NASA
Technical
Memorandum

NASA TM-86535

A PROCESS ACTIVITY MONITOR FOR AOS/VS

By R. A. McKosky, S. W. Lindley, and J. S. Chapman

Management Systems Office
Shuttle Projects Office

January 1986

(NASA-TM-86535) A PROCESS ACTIVITY MONITOR FOR AOS/VS (NASA) 30 p HC A03/MF A01 CSCL 09B

N86-19950

Unclas
G3/60 05511

National Aeronautics and
Space Administration

George C. Marshall Space Flight Center

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. NASA TM - 86535	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
4. TITLE AND SUBTITLE A Process Activity Monitor for AOS/VS		5 REPORT DATE January 1986 6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) R. A. McKosky,* S. W. Lindley,* and J. S. Chapman		8. PERFORMING ORGANIZATION REPORT #
9. PERFORMING ORGANIZATION NAME AND ADDRESS George C. Marshall Space Flight Center Marshall Space Flight Center, Alabama 35812		10. WORK UNIT NO. 11. CONTRACT OR GRANT NO.
12 SPONSORING AGENCY NAME AND ADDRESS National Aeronautics and Space Administration Washington, D.C. 20546		13. TYPE OF REPORT & PERIOD COVERED Technical Memorandum 14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

Prepared by Management Systems Office, MSFC Shuttle Projects Office and
*Rockwell International

16. ABSTRACT

With the ever increasing concern for computer security, users of computer systems are becoming more sensitive to unauthorized access. One of the initial security concerns for the Shuttle Management Information System was the problem of users leaving their workstations unattended while still connected to the system. This common habit was a concern for two reasons: it ties up resources unnecessarily and it opens the way for unauthorized access to the system. The Data General MV/10000 does not come equipped with an automatic time-out option on interactive peripherals. The purpose of this memorandum is to describe a system which monitors process activity on the system and disconnects those users who show no activity for some time quantum.

17. KEY WORDS Elapsed Time, CPU. Process Activity, PID, Threshold, Process Termination	18. DISTRIBUTION STATEMENT Unclassified — Unlimited

19 SECURITY CLASSIF. (of this report) Unclassified	20. SECURITY CLASSIF. (of this page) Unclassified	21. NO. OF PAGES 29	22. PRICE NTIS

For sale by National Technical Information Service, Springfield, Virginia 22151

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

TECHNICAL MEMORANDUM

A PROCESS ACTIVITY MONITOR FOR AOS/VS

I. INTRODUCTION

Managers of computer systems are becoming increasingly aware of the necessity to guard against unauthorized access. A primary security concern for any system is an active terminal left unattended by the user. Some systems are equipped with an automatic time-out option. The Data General MV/10000, however, is not. When selecting the MV/10000 to drive the Shuttle Management Information System (SMIS) for the Marshall Space Flight Center's Shuttle Projects Office, analysts and managers agreed that a time-out feature would need to be incorporated into the system. Such a feature would decrease the chance of unauthorized access to the system and free limited system resources. It is the purpose of this memorandum to describe the process activity monitor and process terminator tasks developed for SMIS, by which users registering no CPU activity for some time quantum are disconnected from the system.

II. SYSTEM DESCRIPTION

The Data General ECLIPSE MV/10000 runs under Advanced Operating System/Virtual Storage (AOS/VS), a multitasking, multiprogramming, demand-paged, virtual storage operating system. It can support users on a time-sharing basis, run batch jobs, or perform control applications on a real-time basis. The user communicates with AOS/VS from the console via Command Line Interpreter (CLI) commands. AOS/VS is unique to 32-bit ECLIPSE MV computers, and has the capacity to run up to 256 processes at a time.

The MV/10000 process tree begins with AOS/VS, designated as Process Identification number (PID) 0. AOS/VS assigns a PID to each other process. AOS/VS has two sons, PMGR, PID 1, and OP, PID 2. PMGR is the peripheral manager. OP is the "master process" because it is operable only from the master console.

EXEC, a son of OP, runs as PID 3. All user processes and the printer queues are sons of EXEC. Most system processes are sons of OP. System processes include peripheral controllers, data base management systems, communication packages and Comprehensive Electronic Office (CEO) office automation software. Figure 1 shows a typical process tree for PID 2 (OP).

III. PROBLEM OVERVIEW

The goal was to develop a task to monitor CPU activity and terminate any inactive user process. During development it was decided to design two tasks which could run independently, the monitor, and the process terminator. The monitor would provide a quick and easy reference to system activity. The process terminator, when activated, would warn the user, then terminate the process after the threshold of inactivity had been passed.

```
OP
2 (OP:OP)...........................:LOCK_CLI
   3 (OP:EXEC)......................:UTIL:EXEC
      19 (OP:LPB)...................:UTIL:XLPT
      20 (OP:LPE1)..................:UTIL:XLPT
      21 (OP:LPE)...................:UTIL:XLPT
      22 (OP:CON40).................:UTIL:XLPT
      32 (OP:CON89).................:UTIL:XLPT
      33 (OP:CON41).................:UTIL:XLPT
      114 (OP:CON57)................:UTIL:XLPT
   4 (OP:INFOS_II)..................:INFOS:INFOS_II
      9 (OP:009).......................:LANG:ORACLE:IOR
      12 (OP:CORBWR)................:LANG:ORACLE:BWR
      13 (OP:CORBIW)................:LANG:ORACLE:BIW
      14 (OP:CORCLN)................:LANG:ORACLE:CLN
      15 (OP:CORARH)................:LANG:ORACLE:ARH
   116 (OP:NETOP)...................:NET:NETOP
      24 (OP:X25_LMGR)..............:NET:X25_LMGR
      117 (OP:SVTA).................:NET:SVTA
      154 (OP:RMA)..................:NET:RMA
      157 (OP:FTA)..................:NET:FTA
```

Figure 1. Process tree for OP.

However, three basic problems needed to be solved. First, a process was required by which only process trees with inactive terminal sons would be terminated. Second, the updates registered by the CEO clock indicate that the process tree of a CEO user is active, when, in fact, it is not. Therefore, it was necessary to determine the inactivity threshold and terminate only those processes below that limit. The third problem concerned exceptions to the rule, that is, certain users who for various reasons would never be terminated.

