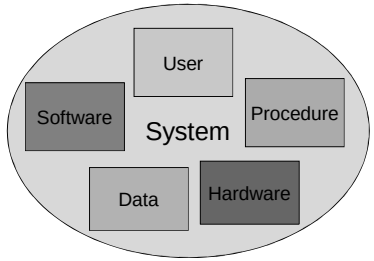
Operating System

11/4/2009

Che-Rung Lee

What is a system?

- A set of interacting of interdependent entities forming an integrated whole.
  - From Wikipedia
- Five components
  - Hardware
  - Software
  - Data
  - Procedure
  - User



Software classification

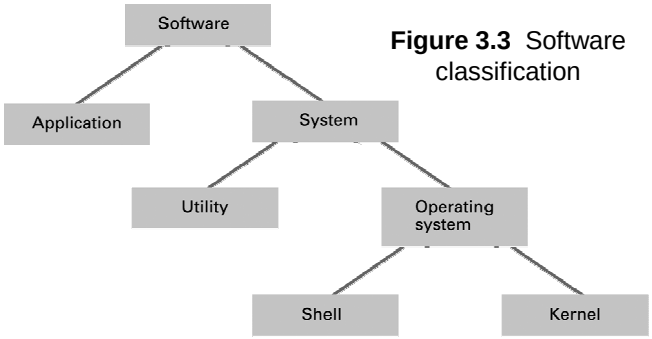
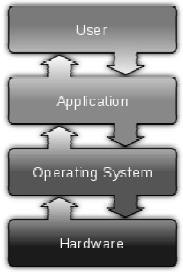


Figure 3.3 Software classification

- Operating system is one kind of software.

Operating system

- One kind of software that controls the overall operation of a computer
  - Unix, Sun Solaris
  - Linux: Ubuntu, Redhat, ...
  - Microsoft Windows
  - Apple Mac OS X
  - Google Chrome OS



Outline

- Components and functions
- Process management
- Handling competition for resources
- Security

Components and Functions

Shell, kernel, file manager,
device drivers, memory manager,
bootstrapping, scheduler, dispatcher

Components of OS

- For **user**: shell, privilege control (security)
- For **data**: file manager
- For **hardware**: device manager, memory manager, and boot manager
- For **software**:
  - Where to store: file manger, registry
  - How to execute: scheduler, process manager

How about procedure?

- OS needs to define a set of rules or working flows for users and hardware/ software developers.
  - For example, you need to *double click* an *icon* to open a program or a file.
  - Design a simple yet useful procedure for a complicated system is not an easy job.
  - This is for books like “How to use computers” to talk about?

Shell

- **Shell**: an interface between users and the operating system
  - Text based
  - Graphical user interface (GUI)
    - Windows, icons, menus, pointers (WIMP)
    - Window manager

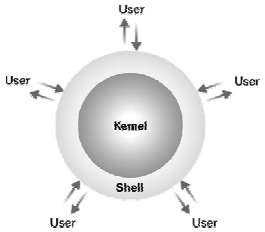
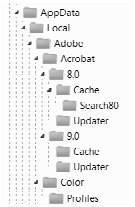


Figure 3.4 The shell as an interface between users and the operating system

File manager

- **File manager**: organizes and maintains the records of files in mass storages
- Hierarchical structure
  - Directory, or folder, directory path
- File descriptor
  - File name, extension, size, updated date, permissions, attributes, …
- File operations
  - Copy, paste, creation, open…

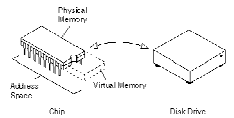


Device manager

- Communicate with the controllers/devices
  - Drive the corresponding peripheral devices
  - Each device driver is uniquely designed for its particular type of device

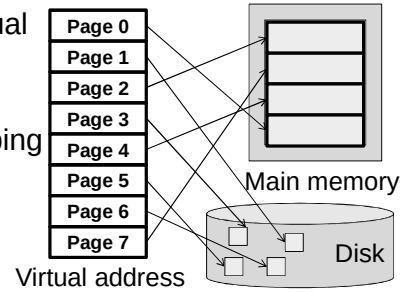
Memory manager

- Coordinates the use of memory
- Virtual memory:
  - Employ the physical memory and disk space
  - Create the *illusion* of a larger memory space
  - To facilitate the mapping, memory is grouped into *pages* (the basic memory unit).
  - Paging: shuffle pages between main memory and disk.



