CLEAR THE FLAG. THEN SHDATA.ONE RETURNS CONTROL TO THE CLI WHICH DISPLAYS A RETURN MESSAGE.

SHDATA.TWO READS ONE PAGE FROM THE SHARED DATA FILE, WAITS FOR SHDATA.ONE TO SET THE FLAG, AND WRITES THE CONTENTS THE BUFFER IN THE SHARED PAGE TO @OUTPUT. IT NEXT CLEARS THE FLAG WORD SHDATA.ONE IS WAITING FOR AND PASSES CONTROL TO THE CLI.

STUDY THE SOURCES TO MAKE SURE YOU FOLLOW THE CODE THEN FILL IN THE ITEMS WHICH HAVE BEEN REPLACED WITH QUESTION MARKS. YOU WILL HAVE TO SUPPLY RELATED SYSTEM CALLS AND THEIR ARGUMENTS, ERROR CODE SYMBOLS, ADDRESSES OF VARIOUS ITEMS IN PACKETS, AND THE APPROPRIATE VALUES IN THOSE PACKETS. WHEN YOU HAVE DETERMINED THE PROPER ENTRIES FOR THE INDICATED ITEMS EDIT THE TWO SOURCE FILES AND MAKE THE CORRECTIONS. COMPILE SHDATA.ONE AND SHDATA.TWO, THEN LINK EACH WITH THE ERROR MODULE.

PREDICT WHAT WILL OCCUR WHEN THE PROGRAMS ARE RUN. EXECUTE SHDATA.ONE AT ONE CONSOLE THEN LOG ON AT A SECOND CONSOLE AND EXECUTE SHDATA.TWO.

USE THE DISPLAY UTILITY TO EXAMINE THE SHARED DATA FILE:

1. YOU SHOULD FIND DATA IN THE FILE. WHY?
2. AT WHAT BITE LOCATION IN THE FILE IS THE FLAG WORD? WHAT IS ITS VALUE?

```
#nolist                              /* SHDATA.ONE.C */
#include <packets:create.h>
#include <packets:block_io.h>
#list

main()
{
  P_CREATE     crepk;                /* packet for file creation   */
  P_???       sppk;                  /* packet for ?SPAGE */

  int  ac0,  ac1,  ac2, error;       /* accumulator and error vars */
  char *byteptr,                     /* general purpose byte pntr  */
       *buffptr;                     /* pointer to shared buffer   */
  int  chnum,                        /* temp storage for channel # */
       flag = 1023;                  /* index to the read flag in  */
                                     /* the shared buffer          */

/* declare a 2048 character buffer, page aligned, in shared
memory */

  ??????? ?????????? char sbuffer[2048];

/* file creation packet spec */

  crepk.cftyp_format = 0;            /* no record format */
  crepk.cftyp_entry  = $FTXT;        /* text type file */
  crepk.ccps         = 0;            /* file ctrl params ignored */
  crepk.ctim         = -1;           /* default time */
  crepk.cacp         = -1;           /* default ACL */
  crepk.cdeh         = 0;            /* reserved */
  crepk.cdel         = 4;            /* element size */
  crepk.cmil         = -1;           /* default max index levels */
  crepk.cmrs         = 0;            /* reserved */


/* SPAGE packet  */

  sppk.psti  = ?;          /* number of blocks to read   */
  sppk.psto  = ?;          /* reserved */
  sppk.pcad  = ????????;           /* W.P. to shared buffer */
  sppk.prnh  = ?;          /* block number */
  sppk.prcl  = 0;
```

```
/* delete the shared file if it exists */

  byteptr = "SDFILE";
  ac0 = byteptr;
  ac1 = 0;
  ac2 = 0;
  if ((error = sys($DELETE,&ac0,&ac1,&ac2)) && (error != ERFDE))
                        errxt(error);

/* create a file for shared data */

  byteptr = "SDFILE";    /* B.P. to filename */
  ac0  = byteptr;        /* ac0 <- B.P. */
  ac1      = 0;          /* reserved */
  ac2  = &crepk;         /* ac2 <- W.P. to create pkt  */
  if (error = sys($CREATE,&ac0,&ac1,&ac2)) errxt(error);

/* open the shared page file */

  byteptr = "SDFILE";            /* B.P. to filename        */
  ac0  = ???????;        /* ac0 <- B.P.          */
  ac1  = ??;             /* system assigns channel #  */
  ac2  = ??;             /* open for read/write      */
  if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);
  chnum = ???;           /* store channel # for SCLOSE */

/* now read a page into shared memory */

  ac0    = 0;            /* reserved set to zero       */
                         /* ac1 has chan # from SOPEN  */
  ac2    = ?????;        /* SPAGE packet address        */
  if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);
```

```
/* move text into shared buffer and set the flag to '$' */

  byteptr = "THIS IS A TEST MESSAGE";
  strcpy(sbuffer, byteptr);
  sbuffer[flag] = '$';

/* wait until SHDATA.TWO resets the flag */

  while(sbuffer[flag] == '$') sleep(1);

/* close shared file */

  ac0 = 0;                /* reserved */
  ac1 = (chnum | ????); /* channel # and release bit  */
  ac2 = 0;                /* reserved          */
  if (error = sys(???????,&ac0,&ac1,&ac2)) errxt(error);

/* return to CLI */

  byteptr = "SHDATA.TWO HAS RECEIVED THE TEST MESSAGE";
  ac0 = 0;
  ac1 = byteptr;
  ac2 = $RFCF + strlen(byteptr);
  if (error = sys($RETURN,&ac0,&ac1,&ac2)) errxt(error);

} /* end main */
```

```
                                        /* SHDATA.TWO.C */
#nolist
#include <stdio.h>
#include <packets:block_io.h>
#list

main( )
{
  P_??? sppk;                           /* packet for ?SPAGE */

  int  ac0,  ac1,  ac2, error;          /* accumulator and error vars */
  char *byteptr,                        /* general purpose byte pntr  */
       *buffptr;                        /* pointer to shared buffer   */
  int  chnum,                           /* temp storage for channel # */
       flag = 1023;                     /* index to flag in buffer */

/* declare a 2048 character buffer, page aligned, in shared
memory */

  ??????? ?????????? char sbuffer[2048];

/* SPAGE packet  */

  sppk.psti  = ?;                       /* number of blocks to read   */
  sppk.psto  = ?;                       /* reserved          */
  sppk.pcad  = ????????;                /* W.P. to shared buffer      */
  sppk.prnh  = ?;                       /* block number          */
  sppk.prcl  = 0;
```

```
/* open the shared page file, retry on ERFDE */

  byteptr = "SDFILE";           /* B.P. to filename          */
  ac1  = ??;                    /* system assigns channel #  */
  ac2  = ??;                    /* open for read/write       */
  do
   {
   ac0 = ???????;
   error = sys(??????,&ac0,&ac1,&ac2);
   if (error == ?????) sleep(1);
   }
  while (error == ?????);
  if (error) errxt(error);
  chnum = ???;          /* store channel # for later  */

/* now read a page into shared memory */

  ac0  = 0;                     /* reserved set to zero      */
  ???  = chnum;                 /* chan # saved from SOPEN   */
  ac2  = ?????;                 /* SPAGE packet address      */
  if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);

/* wait until SHDATA.ONE sets the flag */

  while (sbuffer[flag] != '$') sleep(1);

/* then display the sbuffer */

  printf("%s",sbuffer);

/* now reset the the flag in the shared page */

  sbuffer[flag] = 0;

/* close shared file */

  ac0 = 0;                      /* reserved                  */
  ac1 = (???? | chnum);         /* channel # and release flag */
  ac2 = 0;                      /* reserved                  */
  if (error = sys(???????,&ac0,&ac1,&ac2)) errxt(error);

} /* end main */
```

S309VS RECORD IO LAB

PART ONE: DATASENSITIVE AND VARIABLE

ISSUE THE FOLLOWING COMMANDS TO MOVE NEEDED FILES TO YOUR OWN DIRECTORY:

```
DIRECTORY, :S309VS:C
MOVE/NACL, :UDD:YOURUSERNAME, RECORDIO.C,&
                DATASENS.TXT, VIEW.PR
DIRECTORY/I
```

QPRINT A COPY OF RECORDIO.C. THIS PROGRAM COPIES THE CONTENTS OF THE GENERIC FILE "@DATA" INTO THE FILE "RECORDIO.OUT" ONE RECORD AT A TIME.

