Introduction

This module describes characteristics of the OpenVMS process. Students will learn, conceptually, how processes operate. Students will be able to create processes, get information about them, control them, and delete them. Students will understand the impact of process/image rundown on exit handlers.

Objectives

Students should be able to:

- Identify the three contexts associated with a process.
- Create a process from program control.
- Use the SYS$GETJPI system service to obtain information about a process.
- Control a process through the use of the SYS$HIBER and SYS$WAKE system services.
- Differentiate between forcing image rundown and process deletion.
- Identify when exit handlers are run for a program.

Topics

- The Process
- The Create Process System Service
- Getting Job and Process Information
- Setting Process Characteristics
- Process Management and Control
- Terminating Images and Processes
- Exit Handlers

The Process

The *process* provides an environment, which supports the execution of programs. Traditionally, processes were the primary scheduling entity on OpenVMS systems, although much of this behavior has moved to *threads*. A thread is a sequential flow of execution within the process' address space.

- Processes are created either explicitly or implicitly through calls to the **SYS$CREPRC** system service.
- Process types:
  - Detached
    - Has no owner
    - Modes

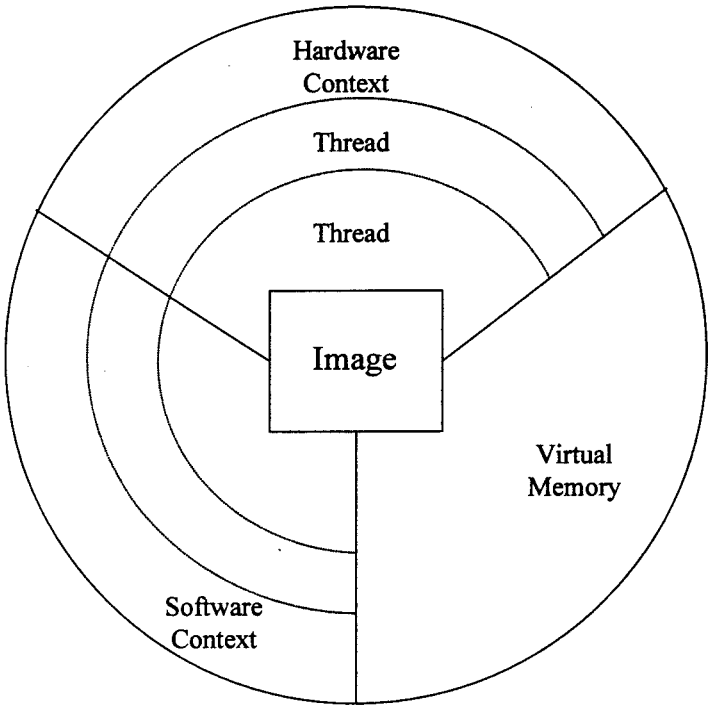
      Interactive

      Batch

      Network

      Other
  - Subprocesses
    - Are owned by a detached process or another
    - Share pooled/deductible quotas
    - Share allocated devices
    - Cannot live without creator
- A *Job* is an accounting entity, which is used to track quotas for all processes created in a tree.

Conceptual View Of A Process With Threads



Software Context

The Software context of a process provides a view of the process to OpenVMS.

Identification

- Process Id (Unique on the system and within a VMScluster)
- Process Name (Unique within UIC group)
- Username
- Account Name
- Terminal Name

Priority

- 0-15
  - Time-share
  - Round-robin
  - Priority Adjustment
  - Working Set List Adjustment
- 16-63
  - Real-time
  - Non-Preemptive
  - No Adjustment

Process Permanent Files

```
SYS$INPUT
SYS$OUTPUT
SYS$ERROR
```

Software Context (continued)

Rights

- User Identification Code
  - [group,member] group= 1->37776_8 member 0->177776_8
  - Mapped to user identifier through rightslist database
- Identifiers

Privileges

Class	Privileges
None	
Normal	TMPMBX, NETMBX
Group	GROUP, GRPPRV
Devour	PRMCEB, PRMMBX, PRMBGL, EXQUOTA, GRPNAM, ALLSPOOL, ACNT, SHMEM
System	ALTPRI, OPER, PSWAPM, WORLD, SECURITY, SYSLCK
Object	DIAGNOSE, SYSGBL, VOLPRO, IMPORT, MOUNT
All	SETPRV, BYPASS, SYSPRV, CMKRNL, CMEXEC, SYSNAM, IMPERSONATE, PFNMAP, LOG_IO, PHY_IO, READALL, SHARE, DOWNGRADE, UPGRADE

Resource Limits and Quotas

- Quotas and limits are divided into 3 categories:

1. Pooled — Charged against job/Reusable
2. Deductible — Charged against process/Non-reusable
3. Non-deductible — Charged against process/Reusable

▲ **Note**
Exhausting quotas may cause process to hang.

Software Context (continued)

Quota Summary

Quota	Class	Description
ASTLM	Non-deductible	Concurrent AST limit
BIOLM	Non-deductible	Buffered I/O limit
BYTLM	Pooled	Buffered byte limit
CPULM	Deductible	CPU time limit for job
DIOLM	Non-deductible	Direct I/O limit
ENQLM	Pooled	Limit for concurrent locks
FILLM	Pooled	Limit for concurrent open files
JTQUOTA	Pooled	Job logical name table quota
PGFLQUOTA	Pooled	Page file quota
PRCLM	Pooled	Process creation limit
TQELM	Pooled	Timer queue entry limit
WSDEFAULT	Non-deductible	Working set list size at image activation
WSQUOTA	Non-deductible	Working set list commitment
WSEXTENT	Non-deductible	Working set list limit is plenty of free memory

Hardware and Virtual Address Space Context

The Hardware context of a process is made up of General Purpose and Internal Registers. Implicit in the definition of the hardware context are process stacks and page tables.

