(SPCC)

Assignment No. 2

Q.1 Explain the Various functions of loaders

→ loaders in computing are essential for loading programs or data into memory So they can be executed or processed. They are part of the System's Overall architecture that facilities the execution of programs and managing resources efficiently

There are several functions of loaders :

① loading program code into memory

Function: the primary role of a loader is to load executable programs or code into memory

② Relocation :-

Function :- Relocation adjusts the memory addresses in the program. The program often assumes Certain memory addresses.

③ symbol Resolution :-

Function :- It resolves symbolic addresses to actual memory addresses. A program may have references to Variables or functions that are defined elsewhere

④ loading Dynamic libraries :-

function :- the loader is responsible for dynamicall linking shared libraries or modules at runtime.

⑤ Memory allocation:

Function :- The loader allocates the necessary memory space for the program and its resources

(6) loading and Initialization of program Variables
function :- The loader intializes global Variables and constants in the program.

(7) Handling program Execution :-
function: It helps set up the execution environment for the program.

(8) Error Handling :-
function :- The loader manages errors that occurs during the loading process

(9) support for memory protection
function :- the lodder ensures that programs do not interfere with each other by managing memory protection.

(10) loading Configuration files.
function: In some Systems, the loader also loads configuration files that the program may need to run

(11) Executable file parsing
function of the process of analyzing and interpreting the structure of executable file (such as .exe, .out, .elf, or, .bin files)

Date

Q.2 Explain the working of Direct linking loaders with neat flowcharts.

→ Direct linking loaders are type of loader the directly links a program into memory, preparing for execution without any significant intermediate steps like relocation.

Working of direct linking loaders :-

Loading :-
The program is loaded into memory as-is, with no changes to its addresses. The loader assumes that all addresses. The loader assumes that all addresses are already correct for the memory location it is loaded into.

Direct linking :-
The loader handles the linking process, which involves Combining all the modules or object files that make up the program into a single executable image. In the case of direct ~~l~~inking loader, this means linking external references between modules directly.

Memory Allocation :-
It assigns a block of memory where the program will reside, typically at the location specified by the User or the operating system.

Relocation :-
In the case of direct linking loaders, the program may already have been relocated or is assumed to not require relocation.

vivo Y20 2025.03.26 23:25

Working of direct linking loaders:-

① Program preparation :- The Source code is written and Compiled into object code (machine code) by Compiler or assembler. The object code usually contains unresolved addresses (labels) that need to be resolved during loading.

② Load Address :- The direct linking loader loads the object code directly into specified memory address. This is typically fixed address or predefined range in memory.

③ Resolution of Addresses :-
In direct linking loaders, all symbolic address are resolved during the linking stage (before loading) This means that all references to Variables or functions are already mapped to actual memory locations when the program is doaded.

④ Load process :-
- The loader reads the object file
- It places the programs code directly into memory
- It does not modify or adjust any addresses as all addresses are already resolved before loading.

⑤ Execution :-
Once the program is in memory, the loader passes control to the program's starting point

vivo Y20 2025.03.26 23:25

Q.3 Explain the design of absolute loaders and explain the data structure in detail.

→ An absolute loader is a type of loader that directly loads the machine code of a program into memory without any relocation or adjustment of addresses In other words, it takes the object code (compiled program) and places it into a predetermined, fixed location in memory

- Absolute loaders are simpler and faster than other types of loaders but are less flexible

Working of Absolute loaders :-

① program complication :- The Source code is first Compiled by a Compiler or assembler into object Code.

② load Address :- the absolute loader knowns the fixed memory location where the program should be loaded

③ loading process :- The loader reads the object code It places the object code directly into the memory starting from the specified location. No address Modification.