IF AN ERROR OCCURS WHEN READING A RECORD, IT CHECKS TO SEE IF THE ERROR WAS "END OF FILE". IF SO, THE PROGRAM TERMINATES NORMALLY.

ON ALL OTHER ERRORS, THE PROGRAM WILL ABORT, RETURNING THE ERROR CODE TO THE PARENT PROCESS.

FILL IN THE FOLLOWING INFORMATION IN THE PACKETS:

I/O PACKET FOR FILE @DATA

OFFSET	OPTIONS OR VALUE
?ISTI	CHANGE RECORD FORMAT TO DATASENSITVE, INPUT ONLY
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	MAX RECORD LENGTH SAME AS NUMBER OF BYTES IN "BUFF"
?IRNH	NEXT SEQUENTIAL RECORD
?IFNP	BYTE POINTER TO FILENAME "@DATA"
?IDEL	DEFAULT DELIMITER TABLE

I/O PACKET FOR FILE RECORDIO.OUT

OFFSET	OPTIONS OR VALUE
?ISTI	CHANGE RECORD FORMAT TO DATASENSITVE, OUTPUT ONLY, DELETE OLD FILE IF IT EXISTS, CREATE NEW FILE
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	MAX RECORD LENGTH SAME AS NUMBER OF BYTES IN "BUFF"
?IRNH	NEXT SEQUENTIAL RECORD
?IFNP	BYTE POINTER TO FILENAME "RECORDIO.OUT"
?IDEL	DEFAULT DELIMITER TABLE

WHAT IS THE SYMBOLIC NAME OF THE PACKET FOR THE FILE @DATA ?

_____

WHAT IS THE SYMBOLIC NAME OF THE PACKET FOR THE FILE RECORDIO.OUT?_____

COMPILE THE SOURCE FILE, USING THE /LINEID SWITCH AND OBTAINING A PRINTED LISTING. LINK THE PROGRAM WITH THE ERROR MODULE.

EXECUTE THE PROGRAM. WHAT ERROR MESSAGE OCCURRED ?

_____

WHAT FILE IS CAUSING THE PROBLEM ?

_____

USE A CLI COMMAND TO SET "DATASENS.TXT" AS THE GENERIC FILE YOUR PROGRAM WILL TRY TO READ. WHAT IS THE COMMAND ?

_____

EXECUTE THE PROGRAM. YOU SHOULD GET THE MESSAGE "IO COMPLETED"

COMPARE THE CONTENTS OF YOUR RECORDIO.OUT FILE AND DATASENS.TXT. THEY SHOULD BE IDENTICAL.

NOW MODIFY THE PROGRAM TO OUTPUT VARIABLE LENGTH RECORDS INSTEAD OF DATASENSITIVE:

NOTE THAT ONLY THE OUTPUT IS TO BE CHANGED - THE INPUT IS STILL TO BE DATASENSITIVE.

IT WILL BE NECESSARY TO SPECIFY THE LENGTH OF EACH RECORD AS IT IS WRITTEN OUT.

AFTER EACH RECORD HAS BEEN READ IN, WHERE IS ITS LENGTH STORED ?

_____

IN ORDER TO OUTPUT THE RECORD IN VARIABLE FORMAT, WHAT OFFSET IN THE OUTPUT PACKET MUST CONTAIN THIS INFORMATION ? _____

ADD SOME CODE AFTER THE ?READ AND BEFORE THE ?WRITE CALL TO GET THE ACTUAL RECORD LENGTH READ IN AND STORE IT IN THE OUTPUT PACKET:

ONE STATEMENT SHOULD BE ENOUGH

_____

CHANGE THE RECORD FORMAT SPECIFICATION IN THE @OUTPUT PACKET TO VARIABLE.

COMPILE AND LINK YOUR PROGRAM.

USING THE CLI COMMAND "X, VIEW, RECORDIO.OUT", YOU CAN SEE THE RECORDS IN YOUR PROGRAM'S OUTPUT FILE. THESE VARIABLE LENGTH RECORDS EACH START WITH FOUR DIGITS THAT REVEAL THE RECORD LENGTH.

SINCE THE "VIEW" PROGRAM OPENS THE SPECIFIED FILE AS THOUGH IT HAD DATASENSITIVE RECORDS, THESE RECORD LENGTHS WILL BE DISPLAYED AS PART OF EACH LINE ON YOUR SCREEN. NOTICE THAT THE LENGTH OF EACH RECORD INCLUDES THE FOUR DIGITS.

```
                                    /* RECORDIO.C */
#nolist
#include <packets:normal_io.h>
#list
#define bufsz 136

main()
{
int  ac0,  ac1,  ac2, error;     /* accumulator and error vars */
char buff[bufsz];

/* declare packet structures */

P_???           outpk, datpk;    /* packets for record I/O   */

  /* I/O packet for file @DATA */

datpk.ich  = 0;                  /* channel # returned by sys */
datpk.isti = ?????|?????|?????; /* change to datasens input  */
datpk.isto = 0;                  /* file type (returned)      */
datpk.imrs = -1;                 /* default physical blk size */
datpk.ibad = ????;               /* buffer bytepointer        */
datpk.ires = 0;                  /* reserved                  */
datpk.ircl = ?????;              /* max record size           */
datpk.irlr = 0;                  /* record length returned    */
datpk.irnw = 0;                  /* reserved                  */
datpk.irnh = 0;                  /* record number             */
datpk.ifnp = ???????;            /* B.P. to filename          */
datpk.idel = ??;                 /* default delimiters        */

  /* I/O PACKET FOR FILE RECORDIO.OUT */

outpk.ich  = 0;                  /* channel # returned by sys */
outpk.isti = ?????|?????|?????|?????|?????; /* recreate for d-s
out */
outpk.isto = 0;                  /* file type                 */
outpk.imrs = -1;                 /* default physical blk size */
outpk.ibad = ????;               /* buffer bytepointer        */
outpk.ires = 0;                  /* reserved                  */
outpk.ircl = ?????;              /* max record size           */
outpk.irlr = 0;                  /* record length returned    */
outpk.irnw = 0;                  /* reserved                  */
outpk.irnh = 0;                  /* record number             */
outpk.ifnp = ????????????????;   /* B.P. to filename          */
outpk.idel = ??;                 /* default delimiters        */
```

```
/* OPEN INPUT FILE */

        ac0 = 0;
        ac1 = 0;
        ac2 = ??????;
        if (error = sys(?????, &ac0, &ac1, &ac2)) errxt(error);

/* OPEN OUTPUT FILE */

        ac0 = 0;
        ac1 = 0;
        ac2 = ??????;
        if (error = sys(?????, &ac0, &ac1, &ac2)) errxt(error);

        do
          {

/* READ A RECORD FROM INPUT FILE */

          ac0 = 0;
          ac1 = 0;
          ac2 = ??????;
          if(!(error = sys(?????,&ac0,&ac1,&ac2)))
            {

/* WRITE THE RECORD TO OUTPUT FILE */

            ac0 = 0;
            ac1 = 0;
            ac2 = ??????;
            if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);
            }
          }
        while (!error);
        if (error != ?????) errxt(error);

        ac0 = 0;
        ac1 = "I/O COMPLETED";
        ac2 = strlen("I/O COMPLETED");
        if (error = sys($RETURN,&ac0,&ac1,&ac2)) errxt(error);

}                                           /* END MAIN */
```

S309VS RECORD IO LAB

PART TWO: DYNAMIC AND FIXED

ISSUE THE FOLLOWING COMMANDS TO MOVE NEEDED FILES TO YOUR OWN DIRECTORY:

```
DIRECTORY, :S309VS:C
MOVE/R/NACL, :UDD:YOURUSERNAME, DYNIO.C, SYSLOG, ERROR.OB
DIRECTORY/I
```

REFER TO THE PRINTED COPY OF THE SOURCE FILE "DYNIO.C" ON THE FOLLOWING PAGES. THIS PROGRAM READS A SYSTEM LOG FILE, OUTPUTTING ONLY RECORDS OF A CERTAIN TYPE.