IV. SOLUTION OVERVIEW

The two tasks developed were PIDACT, the process monitor, and ERP, the process terminator. PIDACT provides a visual display of the status of each PID. ERP can be activated or deactivated at any time. If it is determined that a user is inactive, then the user is warned. After a specified number of warnings, the process tree is terminated. A "VIP Table" was developed to ensure that certain users are not subject to termination.

Together PIDACT and ERP solve the three problems mentioned in the above section. To ensure that only inactive process trees are terminated, the process tree is traversed using the ?PSTAT system call. This traversal enables the task to find the terminal son and father process of the tree. Next, to ensure that inactive CEO processes are terminated, a threshold of CPU time was needed. This was determined by observing and testing of processes. It was found that active processes typically use more than five milliseconds of CPU time per block minute. For example, pressing a NEW LINE takes about 6 milliseconds. The VIP table, called VIP.DAT, which can be modified by a text editor, was designed to ensure that selected users are not terminated.

The remaining system calls needed for PIDACT and ERP are: ?SEND, used to send messages to an inactive PID; ?RUNTM, used to get the run time ticks of a process; ?GPRNM, used to get the program path name of a process; and ?GUNM, used to get the owner, username of the father process. ?TERM, used to terminate the process tree, is the only privileged system call, and requires Superprocess privileges.

V. APPLICATIONS

A. Process Activity Monitor

The task which monitors system activity is called PIDACT. PIDACT divides processes into four groups:

1) Father process or OP

2) Active process

3) Inactive terminal son

4) Unassigned PID.

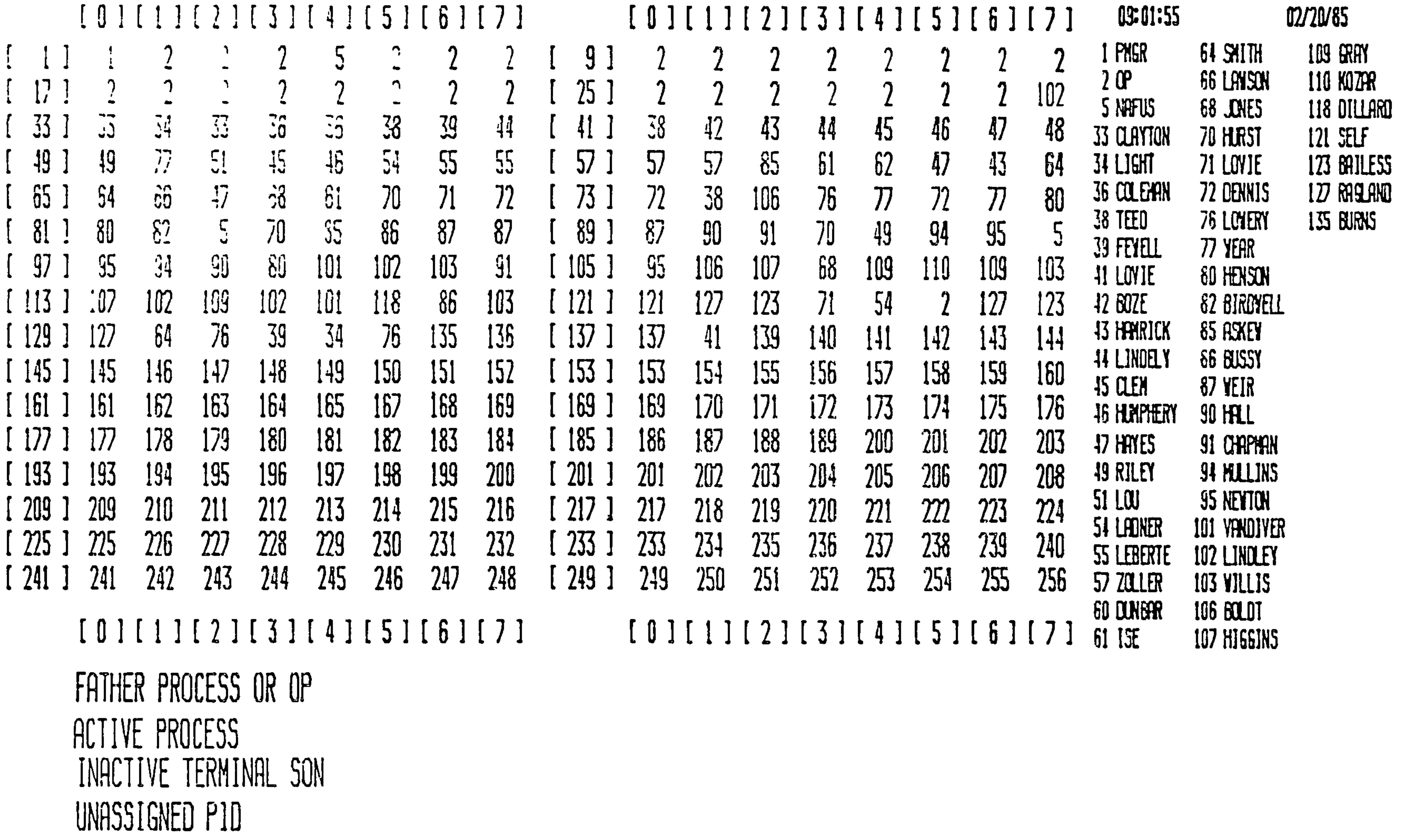
These divisions allow the system manager to easily monitor system activity. On the screen, the father process or OP is displayed in normal video; active processes are blinking; inactive terminal sons are shown in reverse video; and the unassigned PIDS are shown as zeros. Figure 2 shows a typical PIDACT display. The PIDs and usernames of those users logged on are shown on the right. The numbers in square brackets are scales, and help locate a PID quickly. PIDACT updates the screen once a minute, and is date and time stamped. To execute PIDACT requires no special privileges.

To illustrate how PIDACT works, the three process trees shown in Figure 2 shall be examined. First consider PID 44, a father process with an inactive terminal son process at PID 40. This process tree is subject to the process termination task, ERP. Now consider PID 102. PID 102 is a father process with a son process at 114, which is also a father process. PID 114 has two sons, at PID 116 and PID 32. Both son processes are active, therefore this process tree is active and would not be terminated by ERP. Lastly, consider PID 72. PID 72 is a father process with two sons, PID 73 and PID 78. PID 73 is active, PID 78 is an inactive terminal son. This process tree would be subject to termination.

B. Process Terminator

The task which terminates processes is called ERP. Approximately once every eight minutes PIDACT determines the CPU activity of all the processes on the system. If activity is below the threshold, the user is warned. If no significant activity is observed after two successive warnings, the process is terminated by ERP. Superprocess privileges are required to terminate the process. Upon warning a user or terminating a process, ERP records the action in a log. Figure 3 shows an example of an ERP log.

