Evolution of Operating Systems

- Serial Processing
  - No operating system
    - Machines run from a console with display lights, toggle switches, input device, and printer
    - Schedule time, e.g. sign up
    - Setup included loading the compiler and source program, saving compiled program, loading and linking

Evolution of Operating Systems

- Simple Batch Systems
  - Monitor
    - Software that controls the sequence of events
    - Batch jobs together
    - Program branches back to monitor when finished

Job Control Language (JCL)

- Special type of programming language
- Provides instruction to the monitor, e.g.
  - What compiler to use
  - What data to use

Hardware Features

- Memory protection
  - Do not allow the memory area containing the monitor to be altered
- Timer
  - Prevents a job from monopolizing the system

Hardware Features

- Privileged instructions
  - Certain machine level instructions can only be executed by the monitor
- Interrupts
  - Early computer models did not have this capability

Memory Protection

- User program executes in **user mode**
  - Certain instructions may not be executed
- Monitor executes in **system mode**
  - Kernel mode
  - Privileged instructions are executed
  - Protected areas of memory may be accessed

I/O Devices Slow

Read one record from file	15 µs
Execute 100 instructions	1 µs
Write one record to file	15 µs
TOTAL	31 µs

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Figure 2.4 System Utilization Example

More general: Speedup

– Amdahl's Law: $$S = \frac{1}{(1-f) + f/k}$$

Effective Speedup

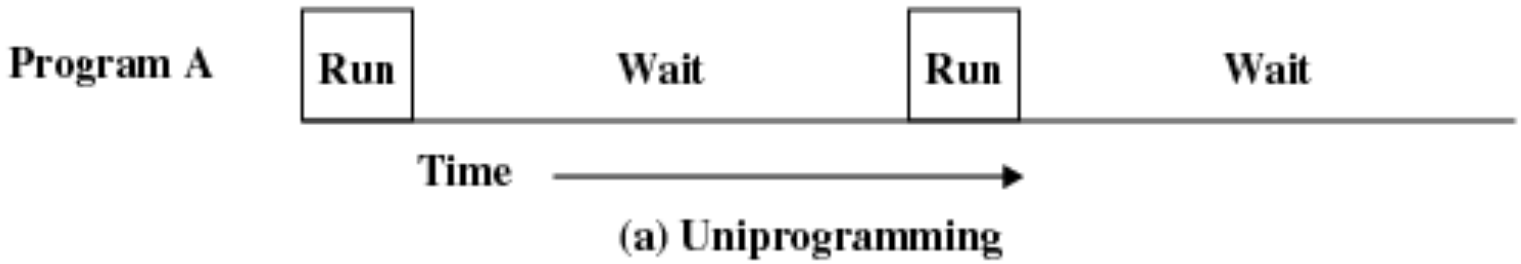
- f = fraction of work in fast mode
- k = speedup while in fast mode

Example:

- assume 10% I/O operation
- if CPU 10x => effective speedup is 5.26
- if CPU 100x => effective speedup is 9.17
  - 90 % of potential speedup is wasted

