Review

- Multiprocessor System Types
  - Multiprocessor System with single core Chip
  - Multiprocessor System with multiple core Chip
  - Symmetric multiprocessing(SMP) – one physical memory shared
  - Non-Uniform memory access (NUMA) – each processor has local memory, but share one physical address space
  - Clustered System
    - Asymmetric clustering – one machine is in hot standby mode – monitoring the active server
    - Symmetric clustering – monitoring each other
- Dual Mode and Multimode Operation
  - System Call
- Multitasking vs. Multiprogramming

Preview

- OS as a Resources Manager
  - Process Management
  - Memory Management
  - File Management
  - Input / Output System Management
  - Deadlock Management
  - Cache Management
- Operating System Structures
  - Monolithic
  - Layered System
  - Microkernels
  - Virtual machine
  - Client-Server module
  - Exokernels

OS as a Resource Manager

- Process (& thread) management
- Memory Management
- File Managements
- Input / Output System Management
- Deadlock Managements
- Cache Management

OS as a Resource Manager
(Process Management)

- The operating system is responsible for the following activities for process management:
  - Creating and deleting both user and system processes
  - Scheduling processes (and threads) on the CPUs
  - Suspending and resuming processes (and threads)
  - Providing mechanisms for process synchronization (semaphore, mutex, conditional variable, …)
  - Providing mechanisms for process communication (PIPE, Message Queue, Shared Memory, FIFO, Socket,…)

OS as a Resource Manager
(Memory Management)

- For a program to be executed, it must be mapped to absolute addresses and loaded into memory.
- As the program executes, it accesses program instructions and data from memory by generating these absolute addresses.
- The CPU reads instructions from main memory during the instruction-fetch cycle and both reads and writes data from main memory during the data-fetch cycle (on a von Neumann architecture).
- Eventually, the program terminates, its memory space is declared available, and the next program can be loaded and executed.

OS as a Resource Manager
(Memory Management)

- The operating system is responsible for the following activities for memory management:
  - Keeping track of which parts of memory are currently being used and which process is using them (for supporting multiprogramming)
  - Allocating and deallocating memory space as needed
  - Deciding which processes (or parts of processes) and data to move into and out of memory (virtual memory).

OS as a Resource Manager
(File Management)

- Operating system provides a uniform, logical view of information storage for user to save a file. A file is a collection of related information defined by its creator.
- The operating system implements the abstract concept of a file by managing mass storage media and the devices that control them.
- In addition, files are normally organized into directories to make them easier to use.
- If a system support multi-user, OS need control which user may access a file and how that user may access it (for example, read, write, append).

OS as a Resource Manager
(File Management)

- The operating system is responsible for the following activities for file management:
  - Creating and deleting files
  - Creating and deleting directories to organize files
  - Supporting primitives for manipulating files and directories.
    - read, write, open, lseek, ….
  - Mapping files onto mass storage (HDD, SSD, USB, Tape,…)
  - Backing up files on stable (nonvolatile) storage media

OS as a Resource Manager
(Mass-Storage Management)

- Modern Computer system use multiple types of secondary storages (HDD, SSD, USB, Magnetic Tape,…).
- The operating system is responsible for the following activities for secondary storage management.
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling (in HDD)
  - Partitioning
  - Protection

OS as a Resource Manager
(Cache Management)

- <u>Cache is a hardware or software component</u> that stores data which might be used again soon. The data stored in a cache might be the result of an earlier computation or a copy of data stored elsewhere.
- A cache hit occurs when the requested data can be found in a cache, while a cache miss occurs when it cannot.
- Because caches have limited size, cache management is an important design problem.
- Careful selection of the cache size and of a replacement policy (with cache miss) can result in greatly increased performance,

OS as a Resource Manager
(Cache Management)

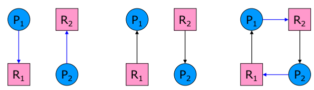
- The movement of information between levels of a storage hierarchy may be either explicit or implicit, depending on the hardware design and the controlling operating-system software.
  - For instance, data transfer from cache to CPU and registers is usually a hardware function, with no operating-system intervention.
  - Data transfer of data from disk to memory is usually controlled by the operating system.

OS as a Resource Manager
(Deadlock Management)

- Deadlocks between processes are happened <u>since limited number of resources which must be shared between processes.</u>
- Processes are sharing resources for finishing their job.
- Some processes might need more than one resources hold to finish its job. (ex. copy data from disk to USB drive)

OS as a Resource Manager
(Deadlock Management: Resource Allocation Graph)

- There are two processes P_1 and P_2. There are two resources R_1 and R_2
- Both P_1 and P_2 need resources R_1 and R_2 to finish their job



a) b) c)

OS as a Resource Manager
(Deadlock Management)

Ex)
- There are two processes P_1, P_2 working on their job.
- There are one CD recorder, and CD ROM, P_1 and P_2 need two resources to finish its job.

