Loadable Executive Images

DECUS Spring ’93 Symposia
VS059

David L. Cathey
Montagar Software Concepts
P. O. Box 260772
Plano, TX 75026-0772
(214) 618-2117
davidc@montagar.com

Session Outline

- Introduction
- Comparison: Loadable Images vs. Drivers
- Loadable Executive Images during SYSBOOT and SYSINIT
- Executive Image Layout
- Building and Executive Loadable Image
- Installing an Executive Loadable Image
- Oops... How to recover from that little bug in your code that keeps your system from booting...

Introduction

- With the exception of the primitive boot images, most of OpenVMS since V5.0 is composed of Loadable Executives.
- Pre-V5.0, SYS.EXE comprised the major portion of the O/S, and other portions (SYSLOAxxx, RMS, F11BXQP, etc) provided their own loader. This was cumbersome.
- Generic executive loader was created (LDR$LOAD_IMAGE) to provide standard loader, and enabled modularizing the OpenVMS Executive. This made fixing pieces of OpenVMS easy without requiring rebuilding the entire SYS image.

Introduction (cont)

- Also reduces the need for "allocate pool", "copy into pool", "fix everything up" technique, re-inventing this wheel for every program requiring code loaded into OpenVMS. This also breaks for OpenVMS/AXP

- New features added, like flexiblility of System Services, paged and non-paged sections, etc.

Loadable Images vs. Drivers

Loadable Images	Drivers
Loaded by SYSBOOT/SYSINIT	Done in startup via SYSGEN
Becomes an actual port of OpenVMS, within any section of the OpenVMS Executive. Usually "invisible".	generally, only effects the I/O Subsystem, and seen as a device on the system.
Code executable in any mode (but generally only accessed by kernel/executive).	Kernel only, in non-paged pool.
Loaded only at system boot.	Can be loaded any time, and if driver permits, reloaded to get a new version.
Appears as a shareable image. Position Independent Code only. Image layout not critical, but has limitations.	Normal image, with a base address of x00000000, PIC code, and PSECTs determining order of sections within image.
Simpler, no more than 6 parts comprise the image.	Complicated with several tables to define and build.

Loadable Executive Images in SYSBOOT/SYSINIT

- Loadable Executive Images are loaded during SYSBOOT, EXEC_INIT, and SYSINIT. For our purposes, we are only interested in SYSINIT.

- Loadable Executive Images are loaded using the LDR$LOAD_IMAGE routine. LDR$LOAD_IMAGE performs similar functions as SYS$IMGACT/SYS$IMGFIX does as port of normal image activation.

- LDR$LOAD_IMAGE takes two parameters:
  - Name of image (assumed to be in SYS$LOADABLE_IMAGES).
  - Flag indicating if pageable sections should be resident (S0PAGING SYSGEN parameter).

- LDR$LOAD_IMAGE performs the following functions:
  - Opens the image, either using the primitive file system, or during SYSINIT, will actually use RMS to open the image, and return using $CMKRNL.

Loadable Executive Images in SYSBOOT/SYSINIT

- LDR$LOAD_IMAGE functions (cont.)
  - Next, the image header is read and the image is checked for system version mismatches.
  - Builds a LDRIMG (Loadable Image Data Block) and initializes with the information from the image header
  - Allocates enough SPTE's to load the image.
  - Invokes LDR$LOAD_NONPAGED twice to load the read-only and read-write image NON-PAGED sections.
  - Invokes LDR$LOAD_PAGED twice to load RO and RW PAGED sections.

Loadable Executive Image in SYSBOOT/SYSINIT

- LDR$LOAD_IMAGE functions (cont.)
  - Invokes LDR$LOAD_NONPAGED twice again to load the initialization section and the address fixup section.
  - Using the GSD of the image, changes the vectored data cells on the image and the system base image.
  - Invokes the initialization routine (if any). If not in SYSINIT, this step is delayed...
  - Returns to caller.

Loadable Executive Images in SYSBOOT/SYSINIT

- If LOADABLE_SYS_IMAGES is set to 1, optional loadable images are able to be loaded. This is the default.

- This includes some of the DDIF, POSIX, ERAPAT, or Customer images to be loaded.