The following is a list of criteria ERP uses to terminate a process:



```
        [ 0 ][ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ][ 7 ]          [ 0 ][ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ][ 7 ]   09:01:55          02/20/85
[   1 ]   1    2    2    2    5    2    2    2   [   9 ]   2    2    2    2    2    2    2    2      1 PMGR        64 SMITH       109 GRAY
[  17 ]   2    2    2    2    2    2    2    2   [  25 ]   2    2    2    2    2    2    2  102      2 OP          66 LAWSON      110 KOZAR
[  33 ]  33   34   33   36   35   38   39   44   [  41 ]  38   42   43   44   45   46   47   48      5 NAFUS       68 JONES       118 DILLARD
[  49 ]  49   77   51   45   46   54   55   55   [  57 ]  57   57   85   61   62   47   43   64     33 CLAYTON     70 HURST       121 SELF
[  65 ]  54   66   47   68   61   70   71   72   [  73 ]  72   38  106   76   77   72   77   80     34 LIGHT       71 LOVIE       123 BAILESS
[  81 ]  80   82    5   70   85   86   87   87   [  89 ]  87   90   91   70   49   94   95    5     36 COLEMAN     72 DENNIS      127 RAGLAND
[  97 ]  95   34   90   80  101  102  103   91   [ 105 ]  95  106  107   68  109  110  109  103     38 TEED        76 LOWERY      135 BURNS
[ 113 ] 107  102  109  102  101  118   86  103   [ 121 ] 121  127  123   71   54    2  127  123     39 FEWELL      77 YEAR
[ 129 ] 127   64   76   39   34   76  135  136   [ 137 ] 137   41  139  140  141  142  143  144     41 LOVIE       80 HENSON
[ 145 ] 145  146  147  148  149  150  151  152   [ 153 ] 153  154  155  156  157  158  159  160     42 BOZE        82 BIRDWELL
[ 161 ] 161  162  163  164  165  167  168  169   [ 169 ] 169  170  171  172  173  174  175  176     43 HAMRICK     85 ASKEY
[ 177 ] 177  178  179  180  181  182  183  184   [ 185 ] 186  187  188  189  200  201  202  203     44 LINDELY     66 BUSSY
[ 193 ] 193  194  195  196  197  198  199  200   [ 201 ] 201  202  203  204  205  206  207  208     45 CLEM        87 WEIR
[ 209 ] 209  210  211  212  213  214  215  216   [ 217 ] 217  218  219  220  221  222  223  224     46 HUMPHERY    90 HALL
[ 225 ] 225  226  227  228  229  230  231  232   [ 233 ] 233  234  235  236  237  238  239  240     47 HAYES       91 CHAPMAN
[ 241 ] 241  242  243  244  245  246  247  248   [ 249 ] 249  250  251  252  253  254  255  256     49 RILEY       94 MULLINS
        [ 0 ][ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ][ 7 ]          [ 0 ][ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ][ 7 ]    51 LOU         95 NEWTON
                                                                                                   54 LADNER     101 VANDIVER
  FATHER PROCESS OR OP                                                                             55 LEBERTE    102 LINDLEY
  ACTIVE PROCESS                                                                                   57 ZOLLER     103 WILLIS
  INACTIVE TERMINAL SON                                                                            60 DUNBAR     106 BOLDT
  UNASSIGNED PID                                                                                   61 ISE        107 HIGGINS
```

Figure 2. PIDACT screen.

```
PID: 36 1ST WARNING LINDLEY          13:55:06 02/25/85
PID: 39 1ST WARNING MCKOSKY          13:55:07 02/25/85
PID: 43 1ST WARNING WEAVER           13:55:07 02/25/85
PID: 51 1ST WARNING ADAMS            13:55:07 02/25/85
PID: 16 1ST WARNING CARTER           13:55:07 02/25/85
PID: 46 1ST WARNING BUSH             14:00:09 02/25/85
PID: 36 2ND WARNING LINDLEY          14:00:09 02/25/85
PID: 39 2ND WARNING MCKOSKY          14:00:09 02/25/85
PID: 43 2ND WARNING WEAVER           14:00:09 02/25/85
PID: 51 2ND WARNING ADAMS            14:00:09 02/25/85
PID: 16 2ND WARNING CARTER           14:00:09 02/25/85
PID:103 1ST WARNING NAFUS            14:00:09 02/25/85
PID: 90 1ST WARNING SHOTTS           14:00:09 02/25/85
PID: 36 TERMINATION LINDLEY          14:05:10 02/25/85
PID: 39 TERMINATION MCKOSKY          14:05:10 02/25/85
PID: 43 TERMINATION WEAVER           14:05:10 02/25/85
PID: 51 TERMINATION ADAMS            14:05:10 02/25/85
PID: 57 1ST WARNING SMITH            14:05:10 02/25/85
```

Figure 3. ERP log.

1) Current CPU time < old CPU time + threshold

2) Current CPU time >= old CPU time

3) USERNAME not in VIP table

4) PID > 3

5) Program name <> OP

6) Father process resolves to EXEC.

ERP was designed specifically to terminate processes based upon inactive leaf nodes in the process tree. Since CEO leaves an inactive CEO word processing (CEO_WP) when completing word processing yet not exiting the CEO control program (CEO_CP), the active CEO_CP will be terminated. This feature could be changed by modifying ERP or writing an additional task to monitor and terminate CEO_WP processes only. In addition, if a user initiates co-processes where they are both leaf nodes in the process tree and only one is active, the process tree is terminated. If the user intends to have an inactive co-process as a leaf node, then he should request that the System Manager place his name in the VIP table.

VI. CONCLUSION

The PIDACT and ERP tasks are part of the SMIS security system. Though security is the primary consideration, the termination of idle processes also frees limited system resources: terminals, memory, and process capacity. The CPU utilization involved in running ERP is an average 0.2 percent. Each idle process utilizes an average of 0.2 percent. Therefore, for SMIS, the overhead for running ERP is well justified.