THE FORMAT OF THE SYSTEM LOG FILE MUST BE STUDIED BEFORE ONE CAN WRITE A PROGRAM TO READ IT. DYNAMIC RECORD FORMAT MUST BE USED.

EACH RECORD IN THE FILE STARTS WITH A 16. BYTE RECORD HEADER. THE FIRST TWO **WORDS** OF EACH HEADER ARE A 32 BIT NUMBER (NOT FOUR DIGITS AS IN VARIABLE RECORD FORMAT) REVEALING THE LENGTH IN **WORDS** OF THAT RECORD.

WORD FIVE OF THE HEADER CONTAINS A CODE WHICH TELLS THE ENTRY TYPE OF THE RECORD. IN THIS PROGRAM WE WILL OUTPUT ONLY TYPE 3 RECORDS AND SKIP OVER ALL OTHERS.

IN THESE TYPE 3 RECORDS, THE 32 BYTES FOLLOWING THE HEADER CONTAIN A PROCESS NAME (PADDED WITH NULLS), WHICH WE WILL OUTPUT. THERE ARE ALSO 8 ADDITIONAL WORDS OF INFORMATION (16. BYTES) WHICH THE PROGRAM WILL NOT OUTPUT, BUT MUST SKIP OVER TO GET TO THE NEXT RECORD.

EDIT YOUR COPY OF THE SOURCE FILE "DYNIO.C".

FILL IN THE FOLLOWING INFORMATION IN THE PACKETS:

I/O PACKET FOR FILE 'SYSLOG'

OFFSET	OPTIONS OR VALUE
?ICH	(CHANNEL NUMBER WILL BE RETURNED BY SYSTEM)
?ISTI	CHANGE RECORD FORMAT TO DYNAMIC, INPUT ONLY
?ISTO	(FILE TYPE WILL BE RETURNED BY SYSTEM)
?IMRS	DEFAULT PHYSICAL BLOCK SIZE
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	MAX RECORD LENGTH SAME AS NUMBER OF BYTES IN "BUFF"
?IRLR	(RECORD LENGTH WILL BE RETURNED BY SYSTEM)
?IRNH	(WILL BE SET BY PROGRAM AT EACH ?READ)
?IFNP	BYTE POINTER TO FILENAME "@DATA"
?IDEL	DEFAULT DELIMITER TABLE

I/O PACKET FOR FILE '@OUTPUT'

OFFSET	OPTIONS OR VALUE
?ICH	(CHANNEL NUMBER WILL BE RETURNED BY SYSTEM)
?ISTI	CHANGE RECORD FORMAT TO FIXED, RECREATE FOR OUTPUT
?ISTO	(FILE TYPE WILL BE RETURNED BY SYSTEM)
?IMRS	DEFAULT PHYSICAL BLOCK SIZE
?IBAD	BYTE POINTER TO BUFFER "BUFF"
?IRCL	33 BYTE FIXED RECORDS
?IRLR	(RECORD LENGTH WILL BE RETURNED BY SYSTEM)
?IRNH	NEXT SEQUENTIAL RECORD
?IFNP	BYTE POINTER TO FILENAME "@OUTPUT"
?IDEL	DEFAULT DELIMITER TABLE

COMPILE THE MAIN PROGRAM USING THE /LINEID SWITCH. LINK THE MAIN PROGRAM TOGETHER WITH THE ERROR MODULE.

EXECUTE THE PROGRAM. YOU SHOULD SEE A BUNCH OF PROCESS NAMES ON YOUR SCREEN.

EACH PROCESS NAME WAS OUTPUT AS A FIXED LENGTH 33. BYTE RECORD. WHY DO YOU SUPPOSE THEY DON'T APPEAR TO BE THE SAME LENGTH ?

TO SEE EXACTLY WHAT IS OUTPUT, IT IS NECESSARY TO HAVE THE OUTPUT GO TO A DISK FILE INSTEAD OF A CONSOLE SCREEN.

CREATE AN EMPTY FILE CALLED DYNIO.OUT .

RUN YOUR PROGRAM AGAIN USING THE CLI COMMAND:

```
) PROC/BLOCK/DEF/OUT=DYNIO.OUT, DYNIO
```

NOW WE CAN EXAMINE THE CONTENTS OF THE OUTPUT FILE USING THE DISPLAY UTILITY:

```
) X, DISPLAY, DYNIO.OUT
```

THE FIRST COLUMN CONTAINS THE WORD ADDRESS OF THE FIRST WORD IN EACH LINE. EACH LINE CONTAINS 8. WORDS (2 CHARACTERS PER WORD) SHOWN IN OCTAL.

THE RIGHT COLUMNS SHOW THE TEXT CHARACTERS WITH NON-PRINTING CHARACTERS REPLACED BY PERIODS. WAS EACH RECORD 33. BYTES LONG AS EXPECTED ? WHAT NON-PRINTING CHARACTERS ARE CONTAINED IN EACH RECORD ? WHERE DID THEY COME FROM ?

****
CHALLENGE 1 -
****

WRITE A PROGRAM TO READ A FILE OF FIXED RECORDS LAST-LINE-FIRST AND WRITE THE RECORDS TO ANOTHER FILE IN REVERSE OF THEIR ORIGINAL ORDER.

THE FILE 'BOTTOM_UP' WHICH CONTAINS FIXED RECORDS OF 80. CHARACTERS, WRITTEN FIRST-LINE-LAST, IS PROVIDED IN THE :S309VS:C DIRECTORY FOR TESTING YOUR PROGRAM.

****
CHALLENGE 2 -
****

WRITE A PROGRAM TO READ A FILE OF FIXED LENGTH 80 BYTE RECORDS, STRIPPING TRAILING BLANKS FROM EACH RECORD BY REPLACING THE FIRST BLANK AFTER THE LAST SIGNIFICANT CHARACTER WITH A NEW-LINE, AND WRITE THE RECORDS TO A DATA SENSITIVE FILE.

