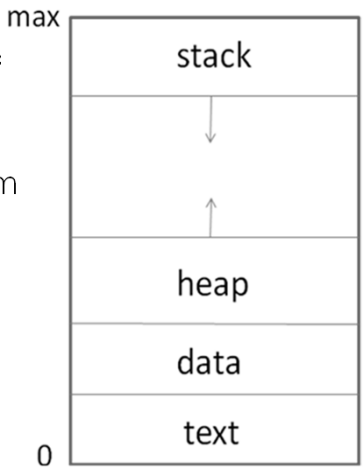
作業系統
Operating Systems

靜宜大學資訊傳播工程學系
劉國有 助理教授
kyliu@pu.edu.tw

Process

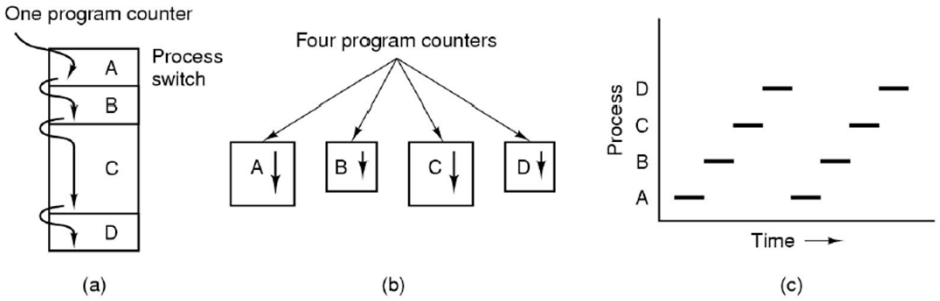
- The most central concept in operating system is the *process*: an abstraction of a running program.
  - Text section: program code
  - Current activity: the value of the program counter and the contents of the processor's registers
  - Stack: contains temporary data (e.g. functions parameters, return addresses, and local variables)
  - Data section: contains global variables
  - Heap: the memory that is dynamically allocated during process run time.



Process

- A program is a passive entity (e.g. an executable file stored on disk)
- A process is an active entity which a program counter specifies the next instruction to execute and a set of associated resources.
- A program becomes a process when an executable file is loaded into memory.
- The CPU is running only one program at any instant of time.
- All modern computers often do several things at the same time.
- In any multiprogramming system, the CPU switches from process to process quickly.
  - At any instant of time, the CPU is running only one process. (pesudoparallelism)

The process Model



- Ex: the difference between a process and a program:
  - Computer scientist: CPU
  - Program 1: birthday cake recipe (input data: cake ingredients)
  - Program 2: first aid book (input data: medicine)

Process Creation

- Events which cause process creation:
  - System initialization
    - When an operating system is booted (how?), several processes are created.
      - Foreground processes: interact with users and perform work for them.
      - Background processes (daemons): handle some activity in the background (e.g. web server).
  - Execution of a process creation system call by a running process
    - Network communication: one process to fetch the data and put them in a shared buffer while another process removes the data and processes them
  - A user request to create a new process
    - Users can start a program by typing a command or clicking an icon
  - Initiation of a batch job
    - Only to the batch systems found on large mainframes

System boot process

- Before any computer can perform useful work, it must first initialize itself using a process known as *bootstrapping*
- BIOS (Basic Input/Output System): holds the instructions needed by the PC to get things started
  - execute the POST (Power On Self Test) which checks the video card (for display error message)
  - then checks that the major input devices (e.g. keyboard, mouse) are connected and functional
  - to verify that the RAM (volatile storage) in the computer works
  - checks the main storage devices connected to the system
  - starts looking for an operating system to load

System boot process

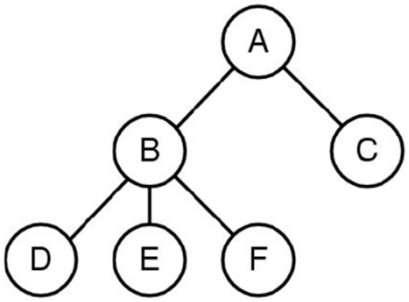
- Search for a drive to BOOT from
  - Floppy or Hard disk
    - Boot at cylinder 0, head 0, sector 1
- Load code in boot sector (Master Boot Record, MBR)
- Execute boot loader (bootstrapping)
- Boot loader loads program to be booted
  - If no OS: "Non-system disk or disk error - Replace and press any key when ready"
- Transfer control to loaded program (i.e., operating system)

