System Programming in Loaders

(Komal, Karuna Hazrati, Priyanka Mittal)

Abstract:

Linkers and loaders have been part of the software toolkit almost as long as there have been computers, since they are the critical tools that permit programs to be built from modules rather than as one big monolith.
Linking and Loading, provides a short historical overview of the linking process, and discusses the stages of the linking process to execute an object program, we needs **Relocation**, **Linking**, **Loading and Allocation**,

INTRODUCTION:

In the beginning of 1947, programmers started to use prehistoric loaders which could take program routines stored on detach tapes and combine and relocate them into a one single program. By the early 1960s, these loaders had evolved into full-fledged linkage editors. Since program memory remained expensive and limited and computers were slow, these linkers contained complex features for creating complex memory overlay structures to cram large programs into small memory, and for re-editing before linked programs to save the time needed to rebuild a program from scratch.
During the 1970s and 1980s there was a progress in linking technology. Linkers tended to become even simpler, as implicit memory moved much of the job of storage space management away from applications and overlays, into the operating system, and as computers became faster and disks became larger, it became simple to recreate a linked program from scratch to replace a few modules rather than to re-link just the changes. In the 1990s linkers have again become more complex, adding support for modern features including dynamically linked shared libraries and the unusual demands of C++. Radical new processor architectures with wide instruction words and compiler-scheduled memory accesses, such as the Intel IA64, will also put new demands on linkers to ensure that the complex requirements of the code are met in linked program.

LOADERS :

A loader is a system program that performs the loading function and also support relocation & linking . The other programs have a separate linker and loader. Basically, a **loader** is a part of an operating system which is responsible for loading programs and libraries for performing various execution. It is one of the basic essential stages in the course of starting a program because its basic purpose is to place programs into memory and then prepares them for further implementation. For Loading a program involves reading the contents of the implementable file containing the program instructions into memory, and then further carrying out other required introductory tasks to prepare the executable for running. Once the loading is done, then the operating system starts the program by passing control to the loaded program code. All operating systems that support program loading have loaders, apart from highly specialized computer systems that only have a fixed set of specialized programs. Rooted systems typically do not have loaders, and instead the code execute directly from ROM. In order to load the operating system itself, as part of executing, a particular execute loader is used. In various operating systems the loader is permanently resident in memory, although some operating systems that support virtual memory may allow the loader to be located in a region of memory that is page able.

Basic Functions of Loader:

Allocation: allocate space in memory for the programs

Linking: Resolve symbolic references between object files and combines two or more separate object programs. It also supplies the information needed to allow references between them.

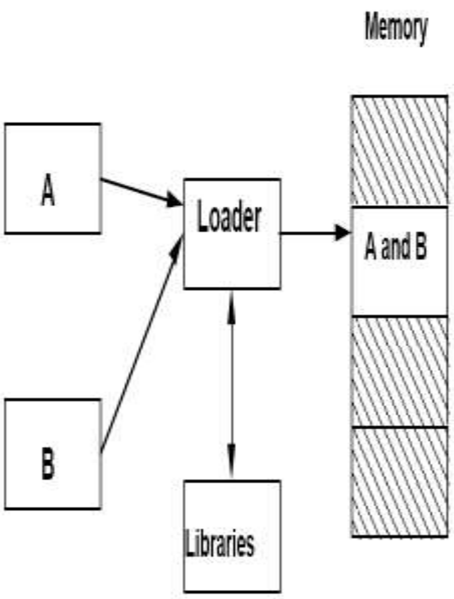

Figure 1 : showing the allocation and linking in the loader

Responsibilities Of loader :

Basically in Unix, the loader is the handler for the system call .the following tasks performed by Unix's loader are as follows :

1. validation (permissions, memory requirements etc.);
2. copying the program image from the disk into main memory;
3. copying the command-line arguments on the stack;
4. initializing registers (e.g., the stack pointer);
5. jumping to the program entry point (_start).