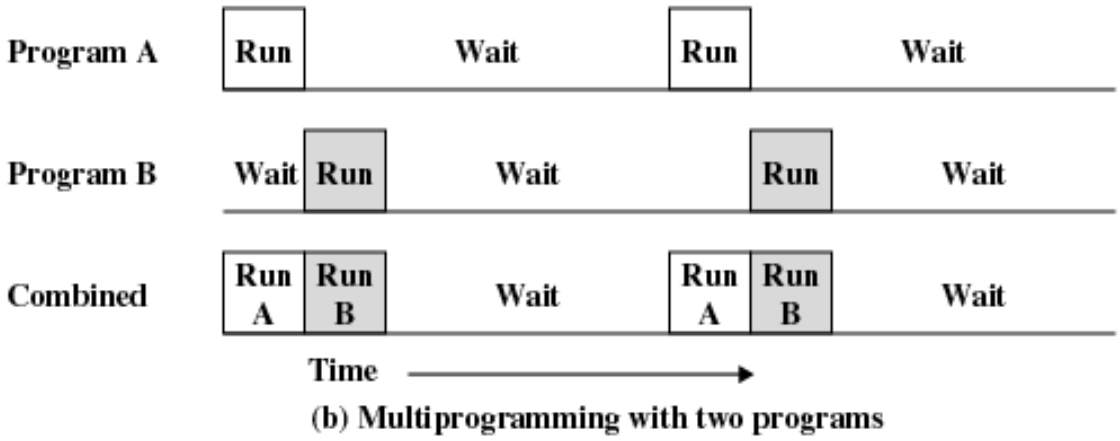
Uniprogramming

- Processor must wait for I/O instruction to complete before preceding



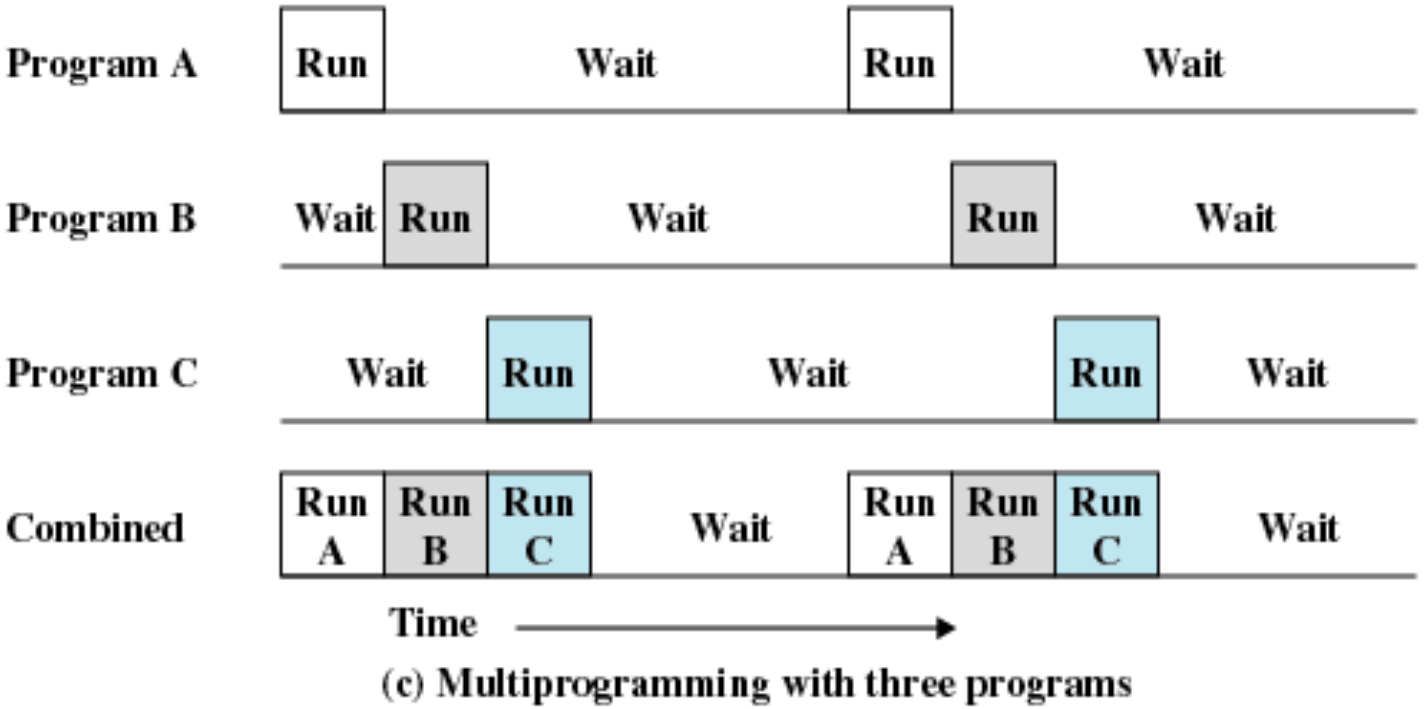
Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

