Operating System

Dr. GuoJun LIU

Harbin Institute of Technology

http://guojunos.hit.edu.cn

Outline

- Memory Management Requirements
  - Relocation
  - Protection
  - Sharing
  - Logical Organization
  - Physical Organization
- Memory Partitioning
  - Fixed Partitioning
  - Dynamic Partitioning
  - Buddy System
  - Relocation
- Paging
- Segmentation

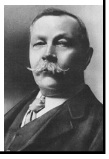
Chapter 07

Memory Management

内存管理

I cannot guarantee that I carry all the facts in my mind. Intense mental concentration has a curious way of blotting out what has passed. Each of my cases displaces the last, and Mlle. Carère has blurred my recollection of Baskerville Hall. Tomorrow some other little problem may be submitted to my notice which will in turn dispossess the fair French lady and the infamous Upwood.

-- *THE HOUND OF THE BASKERVILLES,*
Arthur Conan Doyle



Learning Objectives

- **Discuss the principal requirements for memory management**
- **Understand the reason for memory partitioning and explain the various techniques that are used**
- **Understand and explain the concept of paging**
- **Understand and explain the concept of segmentation**
- **Assess the relative advantages of paging and segmentation**

Memory Management

- **Uniprogramming**
  - one part for **OS**
    - resident monitor, kernel
  - one part for the **program** currently being executed
- Multiprogramming
  - **the "user" part of memory** must be further **subdivided** to accommodate multiple processes
  - The task of subdivision is carried out **dynamically** by OS and is known as **memory management**

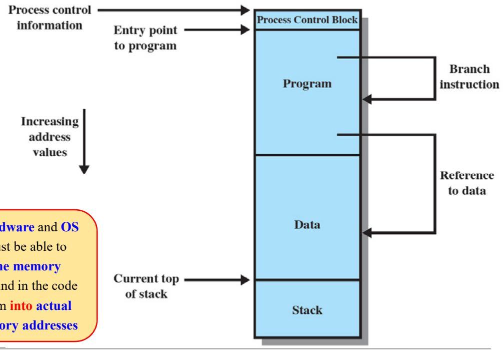
Memory Management Requirements

- Memory management is intended to satisfy the following requirements:
  - Relocation
  - Protection
  - Sharing
  - Logical organization
  - Physical organization

Relocation

- Programmers typically do not know in advance which other programs will be resident in main memory at the time of execution of their program
- Active processes need to be able to be swapped in and out of main memory in order to maximize processor utilization
- Specifying that a process must be placed in the same memory region when it is swapped back in would be limiting
  - may need to relocate the process to a different area of memory

Addressing Requirements for a Process



Processor hardware and OS software must be able to translate the memory references found in the code of the program into actual physical memory addresses

Protection

- Processes need to acquire permission to reference memory locations for reading or writing purposes
- Location of a program in main memory is unpredictable
  - it is impossible to check absolute addresses at compile time to assure protection
- Memory references generated by a process must be checked at run time
- Mechanisms that support relocation also support protection

a program in one process cannot branch to an instruction in another process

Who is responsible for memory protection, processor or OS ?

Sharing

- Advantageous to allow each process access to the same copy of the program rather than have their own separate copy
- Memory management must allow controlled access to shared areas of memory without compromising protection
- Mechanisms used to support relocation support sharing capabilities

Logical Organization

- Memory is organized as linear
  - closely mirrors the actual machine hardware
  - typically, programs are not constructed like this
- Programs are written in modules
  - modules can be written and compiled independently
  - different degrees of protection given to modules
    - read-only, execute-only
  - sharing on a module level corresponds to the user’s way of viewing the problem
- Segmentation is the tool that most readily satisfies requirements

