```
   ty terspy.mar
         .TITLE   TERSPY Terminal Spy Program
         .IDENT   /01/

; Modified by D. Martin, University of Western Ontario
; from
; University of Guelph VAX 11/780

; Terminal Session Logger

; Author: Bob Vera, Digital Equipment Corporation          April 28, 1980

; This system is designed to kidnap characters out of a
; designated terminal's output buffer and record them in a disk file
; which is subsequently printed out on the printer when the session
; is finished. This is useful for producing a hard copy of a
; student's video terminal session of a final run of his
; program or project which may subsequently be handed in with
; his/her assignment.

; Currently the program creates a file with the filename equivalent
; to the user's name and a filetype of ".LOG". The file is sent
; to the print queue "SYS$PRINT" with a request for a flag page
; and deletion of the file once it is printed.

         .PAGE

; Define Unit Control Block Offsets

         $UCBDEF

; Define the Device Data Block Offsets

         $DDBDEF

; Define the terminal UCB offsets

;        $TTYUCBDEF
         $TTYDEF

; Define the IRP offsets

         $IRPDEF

; Define the offsets for the quota list

         $PQLDEF

; Define the offsets for the symbiont manager message

         $SMRDEF

; Define the offsets for the JPI system service

         $JPIDEF

         .PAGE
         .PSECT   DATA,QUAD

; Local Data Structures and definitions

; Define RMS Blocks
; wish to have an entry made then remove the ";" from the FOP
; option to make it a temporary file.

FABBLK: $FAB    FAC = <PUT,BIO>,-                     ;Enable block I/O
;               FOP = TMP,-                           ;Temporary file
                FNM = <SYS$OUTPUT>,-                  ;ASCII filename string
                ORG = SEQ,-                           ;Sequential
                MRS = 128,-                           ;Maximum record size
                RFM = FIX                             ;Fixed length records

RABBLK: $RAB    FAB = FABBLK,-                        ;Associated FAB
                RAC = SEQ,-                           ;Sequential access
                RSZ = 2048,-                          ;Record buffer size
                RBF = RECBUF                          ;Record buffer address

; Define hex equivalents of some ASCII characters

        ESC     =^X1B                                 ;ASCII for Escape
        CR      =^X0D                                 ;ASCII for Return
        LF      =^X0A                                 ;ASCII for Line Feed
        FF      =^X0C                                 ;ASCII for Form Feed
        BS      =^X08                                 ;ASCII for Back Space
        BLNK    =^X20                                 ;ASCII for Space
        ARROW   =^X5E                                 ;ASCII for "^"
        CTRLZ   =^X1A                                 ;ASCII for "<CTRL>Z"
        ZEE     =^X5A                                 ;ASCII for "Z"

; Storage definitions

CHARTIM:        .ASCID  /0 00:00:00.05/               ;ASCII delta time of 1/100 sec
QUADTIM:        .QUAD   0                             ;Binary equivalent stored here

TTNAME:         .ASCID  /SPY_TERMINAL/                ;Logical name of process term
TRANSDESC:                                            ;Translation of above put here
TRANSLEN:       .LONG   63                            ;Length of translated string
TRANSADDR:      .LONG   TRANSNAME                     ;Addr of translated string
TRANSNAME:      .BLKB   63                            ;Translated string

DEVI_T_UNIT:    .WORD   0                             ;Terminal unit number here
DEVI_T_DEVNAM:  .BLKB   100                           ;Terminal device name (TTx)

TT_UCB:         .LONG   0                             ;Terminal's UCB address
TT_WB_NXT:      .LONG   80000000                      ;Address of next char
TT_IRP:         .LONG   0                             ;Address of current IRP
TT_WB_END:      .LONG   0                             ;Address of buffer end

SKIP_CNT:       .WORD   0                             ;Idle search counter
LAST_CHAR:      .LONG   -1                            ;Last character typed
KRNL_CNT:       .LONG   0                             ;Counter for chars transferred
KRNL_PNT:       .LONG   0                             ;Current position in RECBUF
REC_CNT:        .LONG   0                             ;# of chars in RECBUF
                .LONG   0
RECBUF:         .BLKB   2048                          ;I/O Buffer for RMS
OVERFLOW:       .BLKB   200                           ;Buffer overflow area

; Item List for $GETJPI system service

JOB_INFO:       .WORD   12
                .WORD   JPI$_USERNAME                 ;Username
                .LONG   USER
                .LONG   USER_LEN
                .WORD   15
                .WORD   JPI$_PRCNAM                   ;Process name
```

```
USER_LEN:         .LONG   0
USER:             .BLKB   12

PROCESS_LEN:      .LONG   0
                  .LONG   PROCESS
PROCESS:          .BLKB   15

MBXCHAN:          .LONG   0                         ;Mailbox channel

EXITBLK:
          .LONG   0                                 ;Linkage word
          .LONG   MBXAST                            ;Handler address
          .LONG   1                                 ;One argument
          .LONG   REASON                            ;Reason for exit

REASON:   .LONG   0
          .PAGE
          .PSECT  MAINCODE,QUAD

START::   .WORD   0                                 ;And away we go . . .