THE FILE 'PADDED.TXT' WHICH CONTAINS FIXED RECORDS OF 80. CHARACTERS, PADDED WITH TRAILING SPACES, IS PROVIDED IN THE :S309VS:C DIRECTORY FOR TESTING YOUR PROGRAM.

```
        /* DYNIO.C */
#nolist
#include <packets:normal_io.h>
#list

main()
{
int  ac0,  ac1,  ac2, error;    /* accumulator and error vars */
union
  {
  char pname[33];
  struct        /* buff can contain either */
    {                   /* a header or a proc name */
    int reclen;
    short date;
    int time;
    short event;
    } hdr;
  }   buff;
char *buffbp = &buff;
int  recoff;    /* record offset */

/* declare packet structures */

P_???    outpk, logpk;  /* packets for record I/O   */

        /* I/O packet for log file */

logpk.ich  = ?;          /* channel # returned by sys */
logpk.isti = ?????|?????|?????; /* change to dynamic input  */
logpk.isto = ?;          /* file type (returned)      */
logpk.imrs = ??;         /* default physical blk size */
logpk.ibad = ??????;     /* buffer bytepointer   */
logpk.ires = ?;          /* reserved            */
logpk.ircl = ??;         /* max record size         */
logpk.irlr = ?;          /* record length returned    */
logpk.irnw = ?;          /* reserved      */
logpk.irnh = ?;          /* record number set each rd */
logpk.ifnp = "SYSLOG";   /* B.P. to filename       */
logpk.idel = -1;         /* default delimiters       */

/* I/O PACKET FOR @OUTPUT */

outpk.ich  = ?;          /* channel # returned by sys */
outpk.isti = ?????|?????|?????|?????|?????;
                         /* recreate for fixed out */
outpk.isto = ?;          /* file type             */
outpk.imrs = ??;         /* default physical blk size */
outpk.ibad = ??????;     /* buffer bytepointer   */
outpk.ires = ?;          /* reserved            */
outpk.ircl = ??;         /* 33 byte record size       */
outpk.irlr = ?;          /* record length returned    */
outpk.irnw = ?;          /* reserved      */
outpk.irnh = ?;          /* record number       */
outpk.ifnp = ??????????; /* B.P. to filename       */
outpk.idel = -1;         /* default delimiters       */
```

```
/* OPEN INPUT FILE */

  ac0 = 0;
    ac1 = 0;
    ac2 = ??????;
  if (error = sys(?????, &ac0, &ac1, &ac2)) errxt(error);

/* OPEN OUTPUT FILE */

  ac0 = 0;
    ac1 = 0;
    ac2 = ??????;
  if (error = sys(?????, &ac0, &ac1, &ac2)) errxt(error);

  recoff = 0;    /* SELECT FIRST RECORD */

  do
          {
    logpk.irnh = recoff;          /* ADVANCE TO NEXT RECORD */
    logpk.ircl = 16;     /* READ 16 BYTE HEADER */
    ac0 = 0;
     ac1 = 0;
     ac2 = ??????;
    if (!(error = sys(?????,&ac0,&ac1,&ac2)))
      { /* PREPARE TO SKIP REST OF RCD */
      recoff = buff.hdr.reclen*2-16;
      if (buff.hdr.event == 3)  /* BUT IF THIS IS A TYPE 3 RCD */
        {                       /* READ A 32 BYTE PROCNAME     */
        logpk.???? = 32;
        logpk.irnh = 0;
        ac0 = 0;
         ac1 = 0;
         ac2 = ??????;
        if (error = sys(?????,&ac0,&ac1,&ac2)) errxt(error);
                                /* WRITE PROCNAME TO @OUTPUT   */
        buff.pname[32] = '\n';
        ac0 = 0;
        ac1 = 0;
        ac2 = ??????;
        if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);
        recoff = 16;     /* PREPARE TO SKIP REMAINING 16 BYTES */
        }
      }
    } while (!error);
  if (error != ?????) errxt(error);

  ac0 = 0;
  ac1 = "I/O COMPLETED";
  ac2 = strlen("I/O COMPLETED");
  if (error = sys($RETURN,&ac0,&ac1,&ac2)) errxt(error);

}       /* END MAIN */
```

C LANGUAGE SCREEN MANAGEMENT LAB

* MOVE/R/NACL COPIES OF THE FILES 'SCREEN.C' AND 'ERROR.OB' FROM THE :S309VS:C DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'SCREEN' IS TO PRESENT A DATA ENTRY FORM ON THE SCREEN AND ALLOW THE USER TO ENTER THE REQUIRED DATA ONE FIELD AT A TIME. WHEN THE USER COMPLETES ALL THE FIELDS, HE IS ASKED IF HE WANTS TO CHANGE ANY OF THEM. IF SO, HE IS ASKED FOR THE FIELD NUMBER AND ALLOWED TO EDIT THE FIELD. WHEN THE USER HAS NO MORE CHANGES, HE IS ASKED IF HE WISHES TO FILL IN ANOTHER DATA ENTRY SCREEN.

* THE FUNCTION OF 'ERROR' IS TO PROVIDE AN ERROR HANDLER FOR THE SYSTEM CALLS. IT WILL REPORT THE ERROR MESSAGE AND PROGRAM LOCATION WHEN A SYSTEM CALL FAILS TO EXECUTE PROPERLY.

* REFER TO THE PRINTOUT OF 'SCREEN.C' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM CALLS AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
CC/LINEID, SCREEN
CCL, SCREEN, ERROR
```

* GET A HARDCOPY OF THE FILE 'SCREEN.LS'.

* EXECUTE THE SCREEN PROGRAM. YOU SHOULD SEE A BLANK DATA ENTRY FORM ON YOUR SCREEN.

* USING IMAGINARY NAMES AND ADDRESSES, EXPERIMENT WITH FILLING IN THE FIELDS. NOTICE THAT THE FIELDS ARE AUTO-TERMINATING.

* STUDY THE PROGRAM TO UNDERSTAND HOW IT MAKES USE OF THE SCREEN MANAGEMENT EXTENSION AND AUTO-TERMINATING READS.

* TRY ENTERING INVALID RESPONSES TO THE Y/N QUESTIONS AND/OR THE FIELD NUMBER QUESTION. NOTICE HOW THE PROGRAM ACCOMPLISHES THE SAVING OF THE INPUT CURSOR POSITION WHILE IT DISPLAYS THE ERROR MESSAGES. WHICH SCREEN MANAGEMENT FLAG IS REQUIRED FOR THIS PROGRAM FEATURE TO WORK?

* YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE PROGRAM REPEATEDLY WITHOUT ERROR AND UNDERSTAND ITS USE OF SCREEN MANAGEMENT AND AUTO-TERMINATING READS.

```
#nolist                              /* SCREEN.C */
#include <packets:normal_io.h>
#list

/* declare indices for the prompts */

#define mxfld   7
#define ch      mxfld
#define wf      ch+1
#define ag      wf+1
#define yn      ag+1
#define fb      yn+1
#define cl      fb+1
#define ee      cl+1
#define us      ee+1
#define ue      us+1

  int  ac0,  ac1,  ac2, error;  /* accumulators & error vars */
  int  i, edit, chflg;

  P_?????? oupkt, inpkt;        /* extended packets for I/O */
  P_?????? scpkt;               /* screen mgmt packet       */

/* declare arrays of chars for the individual field labels */

  char lbl0[] = "1. First Name",
       lbl1[] = "2. MI",
       lbl2[] = "3. Last Name",
       lbl3[] = "4. Street Address",
       lbl4[] = "5. City",
       lbl5[] = "6. State",
       lbl6[] = "7. Zip";

/* declare buffers for auto terminating reads */

  char firstname[] = "                \177\177",
       initial[]   = " \177\177",
       lastname[]  = "                 \177\177",
       street[]    = "                          \177\177",
       city[]      = "                      \177\177",
       state[]     = "  \177\177",
       zip[]       = "     \177\177",
       ynbuff[]    = " \177\177";

char                        /* declare user prompts */
  chmsg[] = "\013Any changes (Y or N)?",
  wfmsg[] = "\013What number?",
  agmsg[] = "\013Do you wish to enter another form (Y or N)?",
  ynmsg[] = "Please respond with a Y or N\007",
  fbmsg[] = "Please respond with a digit from 1 to 7\007",
  clmsg[] = "\014",      /* clear screen */
  eemsg[] = "\013",      /* erase to end of line */
  usmsg[] = "\024",      /* start underscore */
  uemsg[] = "\025";      /* end underscore */
```

```
/* declare a struture template that describes a field entry */

  typedef struct entry {
        short lcol;              /* label cursor pos col */
        short lrow;              /* label cursor pos row */
        char  *flbp;             /* label text byte pointer */
        short fcol;              /* field cursor pos col */
        short frow;              /* field cursor pos row */
        char  *fbbp;             /* field buffer byte pointer */
        short fbln;              /* field byte length */
                        } entry;

