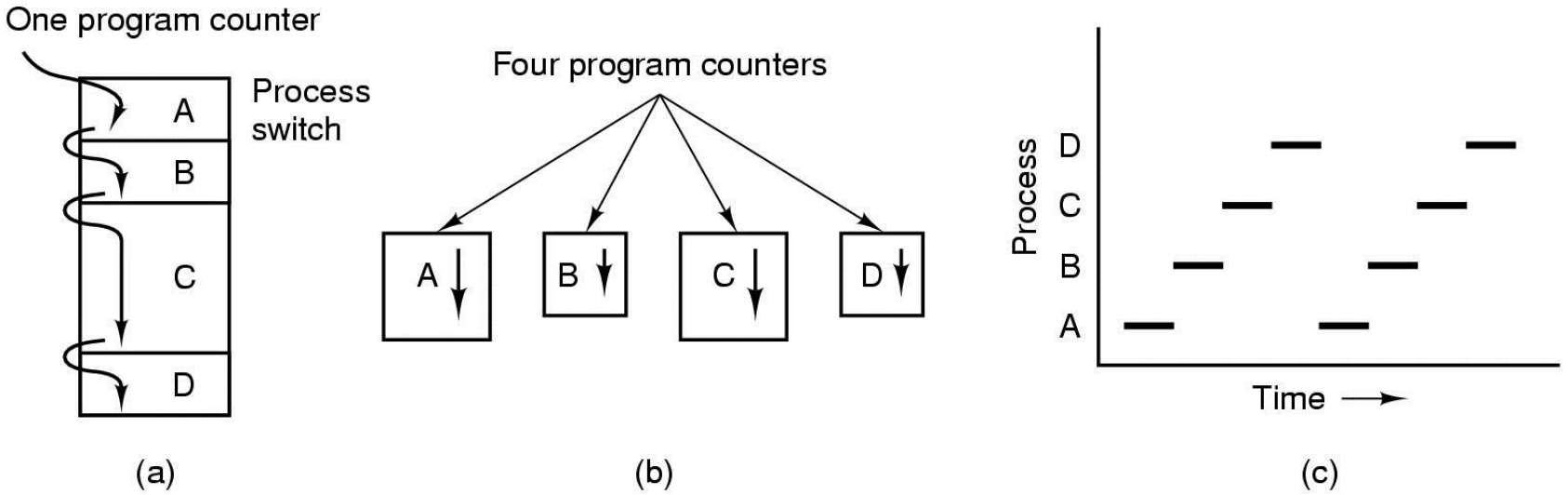
Chapter 2

Processes and Threads

2.1 Processes
2.2 Threads
2.3 Interprocess communication
2.4 Classical IPC problems
2.5 Scheduling

The Process Model

- A process is an executing program
- Multiprogramming
  - The CPU switches from process to process
  - (a) Multiprogramming of four programs
  - (b) Conceptual model of 4 independent processes
  - (c) Only one program active at any instant

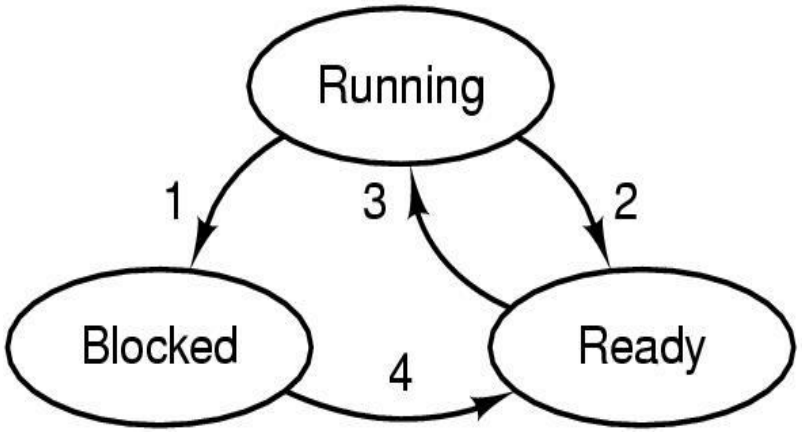


Process

- Program in execution
- Each process has
  - address space (core image)
  - process table entry
    - Contains information about the process to be saved for later run

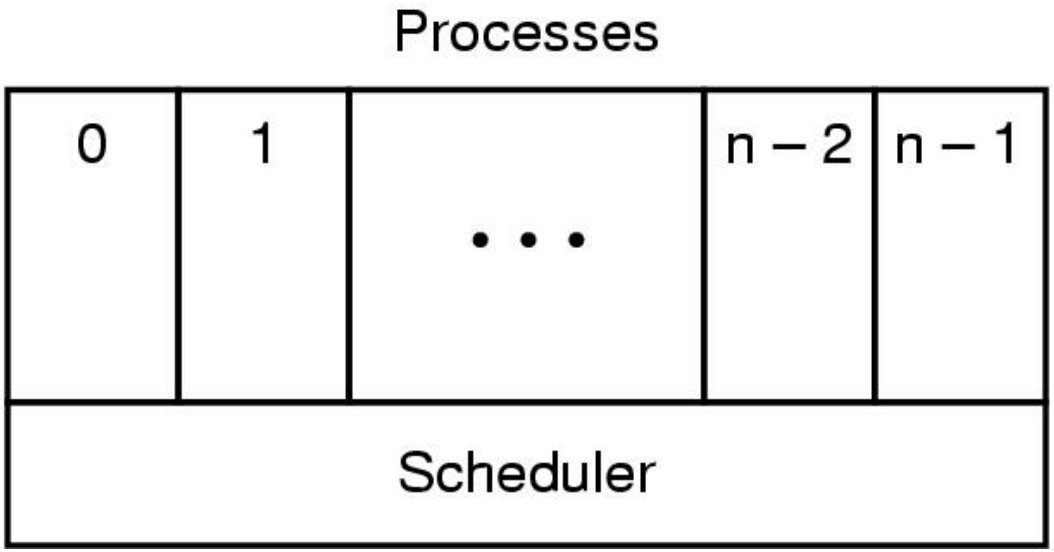
Process States (1)