          $GETJPI_S ITMLST = JOB_INFO               ;Get process and user names

          $CREATE FAB = FABBLK                      ;Create a file with RMS
          BLBC    R0,20$                            ;Error condition
          $CONNECT RAB = RABBLK                     ;Connect an I/O stream to it
          BLBS    R0,10$                            ;Error condition
30$:      PUSHL   R0                                ;Record handling error return
          $CLOSE  FAB = FABBLK
          POPL    R0
20$:      RET                                       ;File handling error return

10$:      $DCLEXH_S DESBLK = EXITBLK                ;Set up exit handler
          $SETPRI_S PRI=#8                          ;Bump up our priority

102$:     $BINTIM_S TIMBUF=CHARTIM,-                ;Get 64 bit equivalent of a
                    TIMADR=QUADTIM                  ;delta time of 1/100 second

103$:     $TRNLOG_S LOGNAM=TTNAME,-                 ;Find out which terminal we
                    RSLLEN=TRANSLEN,-               ;are on. This is the one we
                    RSLBUF=TRANSDESC                ;will monitor
          CMPL    #SS$_NORMAL,R0
          BNEQ    30$

; Here we shall get the terminal we are running on in the form _TTax:
; This will then be used by the routine GET_TT_UCB which will scan the
; I/O data base for this terminal and return the address of it´s
; UCB (Unit Control Block).

104$:     CMPB    TRANSNAME,#^X1B                   ;Is the first char an <ESC>
          BNEQ    1$                                ;No, skip the rest
          SUBL    #4,TRANSLEN                       ;Subtract 4 from the length
          ADDL    #4,TRANSADDR                      ;Move pointer up 4 bytes
1$:       MOVL    TRANSADDR,R0                      ;Point to the terminal name
          ADDL    #4,R0                             ;Pick up the unit number char
          MOVZBL  (R0),R0                           ;Get the ASCII byte
          SUBL    #^X30,R0                          ;Convert the ASCII
          MOVZBW  R0,DEVI_T_UNIT                    ;Store the unit number
          ADDL    #1,TRANSADDR                      ;Point to proper device name
          $CMKRNL_S GET_TT_UCB                      ;Get the UCB addr for the term
          TSTL    TT_UCB
          BNEQ    107$                              ;Exit if not found
          BRW     30$
```

```
; and stealing the characters that they are displaying on the terminal.
; We have a group of buffers for this purpose and the buffer currently
; being used is pointed to by the variable "KRNL_PNT"

107$:   MOVAL   RECBUF+1,KRNL_PNT                     ;Initialize the pointer
        MOVL    #1,REC_CNT
        CLRB    RECBUF

; All set! Scan the IRP's and if we find characters, send them to the
; subprocess. Else try another scan. If we scan 3 times and find nothing
; then go to sleep for 5/100 of a second and start over.

LOOP:   MOVW    #-3,SKIP_CNT                          ;Set up the idle scan counter
REPEAT: $CMKRNL_S GET_TT_NXT                          ;Scan the IRP's
108$:   TSTL    KRNL_CNT                              ;Anything returned?
        BGTR    10$                                   ;Yes, then send them on
        ADDW2   #1,SKIP_CNT                           ;No, count this idle scan
        BLSS    REPEAT                                ;If 3 of them then . . .
        $SCHDWK_S DAYTIM=QUADTIM                      ; . . . go to SLEEP
109$:   $HIBER_S
        BRB     LOOP                                  ;and try again

; This next trick is to make sure we don't overprint our screen. Sometimes
; we miss a form feed. This next section checks to see in the last character
; typed was a carriage return. If so then we must type a line feed if there is
; not one in the first two characters of our new line

10$:    CMPB    LAST_CHAR,#CR                         ;Did we last type a <CR>?
