Prepared By Abhiyan Dev Rimal
https://github.com/abhirimal/be-software-notes

Chapter=3 (2 or 3 Q)
Loader and Linkers

* Introduction

Source Program → [Assembler] → object code → [Linker] → Executable code → [Loader] → Program ready for execution / Memory

→ Loader is a system program that performs loading function
→ many loaders also support relocation and linking.
→ Some systems have linker to perform linking operation and a separate loader to handle relocation and loading.

Process of linking and relocation

- Linking
→ combines two or more ~~specific~~ separate object program.

- Relocating
→ modifies the object program so that it can be loaded at an address different from location originally specified.

- Loading
→ brings object program into memory for execution.

Scanned with CamScanner

Date
Page

11) What is loader? What is the function of loader?

Ans: Loader is a system program that performs loading function.

→ The loader is a special program that takes input of executable files from linker, loads it to main memory, and prepares this code for execution by computer.

→ ~~Loader also~~ (it brings object program into memory.)

→ Loader also allocates memory space to program.

Function of loader

→ bringing an object program into memory

→ starting its execution

Explain linking, relocating and loading from previous page.

→ Types of loader 1) Absolute loader 2) Bootstrap loader 3) Relocating loader 4) Linking loader Direct

12) What is program linking? How different program linking are done. Explain with example.

Ans: Linking combines two or more separate object programs and supplies the information needed to allow references between them.

→ It merges two or more separate object programs and establishes link among them.

→ The main function of linker is to generate executable files. (Linker combines two or more object files generated by compiler/assembler to generate a

Scanned with CamScanner

executable file which have .exe extension)

-> Generally, there are two types of linkers

1. Linkage Editor
2. Dynamic Linker

{Explain them}

3) What is the use of loader and linker in system program?

-> A loader is a system program that performs loading function. loader brings the object program into a memory and starts its execution.

-> An absolute loader completes it's task in a single pass.

-> Absolute loader checks the header record to verify that the correct program is loaded into memory. The loader reads each text record and moves the object code into the indicated address in memory. When end record is encountered, loader jumps to the specified address to begin execution of loaded program.

-> Linker is used to perform linking operation.

-> It is used to combine two or more separate object program and establish link among them. for example: instructions in one control section might need to refer to instructions in another section. As control sections are loaded and relocated independently, assembler is unable to process

Scanned with CamScanner

those references called as external references. The assembler generates the information for each external reference so that loader can perform required linking

-> Thus, loaders ~~takes the~~ help in loading the program into memory and linker links the various programs to provide executable code.

-> Also, loader helps in automatic library search, which helps in handling external references and to use standard subroutines which are required when linking.

Differences

~~Loader~~ Linker	~~Linker~~ Loader
1) The main function of ~~loader is to load executable files into memory~~ linker is to generate executable files.	1) The main function of loader is to load executable files to main memory.
2) The linker takes input of object code generated by compiler/assembler.	2) The loader takes input of executable files generated by linker.
3) Linker are of two types i) Linkage Editor ii) Dynamic Linking	3) Loader are of 4 types i) Absolute loader. ii) Bootstrap loader. iii) Direct Linking Loader iv) Relocating Loader

Scanned with CamScanner

V.Imp

4) What is absolute loader? Write an algorithm for absolute loader.

Ans: The operation of absolute loader is very simple.

-> The object code is loaded to specified location/address in the memory.

-> The address is specified in the START directive.

-> Relocation or linking is not needed.

-> All functions are accomplished in a single pass as follows

a) The Header record of ~~each~~ object program is checked to verify that correct program has been presented for loading.

b) The loader reads each Text record and moves the object code into the indicated address in memory.

c) When the End record is encountered, loader jumps to the specified address to begin execution of loaded program.

P.T.O ->

Scanned with CamScanner

Algorithm for Absolute loader

```
begin
    read Header Record
    verify program name and length
    read first text record
    while record type ≠ E do
        begin
        {
            if object code in characters form, convert
            into internal representation.
            move object code to specified location in
            memory.
            read next object program record.
            end
                jump to address specified in End record
end
```

Scanned with CamScanner

III 5) Simple Bootstrap Loader

→ When a computer is first turned on or restarted, a special type of absolute loader, called a bootstrap loader is executed.