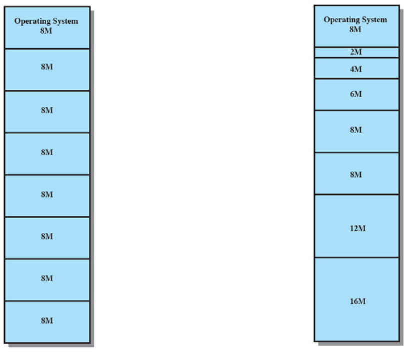
Physical Organization

- Major system concern
  - the flow of information between main and secondary memory
- Be assigned to the individual programmer, impractical and undesirable
  - Memory available for a program plus its data may be insufficient
    - overlaying allows various modules to be assigned the same region of memory but is time consuming to program
  - Programmer does not know how much space will be available
- This task is the essence of memory management

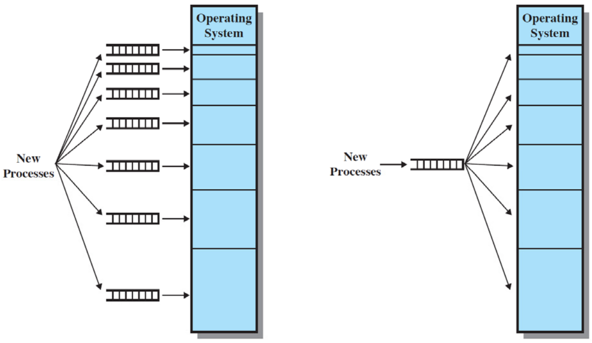
Memory Partitioning

- Memory management brings processes into main memory for execution by the processor
  - involves virtual memory
  - based on segmentation and paging
- Partitioning
  - used in several variations in some now-obsolete operating systems
  - does not involve virtual memory

Fixed Partitioning



Fixed Partitioning



Disadvantages

- The number of partitions specified at system generation time limits the number of active processes in the system
- Small jobs will not utilize partition space efficiently

How to know the main storage requirement of all jobs beforehand ?

Dynamic Partitioning

- Partitions are of variable length and number
- Process is allocated exactly as much memory as it requires
- This technique was used by IBM’s mainframe operating system, OS/MVT

Placement Algorithms

Best-fit

- chooses the block that is closest in size to the request

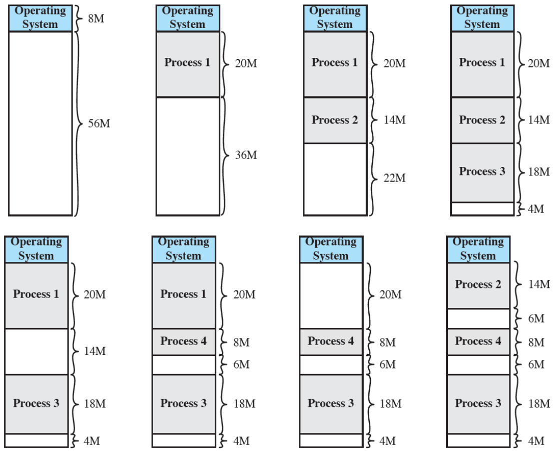
First-fit

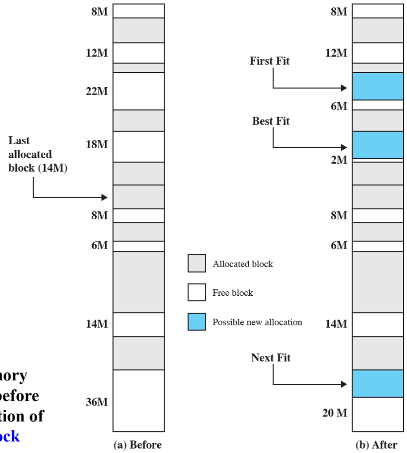
- begins to scan memory from the beginning and chooses the first available block that is large enough

Next-fit

- begins to scan memory from the location of the last placement and chooses the next available block that is large enough

