OPERATING SYSTEM

Linking & Loading

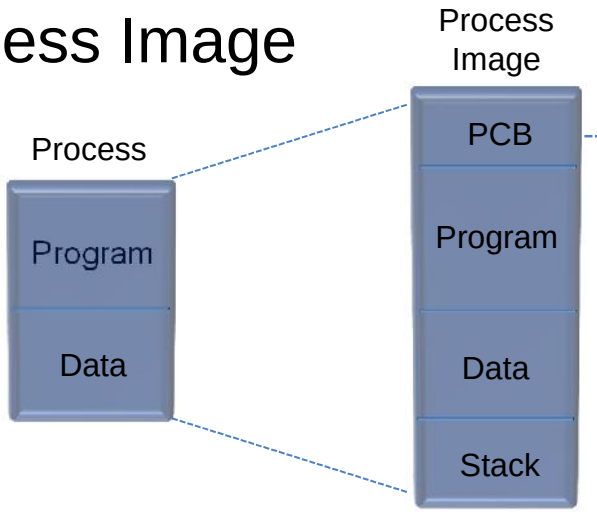
Introduction

Loading & its types

Linking & its types

Review

- Multiprocessing
- Multiprogramming or Multitasking
- Process Image



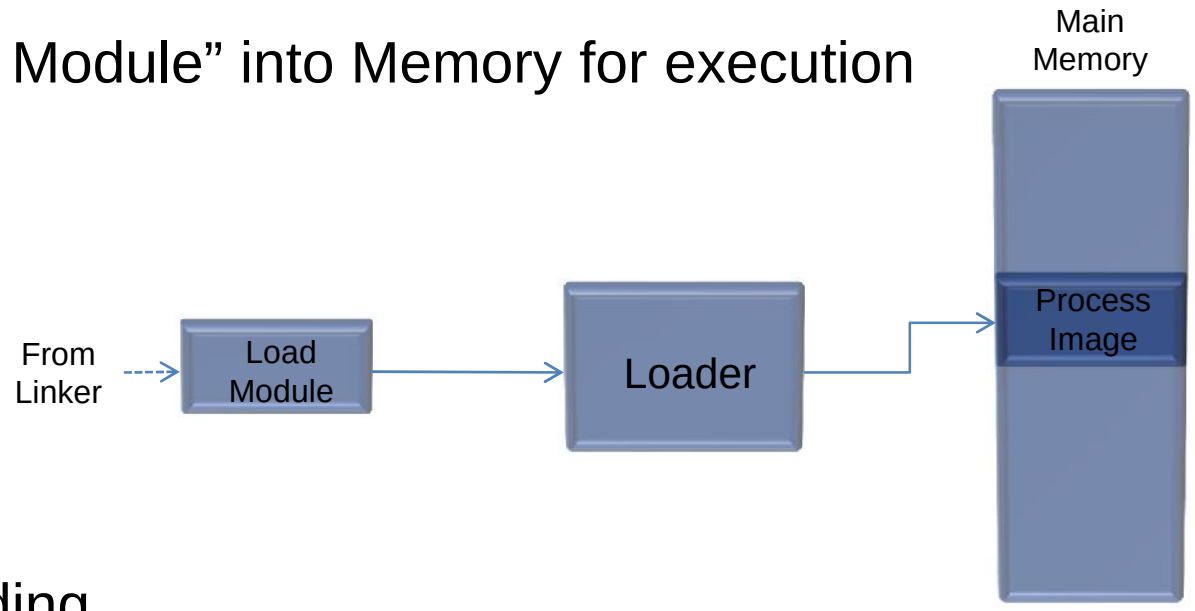
Process Control Block

Registers
Program Counter
Program status word (PSW)
Stack Pointer
Process State
Priority
Scheduling parameters
Process ID
Parent & Children(s) Process
Process group
Signals
Time when process started
CPU time used
Children's CPU time
Time of next alarm

- Memory Management Techniques
- Memory Management Unit
- Programmers' keywords (compiler, assembler, modules, libraries, source code & object code)

Loading

- Loads the "Load Module" into Memory for execution



- Techniques:
  1. Absolute Loading
     - Process is loaded at same location in Memory
     - Advantages:
       - Very simple technique
     - Disadvantages:
       - Programmer have to know about absolute addresses of memory or compiler/assembler has to find physical addresses
       - Absolute addresses keep changing when length of program is altered which result calculation of references again and again

Loading

2. Re-locatable Loading
   - Process can be loaded at different locations in Memory
   - Allows facility of loading, swapping & relocating process in memory (Multiprogramming based systems)
   - Programmer/compiler represent addresses in other address form (relative address starting from 0 location)
   - Loader then convert those relative addresses from code into physical addresses by help of relocation dictionary (contains information about references & their interpretation, it is prepared by compiler/assembler)
   - Advantages
     - Programmer does not need to know or compiler/assembler does not need to find physical addresses
     - Alteration in program is very easy
     - Multi-tasking is possible
   - Disadvantages
     - Whenever a process is swapped, loader have to convert references into physical addresses each time