        BNEQ    20$                                   ;No, skip the whole mess
        MOVL    KRNL_PNT,R0                           ;Look at the first new char
        CMPB    (R0)+,#LF                             ;Is it a <LF>?
        BEQL    20$                                   ;Yes, then don't worry
        CMPL    KRNL_CNT,#1                           ;No, Is there a second char
        BLEQ    30$                                   ;No, can't check it then
        CMPB    (R0),#LF                              ;Yes, Is it a <LF>?
        BEQL    20$                                   ;Yes, thank God
30$:    MOVB    #LF,-2(R0)                            ;No, then insert a <LF>
20$:    ADDL2   KRNL_CNT,REC_CNT                      ;Record the byte count
        CMPL    REC_CNT,#2048                         ;Buffer full??
        BLSS    40$                                   ;No, continue scanning
        $WRITE  RAB = RABBLK                          ;Yes, write it out
70$:    MOVAL   OVERFLOW,R1                           ;Transfer overflow to
        MOVAL   RECBUF+1,R6                           ;beginning of buffer
        MOVL    #1,REC_CNT                            ;Re-initalize buffer count
        MOVL    KRNL_PNT,R0
        ADDL2   KRNL_CNT,R0
        SUBL2   #OVERFLOW,R0                          ;Check if anything in the
        BLEQ    50$                                   ;overflow buffer
60$:    MOVB    (R1)+,(R6)+                           ;Yes, do the move
        INCL    REC_CNT
        SOBGTR  R0,60$
        CLRL    R0
        MOVL    #200,R1
        MOVC5   R0,OVERFLOW,R0,R1,OVERFLOW            ;Clear the overflow buffer
50$:    MOVL    R6,KRNL_PNT                           ;Re-initialize the pointer
        CLRL    KRNL_CNT
40$:    MOVL    KRNL_PNT,R0                           ;Store the last character
        ADDL2   KRNL_CNT,R0                           ;that was sent
        MOVB    -1(R0),LAST_CHAR
        CLRB    (R0)+
        INCL    REC_CNT
        MOVL    R0,KRNL_PNT
        BRW     LOOP                                  ;And Repeat

        .PAGE
```

```
; then this routine reads the mailbox to see how many copies are desired
; and then flushes the record buffer, closes the file, and sends
; a message to the print symbiont manager to print it.

; If zero copies are asked for then the file is simply deleted

; This is now an exit handler to bow out gracefully. /MDM

MBXAST::
        .WORD   0

        $SETPRI_S PRI=#4                                ;Lower our priority again

        $CLOSE  FAB = FABBLK                            ;Close the log file
        $EXIT_S

        .PAGE

; This Kernel Mode routine scans the I/O data base and returns the address
; of the UCB (Unit Control Block) for the terminal whose name is pointed
; to by "TRANSADDR" and whose unit number is in "DEVI_T_UNIT"

GET_TT_UCB::
        .WORD   ^M<R2,R3,R4,R5,R6,R10,R11>              ;Entry point
        MOVL    @#SCH$GL_CURPCB,R4                      ;Lock the I/O Data Base
        JSB     SCH$IOLOCKR
        MOVL    TRANSADDR,R6                            ;Point to the terminal name
        CLRL    TT_UCB                                  ;Clear the return information

; First scan the DDB's (Device Data Blocks) for devices of the type TTx
; (where x is a controller letter).

        MOVAL   L^IOC$GL_DEVLIST-DDB$L_LINK,R11 ;Get addr of addr of first DD
10$:    MOVL    DDB$L_LINK(R11),R11                     ;Get next DDB
        BNEQU   12$                                     ;Is there another?
        CLRL    R10                                     ;Signal UCB addr not found
        BRW     80$                                     ;All gone!
12$:    MOVZBL  DDB$T_NAME(R11),R0                      ;Get length of device name
        INCL    R0
        MOVC3   R0,DDB$T_NAME(R11),W^DEVI_T_DEVNAM ;Copy device name
        CMPB    (R6),DEVI_T_DEVNAM+1                    ;Check for TT device name
        BNEQ    10$                                     ;No, get next DDB
        CMPB    1(R6),DEVI_T_DEVNAM+2                   ;Check second char
        BNEQ    10$                                     ;No
        CMPB    2(R6),DEVI_T_DEVNAM+3                   ;Check the controller letter
        BNEQ    10$

; Now scan through the UCB's for this device looking for the unit
; we desire.