The Effect of Dynamic Partitioning





Example Memory Configuration before and after Allocation of 16-Mbyte Block

Dynamic Partitioning

External Fragmentation

- memory becomes more and more fragmented
- memory utilization declines

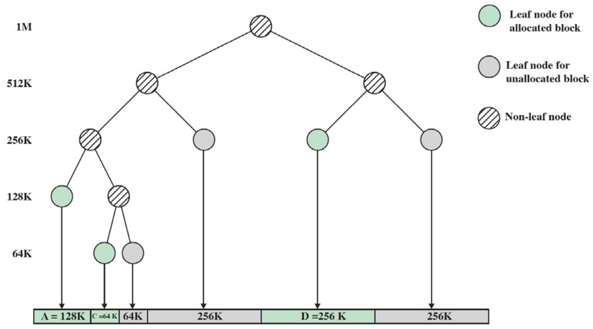
Compaction

- technique for overcoming external fragmentation
- OS shifts processes so that they are contiguous
- free memory is together in one block
- time consuming and wastes CPU time

Buddy System Example

1 Mbyte block	1 M					
Request 100 K	A = 128K	128K		256K	512K	
Request 240 K	A = 128K	128K		B = 256K	512K	
Request 64 K	A = 128K	C = 64K	64K	B = 256K	512K	
Request 256 K	A = 128K	C = 64K	64K	B = 256K	D = 256K	256K
Release B	A = 128K	C = 64K	64K	256K	D = 256K	256K
Release A	128K	C = 64K	64K	256K	D = 256K	256K
Request 75 K	E = 128K	C = 64K	64K	256K	D = 256K	256K
Release C	E = 128K	128K		256K	D = 256K	256K
Release E	512K				D = 256K	256K
Release D	1M					

Tree Representation of Buddy System



Addresses

Logical

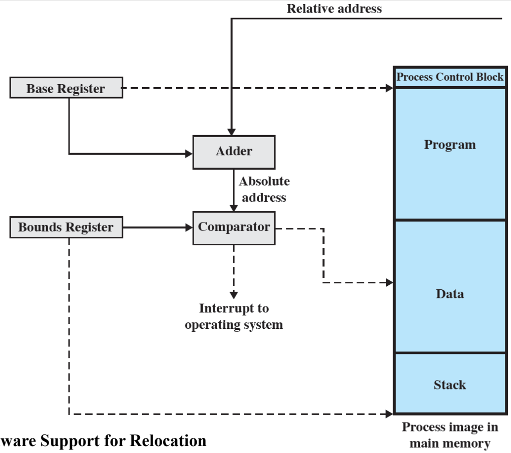
- reference to a memory location independent of the current assignment of data to memory

Relative

- address is expressed as a location relative to some known point

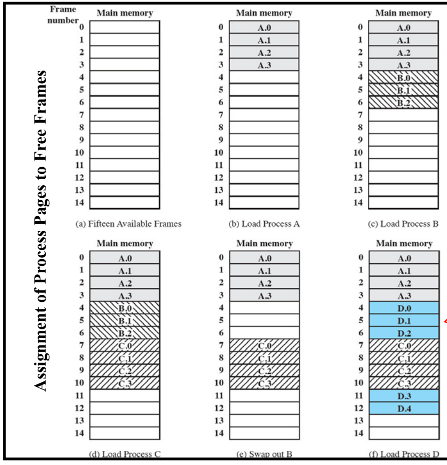
Physical or Absolute

- actual location in main memory



Hardware Support for Relocation

Assignment of Process Pages to Free Frames



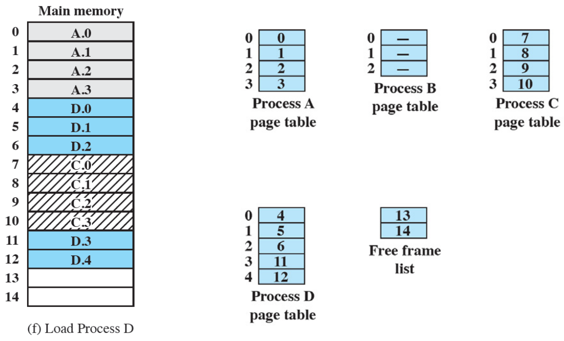
base address register is sufficient ?