- The hardware context of a process is saved when the process/thread enters a wait or blocked state and is restored when the process/thread is made current.

The Virtual Address space of a process is made up of P0, P1 and P2 space (also Page table space) described in Module 1.

The Create Process System Service

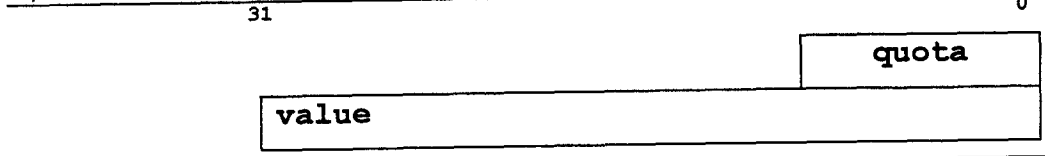
Processes are created on the system, or optionally on another node in a VMScluster, using the **SYS$CREPRC** system service.

Create Process system service format

```
SYS$CREPRC(pidadr, image, input, output, error, prvadr,
quota, prcnam, baspri, uic, mbxunt, stsflg, itmlst, node,
home_rad)
```

Create Process arguments

Argument	Meaning
pidadr	Address to receive process id
image	Name of program to run (SYS$SYSTEM:LOGINOUT.EXE for login process)
input, output, error	Names of process permanent files/devices
prvadr	If not specified, inherits creators. Symbols defined in `$PRVDEF/prvdef.h`.
quota	Passed as an array of structures of the form:



Argument	Meaning
	Quotas default to those specified by **SYSGEN** parameters: **PQL_D*quota***
	Specified value or default may be overridden by **SYSGEN** parameters: **PQL_M*quota***
	Quota types are defined in **$PQLDEF/pqldef.h**
prcnam	Process name
baspri	Base priority (def: 2 (MACRO/BLISS) else 0)
	Need **ALTPRI** to raise
uic	Causes process to be detached
	Requires **IMPERSONATE** privilege
mbxunt	Mailbox unit number for termination accounting message
stsflg	**PRCM_BATCH, PRCM_DETACH, PRCM_DISAWS, PRCM_HIBER, PRC$M_IMGDUMP, PRC$M_INTER, PRC$M_NETWORK, PRC$M_NOACNT, PRC$M_NOPASSWORD, PRC$M_NOUAF, PRCM_PSWAPM, PRCM_SSFEXCU, RC$M_SSRWAIT, PRC$M_SUBSYSTEM, PRCM_TCB, PRCM_HOME_RAD**
	Defined in **$PRCDEF/prcdef.h**
itmlst	Reserved
node	SCS node name on which to create process
home_rad	Identifies home Resource Affinity Domain (RAD) to assign to process.

The Create Process System Service (continued)

Note

SYS$CREPRC does not implicitly map DCL (you need to use **SYS$SYSTEM:LOGINOUT** for DCL to be mapped).

- If you need to issue a DCL command use:

```
lib$spawn(command,input,output,flags,prcnam,pid,
comp_sts,efn,astadr,astarg,prompt,cli,table)
```

LIB$SPAWN Example

```
$ type salmon.c
/* Sample LIB$SPAWN */
#include     <lib$routines.h>
#include     <ssdef>
#include     <descrip.h>
#include     <stdio.h>
#include     <starlet.h>
#define sys_stat(STS) if(!((STS)&1)) sys$exit(STS)
int main(void)
{
     $DESCRIPTOR(sh_proc,"SHOW PROCESS");
     $DESCRIPTOR(dir,"DIRECTORY *.MAR;");
     int status;
     printf("###Show process command###\n");
     status = lib$spawn(&sh_proc);
     sys_stat(status);
     printf("###DIR *.mar; command###\n");
     status = lib$spawn(&dir);
     sys_stat(status);
     return(SS$_NORMAL);
}
$ cc salmon
$ link salmon
$ r salmon
###Show process command###

21-JUL-1996 04:38:22.04   User: ALLEN          Process ID:   00000064
                          Node: ALLEN          Process name: "ALLEN_1"

Terminal:
User Identifier:    [ALLEN]
Base priority:      4
Default file spec:  _DKA300:[ALLEN]
Number of Kthreads: 1
###DIR *.mar; command###
```

LIB$SPAWN Example (continued)

```
Directory _DKA300:[ALLEN]

BUCKET.MAR;3          CFCB.MAR;4          CHANGE_IODB.MAR;8   CHANGE_USER.MAR;12
CH_IO_CR.MAR;1        CL.MAR;1            CS.MAR;3            CS_SUB.MAR;4
DS.MAR;3              ILE.MAR;3           IO.MAR;1            JIB.MAR;1
PFL.MAR;1             PSECT1.MAR;2        PSECT2.MAR;1        PSECT3.MAR;2
TAKE_PRIV.MAR;8       TEST.MAR;2          VCC.MAR;1

Total of 19 files.
$
```

Interactive DCL Example

```
$ type int_dcl.c
/* Sample LIB$SPAWN */
#include     <lib$routines.h>
#include     <ssdef>
#include     <stdio.h>
#include     <starlet.h>
#define sys_stat(STS) if(!((STS)&1)) sys$exit(STS)
int main(void)
{
     int status;
     printf("###Going to dcl###\n");
     status = lib$spawn();
     sys_stat(status);
     printf("###Back from DCL###\n");
     return(SS$_NORMAL);
}
$ cc int_dcl
$ link int_dcl
$
$ run int_dcl
###Going to dcl###
$ show proc
21-JUL-1996 04:48:27.59   User: ALLEN          Process ID:   00000068
                          Node: ALLEN          Process name: "ALLEN_1"

Terminal:
User Identifier:    [ALLEN]
Base priority:      4
Default file spec:  _DKA300:[ALLEN]
Number of Kthreads: 1
```