Process Termination

- Events which cause process termination:
  - Normal exit (voluntary): the processes have done their work
  - Error exit (voluntary): the process discovers a fatal error (e.g., no such file exists)
  - Fatal error (involuntary): an error caused by the process, often due to a program bug (e.g., dividing by zero)
  - Killed by another process (involuntary): the process executes a system call telling the operating system to kill some other process

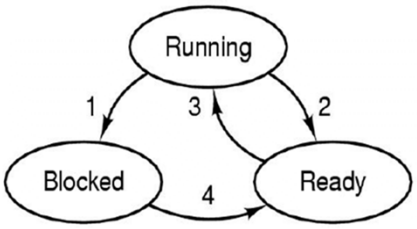
Process Hierarchies

- In some systems, when a process creates another process, the parent process and child process continue to be associated in certain ways
- The child process can itself create more processes, forming a process hierarchy



Process States

- State diagram

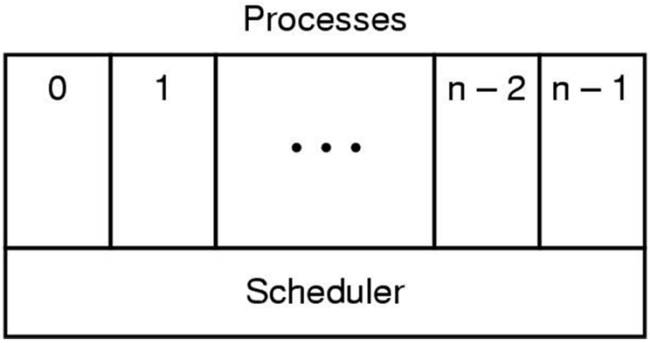


1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

- Running: actually using the CPU at that instant.
- Blocked: unable to run until some external event happens.
- Ready: runnable; temporarily stopped to let another process run.
- Ex: cat ch1 ch2 ch3 | grep tree

Process States

- Lowest layer of a process-structured operating system handles interrupts and scheduling.

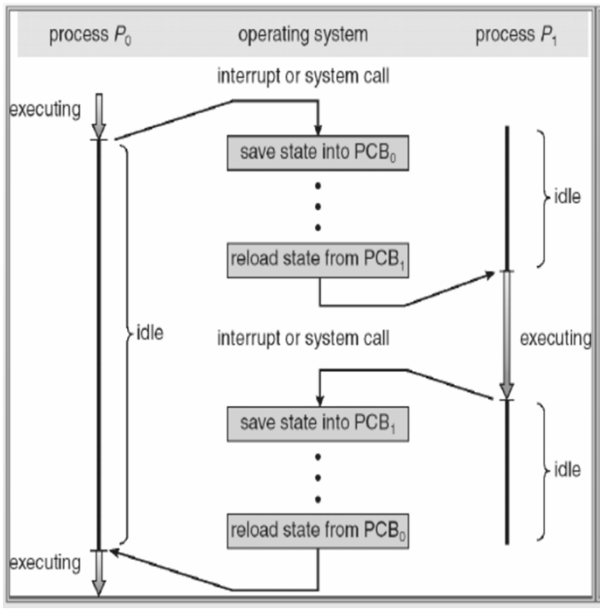


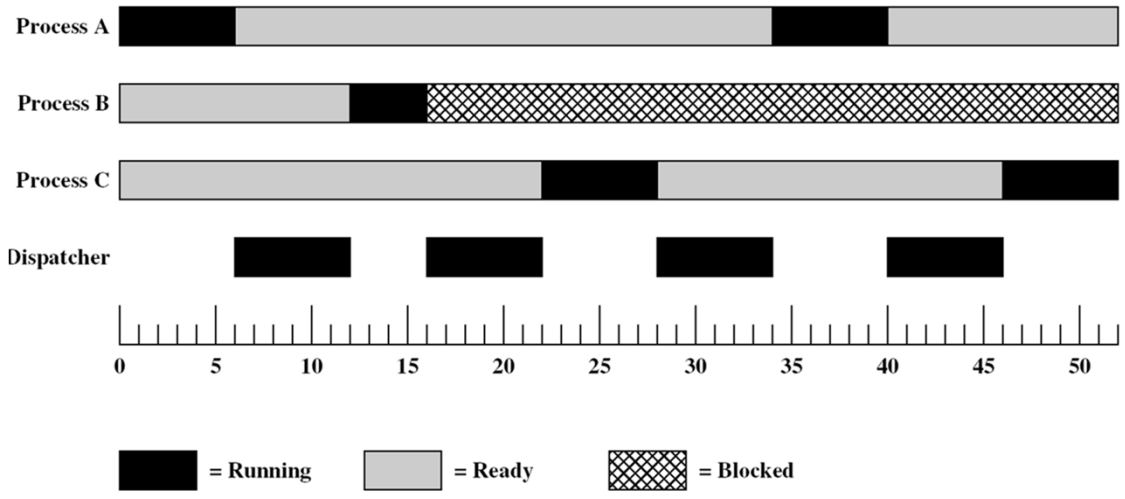
Implementation of Process

- OS maintains a table, called the *process table*, with one entry per process (*process control blocks, PCB*)