Multiprogramming



(c) Multiprogramming with three programs

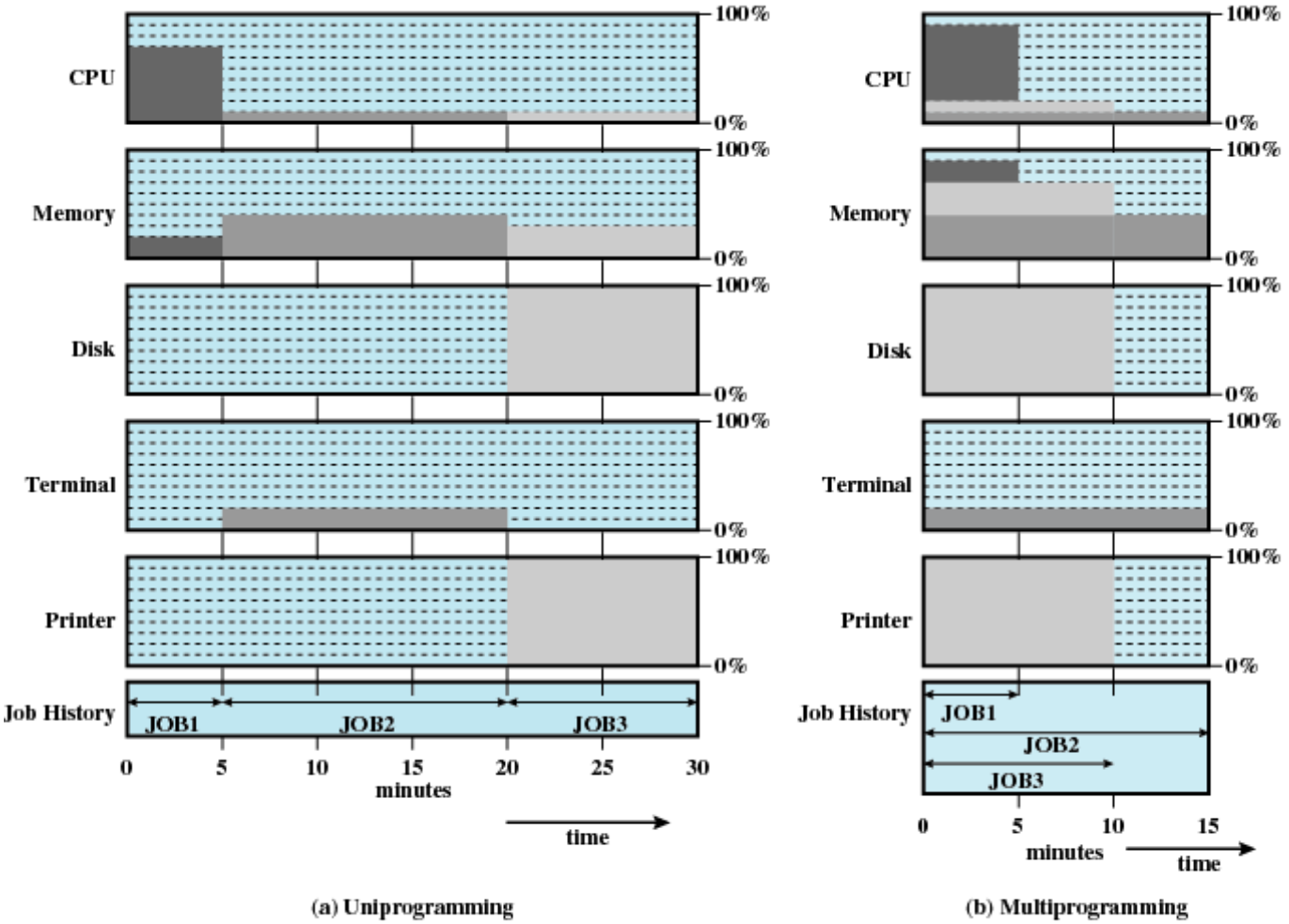
Example

Table 2.1 Sample Program Execution Attributes

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Utilization Histograms

Three jobs, different shades of grey



Sequence 3

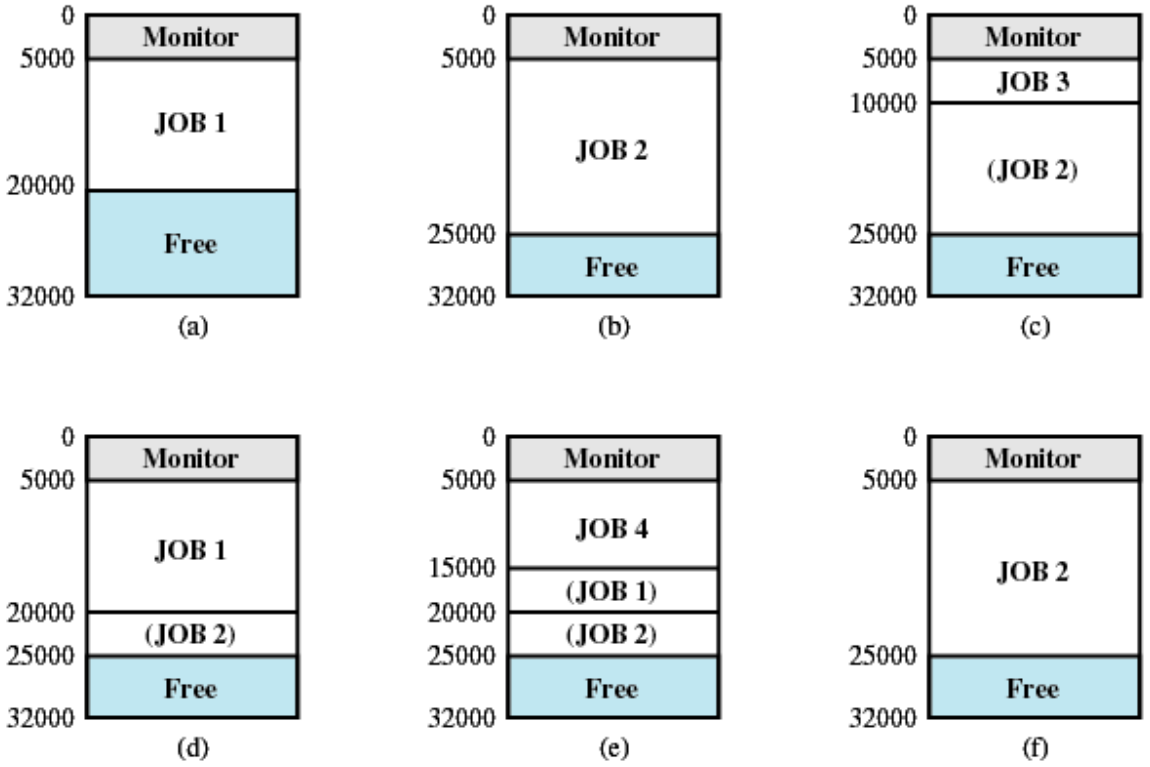
Figure 2.6 Utilization Histograms

Time Sharing

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals

Compatible Time-Sharing System (CTSS)

- First time-sharing system developed at MIT



Example
Memory requirements:
Job1: 15,000
Job2: 20,000
Job3: 5,000
Job4: 10,000

Only write back that portion of memory that is overwritten by newly loaded job.

Figure 2.7 CTSS Operation

Major Achievements

- Denning et.al. [DENN80a] point out 5 major OS advances:
  - Processes
  - Memory Management
  - Information protection and security
  - Scheduling and resource management
  - System structure
- Let's look at each one...

Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

Difficulties with Designing System Software

- Improper synchronization
  - Ensure a process waiting for an I/O device receives the signal
- Failed mutual exclusion
- Nondeterminate program operation
  - Program should only depend on input to it, not on the activities of other programs
- Deadlocks

Process

- Consists of three components
  - An executable program
  - Associated data needed by the program
  - Execution context of the program
    - All information the operating system needs to manage the process