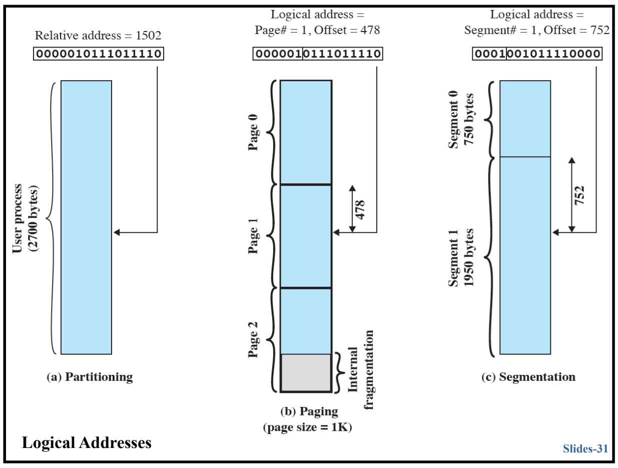
not sufficient unused contiguous frames to hold the process, prevent OS from loading D ?

Page Table

- Maintained by OS for each process
- Contains the frame location for each page in the process
- Processor must know how to access for the current process
- Used by processor to produce a physical address

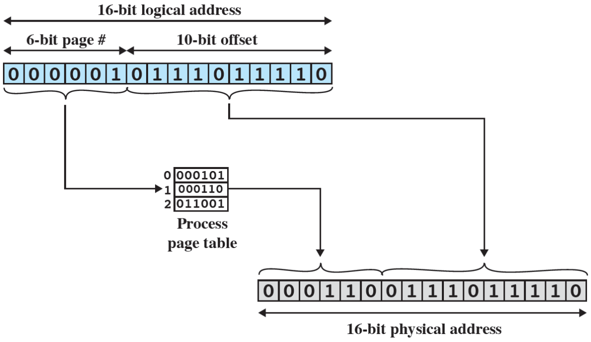
Data Structures





Logical Addresses

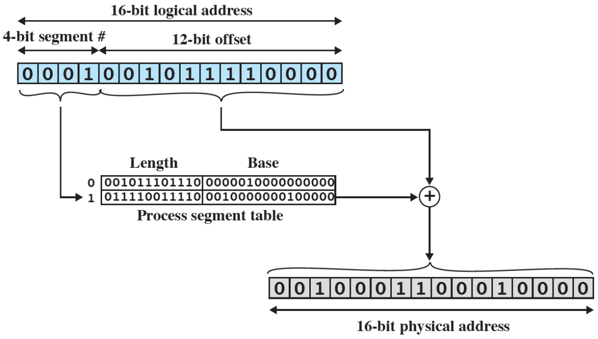
Logical-to-Physical Address Translation



Segmentation

- **A program can be subdivided into segments**
  - may **vary in length**
  - there is **a maximum length**
- **Addressing consists of two parts:**
  - segment number
  - an offset
- **Similar to dynamic partitioning**
- **Eliminates internal fragmentation**