Example of virtual memory

- There 8 pages; each is of 4KB.
  - Main memory is of size 16KB.
  - Programs use virtual address to access data and code
  - OS does the mapping and paging



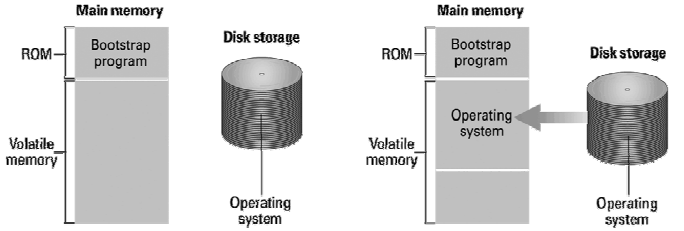
Get it started: bootstrapping

- Loader: a special program places machine programs to main memory for execution
  - Think about the problem 2 of homework 3.
  - Usually part of the OS's scheduler
- Who loads the OS to memory?
  - A "special memory" that contains a "program" to load the OS after computer is powered on.

Read-only memory (ROM) Bootstrap

The booting process

The program counter is initiated with a particular address in ROM where the bootstrap is stored



BIOS and firmware

- The bootstrap program and other basic input/output functions are contained in a special ROM, called **BIOS** (basic input/output system)
- A program stored in ROM is called firmware.
  - Hardware or software?

Process Management

History, today, and future

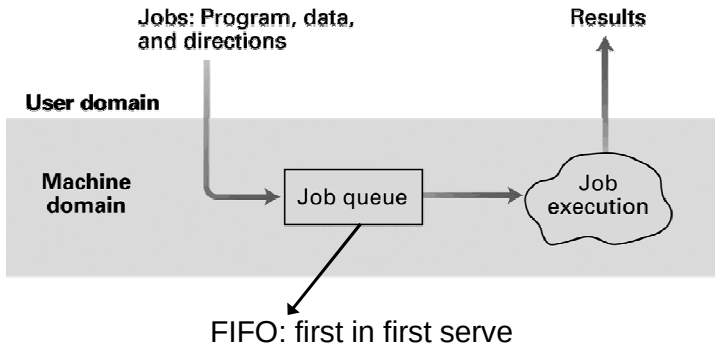
A program vs. a process

- Program: a set of instructions
- Process: the activity of executing a program
- A program can be run multiple times, each instance/activity called a process
- Interprocess communication
  - The communication between processes from running one or more programs

Evolution of shared computing

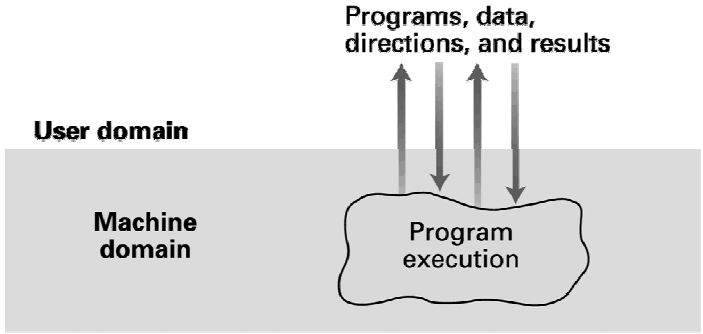
- Batch processing
- Interactive processing: requires real-time processing
- Time-sharing/Multitasking: implemented by Multiprogramming
- New challenges: multicore processors, and small devices