Interactive DCL Example (continued)

```
$ showtime
  21-JUL-1996 04:48:30
$ lo
  Process ALLEN_1 logged out at 21-JUL-1996 04:48:31.57
###Back from DCL###
$
```

Getting Job/Process Information

You may want to obtain information on your process or another process on the system. To do so you can use either **SYS$GETJPI** or **LIB$GETJPI.**

- The **SYS$GETJPI** system service is an asynchronous system service if you get information on a process other than your own. The system service has an implicit wait form using **SYS$GETJPIW.**

SYS$GETJPI format

```
SYS$GETJPI[W](efn, pidadr, prcnam, itmlst, iosb, astadr,
astprm)
```

SYS$GETJPI argument notes

Argument	Meaning
efn, iosb, astadr, and astprm	Allow for synchronization
pidadr, prcnam	Target process' identification. pid of -1 allows for wildcard, reuse until SS$_NOMOREPROC returned. If neither specified, get info on self.
itmlst	List of items you want to get info on. Item lists are used by a variety of system services. SYS$GETJPI uses an item list 3 format. Item list formats are defined in $ILEDEF/iledef.h. Item codes are defined in **JPIDEF/jpidef.h.** Sample item codes: **JPI$_BIOCNT, JPI$_IMAGNAME.**

Item list format

31

Item code	Buffer Length
Buffer address	
Return length	

- SYS$GETJPI requires GROUP or WORLD privilege to get information on a process other than your own.
- **SYS$PROCESS_SCAN** may be called prior to **SYS$GETJPI** to establish a list of criteria to either restrict the search to specific process metrics, like usernames, scheduling states, etc, or expand the search across nodes in a VMScluster. **SYS$PROCESS_SCAN** receives a pidctx and item list 3 argument. The item list identifies the criteria for search. The pidctx is passed to **SYS$GETJPI** in place of the pidadr argument.
- **LIB$GETJPI** is probably easier to use, but you can only obtain 1 piece of information per call.

Create Process Example

$ **type create_proc.c**

```
/* Example of creating a process */
#include <jpidef.h>
#include <iledef.h>
#include <descrip.h>
#include <ssdef.h>
#include <starlet.h>
#include <stdio.h>
#include <pqldef.h>
#define LIST_END 0
#define TERM_ARG 0
#define PID_ARG 1
#define TERM_MAX 15
#define BASE_PRI 4
#define sys_stat(STS) if(!((STS)&1)) sys$exit(STS)
typedef struct pqls
{
    char    item;
    int     value;
} pqls;
int main(void)
{
    $DESCRIPTOR(image,"child");
    $DESCRIPTOR(prcnam,"max");
    long    pid;
    long    my_pid;
    char    terminal[TERM_MAX+1];
    struct dsc$descriptor_s term_d = {TERM_MAX,DSC$K_DTYPE_T,
                                      DSC$K_CLASS_S};
```

Create Process Example (continued)

```
    ile3 jpi_list[] =
            {{sizeof(terminal),JPI$_TERMINAL,0,0},
            {sizeof(my_pid),JPI$_PID},
             {0,LIST_END}};
    int     status;
    pqls    pql_list[] = {{PQL$_WSDEFAULT,1024},
                  {PQL$_WSQUOTA,1024},
                  {PQL$_WSEXTENT,2048},
                  {PQL$_LISTEND}};
/* Fill in the JPI list to get terminal name */
    jpi_list[TERM_ARG].ile3$ps_bufaddr = terminal;
    jpi_list[TERM_ARG].ile3$ps_retlen_addr = &term_d.dsc$w_length;
    jpi_list[PID_ARG].ile3$ps_bufaddr = &my_pid;
/* Init the terminal name descriptor. */
    term_d.dsc$a_pointer = terminal;

/* Get terminal name. */
    status = sys$getjpiw(0,0,0,jpi_list,0,0,0);
    sys_stat(status);
/* Create child. */
    status = sys$creprc(&pid,&image,0,&term_d,&term_d,0,pql_list,
        &prcnam,BASE_PRI,0,0,0);
    printf("Child pid: %08x\nMy pid: %08x\n",pid,my_pid);
    sys_stat(status);
    return(SS$_NORMAL);
}
```

```
$ type child.c
#include <stdio.h>
#include <ssdef.h>
#include <lib$routines.h>
```

Create Process Example (continued)

```
int main(void)
{
    float   stall = 200.0;
    printf("Max says: HI!!!\n");
    fprintf(stderr,"Error check\n");
    lib$wait(&stall);
    return(SS$_NORMAL);
}
$ cc create_proc
$ link create_proc
$ cc child
$ link child
$
$ run create_proc
Max says: HI!!!
Error check
Child pid: 00000060
My pid: 0000005f
$ show sys/sub
OpenVMS V7.0 on node ALLEN  21-JUL-1996 04:06:35.55  Uptime  0 05:48:42
  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
00000060 max             HIB      4        6   0 00:00:00.10        75     79  S
$ write sys$output f$getjpi("60","dfwscnt")
1600
$ write sys$output f$getjpi("60","wsquota")
2112
$ write sys$output f$getjpi("60","wsextent")
16384
$
```