APPENDIX A

```
COMMENT                    PIDACT - PID ACTIVITY MACRO

WRITE [!ASCII 214]
WRITE TO END DISPLAY PERFORM A ^C^B
WRITE .....
STRING [!READ press NEW LINE begin PID ACTIVITY DISPLAY]
WIDE
X/1=IGN/2=IGN PIDACT
NORM
WRITE [!ASCII 214]
```

```
COMMENT                    PROC_ERP
COMMENT                    MACRO TO PROC UP THE ERP PID
COMMENT                    TERMINATION PROCESS

DEL/1=IGN/2=IGN SAVE.ERP.LOG
REN/1=IGN/2=IGN ERP.LOG SAVE.ERP.LOG
CRE ERP.LOG
PROC/NOBL/INP=@NULL/OUT=@NULL/LIST=ERP.LOG/SUPERP ERP
```

```
COMMENT                    WIDE
COMMENT                    MACRO TO PUT DG 460 TERMINAL
COMMENT                    INTO WIDE MODE

        CHAR/CPL=134
        WRITE [!ASCII 236 306 330 260 260 270 265]
        WRITE [!ASCII 236 306 313]
```

```
COMMENT                    NORM
COMMENT                    MACRO TO PUT DG 460 TERMINAL INTO
COMMENT                    80 COLUMN MODE

        CHAR/CPL=80
        WRITE [!ASCII 236 306 330 260 260 264 277]
        WRITE [!ASCII 236 306 312]
```

```
C
C************************************************************
C
        SUBROUTINE CURPOS (N1, N2, IBLK)
C
C                       THIS SUBROUTINE WILL PERFORM
C                       CURSOR POSITIONING FOR THE
C                       DATA GENERAL 410 AND 460 TERMINALS
C                       WERE N1 IS THE ROW AND N2 IS THE COLUMN
C
C                       THE VALUE IBLK IS A FLAG WHICH INDICATES
C                       THAT THE SCREEN IS TO BE ERASED BEFORE
C                       THE CURSOR IS TO BE POSITIONED
C
C                       CALLING PROGRAM SHOULD OUTPUT AFTER CALL
C                       IN THE FOLLOWING FORM:
C                               FORMAT ('#', ....
C                       THIS WILL SUPRESS THEN NEXT FORMAT FROM
C                       OUTPUTTING A CR
C
        CHARACTER N*1(4)
        INTEGER   N1, N2, ITMP1, ITMP2, IBLK, I
C
C                       N     - ARRAY TO CONTAIN ASCII TERM. COMMANDS
C                       N1    - ROW
C                       N2    - COLUMN
C                       ITMP1 - INTERUM CALCULATION FOR ROW
C                       ITMP2 - INTERUM CALCULATION FOR COLUMN
C                       IBLK  - ERASE SCREEN FLAG (1=YES)
C                       I     - LOCAL INDEX
C
C                       CHECK IF SCREEN IS TO BE BLANKED 1ST
C
        IF (IBLK.EQ.1) THEN
            WRITE (*, 101)                          !ERASE SCREEN
 101        FORMAT (1X,'<036><106><105>')
        ENDIF
C
C                       PERFORM INITIAL CALCULATIONS
C
        ITMP1=N1/16                                 !MOD 16 ROW
        ITMP2=N2/16                                 !MOD 16 COLUMN
C
C                       CALCULATE COLUMN POSITION
C
        N(1)=CHAR(ITMP2+48)                         !COLUMN 1ST
        N(2)=CHAR(N2-(ITMP2*16)+48)                 !  IN TWO DIGITS
C
C                       CALCULATE ROW POSITION
C
        N(3)=CHAR(ITMP1+48)                         !ROW 2ND
        N(4)=CHAR(N1-(ITMP1*16)+48)                 !  IN NEXT TWO DIGITS
C
C                       OUTPUT THE POSITION
C
        WRITE (*,102) (N(I),I=1,4)                  !OUTPUT THE FOUR CHAR
 102    FORMAT (1X,'<036><106><120>',4A1,$)         !SUPRESS CR
C
        RETURN
        END
        PROGRAM ERP
C
C                       WARNS AND THEN TERMINATES
C                       INACTIVE PID'S
C
        INTEGER*4 ITIM(256), ICPU(256), IDIS(256)
        INTEGER*4 ETIME, CPUTIM, IERR
C
        DATA ITIM/256*0/                !ELAPSED TIME ARRAY
        DATA ICPU/256*0/                !CPU TIME ARRAY
        DATA IDIS/256*0/                !PID ARRAY
C
C                       SET PROGRAM LIMITS AND FLAGS
C
        IMIN=8                          !MINUTE UPDATE TIME
        IFIRST=0                        !INITIALIZE FIRST LOOP FLAG
        IMINCPU=5*IMIN                  !SET CPU MINIMUM CPU ACTIVITY
C
C                       PERFORM FOR ALL POSSIBLE PIDS
C
 100    DO I=1,256
C
C                       GET ELAPSED TIME AND CPU TIME FOR
C                       THE SELECTED PID
C
            K=I
            CALL RUNTM (K, ETIME, CPUTIM, IERR)
C
C                       CHECK IF PID IS IN USE
C
            IF (IERR.NE.0) THEN
C
C                       PID IS NOT IN USE
C
                IDIS(I)=I                       !USE ACTUAL PID NO.
                ICPU(I)=0                       !ZERO OUT CPU TIME
                ITIM(I)=0                       !ZERO OUT ELAPS TIME
C
              ELSE
C
C                       PID IS IN USE GET THE FATHER'S PID
C                       WHICH IS CLOSEST TO OP.EXEC
C
                K=I
                CALL PDAD (K, IDIS(I))
C
                IF (ICPU(I)+IMINCPU.LT.CPUTIM .OR.
     &              CPUTIM.LT.ICPU(I)         .OR.
     &              ICPU(I).EQ.0) THEN
C
C                       A CHANGE IN CPU TIME HAS OCCURED
C                       OR A NEW PROCESS HAS TAKEN THIS PID
C                       OR THIS IS THE INITIAL RUN
C                       UPDATE ELAPSED TIME, CPU TIME
C                       AND DISPLAY FIELD
C
                    ITIM(I)=ETIME               !UPDATE ELAPSED TIME
C
                  ELSE
C
C                       NO CHANGE IN CPU TIME
C                       CHECK IF THIS PROCESS HAS ANY SONS
```

```
                    K=I
                    CALL PIDSON (K, IFLG)
C
C                       IF NO SONS THAN CHECK FOR
C                       WARNING OR TERMINATION