Continue

Loading

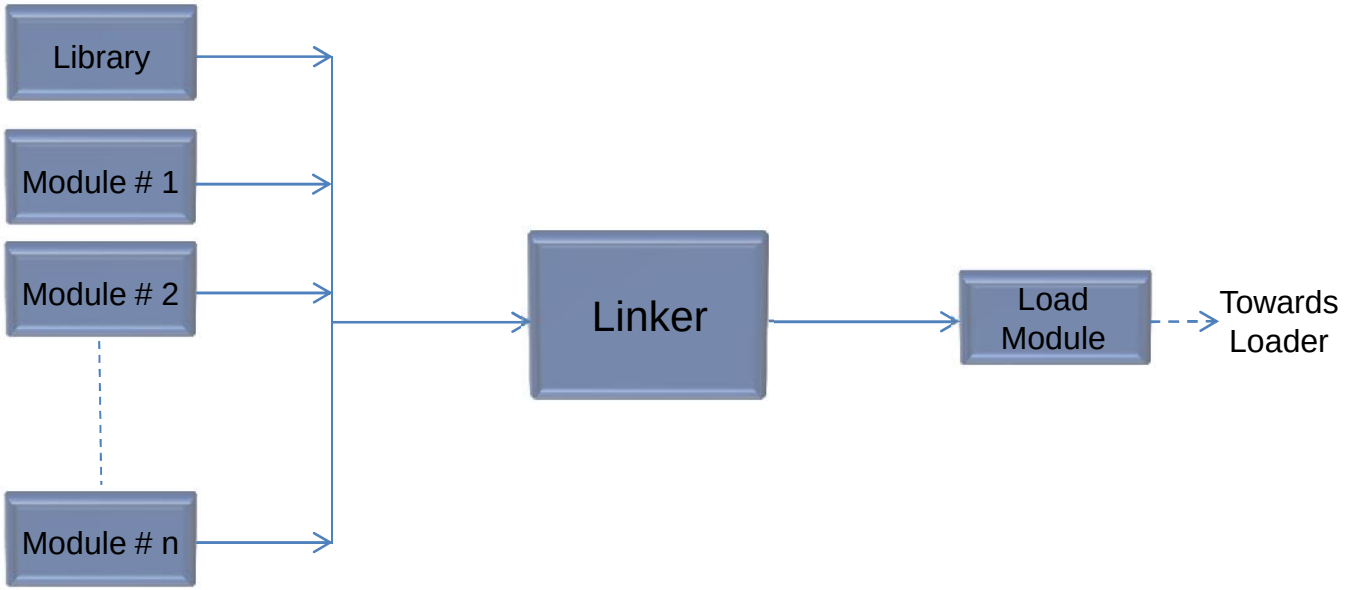
3. Dynamic Run-time Loading

- Also supports multiprogramming based system
- this technique, relative addresses are not converted into physical addresses when loaded
- Rather it remains same until that piece of code is actually to be executed.
- Relative address is converted into Physical address at run time
- Requires special hardware to perform this task (i.e. MMU)
- Advantages
  - No need to convert references into physical address at load time
- Disadvantages
  - Special hardware is required

Continue

Linking

- Linker is used to convert object code (generated by compiler/assembler) into load module for loader
- It combines all modules & external sources (libraries) into one single load module



- It also convert references for modules into relevant addresses

Linking

- Techniques
  1. Linkage Editor
     - Used for Re-locatable & Dynamic run-time loading
     - Its task is to put all modules together and convert symbolic references into relative addresses from its origin (i.e. 0)
     - Advantages
       - Create a way to support multiprogramming based system
       - Provides flexibility to refer unknown physical address at run/load time
     - Disadvantages
       - External modules (Libraries) are put together before load time so only static libraries can be linked

Continue

Linking

2. Dynamic Linker

- In this technique, the external modules are referred but references are not resolved up to load module.
- Those can be resolved at load time or run time
- Provides a feature to use dynamic and/or shared libraries (such as Dynamic Link Library)
- Advantages
  - Change/Updated of libraries is easy rather than changing whole application for a particular module
  - Libraries can be shared among multiple processes
  - Flexibility for programmers to extend functions supported by OS
- Disadvantages
  - Process takes more execution time

Continue

Linking

- Types
  i. Load-Time Dynamic Linking
    - References are resolved by finding target module, load it (if not already loaded in memory or if library is not shared)
  ii. Run-Time Dynamic Linking
    - Most of references are made at load time while, some of references are made at run time.
    - If reference is made for absent module, then it is loaded & reference is resolved
    - Sometimes it is difficult to predict that which modules are to be referred and which modules are not to be
    - Only those modules are loaded which are required
    - This task is performed by loader and/or MMU

Continue

Questions