        MOVAL   DDB$L_UCB-UCB$L_LINK(R11),R10           ;Get addr of addr of UCB
50$:    MOVL    UCB$L_LINK(R10),R10                     ;Get next UCB addr
        BEQLU   60$                                     ;Nothing there
        CMPW    UCB$W_UNIT(R10),DEVI_T_UNIT             ;Check the unit number
        BNEQ    50$

; Got it!!

        MOVL    R10,TT_UCB                              ;Store the UCB address
        BRB     80$                                     ;And return home
60$:    BRW     10$                                     ;Continue to next DDB
80$:    MOVL    @#SCH$GL_CURPCB,R4                      ;Unlock the I/O Data Base
        JSB     SCH$IOUNLOCK
        RET
```

```
; This Kernel Mode routine is the work-horse of the system. This code
; scans the IRP's coming through for this device and decides if it
; is a read or write request. If it is a write request then the entire
; buffer is transferred in a shot. If it is a read request then we work on
; a character by character basis so that the we can emulate the typing
; on the characters on the screen

; One may notice that the code for locking the I/O data base has been
; commented out. This is intended to speed things up for the system
; when a large number of people are using the program. Since the
; code below only reads the data base the system should suffer no ill
; effects. However, if one wishes to feel more secure about the
; situation then simply remove the comment characters (";") in the
; appropriate places. (There are six lines where this must be done).

GET_TT_NXT::
        .WORD   ^M<R2,R3,R4,R5,R6>                      ;Entry point
;       MOVL    @#SCH$GL_CURPCB,R4                      ;Lock the I/O Data Base
;       JSB     SCH$IOLOCKR
        CLRL    R2                                      ;Clear the char count
        MOVL    TT_UCB,R5                               ;Get the UCB address
        EXTV    #UCB$V_BSY,#1,UCB$W_STS(R5),R0          ;Is the unit currently busy?
        BNEQ    300$                                    ;Yes, then check the IRP's
        BRW     1$                                      ;No, then simply return

300$:   MOVL    UCB$L_SVAPTE(R5),R4                     ;Get buffer address
        MOVL    TTY$L_WB_NEXT(R4),R3                    ;Get addr of the next char
        MOVL    TTY$L_WB_END(R4),TT_WB_END              ;Get addr of the last char
        MOVL    UCB$L_IRP(R5),R6                        ;Get addr of the I/O Packet
        MOVZWL  IRP$W_FUNC(R6),R5                       ;Pick up the type of request
        BEQL    7$                                      ;Read if type = 0
        BRW     99$                                     ;Else it is a write

; The next section of code handles read requests. This includes read with
; prompts which require some fancy backtracking

7$:     CMPL    R6,TT_IRP                               ;Is this a new packet?
        BEQL    5$                                      ;No, then pick from before
        MOVL    #80000000,TT_WB_NXT                     ;Yes, reset previous pointer
        DECL    R3                                      ;Decrement our current pointe
        BRB     6$
5$:     SUBL3   R3,TT_WB_END,R4                         ;How many characters added
        CMPL    R4,#-1
        BNEQ    6$
        BRW     10$                                     ;None, just return
6$:     CMPL    R3,#^X200                               ;Check for illegal addr
        BGTRU   2$                                      ;Yes, ignore it and return
        BRW     1$
2$:     SUBL3   TT_WB_NXT,R3,R4                         ;Compute diff from last time
        BGTR    11$                                     ;If positive then ok
        CMPL    R4,#-1                                  ;If -1 a <DEL> char was typed
        BEQL    12$                                     ;And we must simulate it
        BRW     10$                                     ;Else no diff, Return
12$:    MOVL    KRNL_PNT,R5                             ;Simulate <DEL> char
        DECL    REC_CNT                                 ;Reduce the character count
13$:    TSTB    -(R5)                                   ;If last char was Null then
        BEQL    13$                                     ;Find one that wasn't
        CLRB    (R5)                                    ;Delete it
        MOVL    R5,KRNL_PNT                             ;Update the buffer pointer
        BRW     10$                                     ;And return
11$:    CMPL    R4,#20                                  ;Diff <= 20 then move chars
        BLEQ    20$                                     ;as is
        MOVL    R3,R4                                   ;Else initiate backward searc
        BGEQ    40$
        CMPB    (R4),#LF                        ;Like line feeds . . .
        BEQL    45$
        CMPB    (R4),#CR                        ; . . carriase returns . . .
        BEQL    45$
        CMPB    (R4),#FF                        ; . . or form feeds.