C                       IGNORE ANY PIDS WHICH RESOLVE LESS THAN 4
C
                    IF (IFLG.EQ.0 .AND. IDIS(I).GT.3) THEN
                        CALL LIMIT (IDIS(I), ETIME, ITIM(I), IMIN)
                    ENDIF
C
                END IF
C
                ICPU(I)=CPUTIM                  'UPDATE CPU TIME
C
            END IF
C
        END DO
C
        IF (IFIRST.EQ.1) THEN
C
C                       SET UP TO DELAY 5 MINUTES
C
            CALL MDELAY (IMIN)
C
        END IF
C
        IFIRST=1                                'SET INITIAL PASS DONE
C
C                       DO FOREVER
C
        GOTO 100
C
        END
%INCLUDE "RUNTM.F77"
%INCLUDE "PDAD.F77"
%INCLUDE "PIDSON.F77"
%INCLUDE "MDELAY.F77"
%INCLUDE "UNAME.F77"
%INCLUDE "TERM.F77"
%INCLUDE "LIMIT.F77"
%INCLUDE "SEND.F77"
%INCLUDE "VIP.F77"
%INCLUDE "TIMDAT.F77"
```

```
      SUBROUTINE LIMIT (PID, ETIME, OTIME, MIN)
C
C                      DETERMINES THE WARNING OR TERMINATION
C                      STATUS OF THE SELECTED PID
C
      INTEGER*4 PID, ETIME, OTIME, DELTET, MIN
      INTEGER*4 ILIMIT1, ILIMIT2, ILIMIT3
      CHARACTER UNM*32, TMDT*18
C
C                      INITIALIZE LIMIT VALUES
C
      ILIMIT1=1*MIN*60
      ILIMIT2=2*MIN*60
      ILIMIT3=3*MIN*60
C
C                      CALCULATE DELTA ELAPSED TIME
C
      DELTET=ETIME-OTIME
C
C                      CHECK IF IN ACTION STATE
C
      IF (DELTET.GT.ILIMIT1) THEN
C
C                      GET USERNAME OF PID
C
         CALL UNAME (PID, UNM)
C
C                      CHECK IF THIS PID IS EXEMPT
         IFLG=0
         CALL VIP (UNM, IFLG)
         IF (IFLG.EQ.0) THEN
C
C                      PID IS NOT EXEMPT
C
C
             IF (DELTET.GT.ILIMIT1 .AND. DELTET.LE.ILIMIT2) THEN
C
C                      ISSUE 1ST WARNING
C
                 CALL SEND (PID, 1)
                 CALL TIMDAT (TMDT)
                 WRITE (12, 101) PID, UNM(1:15), TMDT
 101             FORMAT (1X,' PID:',I3,' 1ST WARNING ',A15, 2X, A18)
             END IF
C
             IF (DELTET.GT.ILIMIT2 .AND. DELTET.LE.ILIMIT3) THEN
C
C                      ISSUE 2ND WARNING
C
                 CALL SEND (PID, 2)
                 CALL TIMDAT (TMDT)
                 WRITE (12, 102) PID, UNM(1:15), TMDT
 102             FORMAT (1X,' PID:',I3,' 2ND WARNING ',A15, 2X, A18)
             END IF
C
             IF (DELTET.GT.ILIMIT3) THEN
C
C                      TERMINATE PROCESS
C
                 CALL SEND (PID, 3)
                 CALL TIMDAT (TMDT)
                 WRITE (12, 103) PID, UNM(1:15), TMDT
```

```
                END IF
 C
            END IF
 C
        END IF
 C
        RETURN
        END
```

```
C
C**********************************************************
C
       SUBROUTINE MDELAY (MIN)
C
C                      THIS ROUTINE WILL DELAY THE SELECTED
C                      NUMBER OF MINUTES BEFORE RESUMING THE PROCESS
C
       INTEGER*4 MIN
C
C                      SET UP TO DELAY 1 MINUTE
C
              IPID=179               !WDELAY CALL
              IAC0=1000*60*MIN       !DELAY IN MILLISECONDS
              IAC1=0                 !RESERVED
              IAC2=0                 !RESERVED
C
C                      PERFORM WDELAY CALL TO
C                      DELAY MIN MINUTES
C
              IERR=ISYS (IPID, IAC0, IAC1, IAC2)
C
       RETURN
       END
```

```
C
C***********************************************************
C
        SUBROUTINE PDAD (PIDIN, PIDOUT)
C
C                       THIS SUBROUTINE RETURNS THE HIGHEST PID
C                       FATHER BELOW PID 3 IN PIDOUT
C
        INTEGER*4 ISYS, IAC0, IAC1, IAC2
        INTEGER*4 PIDIN, PIDOUT
        CHARACTER UNM*32
C
C                       CHECK FOR A PID LOWER THAN 4
C
        IF (PIDIN.GT.3) THEN
C
C                       SET CALLIN PID NUMBER
C
            IAC1=PIDIN
C
C                       FIND THE FATHER
C
            DO WHILE (IAC1.GT.3)
C
                I=IAC1
C
C                       SET UP TO MAKE FATHER PROCESS CALL
C
                IPID=87         !FATHER PROCESS CALL
                IAC0=I          !PID NO.
                IAC1=0          !RETURN FATHER PID
                IAC2=0          !RETURN LIST
C
C                       THIS CALL WILL RETURN THE FATHER'S
C                       PID IN IAC1
C
                IERR=ISYS (IPID, IAC0, IAC1, IAC2)
C
            END DO
C
            IF (IAC1.LT.3) THEN
                PIDOUT=IAC1
              ELSE
                CALL UNAME(I, UNM)
                IF (UNM(1:3).EQ.'OP ') THEN
                    PIDOUT=2
                  ELSE
                    PIDOUT=I
                ENDIF
            END IF
C
          ELSE
C
            PIDOUT=PIDIN
            IF (PIDOUT.EQ.3) PIDOUT=2
C
        ENDIF
C
        RETURN
        END
```

```
       PROGRAM PIDACT
C
C                       DISPLAYS ACTIVE PID NUMBERS CONTINUOUSLY
C                       ON SCREEN BASED UPON CPU TIME
C
       INTEGER*4 ITIM(256), ICPU(256), IDIS(256), USE(256)
       INTEGER*4 ETIME, CPUTIM, IERR, CNT(4)
       CHARACTER MODE*6(256), BLK*2, REV*2, DIM*2, NRM*4, NUL*2
       CHARACTER UNM*32, TMDT*18
C
C                       INITIALIZE THE FOLLOWING ARRAYS
C                           ICPU - CONTAINING LAST CPU TIME
C                           ITIM - CONTAINING LAST ELAPSED TIME
C                           IDIS - CONTAINS PID NUMBER IF ACTIVE
C
       DATA CNT/4*0/
       DATA ICPU/256*0/, ITIM/256*0/, IDIS/256*0/, USE/256*0/