→ This bootstrap loader loads the first program to be run by the computer - usually an operating system. (OS)

→ It is a program that resides in the computer's EPROM, ROM or another non volatile memory.

→ Bootstrap begins at address 0 in memory of machine.

→ It loads OS at address 80.

→ each byte of object code to be loaded is represented on device F1

→ When the computer is turned on or restarted, the bootstrap loader first performs the power-on self-test, also known as POST

→ If the POST is successful and no issues are found, the bootstrap loader loads the operating system for the computer into memory.

→ The computer can then access, load, and run the operating system.

Scanned with CamScanner

Machine dependent loader feature
- Relocation
- Program linking

Machine Independent loader feature
- Automatic library search
- Loader options
  ↳ Loader Design Option
    - Linking Loader
    - Linkage Editor
    - Dynamic Linking

} V.V.Imp 2 One will be asked

III 6) What is relocation? How relocation is carried out in a loader?

or

What are the main features of machine dependent loader when logically located related parts of programming are linked then what is generated and why is it important?

Ans: Loaders that allow for program relocation are called relocating loaders.

→ A relocating loader is capable of loading a program anywhere in memory.

→ One of the disadvantage of absolute loader is need for programmer to specify the actual address ~~at~~ at which program will be loaded into memory.

Scanned with CamScanner

-> On a simple computer with a small memory the actual address at which the program will be loaded can be specified easily.
-> On a large and more advanced machine several independent programs are run together sharing memory between them. We do not know in advance where a program will be loaded.
-> Hence, we write relocatable program instead of absolute ones.

=> There are two methods for specifying relocation as part of object program.

First method
-> A modification record is used to describe each part of object code that must be changed when the program is relocated.
-> Modification record are used in complex machine
Eg: SIC/XE
-> The format is
M n relocation address length

Second Method (Relocation bit)
-> There are no modification record.
-> It is used for simple machines.
-> The text record is same as before except that there is relocation bit associated with each word of object code.

Scanned with CamScanner

→ The relocation bit are gathered together in a bit mask.

→ This mask is represented as 3 hexadecimal digits.

→ If Relocation bit is 0 : no modification is necessary
Relocation bit is 1 : modification is needed.

Format.

Tn Starting address n length n relocation bits n obj code

Machine Independent Loader feature.

7) What do you understand by automatic library Search?

Ans: → This feature allows programmer to use standard subroutines without explicitly including them in the progm to be loaded.

→ The routines are automatically retrieved from a library as they are needed during linking.

→ The subroutines called by the program are automatically fetched from the library linked with the main program and loaded.

→ The programmer doesnot need to take any action beyond mentioning the subroutine names as external references in the source program.

→ if subroutine fetched contains external references then it is necessary to repeat the library search process until all references are resolved.

Scanned with CamScanner

library function = subroutine

→ if unresolved external references remain after the library search is completed, these must be treated as errors.

(III) 8) Differentiate linking loader from linkage editor.

Ans:

Linking Loader → links at load time

→ A linking loader performs all linking and relocation operations, including automatic library search, and loads the linked program into memory for execution.

→ In this, if object program has to call any library function then it will call the function and load it into the memory.

→ So for every time the program is run the library function will also be called & loaded into memory each time.

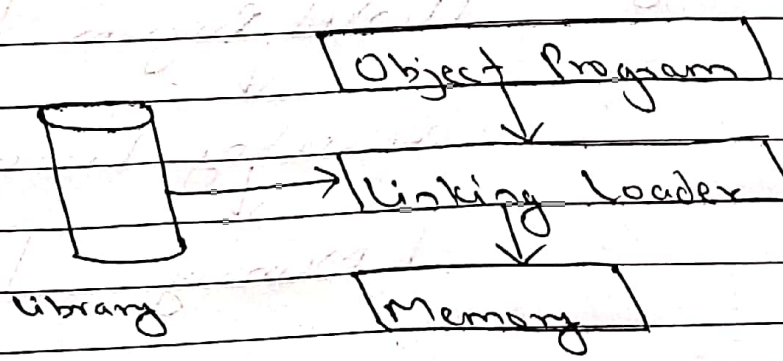


fig: Processing of object program in linking loader

Scanned with CamScanner

Date
Page

Linkage Editor → links before load time

→ it performs linking before the program is loaded for execution.