Batch processing



FIFO: first in first serve

Interactive processing



Text editing, music/movie playing, …

Time-sharing/multitasking

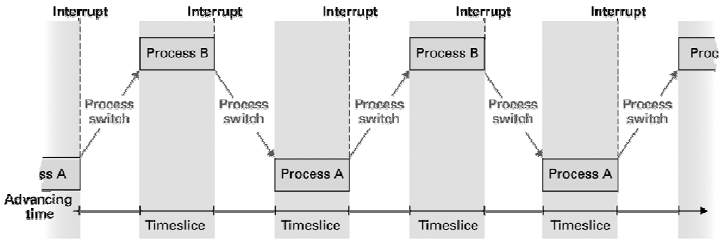


Figure 3.6 Time-sharing between process A and process B

Context (process state)

- Snapshot of the current status of a process
  - A process identifier, or PID
  - Register values, Program Counter value
  - The memory space, I/O, files for the process
  - **State** of the process.
    - Ready: ready for execution.
    - Waiting: waiting for some I/O.
    - Complete: finished process.

Scheduler

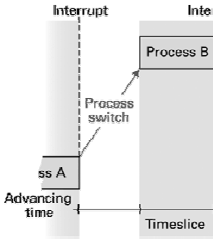
- Determines which processes should be considered for execution based on some priorities or concerns
  - Using process table for administration
- Process table
  - Ready or waiting
  - Priority
  - Non-scheduling information: memory pages, etc.

Dispatcher

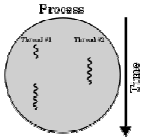
- Gives time slices to a process that is ready
- Executes a context switch when the running process's time slice is over
  - Time slice: a time segment for each execution
  - Interrupt: the signal generated by a hardware timer to indicate the end of a time slice.
  - The Interrupt handler (part of dispatcher) starts after the interrupt to perform context switch

Context switch (process switch)

1. Get an interrupt from timer
2. Go to the interrupt handler
   a. Save the context of process A
   b. Find a process ready to run (Assume that is process B)
   c. Load the context of process B
3. Start (continue) process B



Thread



- A task exist within a process that allows multiple independent instance to be executed concurrently.
  - Multiple threads share resources such as memory, program code, …
  - Each thread has its own program counter, registers, and stack (local memory).
- The context switch of threads is much faster than that of processes.

Exercises

- Suppose an OS allocates time slices in 10 millisecond units and the time required for a context switch is negligible. How many processes can obtain a time slice is one second?
- If it takes one microsecond to perform a context switch and processes use only half of their allotted 10 millisecond time slices, what percent of a CPUs time is spent performing context switches rather than executing processes?

New challenges

- Multicore processor
  - How to assign tasks to processors?
    - Load balance problem
  - How to use processors to handle one task?
    - Parallelization, scaling problem
- Embedded systems, small devices
  - Turkey system: store all programs and data in a persistent memory
  - No BISO and program loader

Handling Competition for Resources

Competition for resources

- What are resources?
  - CPU, memory, files, peripheral devices, …
- In a multitasking system, resources are shared by processes
- Some resources should not be employed by more than one process simultaneously
  - E.g., Printer

Handling competitions

- Define critical regions
  - **Critical Region:** A group of instructions that should be executed by only one process at a time
  - **Mutual exclusion:** Requirement for proper implementation of a critical region

First algorithm

- Use a flag (a global memory address)
  - flag=1: the critical region is occupied
  - flag=0: no process is in the critical region
- Problem:

Process A

```
if (flag == 0) {
   flag = 1;
   /*critical region*/
}
```

Context switch to A

Process B

```
if (flag == 0) {
   flag = 1;
   /*critical region*/
}
```

Context switch to B

  - Both processes get into the critical region

Solutions

- Testing&setting the flag must be completed w/o interruption (atomic)

1. Use disable_Interrupt() to prevent context switch during the flag test and set process.

```
Diable_Interrupt();
if (flag == 0) {
   flag = 1;
   Enable_Interrupt();
   / *critical region*/
}
Enable_Interrupt();
```