/* initialize the entries into a format table */

  entry ftbl[] = {
        {20, 5, lbl0, 20, 4 , firstname, sizeof(firstname) - 1},
        {37, 5, lbl1, 39, 4 , initial  , sizeof(initial)   - 1},
        {45, 5, lbl2, 45, 4 , lastname , sizeof(lastname)  - 1},
        {29, 9, lbl3, 25, 8 , street   , sizeof(street)    - 1},
        {25,13, lbl4, 20, 12, city     , sizeof(city)      - 1},
        {43,13, lbl5, 46, 12, state    , sizeof(state)     - 1},
        {54,13, lbl6, 55, 12, zip      , sizeof(zip)       - 1},
        {20,18,chmsg, 42, 18, ynbuff   , sizeof(ynbuff)    - 1},
        {45,18,wfmsg, 58, 18, ynbuff   , sizeof(ynbuff)    - 1},
        {20,18,agmsg, 64, 18, ynbuff   , sizeof(ynbuff)    - 1},
        {20,20,ynmsg, -1, -1, 0        ,                     0},
        {20,20,fbmsg, -1, -1, 0        ,                     0},
        { 0, 0,clmsg, -1, -1, 0        ,                     0},
        { 0,20,eemsg, -1, -1, 0        ,                     0},
        { 0, 0,usmsg, -1, -1, 0        ,                     0},
        { 0, 0,uemsg, -1, -1, 0        ,                     0},
                };
```

```
main( )
{

  /* @INPUT packet */

  inpkt.ich    = 0;       /* channel # returned by sys */
  inpkt.isti   = ?????|?????|?????|?????;
                          /* chg rcd fmt to D-S, input, ext pkt */
  inpkt.isto   = 0;       /* file type                  */
  inpkt.imrs   = -1;      /* default physical blk size */
  inpkt.ibad   = -1;      /* B.P. to READ   buffer      */
  inpkt.ires   = 0;       /* reserved                   */
  inpkt.ircl   = -1;      /* max rec.size, spec on READ*/
  inpkt.irlr   = 0;       /* record len returned        */
  inpkt.irnw   = 0;       /* reserved                   */
  inpkt.irnh   = 0;       /* record number              */
  inpkt.ifnp   = "@INPUT";        /* B.P. to filename   */
  inpkt.idel   = -1;      /* default delimiters         */
  inpkt.etsp   = ??????; /* W. P. to scrn pkt          */
  inpkt.etft   = 0;       /* field trans pkt ptr        */
  inpkt.etlt   = 0;       /* labeled tape pkt           */
  inpkt.enet   = 0;       /* netowrk ext pkt            */


  /* @OUTPUT packet */

  oupkt.ich    = 0;       /* channel # returned by sys */
  oupkt.isti   = ?????|?????|?????|?????;
                          /* chg rcd fmt to D-S, output, ext pk */
  oupkt.isto   = 0;       /* file type                  */
  oupkt.imrs   = -1;      /* default physical blk size */
  oupkt.ibad   = -1;      /* B.P. to WRITE  buffer      */
  oupkt.ires   = 0;       /* reserved                   */
  oupkt.ircl   = 80;      /* max record size            */
  oupkt.irlr   = 0;       /* record len returned        */
  oupkt.irnw   = 0;       /* reserved                   */
  oupkt.irnh   = 0;       /* record number              */
  oupkt.ifnp   = "@OUTPUT";       /* B.P. to filename   */
  oupkt.idel   = -1;      /* default delimiters         */
  oupkt.etsp   = ??????; /* W. P.  to scrn pkt         */
  oupkt.etft   = 0;       /* field trans pkt ptr        */
  oupkt.etlt   = 0;       /* labeled tape pkt           */
  oupkt.enet   = 0;       /* network ext pkt            */


  /* screen management packet */

  scpkt.????  = ?????|?????|?????|?????;
        /* enable screenedit, initial cursor position,
   do not echo delimiters,display buffer before read */
  scpkt.????  = 0;        /* relative cur pos not used */
  scpkt.????????  = 0;    /* initial cur col for READ  */
  scpkt.????????  = 0;    /* initial cur row for READ  */
```

```
/************************************************************
 *                                                          *
 *        MM   MM    AAA    IIIIIII  NN    N               *
 *        M M M M   A   A     I I    N N   N               *
 *        M  M  M  AAAAAAA    I I    N  N  N               *
 *        M     M  A     A    I I    N   N N               *
 *        M     M  A     A  IIIIIII  N    NN               *
 *                                                          *
 ************************************************************/

              /* OPEN @INPUT */

ac0 = 0;
ac1 = 0;
ac2 = ??????;
if (error = sys(?????, &ac0, &ac1, &ac2)) errxt(error);

              /* OPEN @OUTPUT */

ac0 = 0;
ac1 = 0;
ac2 = ??????;
if (error = sys(?????, &ac0, &ac1, &ac2)) errxt(error);

do {  /* do until again returns 0  */

dispfld(cl);                    /* clear screen */
for(i=0;i<mxfld;i++)            /* display form to fill in */
  dispfld(i);                   /* by displaying each field */

/* loop thru, filling each field from keyboard input */

edit = 0;
for(i=0;i<mxfld;i++)
  rdfld(i,edit);

do                      /* make changes if any are required */
  if (chflg = yorn(ch))
    {
    i = fldnm();
    edit = 1;
    rdfld(i,edit);
    }
while (chflg); /* repeat until no change requested */

} while (yorn(ag)); /* loop thru until again returns 0  */

} /* end main */
```

```
  /*****************************************************
   *                                                   *
   *                  dispfld                          *
   *                                                   *
   *            display specified field                *
   *                                                   *
   *****************************************************/


dispfld(f)
int     f;
{
  scpkt.???????? = ftbl[f].lcol;  /* set column in screen  pkt */
  scpkt.???????? = ftbl[f].lrow;  /* set row in screen  pkt */

  oupkt.????     = ftbl[f].flbp;  /* set buffer bytepointer */
  oupkt.????    |= ????; /* set bit zero in pntr to screen pkt */

  ac0 = 0;
  ac1 = 0;
  ac2 = ??????;
  if (error = sys(??????, &ac0, &ac1, &ac2)) errxt(error);

} /* end dispfld */
```

```
 /************************************************************
  *                                                          *
  *                        rdfld                             *
  *                                                          *
  * if edit flag is off, fill specified buffer with spaces,  *
  * position to specified screen position and display buffer *
  * contents, then read from keyboard into specified buffer. *
  *                                                          *
  ************************************************************/


rdfld(f, eflg)
int   f, eflg;
{
 int    j;              /* local loop counter */

 if (!edit)     /* if not editing, fill buffer with spaces */
   for (j=0;j<ftbl[f].fbln - 2;j++)  ftbl[f].fbbp[j] = ' ';

 ftbl[f].fbbp[ftbl[f].fbln-2] = '\177';  /* store two DELs */
 ftbl[f].fbbp[ftbl[f].fbln-1] = '\177';  /* at buffer end. */
 dispfld(us);                    /* start underscore */
 scpkt.???????? = ftbl[f].fcol; /* set column in screen  pkt */
 scpkt.???????? = ftbl[f].frow; /* set row in screen  pkt */

 inpkt.????     = ftbl[f].fbbp; /* set buffer bp in input pkt */
 inpkt.????     = ftbl[f].fbln; /* set record length */
 inpkt.????    |= ????; /* set bit zero in pntr to screen pkt */

 ac0 = 0;
 ac1 = 0;
 ac2 = ??????;
 if (error = sys(?????, &ac0, &ac1, &ac2)) errxt(error);
 dispfld(ue);                    /* end underscore */

}/* end rdfld */
```

```
/*************************************************************
 *                                                           *
 *                          yorn(f)                          *
 *                                                           *
 *        find out whether the user wants YES or NO          *
 *                                                           *
 *        returns 1 if YES, 0 if NO                          *
 *                                                           *
 *************************************************************/


int yorn(f)   /* find out whether the user wants YES or NO */
  int f;