→ if object program has to call any library function then the linkage editor will call the library function and create a linked program.

↳ linked program = Obj. Program + Library function

→ Linkage Editor will link/save library function when it is called for the first time.

→ If the same library function is called again and again then there is no need to call library function ~~because~~, because it is already present in the linked program.

→ If modification is done in program then only we have to re-run it.

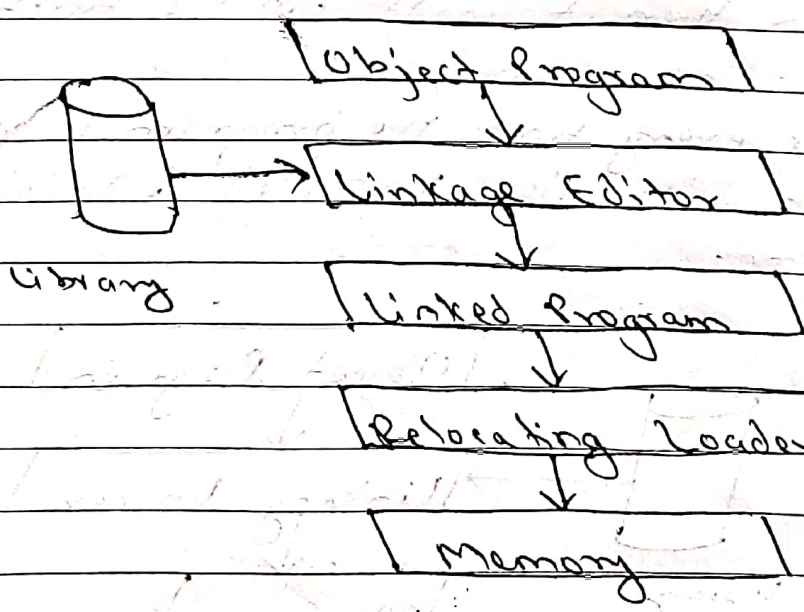


fig: Processing of object program in linkage editor.

Scanned with CamScanner

Difference.

Linking Loader	Linkage Editor
1) Performs all linking & relocation operations, including automatic library search, & loads the ~~linking~~ linked program into memory for execution.	1) Produces a linked version of the program, which is normally written to a file or library for later execution.
2) Loading may require two passes.	2) The loading can be accomplished in one pass.
3) It performs linking operation at load time.	3) It performs linking operation before the load time.
4) There's no need for relocating loader.	4) The relocating loader loads the load module into the memory.
5) Suitable when a program is reassembled for nearly every execution.	5) Suitable when a program is executed many times without being reassembled.
6) It is used when the program is in development stage.	6) It is used when the program development is finished.
7) figure	7) figure.

Scanned with CamScanner

classmate
Date
Page

||||| 9) Dynamic linking (Load on call)

Ans: It performs linking at execution time.
i.e it performs linking after load time.

→ A subroutine or library function is loaded and linked to the rest of the program when it's first called.

→ Several executing programs can now share one copy of subroutine.

→ ~~Eg. a single copy of subroutine or library~~

↳ Eg. a single copy of standard C library can be loaded into memory. All C programs currently in execution can be linked to this one copy instead of linking a separate copy into each object program.

→ Advantages

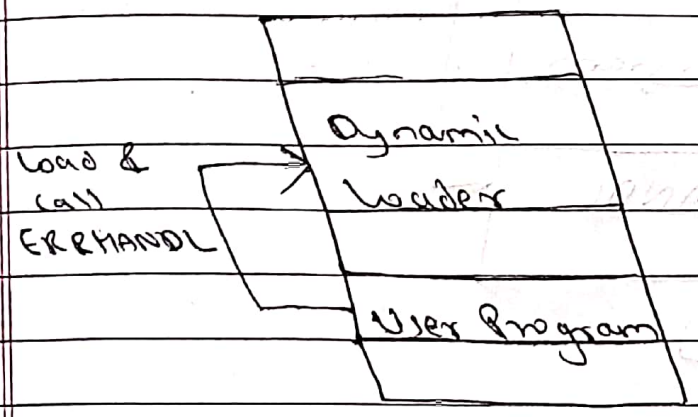
- Avoid the necessity of loading the entire library for each execution.
  i.e loads the routines only when they are needed.
- The time and memory will be saved.