Process

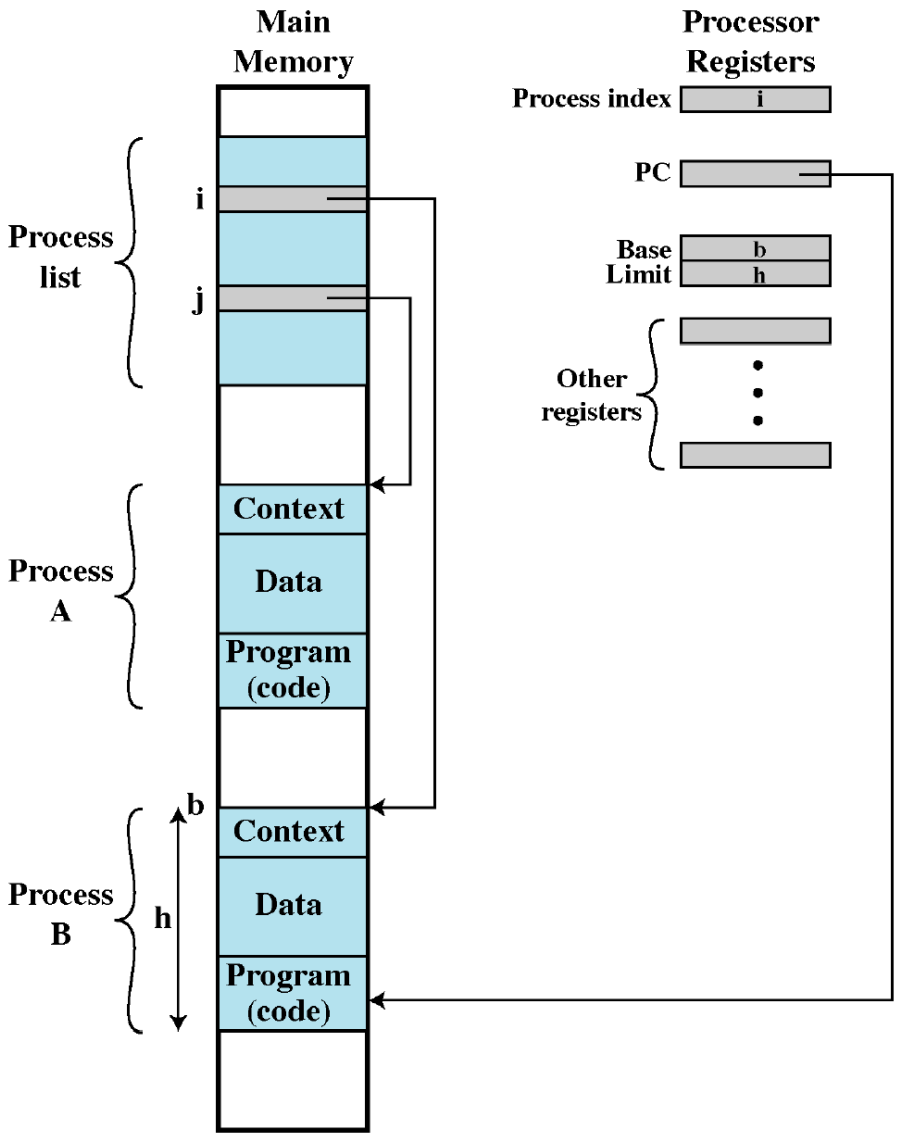


Figure 2.8 Typical Process Implementation

Memory Management

- Process isolation
  - non-interference between independent procs.
- Automatic allocation and management
  - should be transparent to programmer
- Support of modular programming
- Protection and access control
- Long-term storage
  - after computer has been powered down

Virtual Memory

- Allows programmers to address memory from a logical point of view
- No hiatus between the execution of successive processes while one process was written out to secondary store and the successor process was read in