C
C                       SET PROGRAM LIMITS AND FLAGS
C
       IMIN=1                           !MINUTE UPDATE TIME
       IFIRST=0                         !INITIALIZE FIRST LOOP FLAG
       IMINCPU=5*IMIN                   !SET CPU MINIMUM CPU ACTIVITY
C
C                       INITIALIZE DISPLAY CHARACTERISTICS
C
       NUL='<000><000>'                 !NULL CHARACTERS
       BLK='<216><000>'                 !CHARACTER BLINK ON
       REV='<236><304>'                 !REVERSE VIDIO
       DIM='<234><000>'                 !CHARACTER DIM ON
       NRM='<217><236><305><235>'       !BLINK OFF/REVERSE OF/DIM OFF
C
C                       PUT UP FORM
C
       CALL PIDFORM
C
C
C                       PERFORM FOR ALL POSSIBLE PIDS
C
100    DO I=1,256
C
C                       GET ELAPSED TIME AND CPU TIME FOR
C                       THE SELECTED PID
C
           K=I
           CALL RUNTM (K, ETIME, CPUTIM, IERR)
C
C                       CHECK IF PID IS IN USE
C
           IF (IERR.NE.0) THEN
C
C                       PID IS NOT IN USE - SET DISPLAY TO
C                       PID NO. AND SET MODE TO DIM
C
               IDIS(I)=I                       !USE ACTUAL PID NO.
               MODE(I)=NRM//DIM                !SET MODE TO DIM
               ICPU(I)=0                       !CPU TIME
               ITIM(I)=0                       !ELAPSED TIME
               IDIS(I)=0                       !DISPLAY PID
               USE(I)=0                        !USER NAME ARRAY
               CNT(4)=CNT(4)+1                 !UPDATE UNUSED CNT
C
```

```
C
C                       PID IS IN USE GET THE FATHER'S PID
C                       WHICH IS CLOSEST TO OP.EXEC
C
                K=I
                CALL PDAD (K, IDIS(I))
                USE(IDIS(I))=1                  !UPDATE FOR USER DISP
C
                IF (ICPU(I)+IMINCPU.LT.CPUTIM .OR.
     &              CPUTIM.LT.ICPU(I)         .OR.
     &              ICPU(I).EQ.0) THEN
C
C                       A CHANGE IN CPU TIME HAS OCCURED
C                       OR A NEW PROCESS HAS TAKEN THIS PID
C                       OR THIS IS THE INITIAL RUN
C                       UPDATE ELAPSED TIME, CPU TIME
C                       AND DISPLAY FIELD
C
                    ITIM(I)=ETIME               !UPDATE ELAPSED TIME
                    MODE(I)=NRM//BLK            !SET BLINK MODE ON
                    CNT(2)=CNT(2)+1             !UPDATE ACTIVE COUNT
C
                  ELSE
C
C                       NO CHANGE IN CPU TIME
C                       CHECK IF THIS PROCESS HAS ANY SONS
C
                    K=I
                    CALL PIDSON (K, IFLG)
C
C                       IF NO SONS AND NOT OP THEN REVERSE VIDIO
C                       ELSE MAKE DISPLAY NORMAL
C
                    IF (IFLG.EQ.0 .AND. IDIS(I).NE.2) THEN
                        MODE(I)=NRM//REV        !PID HAS NO SONS
                        CNT(3)=CNT(3)+1         !UPDATE INACTIVE COUNT
                      ELSE
                        MODE(I)=NRM//NUL        !PID HAS SON(S)
                        CNT(1)=CNT(1)+1         !UPDATE FATHER COUNT
                    ENDIF
C
                END IF
C
                ICPU(I)=CPUTIM                  !UPDATE CPU TIME
C
            END IF
C
        END DO
C
C                       DISPLAY CURRENT ACTIVE PIDS
C                       IN MATRIX FORM ON SCREEN
C
        DO I=1,241,16
C
            M=I/16+1                            !CALC ROW INDEX
C
            CALL CURPOS (M,5,0)                 !POSITION CURSOR
            WRITE (*,300) (MODE(K),IDIS(K),K=I,I+7)
C
 300        FORMAT ('#',16(A6,I4))
C
            CALL CURPOS (M,43,0)                !POSITION CURSOR
```

```
C
        END DO
C
C                       DISPLAY ACTIVE USER NAMES
C
        PRINT *,NRM
        M=1                                      !INITIAL ROW POSITION
        N=75                                     !INITIAL COLUMN POS
        DO I=1, 256
            IF (USE(I).EQ.1) THEN
                K=I
                CALL UNAME (K, UNM)
                IF ((I.GT.2 .AND. UNM(1:3).NE.'OP ') .OR. I.LE.2) THEN
                    CALL CURPOS (M,N,0)
                    WRITE (*, 400) I, UNM(1:8)
 400                FORMAT ('#',I5,1X,A8)
                    M=M+1
                    IF (M.GT.22) THEN
                        N=N+14
                        M=1
                    END IF
                END IF
            END IF
        END DO
C
C                       BLANK OUT ANY UNUSED FIELDS
C
        DO WHILE (N.LT.120)
            CALL CURPOS (M, N, 0)
            WRITE (*, 500)
 500        FORMAT ('#','              ')
            M=M+1
            IF (M.GT.22) THEN
                N=N+14
                M=1
            END IF
        END DO
C
C                       UPDATE TIME/DATE DISPLAY
C
        CALL TIMDAT (TMDT)
        CALL CURPOS (0,96,0)
        WRITE (*, 600) NRM, TMDT
 600    FORMAT ('#',A4,A18)
C
C                       UPDATE DISPLAY COUNTS
C
        DO I=1,4
            M=I+18
            CALL CURPOS (M,32,0)
            WRITE (*,FMT="('#',I3)") CNT(I)
            CNT(I)=0            !RESET COUNTERS
        END DO
C
C                       ZERO OUT USER DISPLAY TABLE
C
        DO I=1,256
            USE(I)=0
        END DO
C
C                       CHECK FOR INITIAL RUN CONDITION
C                       DO NOT DELAY IF ONLY RUN ONCE
```

```
        IF (IFIRST.EQ.1) THEN
C
C                       SET UP TO DELAY 5 MINUTES