→ Dynamic loader is one part of OS

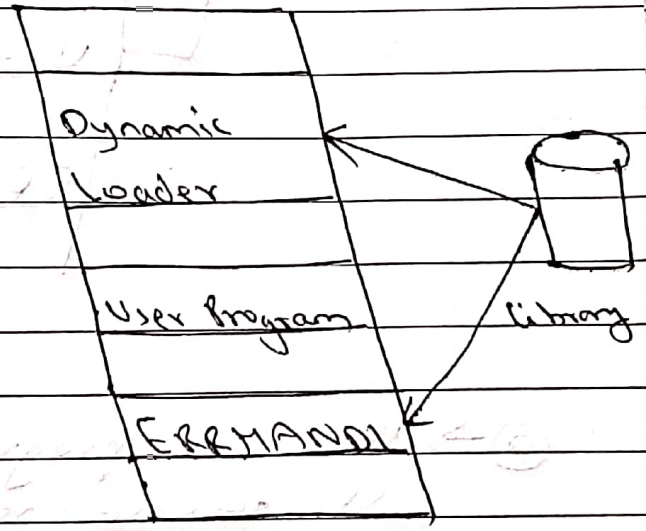
P.T.O →

Scanned with CamScanner

→ It has five steps:

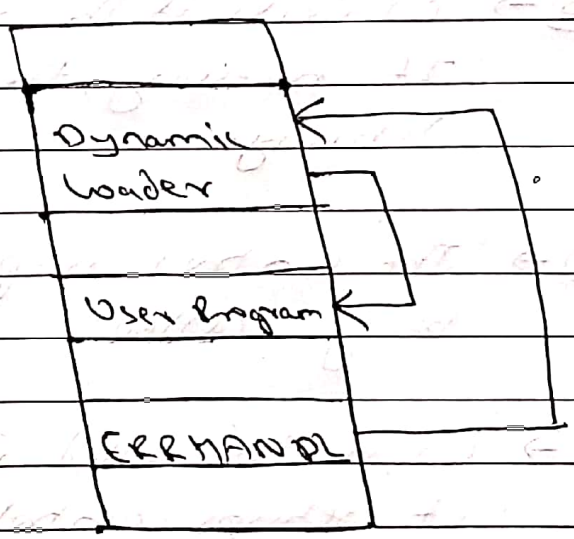


(a)



(b)

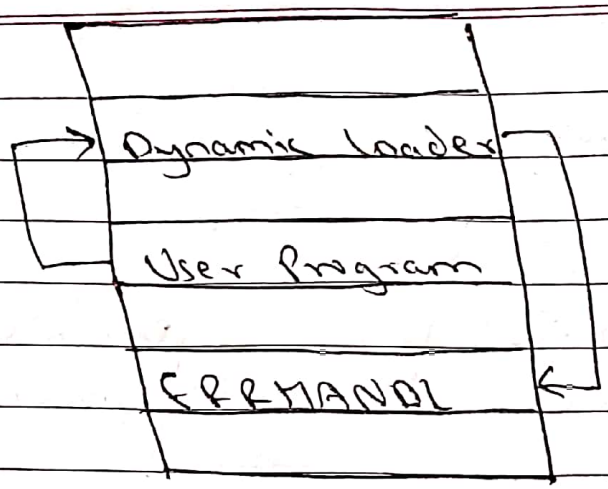
(c)



(d)

P.T.O →

Subroutine: it is a sequence of program instructions that performs a specific task packed as a unit.



(e)

(a) → the user program ~~with a~~ makes a load and call service request to the OS. (ERRMANDL)

(b) → OS checks if the routine is already loaded. If necessary, the routine is loaded from the library.

(c) → The control is passed from OS to the routine being called.

(d) → when the called subroutine completes its processing it returns control to the OS. OS then returns control to the program that issued the request.

(e) → If new/other program will ask for the same library function (subroutine) then the OS doesnot have to load the subroutine.
The control is simply passed from dynamic loader to the called subroutine.

Scanned with CamScanner