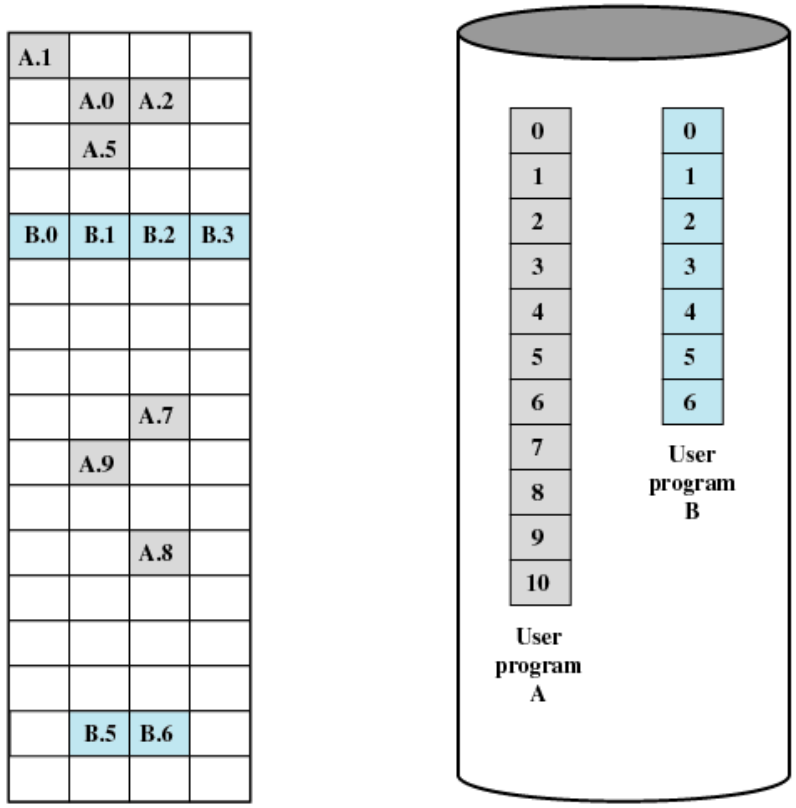
Virtual Memory and File System

- Implements long-term store
- Information stored in named objects called files

Paging

- Allows process to be comprised of a number of fixed-size blocks, called pages
- Virtual address is a page number and an offset within the page
- Each page may be located anywhere in main memory
- Real address or physical address in main memory

Virtual Memory



Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.

Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

Figure 2.9 Virtual Memory Concepts

Virtual Memory Addressing

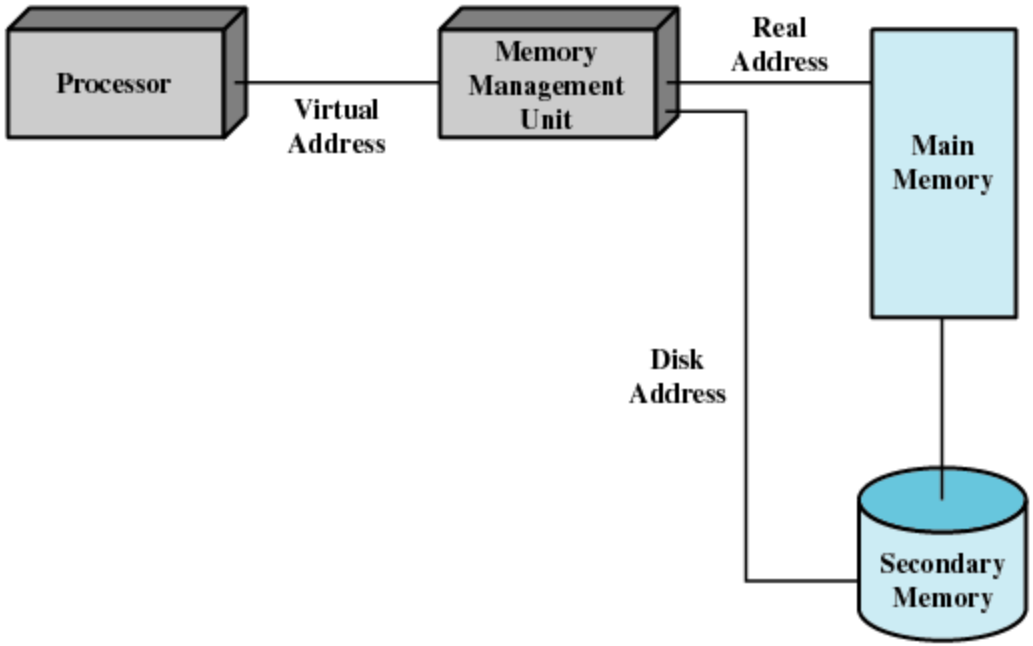


Figure 2.10 Virtual Memory Addressing

Information Protection and Security

- Availability
  - Concerned with protecting the system against interruption
- Confidentiality
  - Assuring that users cannot read data for which access is unauthorized

Information Protection and Security

- Data integrity
  - Protection of data from unauthorized modification
- Authenticity
  - Concerned with the proper verification of the identity of users and the validity of messages or data