Types of Loader:

The various types of loaders are as follows:-

1) **Compile-and-Go Loaders**
2) **Absolute Loader**
3) **Relocation loader**
4) **Program linking loader**

Compile-and-Go Loaders :

A compile and go loader is one in which the assembler itself does the processes of compiling then place the assembled instruction in the designated memory locations. The assembly process is first executed and then the assembler causes a transfer to the first instruction of the program. The assembler run in one part of memory and it place the assembled machine instructions and data, as they are assembled, directly into their assigned memory locations. When the assembly is completed, the assembler causes a transfer to the starting instruction of the program. This type of loader is also called as assemble- and- go.

Advantages of Compile-and-go loaders:

- Simple and easier to implement.

- No additional routines are required to load the compiled code into the memory.

Disadvantages of Compile-and-go loaders:

- Wastage in memory space due to the presence of the assembler.
- There is a need to re-assemble the code every time it is to be run.
- It becomes increasingly difficult to handle large number of segments when the input code is written in a variety of HLL say one routine in pascal and one in FORTRAN and so on.
- Such loader make designing modular programs and systems near impossible.

- **Absolute Loader**

The assembler generates the object code equivalent of the source program bu the output is punched on to the cads forming the object decks instead of loading in memory.

The function of the loader is to read these cads and load them into memory specified by the assembler. The four functions as performed in and absolute loader are :
1.*Allocation*
2.*Linking*
3.*Relocation*

- 4.*Loading*

Advantages of Absolute Loader:

- Simple, easy to design and implement.
- Since more core memory is available to the user there is no memory limit.

Disadvantages of Absolute Loader:

- The programmer must specifically tell the assembler the address where the program is to be loaded.
- When subroutines are referenced, the programmer must specify their address whenever they are called.

Relocation loader

Loaders that allow for program relocation are called relocating loaders/relative loaders. There are two methods for this purpose. In the first method, a modification record is used to describe each part of the object code that must be changed when the program is relocated. There is one modification record for each calue that must be changed during relocation. Each modification record specifies the starting address and length of the field whose value is to be altered. It then describes the modification to be performed. This method is not well suited for all machine architectures, On a machine that primarily uses direct addressing and hads a fixed instruction format, a different technique is used. The text records are same as before except that there is a relocation bit with each word of object code. These relocation bits are gathered together into a bit mask following a length indicator in each text record. If the bit is set to 1, the programs's starting address must be added to the word when program is relocated. If bit is 0, then no modification is necessary. The bits for unused words are also set to 0.

Program linking loader

Let an application program 'A' consists of a set of program units. A program units interacts with another by using the addresses latter's instructions and data in its own instructions. To enable this, the program units must contain public defintions and external references. A public definition in a program unit may be reference in others. An external reference is a reference to symbol, which is not defined in the program unit containing the reference.

EXTRN and ENTRY statements are used to deeal with the above cases. ENTRY statement lists the public definitions of program unit, whcih may be referenced in other program units. EXTRN statement lists the symbols to which the external references are made in the program units.

Before executing the program A, for each of its program units, every external reference shoyld bound to the correct link time address. This is called "LINKING".Once it is linked, it is said to be resolved.

A binary program is a machine language program consisting of a set of program units, such that each program unit has been relocated to the memory area starting at its link origin and linking has been performed for each external reference in the program units.

CONCLUSION :

Linkers and loaders play a vital role in completion of a program. Programs can even be run/executed without loaders i.e., directly from the ROM but to load the operating system itself, a loader is required. Back in the days the loader had many limitations but over the years it has been improvised and worked upon and now it’s able to handle a lot more complex functions like adding support for modern featuresincluding dynamically linked shared libraries and the unusual demandsofC++. Loader has many functions namely allocation, linking. Hence loaders are very important for a programmer.

References :

1. "exec". *The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition*. The Open Group. Retrieved2008-06-23.
2. **Jump up^** IBM Corporation (1972). *IBM OS MVT Supervisor*.
3. **Jump up^** IBM Corporation (1972). *IBM OS Linkage Editor and Loader*.