Logical-to-Physical Address Translation

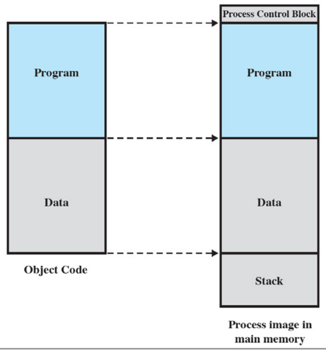


Technique	Description	Strengths	Weaknesses
Fixed Partitioning	Main memory is divided into a number of static partitions at system generation time. A process may be loaded into a partition of equal or greater size.	Simple to implement; little operating system overhead.	Inefficient use of memory due to internal fragmentation; maximum number of active processes is fixed.
Dynamic Partitioning	Partitions are created dynamically, so that each process is loaded into a partition of exactly the same size as that process.	No internal fragmentation; more efficient use of main memory.	Inefficient use of processor due to the need for compaction to counter external fragmentation.
Simple Paging	Main memory is divided into a number of equal-size frames. Each process is divided into a number of equal-size pages of the same length as frames. A process is loaded by loading all of its pages into available, not necessarily contiguous, frames.	No external fragmentation.	A small amount of internal fragmentation.
Simple Segmentation	Each process is divided into a number of segments. A process is loaded by loading all of its segments into dynamic partitions that need not be contiguous.	No internal fragmentation; improved memory utilization and reduced overhead compared to dynamic partitioning.	External fragmentation.
Virtual Memory Paging	As with simple paging, except that it is not necessary to load all of the pages of a process. Nonresident pages that are needed are brought in later automatically.	No external fragmentation; higher degree of multiprogramming; large virtual address space.	Overhead of complex memory management.
Virtual Memory Segmentation	As with simple segmentation, except that it is not necessary to load all of the segments of a process. Nonresident segments that are needed are brought in later automatically.	No internal fragmentation, higher degree of multiprogramming; large virtual address space; protection and sharing support.	Overhead of complex memory management.

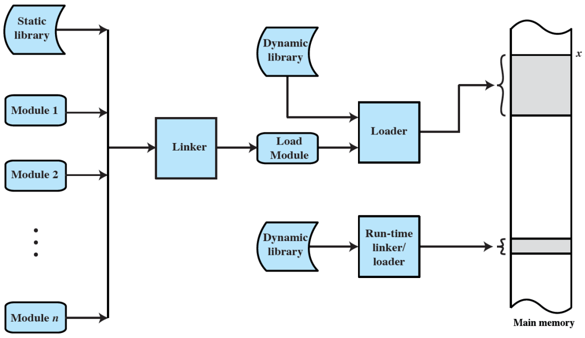
Summary

- **Memory Management**
  - one of **the most important and complex** tasks of an OS
  - needs to be treated as a resource to be **allocated to and shared** among a number of active processes
  - desirable to **maintain** as **many processes in main memory** as possible
  - desirable to **free programmers** from **size restriction** in program development
  - basic tools are **paging** and **segmentation** (possible to **combine**)
    - paging – small **fixed-sized** pages
    - segmentation – pieces of **varying size**

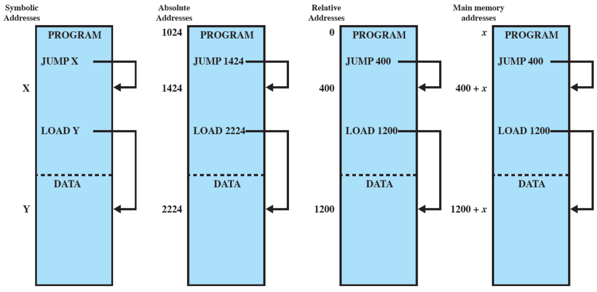
The Loading Function



A Linking and Loading Scenario



Absolute and Relocatable Load Modules



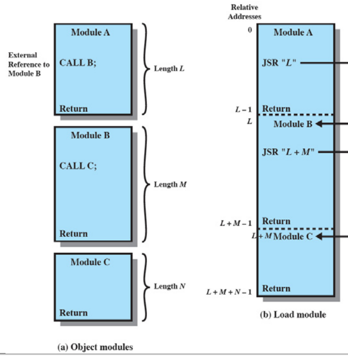
(a) Object module

(b) Absolute load module

(c) Relative load module

(d) Relative load module loaded into main memory starting at location *x*

The Linking Function



(a) Object modules

(b) Load module