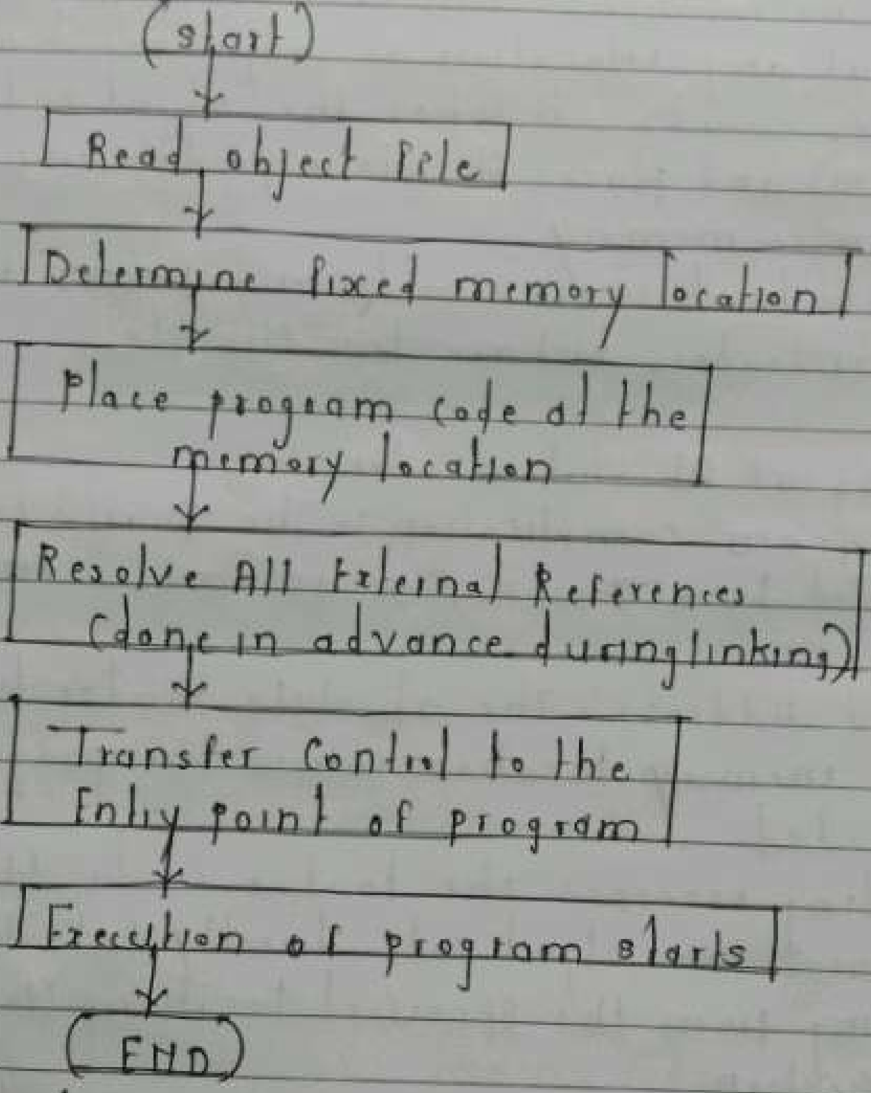
④ Execution :- After loading, the control is transferred to the entry point of the program

Data Structure in Absolute loaders :-

① object code Representation :-
The object code generated by the Compiler/assembler typically consists of a Series of binary instructions

typically the end point

Flowchart for Direct linking Loader



(start)
Read object file
Determine fixed memory location
Place program code at the memory location
Resolve All External References (done in advance during linking)
Transfer Control to the Entry point of program
Execution of program starts
(END)

key points :-

- The loader does not need to perform address relocation (it is already done by during linking)
- Direct linking loaders are typically fast and simple but are less flexible.
- This type of loader works best when the program size is known and there are no overlapping memory issues

along with metadata describing where and how the program should be loaded

- Code Segment :- This contains the machine code or binary instructions
- Data Segment :- This holds intialized data Variables.
- Symbol Table (optional) :- Some object files might include a symbol table that maps Symbolic names

Object file Structure Example :-

Code Segment	Data Segment	Symbol Table
Machine code instructions	Initialized Data	Symbol -> Addr (If available)

2. Memory layout :-
The memory layout for the loaded program can be represented in the following structure. Since absolute loaders work with fixed memory locations, the program must be loaded exactly at the pre-determin address

Memory layout Example :-

Program code (Machine code)	Unused / Reserved Space	Data / Variables (Initial Data)

3. loader Table (optional) :-
Absolute loaders don't require relocation tables (since no relocation is done) It may use a loader.

table to map logical addresses to physical memory addresses for reference

logical Address	Physical Address
0X004000	0X004000
0X004004	0X004004

Advantages :-	Disadvantages
① simplicity	① limited flexibility
② speed	② Memory wastage
③ No relocation Needed	③ No suitable for dynamic sy

Q. 4 Explain the significance and differences between linkage editor and linking editor.

$\longrightarrow$ Linkage editor :-

Significance :- The linkage editor is responsible for taking object modules and combining them into a single executable program. It resolves symbolic references between different object modules, such as functions or variables that are defined in one module but referenced in another.

linking Editor.

Significance :- the linking editor is similar to the linkage editor and in some contexts, the terms are synonymous. It is used to combine object files and libraries to create an executable.

Linking Editor	Linkage Editor
1) performs all linking and relocation operations, including automatic library Search and loads the linked program into memory for execution.	produces a linked version of the program which is normally written to a file or library for later execution.
2) Suitable when a Program is reassembled for nearly every execution	Suitable when a program is executed many times without being reassembled
3) Searching is performed more than once	Searching is performed only once
4) It perform linking Operations at load time	It perform linking operations before the program is loaded for execution.
5) no need of relocating loader	relocating loader loads the load module into the memory
6) When program is in development stage the at that time.	When the program development is finished or when the library is built.

Q.5 State the advantages and disadvantages of absolute loaders

→ Advantages

① cleaner code structure.
- Absolute imports allow you to import modules using absolute paths rather than relative paths.
- It leads to cleaner, more readable code, especially in large projects.

② Better Maintainibility :-
- As the project grows, refactoring becomes easier because absolute paths don't require changes when files are moved around the directory structure

③ Improved Developer productivity:-
- It reduces the time spent on managing relative imports, which can become a hassle in large codebases. with absolute imports, developers can quickly locate files and dependencies

④ Consisten paths :-
Using absolute paths ensures that every developer on the project is importing modules in a Consistent manner, making collaboration easier.

Disadvantages

① Initial setup Complexity :-
Setting up absolute imports in a project. It Configure the paths

② potential conflicts with Node Modules :-

③ Learning curve :- those unfamilier with project structure

④. Dependencies on Build tool Configuration