C
            CALL MDELAY (IMIN)
C
        END IF
C
        IFIRST=1                                        !SET INITIAL PASS DONE
C
C                       DO FOREVER
C
        GOTO 100
C
        END
%INCLUDE "CURPOS.F77"
%INCLUDE "RUNTM.F77"
%INCLUDE "PDAD.F77"
%INCLUDE "PIDSON.F77"
%INCLUDE "MDELAY.F77"
%INCLUDE "PIDFORM.F77"
%INCLUDE "UNAME.F77"
%INCLUDE "TIMDAT.F77"
```

```
C
C*** *******************************************************
C
        SUBROUTINE PIDFORM
C
C                         THIS SUBROUTINE WILL LAYOUT A FORM
C                         FOR THE PID ACTIVITY REPORT
C
        CHARACTER MODE*6(4), LEGEND*22(4)
C
C                         DG 400 SERIES CONTROLL CODES
C                         FOR:
C                                 NORMAL
C                                 BLINK ON
C                                 REVERSE VIDIO
C                                 DIM ON
C
        DATA MODE/'<217><236><305><235><000><000>',
     &            '<217><236><305><235><216><000>',
     &            '<217><236><305><235><236><304>',
     &            '<217><236><305><235><234><000>'/
C
C                         EXPLANATION LEGEND
C
        DATA LEGEND/'FATHER PROCESS OR OP   ',
     &              'ACTIVE PROCESS         ',
     &              'INACTIVE TERMINAL SON ',
     &              'UNASSIGNED PID         '/
C
C                         OUTPUT TOP PID LEGEND
C
        CALL CURPOS (0,5,1)
        WRITE (*, 101)
 101    FORMAT ('#',' [0] [1] [2] [3] [4] [5] [6] [7]')
C
        CALL CURPOS (0,43,0)
        WRITE (*, 101)
C
C                         OUTPUT SIDE PID LEGENDS
C
        DO I=1, 256,16
            J=I+8
            WRITE (*, 201)I, J
 201        FORMAT (1X,'[',I3,']',33X,'[',I3,']')
        END DO
C
C                         OUTPUT BOTTOM PID LEGEND
C
        CALL CURPOS (17,5,0)
        WRITE (*, 101)
C
        CALL CURPOS (17,43,0)
        WRITE (*, 101)
C
C                         OUTPUT EXPLANATION LEGENDS
C
        DO I=1,4
            K=I+18
            CALL CURPOS (K,10,0)
            WRITE (*, 301) MODE(I), LEGEND(I)
 301        FORMAT ('#',A6,A22)
        END DO
        END
```

```
C
C***********************************************************
C
        SUBROUTINE PIDSON (PID, FLAG)
C
C                        THIS ROUTINE DETERMINES IF THIS PID
C                        HAS ANY SONS
C                            IF YES THEN FLAG=1
C                                   ELSE FLAG=0
C
        INTEGER*4 ISYS, IAC0, IAC1, IAC2
        INTEGER*4 PID, FLAG
        INTEGER*2 STAT(200)
C
C                        PERFORM PSTAT CALL TO DETERMINE
C                        IF SELECTED PID HAS ANY SONS
C
        IPID=5
        IAC0=PID
        IAC1=0
        IAC2=WORDADDR(STAT)
C
        IERR=ISYS(IPID, IAC0, IAC1, IAC2)
C
C                        CHECK BIT PATTERN FOR ANY SONS
C
        FLAG=0
        DO J=2,17
            FLAG=FLAG+STAT(J)
        END DO
C
C                        IF SONS EXIST THEN MAKE FLAG = 1
C
        IF (FLAG.NE.0) FLAG=1
C
        RETURN
        END
```

```
C
C*************************************************************
C
        SUBROUTINE RUNTM (PID, ETIME, CPUTIM, IERR)
C
C                       GETS PID NUMBER AND RETURNS ELAPSED TIME
C                       IN SECONDS AND CPU TIME IN MILLISECONDS
C
        INTEGER*4 ISYS, IAC0, IAC1, IAC2
        INTEGER*4 PAC(4)
        INTEGER*4 PID, ETIME, CPUTIM, IERR
C
C                       SET UP TO MAKE SYSTEM RUN TIME CALL
C
        IPID=24                 !RUNTIME CALL
        IAC0=PID                !PID NO.
        IAC1=0                  !USING PID
        IAC2=WORDADDR(PAC)      !RETURN LIST
C
C                       PERFORM RUNTIME CALL TO GET
C                       ELAPSED TIME AND CPU TIME
C
        IERR=ISYS (IPID, IAC0, IAC1, IAC2)
C
        ETIME=PAC(1)            !RETURN ELAPSED TIME
        CPUTIM=PAC(2)           !RETURN CPU TIME
C
        RETURN
        END
        SUBROUTINE SEND (PID, NUM)
C
C                       THIS SUBROUTINE RETURNS SENDS WARNING
C                       MESSAGES TO THE SELECTED PID NO.
C                       UPON THE THIRD WARNING BEING SENT
C                       THIS ROUTINE WILL CALL FOT THE TERMINATION
C                       OF THE SELECTED PID
C
        INTEGER*4 ISYS, IAC0, IAC1, IAC2
        INTEGER*4 PID, NUM
        CHARACTER MESS*47, WARN*47(3)
C
        DATA WARN/'<BEL>1ST WARNING TERMINAL INACTIVE FOR 5 MIN
     &            '<BEL>FINAL WARNING BEFORE LOG OFF - INACTIVE 10 MIN
     &            '<BEL>TERMINATION - INACTIVE 15 MIN
C
C                       SET MESSAGE LENGTH
C
        LEN=47
C
C                       GET SELECTED MESSAGE
C
        MESS=WARN(NUM)
C
C                       SET UP TO MAKE ?SEND CALL
C
                IPID=206                !SEND CALL
                IAC0=PID                !PID NO.
                IAC1=BYTEADDR(MESS)     !MESSAGE POINTER
                IAC2=LEN                !MESSAGE LENGTH
C
C                       SEND THE MESSAGE TO THE SELECTED PID
C
                IERR=ISYS (IPID, IAC0, IAC1, IAC2)
C
C                       CHECK FOR TERMINATION
C
C*****************************************************************
C*      REMOVE COMMENTS TO ACTIVATE TERMINATION OPERATION        *
C*****************************************************************
C                                                                *