Create Process Example (continued)

```
$ mcr sysgen
SYSGEN>  SHOW PQL_MWSDEFAULT
Parameter Name          Current    Default     Min.      Max.    Unit  Dynamic
--------------          -------    -------    -------   -------  ----  -------
PQL_MWSDEFAULT             1600        512        -1        -1 Pagelets
 internal value             100         32        32        -1 Pages
SYSGEN>  SHOW PQL_MWSQUOTA
Parameter Name          Current    Default     Min.      Max.    Unit  Dynamic
--------------          -------    -------    -------   -------  ----  -------
PQL_MWSQUOTA               2112       1024        -1        -1 Pagelets   D
 internal value             132         64        64        -1 Pages      D
SYSGEN>  SHOW PQL_MWSEXTENT
Parameter Name          Current    Default     Min.      Max.    Unit  Dynamic
--------------          -------    -------    -------   -------  ----  -------
PQL_MWSEXTENT             16384       2048        -1        -1 Pagelets   D
 internal value            1024        128       128        -1 Pages      D
SYSGEN>  EXIT
$
$
```

Wildcard GETJPI Example

```
$
$ type swapped.c
/* Example of wildcarding with SYS$GETJPI.
   Note: this method is dangerous for performance
   since it obtains the image name a process is
   running, which is stored in P1 space.  Getting
   this information will cause an outswapped process
   to be inswapped.
*/
#include <jpidef.h>
#include <iledef.h>
#include <descrip.h>
#include <ssdef.h>
#include <starlet.h>
#include <stdio.h>
#define LIST_END 0
#define PROCESS_MAX 15
#define IMAGE_MAX 255
#define sys_stat(STS) if(!((STS)&1)) sys$exit(STS)
#define PID_ITEM 0
#define PROCESS_ITEM 1
#define IMAGE_ITEM 2
```

Wildcard GETJPI Example (continued)

```
int main(void)
{
    char    process[PROCESS_MAX+1];
    unsigned short process_len;
    char    image[IMAGE_MAX+1];
    unsigned short image_len;
    long    pid;
    ile3 jpi_list[] =
            {{sizeof(pid),JPI$_PID,0,0},
             {PROCESS_MAX,JPI$_PRCNAM},
             {IMAGE_MAX,JPI$_IMAGNAME},
             {0,LIST_END}};
 /* Wildcard process id. */
    long    pidctx = -1;
    int     status;
 /* Fill in the JPI list to get pid, process name, and image name */
    jpi_list[PID_ITEM].ile3$ps_bufaddr = &pid;
    jpi_list[PROCESS_ITEM].ile3$ps_bufaddr = process;
    jpi_list[PROCESS_ITEM].ile3$ps_retlen_addr = &process_len;
    jpi_list[IMAGE_ITEM].ile3$ps_bufaddr = image;
    jpi_list[IMAGE_ITEM].ile3$ps_retlen_addr = &image_len;
 /* Get info until out of processes. */
    while((status = sys$getjpiw(0,&pidctx,0,jpi_list,0,0,0))
       != SS$_NOMOREPROC)
    {
      sys_stat(status);
     /* Make string a C string. */
      process[process_len] = '\0';
      image[image_len] = '\0';
     /* Spill info. */
```

Wildcard GETJPI Example (continued)

```
        printf("Process: %-15s  PID: %08x\n  Image: %s\n\n",
                process,pid,image);
    }
    return(SS$_NORMAL);
}
$ cc swapped.c
$ link swapped
$ show sys
OpenVMS V7.0  on node ALLEN  21-JUL-1996 04:08:53.19  Uptime  0 05:50:59
  Pid     Process Name    State  Pri      I/O       CPU         Page flts  Pages
00000041 SWAPPER         HIB     16        0    0 00:00:02.21          0      0
00000045 IPCACP          HIB     10       11    0 00:00:00.09        247     18
00000046 ERRFMT          HIB      8      208    0 00:00:00.54         73     30
00000047 OPCOM           HIB      7       79    0 00:00:00.75        138     41
00000048 AUDIT_SERVER    HIB      9       85    0 00:00:00.31        255     73
00000049 JOB_CONTROL     HIB     10       66    0 00:00:00.14        230     29
0000004A SECURITY_SERVER HIB     10       46    0 00:01:17.18       2906    104 M
0000004C DECW$SERVER_0   HIB      8     1434    0 00:03:03.66       1831    178
0000004D DECW$SESSION    HIB      6      358    0 00:00:04.73       2923     51 M
0000004F DECW$MWM        LEFO     4       --  swapped  out  --                32
00000050 ALLEN           LEFO     4       --  swapped  out  --                30
00000051 DECW$TE_0051    COM      4      400    0 00:03:39.74       4178    276 M
00000052 _FTA3:          HIB      7    94998    0 00:07:53.00      57214     60
0000005B NETACP          HIBO     8       --  swapped  out  --                32
0000005C EVL             HIB      6       51    0 00:00:00.17        114     40 N
0000005D REMACP          HIBO     9       --  swapped  out  --                21
0000005F _RTA1:          CUR      4     2098    0 00:00:21.53       3254    123
00000060 max             HIBO     4       --  swapped  out  --                21  S
$ r swapped
%SYSTEM-F-NOPRIV, insufficient privilege or object protection violation
$ set proc/priv=world
```