- LDR$ALTERNATE_LOAD is called within EXEC$INIT and SYSINIT.
  - Tests LOADABLE_SYS_IMAGES, return if zero.
  - Opens and reads SYS$LOADABLE_IMAGES:VMS$SYSTEM_IMAGES.DATA
  - If proper initialization phase, LDR$LOAD_IMAGE is called for each entry in the file.
  - When all images are processed, the file is closed and control returned to the caller.

Loadable Executive Image Layout

- A SHAREABLE image, with an entry point referencing the address of the initialization routine (DOINIT from STARLET.OLB).

- Contains up to 6 Image Section Descriptors (ISD's), no more than one each of the following:

  - PAGED Read-Only Section (ISD$V_RESIDENT=0, ISD$V_WRT=0)
  - PAGED Read-Write Section (ISD$V_RESIDENT=0, ISD$V_WRT=1)
  - NON-PAGED Read-Only Section (ISD$V_RESIDENT=1, ISD$V_WRT=0)
  - NON-PAGED Read-Only Section (ISD$V_RESIDENT=1, ISD$V_WRT=1)
  - Initialization Section (ISD$V_INITIALCODE=1)
  - Address Fixup Vectors (ISD$V_FIXUPVEC=1)

Loadable Executive Image Layout

- Loadable Image extra restrictions:
  - Position Independent Code - care must be taken with offset and usage of addresses. Use general addressing where appropriate (G^ or @#).
  - Be careful about what is in initialization code and what is used in the rest of the code. Initialization code is unmapped after initialization is complete!
  - Use the DECLARE_PSECT macro to declare image sections correctly for the LINKER to properly order and collect them.
  - Linked with the VECTOR_TABLE option to enable universal symbol resolution using the base image.
  - Use the LINK command and options file, or else system will probably BUGCHECK at boot time. See DOD_ERAPAT_LNK.COM.

Loadable Executive Images Layout

- All pages are mapped as User-read/Kernel-write or simply User-read/No-write. However, since page protection is set during the LDR$LOAD_XXX phase, this can be overridden by the initialization routine.

- Care must be taken with writing the initialization code, as it is executed in SYSINIT context (simple process), kernel mode, and via a JSB entry point.

Building A Loadable Executive Image

- Generally will have to be written in MACRO, however depending up the application, some portions may be written in higher level languages if proper care is taken in setting up the PSECT's, and collecting them at link time.

- Determine the environment of the code which will be loaded into the Executive:

  - Elevated IPL code must use Non-paged sections
  - Executive mode can used paged sections if desired.
  - As with any privileged code, care must be taken to avoid security breaches.
  - If replacing or using current symbols defined in VMS, they must be defined as a VECTORED symbols using one of:

    - SYSTEM_SERVICE - For a System Service (Such as SYS$ERAPAT)
    - UNIVERSAL_ENTRY - For a ".entry" routine entry point
    - UNIVERSAL_SYMBOL - For a JSB entry point or vectored data cell.

Building A Loadable Executive Image

- A basic program will look like:

```
DECLARE_PSECT            EXEC$INIT_CODE
INITIALIZATION_ROUTINE   KICKOFF
kickoff:
        ; do some interesting stuff...
        movl    #SS$_NORMAL,r0
        rsb

DECLARE_PSECT            EXEC$NONPAGED_CODE

routine_name:
        ; do some fun stuff here...
        rsb

DECLARE_PSECT            EXEC$NONPAGED_DATA

some_data: .long         0

        .end
```

Building a Loadable Executive Image

- Some sample code, SYS$EXAMPLES:DOD_ERAPAT.MAR reduced to its barest form:

```
;
;Include files:
;
        .LIBRARY /SYS$LIBRARY:LIB/      ; Automate inclusion of this macro
                                        ; library during the assembly.
        ; more include files and definitions...
        $SYSVECTORDEF                   ; Define system service vector offsets
;
; Own Storage:
;
        DECLARE_PSECT   EXEC$PAGED_DATA
COUNT_VECTOR:                           ; Iteration counts for medium types
        ; misc data cells...
        DECLARE_PSECT   EXEC$PAGED_CODE
        SYSTEM_SERVICE  ERAPAT,-        ; $ERAPAT entry point
                <R4>,MODE=KERNEL,NARG=3
        ; misc code...
EXIT:   MOVQ    (SP)+,R2                ; Restore R2/R3
        RET                             ; Return.
        .END
```

