A Shareable Image Library in the BEPC Control System

C. Wang J. Zhao
Institute of High Energy Physics, Chinese Academy of Sciences
P.O. Box 918-10, Beijing 100039, China

Abstract

The paper describes the method and procedure for developing the shareable image library within the BEPC control system.

1 Introduction

There are many application processes and object module libraries of the BEPC control system in the VAX/VMS environment. Each process must be linked to these object module libraries to create an executable image (see figure 1). When processing object modules, the linker has to resolve symbolic references, sort program sections into image sections and initialize the image section contents, total link processing time will increase , each process needs to have its own copy of these modules, so each process takes up more space in physical memory and more disk storage. If these object modules can be linked into one shareable image(see figure 2), the above mentioned problems can be avoided.

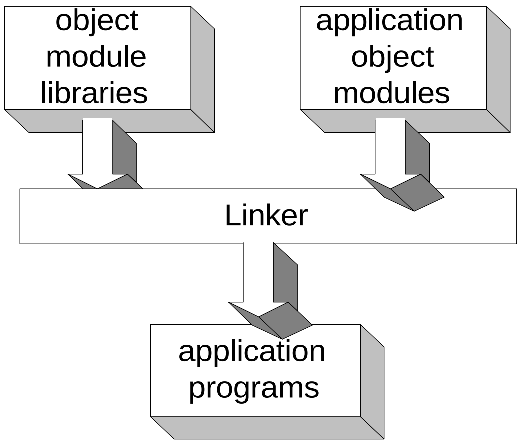

Figure 1. The executable images linked to the object module libraries.

2 A shareable image library

2.1 A shareable image

A shareable image is the product of a link operation. It is not directly executable, that is, it cannot be executed by means of the DCL command RUN. To execute, a shareable image must first be included as input in a link operation that produces an executable image of an application program. When that executable image is run, the shareable image is also activated by the image activator.

A shareable image file consists of an image header, one or more image sections, and a symbol table, which appears at the end of the file. The symbol table is, in fact, an object module whose records contain definitions of universal symbols in the shareable image. For a shareable image a universal symbol is what a global symbol is for a module, that is, it is a symbol that can be used to satisfy references in external modules.

Shareable images can provide the following benefits:

- Reduce total linking processing time.
- Avoid relinking entire applications.
- Conserve disk space.
- Conserve physical memory.
- Reduce I/O paging.
- Implement memory-resident database.

2.2 Method of creating a shareable image

There are two means of creating a shareable image:

- Declaring Universal Symbols.
- A transfer vector.

Here, we only discuss the second way for simplicity.

A transfer vector labels a visual register cell by a label, in which the next cell address of a visual register or a cell displacement is contained. This second cell is a pointer to the actual directive to receive control. That is to say, a transfer vector defines an entry point for each subroutine. The reasons for using a transfer vector are as follows:

- A shareable image may conveniently be modified and increased.
- The other processes linked against a shareable image need not be relinked.

A transfer vector must be written in MACRO. Specify the .TRANSFER directive to declare the symbol which is its argument, this symbol is a universal symbol by default. The instructions are as follows:

```
(1) .transfer  F00      ; Begin transfer vector to F00
(2) .mask    F00        ; Store register save mask
(3) jump   L^F00+2  ; Jump to routine
```

Whatever language the procedure is written in, the transfer vector has to be specified as above. To ensure upward compatibility, follow these guidelines when creating a transfer vector:

- Preserve the order and placement of entries in a transfer vector.
- Add new entries to the end of a transfer vector.

When including universal data in a transfer vector file, use padding to leave adequate room for future growth between the end of the transfer vector and the beginning of the list of universal data declarations.

A transfer vector is included in a link operation as any other object module, the linker will automatically use the transfer vector when specifying LINK/SHAREABLE to

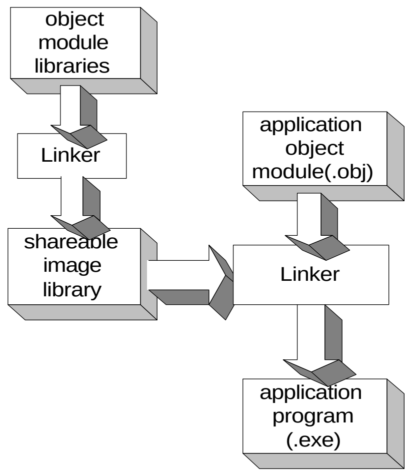

Figure 2. The executable images linked to the shareable image library.

create a shareable image. The linker will link the object modules with the transfer vectors and the object modules contained in the shareable image. However, to ensure upward compatibility, you must make sure that the transfer vector always appears in the same location of the image. The best way to accomplish this is to make the transfer vector always appear at the beginning of the image by forcing the linker to process it first. So, using the CLUSTER-option, put the transfer vector file in a named cluster, so that the transfer vector will appear at the beginning of the file. Besides, to enable images which are linked against a shareable image to run with various versions of the shareable image, you must specify the identification number of the image by using the GSMATCH-option to achieve it.

3 The shareable image library in the BEPC control system

The shareable image library is different from the object module library, in which the symbol table of the shareable image is placed. There are about 20 object module libraries in the BEPC control system, which refer to each other. Because all application programs must be linked to these libraries to create executable images, it takes a long time to link these programs. If one module is modified, all programs, which refer the module, must be relinked. To avoid these shortcomings, we create a transfer vector which defines an entry for each object module. Then, if you include the object module with the transfer vector in a named cluster, these object module libraries are still included in a link operation as input. When creating shareable images, we found that the attributes of some program sections had to be reset. If you do not reset the shareable attribute for program sections which are writable, you must INSTALL the shareable images to run the program. The shareable attribute [SHR] determines whether multiple processes have shared access to the memory. For example, if modules are data blocks, which are declared as EXTERNAL in the subroutine, we use the Linker ′PSECT-ATTR′ command to fix up various PSECT′s to make them shareable or read-only, etc. We create one shareable image library, which is called LIBSHR. We put shareable modules into this library. All programs only need to link to this library. When processing the shareable image library, the linker will look for the entire symbol table in the library. So, all programs only reserve the address of their referred modules. Each program takes up less space in physical memory and less disk storage. When applications are executed, referenced modules will be modules from the shareable library. Because the version number of the shareable image is never changed, all programs linked with the old version of the shareable image don′t need to be relinked when modifying or increasing modules of the library, but the shareable image needs to be relinked. When running these programs, the executable image with the old version of the shareable image can map to the new version of the shareable image.

Up to now, the shareable image library in the BEPC control system has been setup. It proved to be safe and reliable when running. Using the shareable image conserves disk space and reduces linker processing time. It provides convenience for maintenance and running.

Acknowledgment

The authors wish to take this opportunity to thank Dr. J. Smedinghoff, who gave us a lot of help. We would like to thank Prof. Huiying Luo and Wei Liu for their help.

References

[1] 〈Open VMS Linker Utility Manual〉.
[2] J.Smedinghoff, Fermilab Accelerator, Division Internal Report.