1. T_1: P_1 request CD ROM and granted
2. T_1: P_2 request CD recorder and granted
3. T_2: P_1 request CD recorder without release CD ROM. Since CD recorder is hold by P_2. P_1 be waiting state (block state) waiting for CD recorder leased by P_2
4. T_3: P_2 request CD ROM but it is not released by P_1, P_2 go to waiting state (block state)
5. P_1 and P_2 will stay in block state forever!

OS as a Resource Manager
(Deadlock Management)

- Four Strategies for Dealing Deadlock
  1. Just ignore
  2. Detection and Recover (detection algorithm)
  3. Dynamic Avoidance by careful allocation (banker's Algorithm)
  4. Prevention – by negating one of the four conditions necessary to cause deadlock

     (Four necessary conditions to cause deadlock
     1. Mutual exclusion
     2. Circular wait
     3. Hold and wait
     4. No Preemptive)

OS as a Resource Manager
(Input/Output)

- Operating System manages all kinds of I/O devices such as keyboards, monitors, printers, and so on.
- Operating System has I/O subsystems for managing I/O devices.
- I/O subsystems consist of
  - A memory-management components – Buffering, Spooling, and Caching
  - General device drivers interface
  - Drivers for specific hardware
  - Processor for specific hardware

OS as a Resource Manager
(Input/Output)

- Operating System controls all I/O devices by
  - Issue commands to Devices (read/write)
  - Catch interrupts from Devices (when device is ready to read or write)
  - Handle error
- OS also provide interface between the devices and the rest of the system

Structure of Operating System

- Operating System Structures
  - Monolithic
  - Layered System
  - Microkernels
  - Virtual machine
  - Client-Server module
  - Exokernels

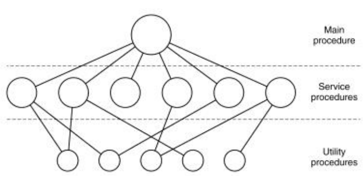
Structure of Operating System
(Monolithic System)

- It is written as a collection of procedures, each of which can call any of the other ones whenever it needs to.
- Each procedure in the system has a well-defined interface in terms of parameters and results, and each one is free to call any other one, if the latter provides some useful computation that the former needs.
- It is possible to have some structure for a monolithic system

Structure of Operating System
(Monolithic System)

- Possible structure for a monolithic system
  - A main program –invoke service functions
  - Service functions – carry out the system calls
  - Utility functions – helps service functions

Structure of Operating System
(Monolithic System)



Structure of Operating System
(Layered System)

- Operating system is divided into several layers and each layer works for different rule.
  - Layer 0 – process management
  - Layer 1 – memory management
  - Layer 2 – inter-process communication
  - Layer 3 – Input /Output management
  - Layer 4 – Deadlock management
  - Layer 5 – user program
  - Layer 6 – system operator process

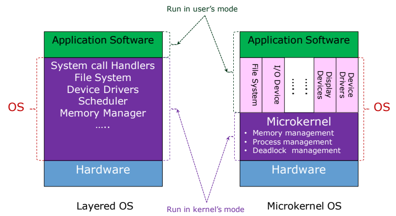
Structure of Operating System
(Microkernels)

- With the layered approach, the designers have a choice where to draw the kernel-user boundary.
- Traditionally, all the layers went in the kernel, but it is not necessary since one bug in kernel might cause entire system down.
- Various researchers studied the number of bugs (logical) per 1000 lines of code.
- About 5 million lines of codes for layered kernel likely to contains 10000 to 50000 kernel bugs.
- Not all of bugs are fatal.

Structure of Operating System
(Microkernels)

- The basic idea of the microkernel is to achieve high reliability by splitting the operating system up into small well-defined module.
- Only one of module (microkernel) run in kernel mode and the rest run as users mode.
- Ex) I/O device driver becomes part of OS which might cause system down.
  - A bug in the audio driver will cause the sound to be garbled or stop, but not crash the computer.

Structure of Operating System
(Microkernels)



Structure of Operating System
(Virtual Machine)

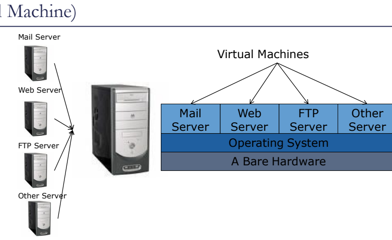
- Virtual machine monitor(Hypervisor) runs on the bare hardware and does multiprogramming, by providing several virtual machines.
- Each virtual machines are exact copies of the bare hardware, including kernel/user mode, I/O and everything else the real machine has.
- Different virtual machine can run different operating system.

Structure of Operating System
(Virtual Machine)

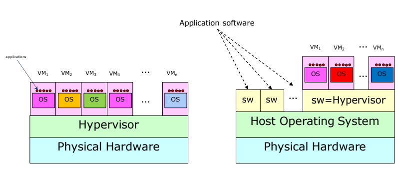
Example of virtualization

- Many companies have traditionally run their mail servers, Web servers, FTP servers and other servers on separate computer with different operating system.
- Several server can run on the same machine without having a crash of one server bring down the rest since each server running on different virtual machine.

Structure of Operating System
(Virtual Machine)



Structure of Operating System
(Virtual Machine)



Extra topics

- Protection and Security
- Virtualization