        IF (NUM.EQ.3) THEN              !                        *
            CALL TERM(PID)              !                        *
        END IF                          !                        *
C                                                                *
C*****************************************************************
C
        RETURN
        END
```

```
      SUBROUTINE TERM (PID)
C
C                    THIS SUBROUTINE TURNS ON SUPERPROCESS AND THEN
C                    TERMINATES THE SELECTED PID, ALL SON PROCESSES
C                    ARE ACCORDINGLY ALSO TERMINATED
C
      INTEGER*4 ISYS, IPID, IAC0, IAC1, IAC2, PID
C
C                    SET UP TO TURN ON SUPERPROCESS
C
              IPID=43                     !SUPROC CALL
              IAC0=-1                     !TURN ON
              IAC1=0                      !UNDEFINED
              IAC2=0                      !UNDEFINED
C
C                    TURN ON SUPERPROCESS
C
              IERR=ISYS (IPID, IAC0, IAC1, IAC2)
C
C                    SET UP TO MAKE ?GTERM CALL
C
              IPID=45                     !TERM CALL
              IAC0=PID                    !PID NO.
              IAC1=0                      !CONTAINS PID
              IAC2=0                      !NO MESSAGE
C
C                    TERMINATE THE SELECTED PID
C
              IERR=ISYS (IPID, IAC0, IAC1, IAC2)
C
C                    SET UP TO TURN OFF SUPERPROCESS
C
              IPID=43                     !SUPROC CALL
              IAC0=1                      !TURN OFF
              IAC1=0                      !UNDEFINED
              IAC2=0                      !UNDEFINED
C
C                    TERMINATE THE SELECTED PID
C
              IERR=ISYS (IPID, IAC0, IAC1, IAC2)
C
      RETURN
      END
```

```
C
C************************************************************
C
      SUBROUTINE TIMDAT (TMDT)
C
C                       THIS SUBROUTINE WILL RETURN THE CURRENT
C                       SYSTEM TIME AND DATE IN CHARACTER FORMAT
C                       IN STRING TMDT (OF LENGTH 18)
C
      INTEGER IDATE(3), ITIME(3), IBLD(6)
      CHARACTER TMDT*18
C
C                       GET SYSTEM DATE AND TIME FOR HEADER
C
      CALL DATE (IDATE)
      CALL TIME (ITIME)
C
C                       SET UP DATE TO BE IN MM/DD/YY FORM
C
      ITMP=IDATE(1)-1900
      IDATE(1)=IDATE(2)
      IDATE(2)=IDATE(3)
      IDATE(3)=ITMP
C
      DO I=1,3
          IBLD(I)=ITIME(I)
          IBLD(I+3)=IDATE(I)
      END DO
C
      DO I=1,6
          M=(I-1)*3+1
          N=M+1
          ITMP=IBLD(I)/10
          IBLD(I)=IBLD(I)-(ITMP*10)
          TMDT(M:M)=CHAR(ITMP+48)
          TMDT(N:N)=CHAR(IBLD(I)+48)
      END DO
C
      TMDT(3:3)=':'
      TMDT(6:6)=':'
      TMDT(9:9)=' '
      TMDT(12:12)='/'
      TMDT(15:15)='/'
      TMDT(18:18)=' '
C
      RETURN
      END
```

```
        SUBROUTINE UNAME (PID, UNM)
C
C                       THIS SUBROUTINE WILL RETURN THE USERNAME OF
C                       THE CURRENT PROCESS IN THE CHARACTER
C                       STRING UNM, THE STRING WILL BE TERMINATED
C                       WITH A <NULL>
C
        CHARACTER UNM*32
        INTEGER*4 ISYS, IAC0, IAC1, IAC2, IFLG, PID
C
C                       DETERMINE IF THIS IS TO BE THE
C                       CALLING TASK'S PID
C
        IF (PID.LT.0) THEN
            IFLG=1
          ELSE
            IFLG=0
        ENDIF
C
        IPID=58                 !?GUNM CALL
        IAC0=PID                !PID NO. OR -1
        IAC1=IFLG               !USING PID OR -1
        IAC2=BYTEADDR(UNM)      !RETURN LIST
C
C                       PERFORM SYSTEM CALL TO GET USERNAME
C
        IERR=ISYS (IPID, IAC0, IAC1, IAC2)
C
C                       BLANK THE STRING AFTER THE USERNAME
C
        IFLG=0
        DO I=1,32
            IF (UNM(I:I).EQ.'<000>') IFLG=1
            IF (IFLG.EQ.1) UNM(I:I)=' '
        END DO
C
        RETURN
        END
```

```
      SUBROUTINE VIP (UNAME, IFLG)
C
C                       THIS ROUTINE WILL DETERMINE IF THE
C                       USERNAME PASSED TO IT EXIST IN THE
C                       VIP.DAT FILE, IF YES THEN IFLG=1
C                                                ELSE IFLG=0
C
      CHARACTER UNAME*32, VNAME*32
C
C                       INITIALIZE RETURN FLAG TO 0
C
      IFLG=0
C
C                       OPEN VIP FILE
C
      OPEN (UNIT=21, STATUS='OLD
     &      FILE='VIP.DAT',
     &      IOSTAT=IERR1, RECFM='DS', FORM='FORMATTED',PAD='YES',
     &      ERR=999)
C
C                       READ ONE (1) LINE
C
 100  READ (21,FMT=101, IOSTAT=IERR2, ERR=999, RETURNRECL=IL) VNAME
 101  FORMAT (A32)
C
C                       CHECK FOR ZERO RECORD LENGTH
C
      IF (IL.EQ.0) GOTO 999
C
C                       CHECK IF NAME'S MATCH
C
      IF (VNAME.NE.UNAME) GOTO 100
C
C                       A MATCH HAS BEEN FOUND
C                       SET IFLG TO 1
C
      IFLG=1
C
C                       CLOSE THE VIP FILE AND RETURN TO CALLER
C
 999  CLOSE (UNIT=21)
C
      RETURN
      END
```

APPROVAL

A PROCESS ACTIVITY MONITOR FOR AOS/VS

By R. A. McKosky, S. W. Lindley, and J. S. Chapman

The information in this report has been reviewed for technical content. Review of any information concerning Department of Defense or nuclear energy activities or programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.

S. R. Reinartz

S. R. REINARTZ
Manager, Shuttle Projects Office

☆U S. GOVERNMENT PRINTING OFFICE 1986—631-058/20077