Previous lecture review

- to parallelize execution each device is connected to a controller; all controllers are joined by common bus; communication between controllers and CPU is interrupt based
- the storage is hierarchical - fastest types of storage are accessed first; to exploit locality of reference caching mechanism is used; faster types of storage tend to be more expensive and less reliable
- protection needs to be used in multiprogramming and timesharing OSes:
  - I/O protection
  - memory protection
  - CPU protection

Lecture 3: OS objectives and organization

- OS objectives
  - process management
  - memory management
  - disk/file management
  - networking, command interpreting
- system calls - OS interface to application programs
- OS design approaches:
  - monolithic kernel
  - layering
  - microkernel
  - virtual machine

Process management

- OS manages many kinds of activities:
  - user programs
  - system programs: printer spoolers, name servers, file servers, etc.
- a running program is called a process
  - a process includes the complete execution context (code, data, PC, registers, OS resources in use, etc.)
  - a process is not a program
    - program - a sequence of instructions (passive)
    - process - one instance of a program in execution (acitve);
  - many processes can be running the same program and one program may cause to create multiple processes
- from OS viewpoint process is a unit of work; OS must:
  - create, delete, suspend, resume, and schedule processes
  - support inter-process communication and synchronization, handle deadlocks

Memory management

- primary (main) memory (RAM)
  - provides direct access storage for CPU
  - processes must be in main memory to execute
- OS must:
  - mechanics
    - keep track of memory in use
    - keep track of unused (“free”) memory
    - protect memory space
    - allocate, deallocate space for processes
    - swap processes: memory <–> disk
  - policies
    - decide when to load each process into memory
    - decide how much memory space to allocate to each process
    - decide when a process should be removed from memory

Disk management

- The size of the disk is much greater than main memory and, unlike main memory, disk is persistent (endures system failures and power outages)
- OS hides peculiarities of disk usage by managing disk space at low level:
  - keeps track of used spaces
  - keeps track of unused (free) space
  - keeps track of “bad blocks”
- OS handles low-level disk functions, such as:
  - schedules of disk operations
  - and head movement

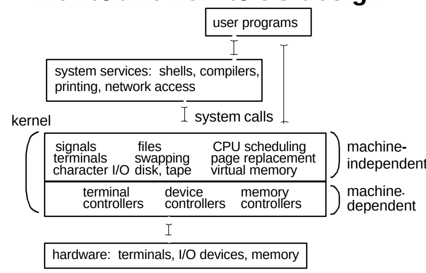
File management

- disks provide long-term storage, but are awkward to use directly
- file - a logical named persistent collection of data maintained by OS
- file system - a logical structure that that is maintained by OS to simplify file manipulation; usually directory based
- OS must:
  - create and delete files and directories
  - manipulate files and directories - read, write, extend, rename, copy, protect
  - provide general higher-level services - backups, accounting, quotas
- note the difference between disk management and file system management

System calls

- system calls provide the interface between a process and the operating system.
- It is a way of transferring control to the OS so that it can carry out a certain function for the process.
- Example: a program that opens a text file and prints on the screen uses the following system calls:
  - open a file - if the file could not be opened - inform the program
  - read a line of file
  - print the line just read on the screen
  - continue the last two system calls until the end of the file is reached
  - close file

Monolithic kernel OS design



the *kernel* is the protected part of the OS that runs in monitor mode

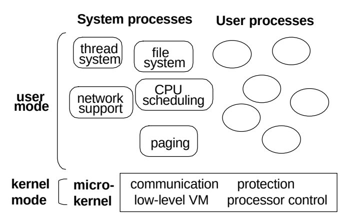
critical OS data structures and device registers are protected from user programs can use privileged instructions

- advantages: speed and ease of operation (everything is at hand)
- disadvantages:
  - hard to develop, maintain, modify and debug
  - kernel gets bigger as the OS develops

Layered Design

- divide OS into layers
- each layer uses services provided by next lower layer
  yet the implementation of these services are hidden from the upper layer
- THE Operating system layer structure:
  - user programs
  - buffering for input and output devices
  - operator-console device driver
  - memory management
  - CPU scheduling
  - hardware
- advantages: easier development and implementation
- disadvantages: not always easy to break down on layers, slower (each level adds overhead)
  - ex: CPU scheduler is lower than virtual memory driver (driver may need to wait for I/O) yet the scheduler may have more info than can fit in memory
- examples: THE, OS/2

Microkernel



• small kernel implements communication (usually messages)
• when system services are required microkernell calls other parts of OS running in user modes and passes the request there

- advantages: reliability, ease of development, modularity - parts can be replaced and tailored to the architecture, user requirements etc.
- disadvantages: slow
- examples: Mach(US), MacOS X, Windows NT

Virtual Machine

- system calls can be considered an enhancement of hardware's instruction set
  - extend further – virtual machine
- each user task is provided with an abstract (virtual machine) which OS + hardware implement
  - IBM – pioneered
  - Java VM – modern example
- JVM
  - Java source code is translated into an architecture independent java bytecode
  - bytecode is executed by JVM
  - JVM can be implemented purely in software or in hardware
  - JVM verifies bytecode's correctness and then either interprets (tranlates the code into machines instructions one by one)
  - or just-in-time (JIT) comples to optimize

adv. – portability at binary-level, security, greater language flexibility

dis. – speed(?)