Wildcard GETJPI Example (continued)

```
$ rswapped
Process: SWAPPER          PID: 00000041
  Image:

Process: IPCACP           PID: 00000045
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]IPCACP.EXE;1

Process: ERRFMT           PID: 00000046
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]ERRFMT.EXE;1

Process: OPCOM            PID: 00000047
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]OPCOM.EXE

Process: AUDIT_SERVER     PID: 00000048
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]AUDIT_SERVER.EXE;1

Process: JOB_CONTROL      PID: 00000049
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]JBC$JOB_CONTROL.EXE;1

Process: SECURITY_SERVER  PID: 0000004a
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]SECURITY_SERVER.EXE;1

Process: DECW$SERVER_0    PID: 0000004c
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]DECW$SERVER_MAIN.EXE;1

Process: DECW$SESSION     PID: 0000004d
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]DECW$SESSION.EXE

Process: DECW$MWM         PID: 0000004f
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]DECW$MWM.EXE;1

Process: ALLEN            PID: 00000050
  Image:
```

Wildcard GETJPI Example (continued)

```
Process: DECW$TE_0051      PID: 00000051
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]DECW$TERMINAL.EXE

Process: _FTA3:            PID: 00000052
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]RTPAD.EXE

Process: NETACP            PID: 0000005b
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]NETACP.EXE;1

Process: EVL               PID: 0000005c
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]EVL.EXE

Process: REMACP            PID: 0000005d
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]REMACP.EXE;1

Process: _RTA1:            PID: 0000005f
  Image: _DKA300:[ALLEN]SWAPPED.EXE;5

Process: max               PID: 00000060
  Image: _DKA300:[ALLEN]CHILD.EXE;4
```

Wildcard GETJPI Example (continued)

```
$ sh sys
OpenVMS V7.0  on node ALLEN  21-JUL-1996 04:09:23.68  Uptime  0 05:51:30
  Pid     Process Name    State  Pri      I/O       CPU         Page flts  Pages
00000041 SWAPPER         HIB     16        0    0 00:00:02.22          0      0
00000045 IPCACP          HIB     10       11    0 00:00:00.09        249     20
00000046 ERRFMT          HIB      8      208    0 00:00:00.54         73     30
00000047 OPCOM           HIB      7       79    0 00:00:00.75        139     42
00000048 AUDIT_SERVER    HIB      9       85    0 00:00:00.31        256     74
00000049 JOB_CONTROL     HIB     10       66    0 00:00:00.14        230     29
0000004A SECURITY_SERVER HIB     10       46    0 00:01:17.31       2934    107 M
0000004C DECW$SERVER_0   HIB      6     1434    0 00:03:05.70       1832    179
0000004D DECW$SESSION    HIB      6      358    0 00:00:04.73       2941     76 M
0000004F DECW$MWM        LEF      4      150    0 00:00:02.36        763     32
00000050 ALLEN           LEF      4      154    0 00:00:01.02        488     30
00000051 DECW$TE_0051    COM      4      400    0 00:03:40.55       4178    276 M
00000052 _FTA3:          HIB      7    95126    0 00:07:53.06      57222     52
0000005B NETACP          HIB      8       54    0 00:00:00.22         62     35
0000005C EVL             HIB      6       51    0 00:00:00.17        114     40  N
0000005D REMACP          HIB      9       20    0 00:00:00.02         35     23
0000005F _RTA1:          CUR      4     2229    0 00:00:21.97       3503    109
00000060 max             HIB      4        6    0 00:00:00.10         77     19  S
$
```

Process SCAN Example

```
$ type not_swapped.c
/* Example of using SYS$PROCESS_SCAN to limit search of
   processes to those that are resident.
*/
#include <jpidef.h>
#include <iledef.h>
#include <pscandef.h>
/* Note: Need to compile against SYS$LIB_C for this. */
#include <pcbdef.h>
#include <descrip.h>
#include <ssdef.h>
#include <starlet.h>
#include <stdio.h>
#define LIST_END 0
#define PROCESS_MAX 15
#define IMAGE_MAX 255
#define sys_stat(STS) if(!((STS)&1)) sys$exit(STS)
#define PID_ITEM 0
#define PROCESS_ITEM 1
#define IMAGE_ITEM 2
/* SYS$PROCESS_CONTEXT uses a variation of ile3 which
   is not compatible with the standard, since the 3rd
   and 4th arguments are not pointers.
*/
typedef struct _imed_ile3
{
    unsigned short int ile3$w_length;
    unsigned short int ile3$w_code;
    unsigned int ile3$l_value;
    unsigned int ile3$l_specific;
} imed_ile3;
#define STS_ITEM  0
```

Process SCAN Example (continued)

```
int main(void)
{
/* Match any process whose resident bit is set. */
    imed_ile3 pscan_list[]=
      {{0,PSCAN$_STS,PCB$M_RES,PSCAN$M_BIT_ANY},{0,LIST_END}};
    char     process[PROCESS_MAX+1];
    unsigned short process_len;
    char     image[IMAGE_MAX+1];
    unsigned short image_len;
    long     pid;
    ile3 jpi_list[] =
            {{sizeof(pid),JPI$_PID,0,0},
             {PROCESS_MAX,JPI$_PRCNAM},
             {IMAGE_MAX,JPI$_IMAGNAME},
             {0,LIST_END}};
    long     pidctx;
    int      status;

/* Fill in the JPI list to get pid, process name, and image name. */
       jpi_list[PID_ITEM].ile3$ps_bufaddr = &pid;
       jpi_list[PROCESS_ITEM].ile3$ps_bufaddr = process;
       jpi_list[PROCESS_ITEM].ile3$ps_retlen_addr = &process_len;
       jpi_list[IMAGE_ITEM].ile3$ps_bufaddr = image;
       jpi_list[IMAGE_ITEM].ile3$ps_retlen_addr = &image_len;

/* Set search criterion to be all resident processes. */
     status = sys$process_scan(&pidctx, pscan_list);
     sys_stat(status);
```