Process management	Memory management	File management
Registers	Pointer to text segment	UMASK mask
Program counter	Pointer to data segment	Root directory
Program status word	Pointer to bss segment	Working directory
Stack pointer	Exit status	File descriptors
Process state	Signal status	Effective uid
Time when process started	Process id	Effective gid
CPU time used	Parent process	System call parameters
Children's CPU time	Process group	Various flag bits
Time of next alarm	Real uid	
Message queue pointers	Effective uid	
Pending signal bits	Real gid	
Process id	Effective gid	
Various flag bits	Bit maps for signals	
	Various flag bits	

How the multiple processes is maintained on one CPU?

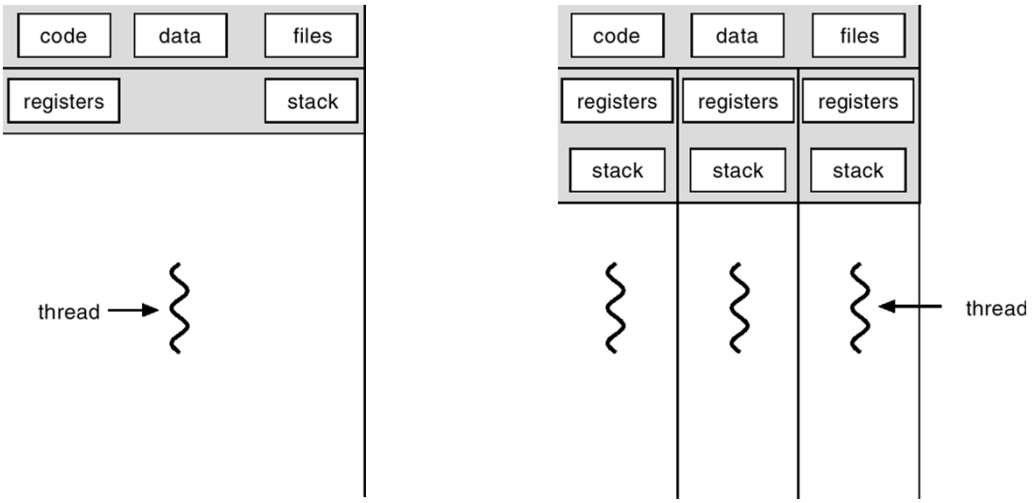




Threads

- In traditional operating systems, each process has an address space and a single thread of control
- Actually, it is desirable to have multiple threads of control in the same address space running in quasi-parallel
- Instead of owning an address space, the thread only has its own program counter, registers, and stack.

Per process items	Per thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

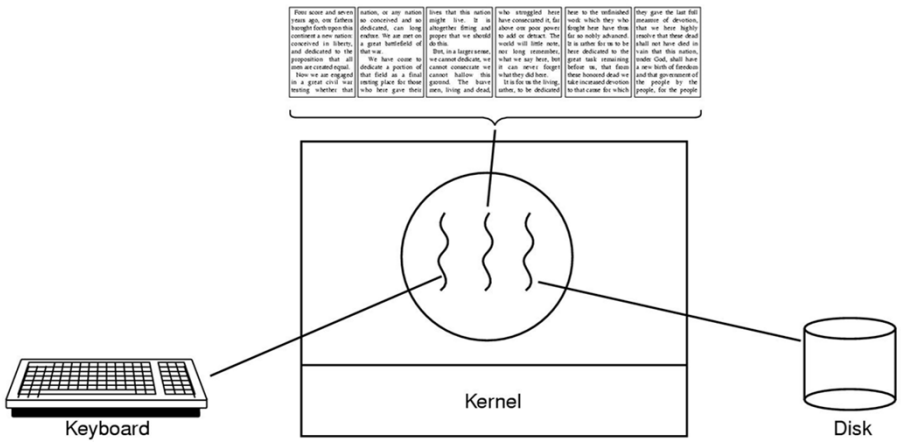


Thread Usage

- Three reasons for having threads
  - in many applications, multiple activities are going on at once
  - threads are easier (i.e., faster) to create and destroy than processes
  - speeding up the application

Thread Usage

- A word processor with three threads



Thread Usage

- A multithreaded Web server