        BEQL    45$
        INCL    R4                              ;Isnore any others
45$:    SUBL3   R4,R3,R5
        CMPL    R5,#40
        BGTR    10$
        BRB     55$
20$:    MOVL    TT_WB_NXT,R4                    ;Point to where we left off
55$:    MOVL    KRNL_PNT,R5                     ;Point to where we are soins
30$:    MOVB    (R4)+,(R5)                      ;Move a char
        INCL    R2                              ;Count it
        CMPB    (R5),#CTRLZ                     ;If it was a <CTRL>Z then
        BNEQ    80$                             ;Emulate it with "^Z"
        MOVB    #ARROW,(R5)+
        MOVB    #ZEE,(R5)+
        MOVB    #CR,(R5)+
        ADDL    #2,R2
        BRB     90$
80$:    CMPB    (R5)+,#LF                       ;If it is a line feed make
        BNEQ    90$                             ;sure it has a matchins
        CMPB    -2(R5),#LF                      ;carriase return
        BNEQ    100$
        TSTB    -(R5)
        DECL    R2
        BRB     90$
100$:   CMPB    -2(R5),#CR
        BEQL    90$
        MOVB    -1(R5),(R5)+
        MOVB    #CR,-2(R5)
        INCL    R2
90$:    CMPL    R4,R3                           ;Are we at the end?
        BLSS    30$                             ;No, continue
        CLRB    (R5)+
10$:    MOVL    R3,TT_WB_NXT                    ;Store the new pointer
1$:     MOVL    R2,KRNL_CNT                     ;Store the char counter
        MOVL    R6,TT_IRP                       ;Store the IRP address
;       MOVL    @#SCH$GL_CURPCB,R4              ;Unlock the I/O Data Base
;       JSB     SCH$IOUNLOCK
        RET                                     ;And return

; Here we handle the write requests. We take the address of the end
; of the buffer, subtract the number of bytes transferred, and use
; that as the startins address

99$:    CLRL    KRNL_CNT
        CMPL    R6,TT_IRP                       ;New packet?
        BEQL    95$                             ;No, Then no action!
        MOVZWL  IRP$W_BCNT(R6),R5               ;Yes, set the byte count
        CMPL    TT_WB_END,#80000000             ;Buffer in system space?
        BLSSU   95$                             ;No, then isnore it
        MOVL    TT_WB_END,R2                    ;Compute address of buffer
        SUBL2   R5,R2                           ;besinnins and transfer
        MOVL    KRNL_PNT,R3                     ;entire buffer.
        MOVL    R5,KRNL_CNT
210$:   MOVB    (R2)+,(R3)+
        SOBGTR  R5,210$

95$:    MOVL    R6,TT_IRP
;       MOVL    @#SCH$GL_CURPCB,R4              ;Unlock the I/O Data Base
```

```
        MOVB    #CR,-2(R5)
        INCL    R2
90$:    CMPL    R4,R3
        BLSS    30$                             ;Are we at the end?
        CLRB    (R5)+                           ;No, continue
10$:    MOVL    R3,TT_WB_NXT
1$:     MOVL    R2,KRNL_CNT                     ;Store the new pointer
        MOVL    R6,TT_IRP                       ;Store the char counter
;       MOVL    @#SCH$GL_CURPCB,R4              ;Store the IRP address
;       JSB     SCH$IOUNLOCK                    ;Unlock the I/O Data Base
        RET
                                                ;And return
; Here we handle the write requests. We take the address of the end
; of the buffer, subtract the number of bytes transferred, and use
; that as the starting address

99$:    CLRL    KRNL_CNT
        CMPL    R6,TT_IRP
        BEQL    95$                             ;New packet?
        MOVZWL  IRP$W_BCNT(R6),R5               ;No, Then no action!
        CMPL    TT_WB_END,#80000000             ;Yes, set the byte count
        BLSSU   95$                             ;Buffer in system space?
        MOVL    TT_WB_END,R2                    ;No, then ignore it
        SUBL2   R5,R2                           ;Compute address of buffer
        MOVL    KRNL_PNT,R3                     ;beginning and transfer
        MOVL    R5,KRNL_CNT                     ;entire buffer.
210$:   MOVB    (R2)+,(R3)+
        SOBGTR  R5,210$

95$:    MOVL    R6,TT_IRP
;       MOVL    @#SCH$GL_CURPCB,R4
        JSB     SCH$IOUNLOCK                    ;Unlock the I/O Data Base
        RET
        .END    START                           ;And return
$
```