Building A Loadable Executive Image

- Once it compiles (Hooray!), it needs to be linked. Again, you can see SYS$EXAMPLES:DOD_ERAPAT_LNK.COM and save some time.

```
$ LINK/CONTIG /NOSYSSHR/NOTRACEBACK/SHARE=MY_STUFF              SYS$INPUT/OPTION
MY_STUFF,-
SYS$LIBRARY:STARLET/INCLUDE:(SYS$DOINIT),-
SYS$SYSTEM:SYS.STB/SELECTIVE,-
VECTOR_TABLE=SYS$SYSTEM:SYS.STB
COLLECT=NONPAGED_READONLY_PSECTS/ATTRIBUTES=RESIDENT,      EXEC$NONPAGED_CODE
COLLECT=NONPAGED_READWRITE_PSECTS/ATTRIBUTES=RESIDENT,     EXEC$NONPAGED_DATA
COLLECT=PAGED_READONLY_PSECTS,             EXEC$PAGED_CODE
COLLECT=PAGED_READWRITE_PSECTS,            EXEC$PAGED_DATA
COLLECT=INITIALIZATION_PSECTS/ATTRIBUTES=INITIALIZATION_CODE,-
                EXEC$INIT_CODE,-
                EXEC$INIT_000,-
                EXEC$INIT_001,-
                EXEC$INIT_002,-
                EXEC$INIT_PFNTBL_000,-
                EXEC$INIT_PFNTBL_001,-
                EXEC$INIT_PFNTBL_002,-
                EXEC$INIT_SSTBL_000,-
                EXEC$INIT_SSTBL_001,-
                EXEC$INIT_SSTBL_002
```

Building A Loadable Executive Image

- Allocating space within the image
  - Some work requires allocating work area for each "task" that cannot be shared with other "tasks". To allocate memory, use normal routines that would be used for system programming.

  - For System (S0) space:
    - EXE$ALONONPAGED
    - EXE$ALONPAGVAR
    - EXE$ALOPAGED
    - EXE$ALOPHYCNTG
  - For process (P1) space:
    - EXE$ALOP1PROC
    - RMS$GET_SPACE
  - Make sure to return unneeded pool space!

Installing an Executive Loadable Image

- Move the image to SYS$LOADABLE_IMAGES (SYS$COMMON:[SYS$LDR])
- Use SYSMAN to add the image to the loaded image database in SYS$UPDATE:VMS$SYSTEM_IMAGES.IDX:

```
$ MCR SYSMAN
SYSMAN> SYS_LOAD ADD _LOCAL_ MY_STUFF /LOAD=SYSINIT -
        /SEVERITY=WARNING /MESSAGE="It didn't work... Nuts..."
SYSMAN> EXIT
```

- From the system images, generate the data file that is read by LDR$ALTERNATE_LOAD:

```
$ @SYS$UPDATE:VMS$SYSTEM_IMAGES
```

- Insure that LOAD_SYS_IMAGES is set to "1"

```
$ MCR SYSGEN SHOW LOAD_SYS_IMAGES
Parameter Name         Current    Default     Min.
--------------         -------    -------    -------  ...
LOAD_SYS_IMAGES              1          1          0
```

Installing an Executive Loadable Image

- Reboot the system...

```
$ @SYS$SYSTEM:SHUTDOWN 0 SHUTDOWN YES NO LATER YES NONE
```

- Pray...

- Learn to read interesting crash dump, and maybe you might even get to use XDELTA!

Recovering From That Little Bug In Your Image...

- Reboot the system using conversation boot:

```
>>> B/1     ! or similar depending upon console type
        or
>>> boot -bflg 0,1  ! for AXP processors
```

- Once at SYSBOOT, then disable the alternate loadable images:

```
SYSBOOT> SET LOAD_SYS_IMAGES 0
SYSBOOT> CONTINUE
```

- Fix the image, or if it is not convenient to fix at this time, remove from the loadable image list:

```
$ MCR SYSMAN
SYSMAN> SYS_LOAD REMOVE _LOCAL_ MY_STUFF
SYSMAN> EXIT
$ @SYS$UPDATE:VMS$SYSTEM_IMAGES
```

Questions and Answers