  {
  int flag;

  dispfld(ee);                              /* erase error msg */
  dispfld(f);                               /* ask question */

  do
    {
    edit = 0;                               /* read response */
    rdfld(f,edit);

    switch(ftbl[f].fbbp[0]) {               /* validate response */
        case 'Y':
        case 'y': flag = 1; break;
        case 'N':
        case 'n': flag = 0; break;
        default : flag = -1;                /* if response is
invalid, */
                  dispfld(yn);      /* display error message */
        } /* end switch */

    }
  while (flag == -1);   /* repeat until response is vaild */

  return(flag);
} /* end yorn routine */
```

```
/*************************************************************
 *                                                           *
 *                          fldnm()                          *
 *                                                           *
 *             returns field number to be changed            *
 *                                                           *
 *************************************************************/

int fldnm()
  {
  int fld;

  dispfld(ee);          /* erase error msg */
  dispfld(wf);          /* ask which field to change */

  do
    {
    edit = 0;           /* get response */
    rdfld(wf,edit);

    fld = atoi(ftbl[wf].fbbp);  /* validate response */

    if (fld < 1 || fld > mxfld) /* if response is invalid, */
      dispfld(fb);              /* display error message */

    }
  while (fld < 1 || fld > mxfld); /* repeat until valid */

  return (fld-1);
  }             /* end of routine fldnm */
```

* MOVE/R/NACL COPIES OF THE FILES 'PROC.C' AND 'ERROR.OB' FROM THE :S309VS:C DIRECTORY TO YOUR WORKING DIRECTORY.

* THE FUNCTION OF 'PROC' IS TO CREATE A SUBORIDNATE PROCESS RUNNING THE CLI PROGRAM, AND TO PASS AN INNITIAL IPC MESSAGE CONTAINING A CLI COMMAND TO THE NEW PROCESS. THE NEW PROCESS IS TO EXECUTE THE CLI COMMAND AND IMMEDIATELY TERMINATE.

* REFER TO THE PRINTOUT OF 'PROC.C' ON THE FOLLOWING PAGES. YOU WILL HAVE TO REPLACE ALL QUESTION MARKS (?) WITH THE CORRECT VALUES. THESE INCLUDE THE ACTUAL SYSTEM CALLS AND THE CONTENTS OF THE PACKETS.

* AFTER UPDATING OF THE FILE EXECUTE THE FOLLOWING COMMANDS:

```
CC/LINEID, PROC
CCL, PROC, ERROR
```

* GET A HARDCOPY OF THE FILE 'PROC.LS'.

* EXECUTE THE PROC PROGRAM. YOUR ORIGINAL CLI PROCESS BECOMES BLOCKED UNTIL THE 'PROC' PROGRAM TERMINATES. THE 'PROC' PROGRAM CREATES THE SUBORDINATE CLI AND THE 'PROC' PROCESS BECOMES BLOCKED WAITING FOR THE NEW CLI TO TERMINATE. THE NEW CLI EXECUTES THE COMMAND IN THE IPC MESSAGE ASSOCIATED WITH THE ?PROC PACKET AND THEN TERMINATES. THEN THE 'PROC' PROCESS BECOMES UNBLOCKED AND TERMINATES ITSELF, WHICH NOW UNBLOCKS YOUR ORIGINAL CLI PROCESS.

* USE THE CLI TYPE COMMAND TO EXAMINE THE CONTENTS OF 'OUTFILE'. IT SHOULD CONTAIN THE MESSAGE "HELLO FROM YOUR CREATOR".

* YOU HAVE COMPLETED THIS LAB WHEN YOU CAN EXECUTE THE PROGRAM REPEATEDLY WITHOUT ERROR

****
* CHALLENGE *
****

* MODIFY THE PROGRAM SO THAT THE NEW CLI DOES NOT TERMINATE AFTER COMPLETING THE CLI COMMAND IN THE IPC MESSAGE, BUT GIVES THE CLI PROMPT INSTEAD. YOU WILL NEED TO PASS ?RFCF (CLI FORMAT) IN THE USER FLAGS WORD, AND TO ADD 'CLI,' TO THE BEGINNING OF THE IPC MESSAGE TEXT STRING.

```
#nolist                              /* PROC.C */
#include <packets:process.h>
#include <packets:ipc.h>
#list

main()
{

  int ac0, ac1, ac2, error;

  P_???? ppkt;
  P_????? ipcpk;

  char msg[] = "WRITE/L=OUTFILE,HELLO,FROM,YOUR,CREATOR";

  /* PACKET FOR ?PROC */

  ppkt.pflg = ?????;     /* block creator, swappable */
  ppkt.ppri = ??;        /* priority same as creator */
  ppkt.???? = "CLI.PR"; /* byte ptr to program name */
  ppkt.pipc = ??????;    /* address of ipc header */
  ppkt.???  = ??;        /* default process name */
  ppkt.???? = ??;        /* max addr space same as creator */
  ppkt.pdir = ?;      /* initial directory = creator's current */
  ppkt.???? = ??;        /* @CONSOLE same as creator */
  ppkt.???? = ??;        /* default max system calls */
  ppkt.???? = ??;        /* max WSS same as creator */
  ppkt.???? = ??;        /* username same as creator */
  ppkt.???? = ??;        /* privileges same as creator */
  ppkt.???? = ??;        /* sons quota = remainder of creator's */
  ppkt.???? = ??;        /* min WSS same as creator */
  ppkt.???? = ??;        /* @INPUT same as creator */
  ppkt.???? = ??;        /* @OUTPUT same as creator */
  ppkt.???? = ??;        /* @LIST same as creator */
  ppkt.???? = ??;        /* @DATA same as creator */
  ppkt.???? = ??;        /* CPU limit = remainder of creator's */

  /* HEADER FOR INITIAL IPC MESSAGE */

  ipcpk.isfl = ?;        /* no system flags */
  ipcpk.iufl = ?;        /* no user flags = do CLI cmd and quit */
  ipcpk.idph = ?;        /* global destination port */
  ipcpk.iopn = ?;        /* local origin port */
  ipcpk.???? = sizeof(msg)/2;   /* message length (words) */
  ipcpk.iptr = ????;     /* message (word) address */

  /* CREATE SUBORDINATE PROCESS */

  ac0 = 0;
  ac1 = 0;
  ac2 = ?????;
  if (error = sys(?????,&ac0,&ac1,&ac2)) errxt(error);

}                              /* end main */
```

C LANGUAGE CONTROL IPC LAB

* MOVE/NACL A COPY OF THE FILE CONTROL.IPC.C AND ERROR.OB FROM THE :S309VS:C DIRECTORY TO YOUR INITIAL WORKING DIRECTORY.

* REFER TO THE PRINTOUT OF CONTROL.IPC.C ON THE FOLLOWING PAGES.

* THIS PROGRAM ACCEPTS ANY IPC MESSAGE SENT TO THE FILE 'TRYIPC' FROM ANY CONSOLE. TO DO THIS THE CONSOLE DOES NOT HAVE TO BE ASSOCIATED WITH THE RECEIVING PROCESS.

* WHAT IS THE CLI COMMAND WHICH ALLOWS YOU TO INTERACTIVELY SEND AN IPC MESSAGE TO A PROCESS?

---

* EACH MESSAGE RECEIVED IS DISPLAYED ON THE PROCESS' CONSOLE, THEN AN ACKNOWLEDGEMENT IS RETURNED TO THE SENDER.

* THIS PROGRAM ACCEPTS ANY IPC MESSAGE SENT TO THE FILE 'TRYIPC' FROM ANY CONSOLE. TO DO THIS THE CONSOLE DOES NOT HAVE TO BE ASSOCIATED WITH THE RECEIVING PROCESS.

* EACH MESSAGE RECEIVED IS DISPLAYED ON THE PROCESS' CONSOLE, THEN AN ACKNOWLEDGEMENT IS RETURNED TO THE SENDER.

* EDIT YOUR COPY OF THE SOURCE FILE AND REPLACE THE QUESTION MARKS WITH THE CORRECT SYMBOLS.

* COMPILE WITH THE /LINEID SWITCH AND LINK YOUR PROGRAM WITH THE ERROR MODULE.

* EXECUTE YOUR PROGRAM.

* LOG ON AT ANOTHER TERMINAL AND SEND MESSAGES TO YOUR PROGRAM:

```
) CONTROL TRYIPC YOUR MESSAGE GOES HERE
```

EACH TIME YOU SHOULD GET THE RESPONSE:

```
FROM PID XX: I GOT THE MESSAGE
```

EACH MESSAGE WILL BE DISPLAYED ON YOUR RECEIVING PROCESS' CONSOLE. NOTE THAT THE CLI COMMAND USED TO SEND THESE MESSAGES EDITS THEM SOMEWHAT BEFORE THEY ARE SENT.

* WHEN YOU HAVE SENT AS MANY MESSAGES AS YOU WANT, ENTER CONTROL-C, CONTROL-B AT YOUR RECEIVING CONSOLE.

* ---------- CHALLENGE ----------

MODIFY THE PROGRAM SO THAT IT WILL TERMINATE NORMALLY BY ITSELF IF IT RECEIVES A MESSAGE WHICH IS LESS THAN TWO WORDS (4 BYTES) IN LENGTH.

(WE CHOSE TWO WORDS BECAUSE THE CLI WON'T LET US SEND A ZERO LENGTH MESSAGE.)

```
#nolist                              /* CONTROL.IPC.C */
#include <packets:create.h>
#include <packets:ipc.h>
#list

#define bufsz 136                    /* buffer size in bytes */
#define bufflen bufsz/2              /* buffer length in words */

main( )
{
int ac0,ac1,ac2,error;
char *bytptr, buff[bufsz];           /* byte pointers */

P_?????????? crepk;                  /* ipc creation packet */
P_???? ipcpk;                        /* ipc header */

/* packet for creation of ipc file */

  crepk.cftyp_format = 0;                    /* reserved */
  crepk.cftyp_entry = ?????;         /* file is type ipc */
  crepk.cpor = ??;                   /* local ipc port number */
  crepk.ctim = ??;                   /* default time */
  crepk.cacp = ??;                   /* default access controls */
```

```
/* CREATE IPC FILE FOR OTHER PROCESS TO LOOK UP */

  bytptr = "TRYIPC";
  ac0 = bytptr;                    /* byte pointer to ipc file */
  ac1 = 0;                         /* reserved */
  ac2 = &crepk;                    /* addr of creation packet */
  if (error = sys($CREATE,&ac0,&ac1,&ac2)) errxt(error);

/* WAIT FOR A MESSAGE FROM ANY SENDER */

  ipcpk.isfl = ?;                  /* system flags - none */
  ipcpk.???? = 0;                  /* user flags from sender */
  ipcpk.ioph = ?;                  /* global origin - any sender */
  ipcpk.idpn = ??;       /* local dest port same as ipc file */
  ipcpk.ilth = ???????;            /* buffer length (words) */
  ipcpk.iptr = ?????;              /* buffer word address */
  ac0 = 0;
  ac1 = 0;
  ac2 = ??????;
  if (error = sys(?????,&ac0,&ac1,&ac2)) errxt(error);

/* DISPLAY THE MESSAGE */

  buff[ipcpk.ilth*2] = '\000';/* store null byte at end of msg */
  printf("%s\n",buff);

/* GET THE SENDER'S PID */

  ??? = 0;
  ??? = ipcpk.????;                /* sender's global origin port */
  ??? = 0;
  if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);

/* ACKNOWLEDGE RECEIPT OF THE MESSAGE TO SENDER'S CONSOLE */

  ??? = ???;                       /* sender's pid from above */
  bytptr = "I GOT THE MESSAGE\n";
  ??? = bytptr;
  ??? = strlen(bytptr);
  if (error = sys(?????,&ac0,&ac1,&ac2)) errxt(error);

}                                  /* end main */
```

C LANGUAGE TALK IPC LAB

USING THE FOLLOWING CLI COMMANDS, GET THE NEEDED FILES INTO YOUR DIRECTORY:

```
DIRECTORY, :S309VS:C
MOVE/R/NACL, :UDD:YOURUSERNAME, TALK<ONE,TWO>.C, ERROR.OB
DIRECTORY/I
```

EXAMINE THE PRINTOUTS OF TALKONE.C AND TALKTWO.C ON THE FOLLOWING PAGES AND FILL IN THE ITEMS WHICH HAVE BEEN REPLACED WITH QUESTION MARKS. YOU WILL HAVE TO SUPPLY VARIOUS SYSTEM CALLS AND THEIR ARGUMENTS, ERROR CODE SYMBOLS, ADDRESSES OF VARIOUS ITEMS IN PACKETS, AND THE APPROPRIATE VALUES IN THOSE PACKETS.

WHEN YOU HAVE DETERMINED THE CORRECT INFORMATION FOR EACH OF THESE ITEMS, EDIT YOUR TWO SOURCE FILES AND MAKE THE CORRECTIONS.

COMPILE EACH SOURCE FILE USING THE /LINEID SWITCH AND LINK IT WITH THE ERROR ROUTINE.

LOG ON AT ANOTHER CONSOLE. EXECUTE TALKONE AT ONE CONSOLE AND TALKTWO AT THE SECOND. YOU SHOULD BE ABLE TO SEND MESSAGES BACK AND FORTH.

TO STOP EACH OF THESE PROGRAMS, ENTER CONTROL-D (END-OF-FILE).

********** CHALLENGE **********

WORKING WITH ANOTHER STUDENT USE THE PROGRAMS TO CONVERSE.

HINT:

WHEN YOU EXECUTED THE PROGRAMS BY YOURSELF, WHAT WAS IT THAT ENSURED YOU ONLY GOT YOUR OWN MESSAGES EVEN THOUGH OTHERS MAY HAVE BEEN EXECUTING THE SAME PROGRAMS ????

****

********** CHALLENGE **********

PREPARE A FILE FULL OF MESSAGES WHICH COMPRISE ONE HALF OF A CONVERSATION.

CAUSE TALKONE TO EXECUTE, SENDING THE MESSAGES FROM THIS FILE, AND STORING ANY MESSAGES IT MAY RECEIVE IN AN EMPTY FILE OF YOUR CHOICE.

EXECUTE TALKTWO AT YOUR CONSOLE, AND RESPOND TO THE MESSAGES COMING FROM YOUR TALKONE.

==========>> YOU MAY USE ONLY ONE CONSOLE !! <<==========

AFTER YOU HAVE RESPONDED TO THE LAST MESSAGE IN YOUR FILE, YOUR TALKTWO PROCESS SHOULD TERMINATE AUTOMATICALLY.
EXAMINE THE FILE IN WHICH TALKONE STORED THE MESSAGES IT RECEIVED.

****

PREPARE A SECOND FILE WITH THE RESPONSES TO THE MESSAGES IN THE FIRST FILE. CAUSE BOTH TALKONE AND TALKTWO TO EXECUTE, TAKING INPUT FROM THESE TWO MESSAGE FILES, AND STORING ANY RECEIVED MESSAGES IN TWO NEW EMPTY FILES OF YOUR CHOICE.

==========>> YOU MAY NOT USE ANY CONSOLE !! <<==========
(EXCEPT TO START AND STOP THE PROCESSES)

****

```
#nolist                         /* TALKONE.C */
#include <packets:ipc.h>
#include <packets:normal_io.h>
#list