Scheduling and Resource Management

- Fairness
  - Give equal and fair access to resources
- Differential responsiveness
  - ...but, OS also needs to discriminate among different classes of jobs
- Efficiency
  - Maximize throughput, minimize response time, and accommodate as many uses as possible

Key Elements of Operating System

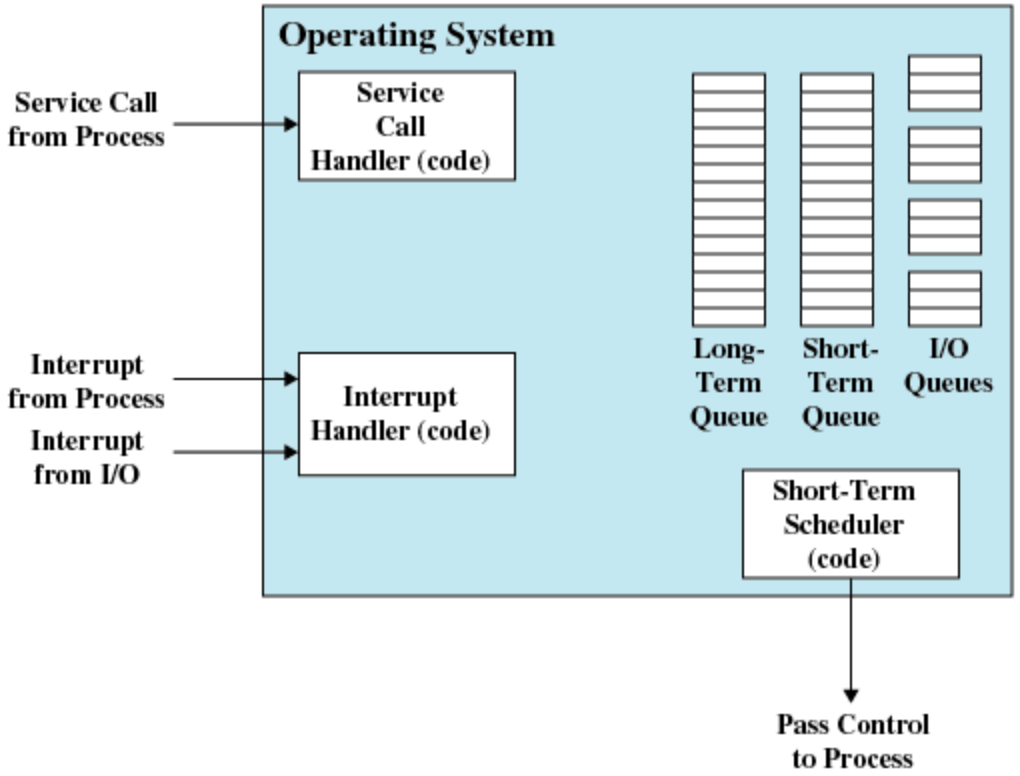


Figure 2.11 Key Elements of an Operating System for Multiprogramming

System Structure

- View the system as a series of levels
- Each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions
- This decomposes a problem into a number of more manageable subproblems

Process Hardware Levels

- Level 1
  - Lowest level
  - Electronic circuits
  - Objects are registers, memory cells, and logic gates
  - Operations are clearing a register or reading a memory location
- Level 2
  - Processor's instruction set
  - Operations such as add, subtract, load, and store

Process Hardware Levels

- Level 3
  - Adds the concept of a procedure or subroutine, plus call/return operations
- Level 4
  - Interrupts

Concepts with Multiprogramming

- Level 5
  - Process as a program in execution
  - Suspend and resume processes
- Level 6
  - Secondary storage devices
  - Transfer of blocks of data
- Level 7
  - Creates logical address space for processes
  - Organizes virtual address space into blocks

Deal with External Objects

- Level 8
  - Communication of information and messages between processes
- Level 9
  - Supports long-term storage of named files
- Level 10
  - Provides access to external devices using standardized interfaces

Deal with External Objects

- Level 11
  - Responsible for maintaining the association between the external and internal identifiers
- Level 12
  - Provides full-featured facility for the support of processes
- Level 13
  - Provides an interface to the operating system for the user