Process SCAN Example (continued)

```
/* Get info until out of processes. */
    while((status = sys$getjpiw(0,&pidctx,0,jpi_list,0,0,0))
       != SS$_NOMOREPROC)
    {
      sys_stat(status);
      process[process_len] = '\0';
      image[image_len] = '\0';
      printf("Process: %-15s  PID: %08x\n  Image: %s\n\n",
            process,pid,image);
    }
    return(SS$_NORMAL);
}
$
```

Process SCAN Example (continued)

```
$ cc not_swapped+sys$library:sys$lib_c/lib
$ link not_swapped
$ sh sys
OpenVMS V7.0  on node ALLEN  21-JUL-1996 04:12:20.58  Uptime  0 05:54:27
  Pid    Process Name    State  Pri      I/O       CPU         Page flts  Pages
00000041 SWAPPER         HIB     16        0    0 00:00:02.30         0      0
00000045 IPCACP          HIB     10       11    0 00:00:00.09       249     18
00000046 ERRFMT          HIBO     8       --  swapped  out  --              30
00000047 OPCOM           HIBO     7       --  swapped  out  --              32
00000048 AUDIT_SERVER    HIB      9       85    0 00:00:00.31       256     74
00000049 JOB_CONTROL     HIB     10       71    0 00:00:00.16       250     25
0000004A SECURITY_SERVER HIB     10       46    0 00:01:18.02      2990    114 M
0000004C DECW$SERVER_0   HIB      8     1434    0 00:03:09.92      1862    180
0000004D DECW$SESSION    HIB      6      358    0 00:00:04.75      2971     71 M
0000004F DECW$MWM        LEFO     4       --  swapped  out  --              32
00000050 ALLEN           LEF      4      154    0 00:00:01.02       488     30
00000051 DECW$TE_0051    COM      4      400    0 00:03:43.17      4178    276 M
00000052 _FTA3:          HIB      7    95422    0 00:07:53.21     57229     46
0000005B NETACP          HIB      8       54    0 00:00:00.23        67     47
0000005C EVL             HIB      6       51    0 00:00:00.17       114     30  N
0000005D REMACP          HIBO     9       --  swapped  out  --              23
0000005F _RTA1:          CUR      4     3440    0 00:00:32.93      6447    118
$ r not_swapped
Process: SWAPPER          PID: 00000041
  Image:

Process: IPCACP           PID: 00000045
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]IPCACP.EXE;1

Process: AUDIT_SERVER     PID: 00000048
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]AUDIT_SERVER.EXE;1
```

Process SCAN Example (continued)

```
Process: JOB_CONTROL      PID: 00000049
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]JBC$JOB_CONTROL.EXE;1

Process: SECURITY_SERVER  PID: 0000004a
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]SECURITY_SERVER.EXE;1

Process: DECW$SERVER_0    PID: 0000004c
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]DECW$SERVER_MAIN.EXE;1

Process: DECW$SESSION     PID: 0000004d
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]DECW$SESSION.EXE

Process: ALLEN            PID: 00000050
  Image:

Process: DECW$TE_0051     PID: 00000051
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]DECW$TERMINAL.EXE

Process: _FTA3:           PID: 00000052
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]RTPAD.EXE

Process: NETACP           PID: 0000005b
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]NETACP.EXE;1

Process: EVL              PID: 0000005c
  Image: ALLEN$DKA300:[SYS0.SYSCOMMON.][SYSEXE]EVL.EXE

Process: _RTA1:           PID: 0000005f
  Image: _DKA300:[ALLEN]NOT_SWAPPED.EXE;15

$
```

Setting Process Characteristics

Several process characteristics may be modified, through system service calls.

Changing default directory

```
SYS$SETDDIR(new_dir,len,cur_dir)
```

Changing default file protection

```
SYS$SETDFPROT(new_def_prot,cur_def_prot)
```

Changing base priority

(Requires ALTPRI privilege to raise above authorized value)

```
SYS$SETPRI(pidadr,prcnam,pri,prvpri,nullarg,nullarg)
```

Changing process name

```
SYS$SETPRN(prcnam)
```

Changing privileges

(Requires SETPRV privilege to turn on)

```
SYS$SETPRV(enblflg,prvadr,prmflg,prvprv)
```

Changing resource wait mode

(for quota exhaustion)

```
SYS$SETRWM(waitflg)
```

Process Management/Control

- Process execution can be suspended (not a synchronization choice) and resumed using:

```
SYS$SUSPND(pidadr,prcnam,flags)
```

flags include: bit 0, if set kernel and executive mode Asynchronous System Traps (ASTs) are blocked, else only supervisor/user mode ASTs are blocked.

```
SYS$RESUME(pidadr,prcnam)
```

If resume issued prior to suspend, no suspend.

- Processes may synchronize with one another using a hibernation/wake combination. This method of synchronization requires coordination between the cooperating processes.

Note

ASTs are deliverable to a hibernating process.