#define bufsz 80                /* buffer size in bytes */
#define bufflen bufsz/2             /* buffer length in words */

main()
{
int ac0,ac1,ac2,error,i;
char *bytptr, buff[bufsz];          /* byte pointers */

P_????? ipcpk;          /* ipc header */
P_???   inpkt;          /* packet for record I/O   */

  /* I/O packet for file @INPUT */

inpkt.ich  = 0;                  /* channel # returned by sys */
inpkt.isti = $ICRF|$RTDS|$OFIO; /* change to datasens in/out */
inpkt.isto = 0;                  /* file type (returned)      */
inpkt.imrs = -1;                 /* default physical blk size */
inpkt.ibad = buff;               /* buffer bytepointer        */
inpkt.ires = 0;                  /* reserved                  */
inpkt.ircl = bufsz;              /* max record size           */
inpkt.irlr = 0;                  /* record length returned    */
inpkt.irnw = 0;                  /* reserved                  */
inpkt.irnh = 0;                  /* record number             */
inpkt.ifnp = "@INPUT";           /* B.P. to filename          */
inpkt.idel = -1;                 /* default delimiters        */

/* OPEN INPUT FILE */

  ac0 = 0;
  ac1 = 0;
  ac2 = &inpkt;
  if (sys($OPEN, &ac0, &ac1, &ac2)) errxt(error);

/* LOOK UP TALKTWO'S IPC FILE */

  do
    {
    ??? = "FILEB";
    ??? = 0;
    ??? = 0;
    error = sys(??????,&ac0,&ac1,&ac2);
    }
  while (error == ?????);  /* if it doesn't exist, try again */
  if (error) errxt(error);
  ipcpk.???? = ???;         /* global destination */

/* DISPLAY INITIAL PROMPT */

  printf("TYPE A MESSAGE AND WAIT FOR A RESPONSE\n");
```

```
  do
    {

/* READ A LINE FROM KEYBOARD */

    ac0 = 0;
    ac1 = 0;
    ac2 = &inpkt;
    error = sys($READ,&ac0,&ac1,&ac2);
    if (error&&(error != EREOF)) errxt(error);
    if (error == EREOF)
      i = 0;
    else
      i = inpkt.irlr;       /* get its length */
    if (i%2) i += 1;        /* if it's odd make it even */
    i /=2;          /* make it a word length */

/* SEND IT AS AN IPC MESSAGE */

    ipcpk.isfl = ?;         /* system flags - none */
    ipcpk.iufl = ?;         /* no user flags */
    ipcpk.iopn = ??;        /* any local origin port */
    ipcpk.???? = i;         /* buffer length (words) */
    ipcpk.iptr = ?????;          /* buffer word address */
    ac0 = 0;
    ac1 = 0;
    ac2 = ??????;
    if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);

    if (ipcpk.????)         /* if msg sent was non zero length */
      {
/* WAIT FOR AN IPC MESSAGE */

      ipcpk.???? = ???????;         /* buffer length (words) */
      ipcpk.???? = ?????;           /* buffer word address */
      ac0 = 0;
      ac1 = 0;
      ac2 = ??????;
      if (error = sys(?????,&ac0,&ac1,&ac2)) errxt(error);

/* DISPLAY THE MESSAGE */

      if (ipcpk.????)       /* if msg rcvd was non-zero length */
        {
        ac0 = 0;
        ac1 = 0;
        ac2 = &inpkt;
        if (error = sys($WRITE,&ac0,&ac1,&ac2)) errxt(error);
        }
      }
    }
  while (ipcpk.ilth);

}                   /* end main */
```

```
#nolist                    /* TALKTWO.C */
#include <packets:create.h>
#include <packets:ipc.h>
#include <packets:normal_io.h>
#list

#define bufsz 80           /* buffer size in bytes */
#define bufflen bufsz/2          /* buffer length in words */

main( )
{
int ac0,ac1,ac2,error,i;
char *bytptr, buff[bufsz];       /* byte pointers */

P_?????????? crepk;        /* ipc creation packet */
P_???? ipcpk;              /* ipc header */
P_???   inpkt;             /* packet for record I/O   */

  /* I/O packet for file @INPUT */

inpkt.ich  = 0;                    /* channel # returned by sys */
inpkt.isti = $ICRF|$RTDS|$OFIO;    /* change to datasens in/out */
inpkt.isto = 0;                    /* file type (returned)      */
inpkt.imrs = -1;                   /* default physical blk size */
inpkt.ibad = buff;                 /* buffer bytepointer        */
inpkt.ires = 0;                    /* reserved                  */
inpkt.ircl = bufsz;                /* max record size           */
inpkt.irlr = 0;                    /* record length returned    */
inpkt.irnw = 0;                    /* reserved                  */
inpkt.irnh = 0;                    /* record number             */
inpkt.ifnp = "@INPUT";             /* B.P. to filename          */
inpkt.idel = -1;                   /* default delimiters        */

/* OPEN INPUT FILE */

  ac0 = 0;
  ac1 = 0;
  ac2 = &inpkt;
  if (sys($OPEN, &ac0, &ac1, &ac2)) errxt(error);
```

```
/* packet for creation of ipc file */

  crepk.cftyp_format = 0;         /* reserved */
  crepk.cftyp_entry = ?????;     /* file is type ipc */
  crepk.cpor = ??;         /* local ipc port number */
  crepk.ctim = ??;         /* default time */
  crepk.cacp = ??;         /* default access controls */

/* CREATE IPC FILE FOR OTHER PROCESS TO LOOK UP */

  ??? = "FILEB";           /* byte pointer to ipc file */
  ??? = 0;                 /* reserved */
  ??? = ??????;            /* addr of creation packet */
  if (error = sys(???????,&ac0,&ac1,&ac2)) errxt(error);

/* IPC HEADER */

  ipcpk.???? = ?;          /* system flags - none */
  ipcpk.???? = ?;          /* no user flags */
  ipcpk.???? = ?;          /* global origin - any sender */
  ipcpk.???? = ??;         /* local dest port same as ipc file */
  ipcpk.???? = ???????;          /* buffer length (words) */
  ipcpk.???? = ?????;            /* buffer word address */

/* DISPLAY INITIAL PROMPT */

  printf("WAIT FOR A MESSAGE AND TYPE A RESPONSE\n");
```

```
  do
    {
/* WAIT FOR A MESSAGE */

    ipcpk.???? = ???????;         /* buffer length (words) */
    ipcpk.???? = ?????;          /* buffer word address */
    ac0 = 0;
    ac1 = 0;
    ac2 = ??????;
    if (error = sys(?????,&ac0,&ac1,&ac2)) errxt(error);

/* DISPLAY THE MESSAGE */

    if (ipcpk.????)       /* if msg rcvd was non-zero length */
      {
      ac0 = 0;
      ac1 = 0;
      ac2 = &inpkt;
      if (error = sys($WRITE,&ac0,&ac1,&ac2)) errxt(error);

/* READ A LINE FROM KEYBOARD */

      ac0 = 0;
      ac1 = 0;
      ac2 = &inpkt;
      error = sys($READ,&ac0,&ac1,&ac2);
      if (error&&(error != EREOF)) errxt(error);
      if (error == EREOF)
        i = 0;
      else
        i = inpkt.irlr;           /* get its length */
      if (i%2) i += 1;            /* if it's odd make it even */
      i /=2;              /* make it a word length */

/* SEND IT AS AN IPC MESSAGE */

      ipcpk.???? = i;             /* buffer length (words) */
      ipcpk.???? = ?????;         /* buffer word address */
      ac0 = 0;
      ac1 = 0;
      ac2 = ??????;
      if (error = sys(??????,&ac0,&ac1,&ac2)) errxt(error);
      }
    }
  while (ipcpk.ilth);

}                         /* end main */
```

Data General

Data General Corporation, 4400 Computer Drive, Westboro, MA 01580

(617) 366-8911