2. A machine instruction called “test-and-set” which cannot be interrupted

- Semaphore: a properly implemented flag

Another problem: deadlock

- Example:
  - A is in critical region 1, and waits to enter critical region 2
  - B is in critical region 2, and waits to enter critical region 1

Process A

```
if (test_set(flag1)) {
   /*critical region 1*/
   if (test_set(flag2)) {
      /*critical region 2*/
   }}
```

Context switch to A

Process B

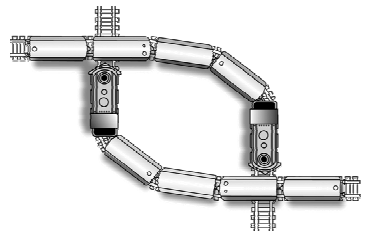
```
if (test_set(flag2)) {
   /*critical region 2*/
   if (test_set(flag1)) {
      /*critical region 1*/
   }}
```

Context switch to B

Conditions for deadlock

1. Competition for non-sharable resources
2. Resources requested on a partial basis
3. An allocated resource can not be forcibly retrieved

Remove any one of the conditions can resolve the deadlock.



Solutions

Which condition is removed?

1. Kill one of the process
2. Process need to request all the required resources at one time
3. Spooling
   - For example, stores the data to be printed and waits the printer available
4. Divide a file into pieces so that it can be altered by different processes

Exercises

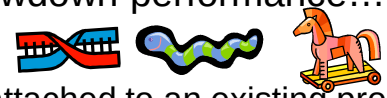
- There is a bridge that only allows one car to pass. When two cars meet in middle, it causes “deadlock”. The following solutions remove which conditions
  1. Do not let a car onto the bridge until the bridge is empty.
  2. If cars meet, make one of them back up.
  3. Add a second lane to the bridge.
- What’s the drawback of solution 1?

Security

Security

- Attacks
  - Malware
  - Spyware and phishing
  - Adware and spam
  - Abnormal behaviors
- Defenses
  - User management
    - Privilege control
  - Protections
    - Antivirus software
    - Auditing software
    - Firewall, spam filter
  - Encryption

Malware

- Infect programs/computers, erase data, slowdown performance…
- Types 
  - Virus: attached to an existing program
  - Worm: a stand alone program
  - Trojan horse: disguised as valid files or programs

Spyware and phishing

- Spyware: collects information about users without their knowledge.
  - Keylogger: log the keys struck on a keyboard
  - Login sniffing: simulates the login process to get valid user name and password.
  - Network sniffing: intercept network messages
- Phishing: acquires information by masquerading as a trustworthy entity in an electronic communication. 

Adware and spam

- Adware: automatically plays, displays, or downloads advertisements to a computer after the software is installed on it or while the application is being used.
- Spam: sends unsolicited bulk messages indiscriminately.
  - Email spam

Abnormal behaviors

- Dictionary attack: trying passwords derived from a list of words in a dictionary.
- Denial of service attack: overloading a computer (server) with messages to make a computer resource unavailable to its intended users.
- Spoofing attack: masquerading as a party other than one's self

User management

- To protect the computer's resource from access by unauthorized personnel.
- User authentication process:
  - Username, password, fingerprint, …
- Super user / administrator / root
  - A kind of user having higher privilege to control machines and operating system.

Privilege control

- To prevent malicious programs to execute dangerous instructions.
- Privilege levels:
  - Nonprivilege mode: only "safe" instructions
    - For example, to access some part of memory.
  - Privilege mode: all kinds of instructions
    - Those instructions that can be only executed in the privilege mode are called **privilege instructions**.

Protections

- **Antivirus software**: detecting and removing the presence of known viruses and other infections.
- **Auditing software**: detecting and preventing abnormal situations
- **Firewall**: filtering messages passing through computers.
  - **Spam filter**: firewall for email spam

Related courses

- Operation system
  - 作業系統，計算機系統管理，平行程式
- Security
  - 計算機系統管理，密碼與網路安全概論

References

- http://www.wikipedia.org/
- Textbook chap3, sec 4.5 (security)