- A Process may hibernate through:

```
SYS$HIBER() or use of PRC$M_HIBER flag on SYS$CREPRC
```

- A Process may be wakened through:

```
SYS$WAKE(pidadr,prcnam)
```

- or may schedule a wake up through:

```
SYS$SCHDWK(pidadr,prcnam,daytim,reptim)
```

Hiber/Wake Example

$ **type create_proc2.c**

```
/* Example of using hibernation to coordinate process
   execution.

*/
#include <jpidef.h>
#include <iledef.h>
#include <prcdef.h>
#include <descrip.h>
#include <ssdef.h>
#include <starlet.h>
#include <stdio.h>
#include <pqldef.h>
#define LIST_END 0
#define TERM_ARG 0
#define PID_ARG 1
#define TERM_MAX 15
#define BASE_PRI 4
#define sys_stat(STS) if(!((STS)&1)) sys$exit(STS)
typedef struct pqls
{
    char    item;
    int     value;
} pqls;
```

Hiber/Wake Example (continued)

```
int main(void)
{
    $DESCRIPTOR(image,"sleepy_child");
    $DESCRIPTOR(prcnam,"max");
    long    pid;
    long    my_pid;
    char    terminal[TERM_MAX+1];
    struct dsc$descriptor_s term_d = {TERM_MAX,DSC$K_DTYPE_T,DSC$K_CLASS_S};
    ile3 jpi_list[] =
            {{sizeof(terminal),JPI$_TERMINAL,0,0},
            {sizeof(my_pid),JPI$_PID},
             {0,LIST_END}};
    int     status;
    pqls    pql_list[] = {{PQL$_WSDEFAULT,1024},{PQL$_WSQUOTA,1024},
                {PQL$_WSEXTENT,2048},{PQL$_LISTEND}};

/* Fill in the JPI list to get terminal name */
    jpi_list[TERM_ARG].ile3$ps_bufaddr = terminal;
    jpi_list[TERM_ARG].ile3$ps_retlen_addr = &term_d.dsc$w_length;
    jpi_list[PID_ARG].ile3$ps_bufaddr = &my_pid;
/* Init the terminal name descriptor. */
    term_d.dsc$a_pointer = terminal;
/* Get terminal name. */
    status = sys$getjpiw(0,0,0,jpi_list,0,0,0);
    sys_stat(status);
/* Create child. */
    status = sys$creprc(&pid,&image,0,&term_d,&term_d,0,pql_list,
        &prcnam,BASE_PRI,0,0,0,0);
    sys_stat(status);
    printf("Child pid: %08x\nMy pid: %08x\n",pid,my_pid);
    printf("Waking max and going to sleep!\n");
```

Hiber/Wake Example (continued)

```
/* Wait for child to finish up. */
    status = sys$wake(&pid,0);
    sys_stat(status);
    sys$hiber();
    printf("Ahhh, I feel refreshed now!\n");
    return(SS$_NORMAL);
}
$
$ type sleepy_child.c
/* Example of child using wake system service
   to wake creator.*/
#include <stdio.h>
#include <ssdef.h>
#include <jpidef.h>
#include <lib$routines.h>
#include <starlet.h>
#define sys_stat(STS) if(!((STS)&1)) sys$exit(STS)
int main(void)
{
    float    stall = 10.0;
    int      status;
    unsigned int   dads_pid;
    int      owner = JPI$_OWNER;

/* Go to sleep. */
    sys$hiber();
/* Determine creator. */
    status = lib$getjpi(&owner,0,0,&dads_pid);
    sys_stat(status);
```

Hiber/Wake Example (continued)

```
    printf("\tMax says: HI!!!\n");
    printf("\tLet dad sleep for 10 seconds.\n");
    status = lib$wait(&stall);
    sys_stat(status);
    printf("\tWaking dad!\n");
/* Wake creator. */
    status = sys$wake(&dads_pid,0);
    sys_stat(status);
    sys$exit(SS$_NORMAL);
}
$
```

Hiber/Wake Example (continued)

```
$ cc create_proc2
$ link create_proc2
$
$ cc sleepy_child
$ link sleepy_child
$
$ run create_proc2
Child pid: 0000005d
My pid: 0000005c
Waking max and going to sleep!
     Max says: HI!!!
     Let dad sleep for 10 seconds.
     Waking dad!
Ahhh, I feel refreshed now!
$
$ run create_proc2
Child pid: 0000005e
My pid: 0000005c
Waking max and going to sleep!
     Max says: HI!!!
     Let dad sleep for 10 seconds.
     Waking dad!
Ahhh, I feel refreshed now!
$
```

Terminating Images/Processes

Images may be rundown using the **SYS$FORCEX** system service. If DCL is mapped control will be returned, otherwise the process will normally be terminated.

- **SYS$FORCEX** will cause the **SYS$EXIT** system service to be called on behalf of the target process. If the image has declared exit handlers they will be called, allowing the process to perform any required *cleanup* operations.

SYS$FORCEX format

```
SYS$FORCEX(pidadr, prcnam, exit_code)
```

- Processes may be terminated by calling the **SYS$DELPRC** system service. This is the effect of the DCL **$STOP/IDENTIFICATION** or **$STOP *prcnam*** command.
- Exit handlers are not called when a process is deleted.

SYS$DELPRC format

```
SYS$DELPRC(pidadr,prcnam)
```

Exit Handlers

Exit handlers may be declared to perform *cleanup* operations for a process. To declare an exit handler call the **SYS$DCLEXH** system service.

- Exit handlers are called from outer access modes (user) to inner (executive) in reverse order of declaration.

SYS$DCLEXH format

```
SYS$DCLEXH(desblk)
```

Descriptor block (*desblk*) format

Forward link (for OpenVMS to fill in)	
Exit handler address	
mbz	Arg count
Address of condition value (for Open VMS to write)	
Additional arguments . . .	

- Exit handlers may be cancelled using the **SYS$CANEXH** system service. **SYS$CANEXH** has the same argument as **SYS$DCLEXH**.

Exit Handler Example

```
$ type handle_with_care.c
/*  Example of exit handler usage.*/
#include <starlet.h>
#include <stdio.h>
#include <ssdef.h>
#include <descrip.h>
#include <lib$routines.h>
#include <errno.h>
#define sys_stat(STS) if(!((STS)&1)) sys$exit(STS)
#define FILL_SIZE 3
/* Descriptor block for sys$dclexh with no extra args. */
typedef struct _desblkdef
{
    void    *flink;
    int     (*func)();
    int     argc;
    int     *sts;
}   base_desblk;
/* Extended desblk supporting a file pointer pointer arg. */
typedef struct     extended_desblk
{
    base_desblk    bd;
    FILE    **fp;
} h2_desblk;
/* Exit handler functions. */
int handler1(int *sts);
int handler2(int   *sts,FILE **fp);
```

Exit Handler Example (continued)

```
int main(void)
{
    int     status;
    float   stall = 60.0;
/* NOTE: All of these variables MUST be static to be
   available for the exit handlers. */
    static base_desblk   h1_desb;
    static h2_desblk     h2_desb = {{0,0,{2}}};
    static int     h1_sts;
    static int     h2_sts;
    static FILE    *file;

/* Initialize first exit handler descriptor. */
    h1_desb.func = handler1;
    h1_desb.sts = &h1_sts;
    h1_desb.argc = 1;
/* Initialize second exit handler descriptor. */
    h2_desb.bd.func = handler2;
    h2_desb.bd.sts = &h2_sts;
    h2_desb.fp = &file;
/* Declare exit handlers. */
    status =sys$dclexh(&h1_desb);
    sys_stat(status);
    status =sys$dclexh(&h2_desb);
    sys_stat(status);
/* Open a file. */
    file = fopen("junk.dat","w");
    if(!file)
    {
      fprintf(stderr,"Cannot open junk.dat\n");
      return(errno);
    }
```

Exit Handler Example (continued)

```
/* Write a couple of records and stall. */
    fprintf(file,"Record 1\n");
    fprintf(file,"Record 2\n");
    lib$wait(&stall);
    fprintf(file,"Record 3\n");
    return(SS$_NORMAL);
}
/* First exit handler just displays exit status. */
#define MAX_MSG 255
int  handler1(int *sts)
{
    char    msg_a[MAX_MSG];
    char    *cptr = msg_a;
    struct dsc$descriptor_s msg = {sizeof(msg_a),DSC$K_DTYPE_T,
                        DSC$K_CLASS_S};
    int     status;


    msg.dsc$a_pointer = msg_a;
    printf("In handler 1\nReason for exit:\n");
/* Translate message number into string. */
    status = sys$getmsg(*sts,&msg.dsc$w_length,&msg,0,0);
    sys_stat(status);
/* Truncate before text. */
    msg.dsc$w_length = 0;
    while(*cptr++ != ',') msg.dsc$w_length++;
/* Display message. */
    lib$put_output(&msg);
    return(SS$_NORMAL);
}
```

Exit Handler Example (continued)

```
/* Second handler displays message in file
   if not normal completion.*/
#define MAX_NAME 255
int handler2(int   *sts,FILE **fp)
{
     printf("In handler2\n");
     if(*sts != SS$_NORMAL)
     {
       fprintf(*fp,"Junk.dat closed abnormally\n");
     }
     return(SS$_NORMAL);
}
$ rhandle_with_care
^Y
$ exit
In handler2
In handler 1
Reason for exit:
SYSTEM-W-CLIFRCEXT
$ type junk.dat
Record 1
Record 2
Junk.dat closed abnormally
$ rhandle_with_care
^Y
$ stop
$ type junk.dat
Record 1
Record 2
$
```

Force Exit Example

```
$ type kill.c
/* Simple force exit example. */
#include <stdio.h>
#include <ssdef.h>
#include <starlet.h>
int main(void)
{
    int     pid;
    int     status;

/* Get pid, burden is on user to include leading 0x. */
    printf("Enter pid to kill (0xpid):");
    scanf("%i",&pid);
/* Kill the process. */
    status = sys$forcex(&pid,0,SS$_VETO);
    return(status);
}
$
$ cc kill
$ link kill
$ run/process=to_be_killed/output="f$getjpi("","TERMINAL")-
_$ handle_with_care
%RUN-S-PROC_ID, identification of created process is 00000059
$ sh sys/sub
OpenVMS V7.0  on node ALLEN  21-JUL-1996 21:00:25.02  Uptime  0 04:38:03
  Pid    Process Name    State  Pri      I/O       CPU        Page flts  Pages
00000059 TO_BE_KILLED    HIB      5       13    0 00:00:00.09         70     79 S
```

Force Exit Example (continued)

```
$
$ rkill
Enter pid to kill (0xpid):0x59
In handler2
In handler 1
Reason for exit:
%SYSTEM-F-VETO
$
$ show sys/sub
$
$ type junk.dat
Record 1
Record 2
Junk.dat closed abnormally
$
```

Delete Process Example

```
$ run/process=to_be_killed/output='f$getjpi("","TERMINAL") handle_with_care
%RUN-S-PROC_ID, identification of created process is 0000005A
$ type my_stop.c
/* Simple delete process example. */
#include <stdio.h>
#include <ssdef.h>
#include <starlet.h>
int main(void)
{
    int     pid;
    int     status;

/* Get pid, burden is on user to include leading 0x. */
    printf("Enter pid to stop (0xpid):");
    scanf("%i",&pid);
/* Kill the process. */
    status = sys$delprc(&pid,0);
    return(status);
}
$ cc my_stop
$ link my_stop
$ show sys/sub
OpenVMS V7.0  on node ALLEN  21-JUL-1996 21:01:27.29  Uptime  0 04:39:05
  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
0000005A TO_BE_KILLED    HIBO     5       --  swapped  out  --             23  S
$ r my_stop
Enter pid to stop (0xpid):0x5a
$ show sys/sub
$ type junk.dat
Record 1
Record 2
$
```