- State diagram showing the possible three process states
  - Running
    - Actually using the CPU at that instant
  - Ready
    - Runnable; temporarily stopped to let another process run
    - The CPU needs to be shared among processes.
  - Blocked
    - Unable to run until some external event happens such as the disk has been read or the character typed.
    - The suspension is inherent in the problem.



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

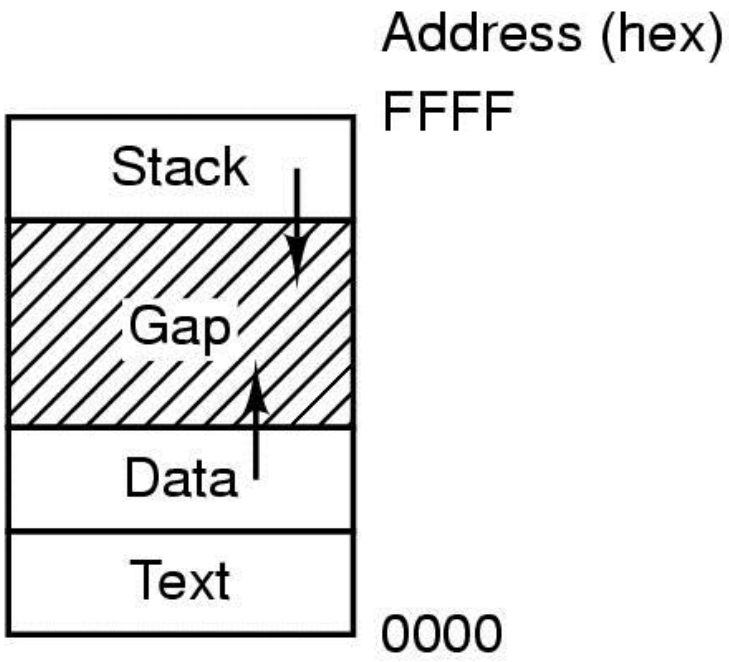
Process States (2)



- Lowest layer of process-structured OS
  - handles interrupts, scheduling
- Above that layer are sequential processes

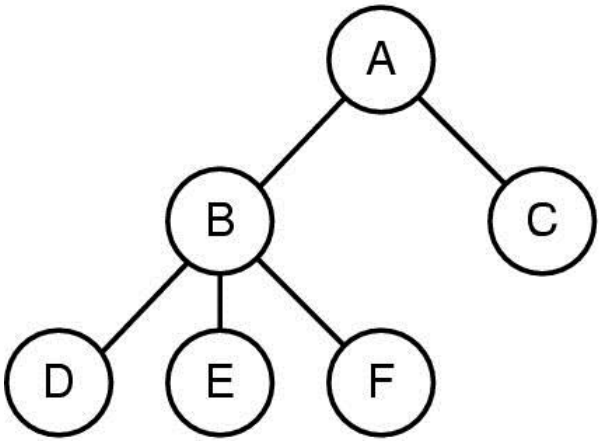
Address Space of a Process

- A process has three segments in memory
  - text
  - data
  - stack



Process Hierarchies

- Parent creates a child process, child processes can create its own process, forming a process hierarchy
- A process tree
  - A created two child processes, B and C
  - B created three child processes, D, E, and F
  - E.g. process tree with init at the root



- In UNIX, a process and its children form a process group.
  - processes created in the current window
- Windows has no concept of process hierarchy
  - all processes are created equal

Protection

- A process started has the UID (User Identification) of the person who started it.
  - Each person authorized to use a system is assigned a UID (User Identification) by the system administrator.
  - A child process has the same UID as its parent.
  - One UID, called the superuser (in UNIX), has special power.
- Users can be members of groups, each of which has a GID(Group Identification).

Process Table

- process table
  - One entry per process (also called process control block)
  - Contains information about the process' state, its program counter, stack pointer, memory allocation, the status of its open files, its accounting and scheduling information, and everything else about the process that must be saved when the process is switched from running to ready or blocked state so that it can be restarted later as if it had never been stopped.

Process management	**Memory management**	**File management**
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Process Creation

- A new process can only be created by having an existing process execute a process creation system call.
- Principal events that cause process creation
  - System initialization
    - Daemons
      - Processes in the background that handle some activity such as email, Web pages, news, printing, etc.
  - Execution of a process creation system call by a running process
    - fork system call
  - User request to create a new process
    - Typing a command or clicking an icon
  - Initiation of a batch job
    - large mainframes

Process Termination

- Conditions which terminate processes
  - Normal exit (voluntary)
    - exit
  - Error exit (voluntary)
    - Bad input parameters
  - Fatal error (involuntary)
    - Often due to a program bug
    - Illegal instructions, referencing nonexistent memory, dividing by zero
  - Killed by another process (involuntary)
    - kill

System Calls for Process Management

- Process creation/termination
- Overlay its program with a different one
- Wait for a child process to terminate
- Request more memory/release unused memory

System Calls for Process Management

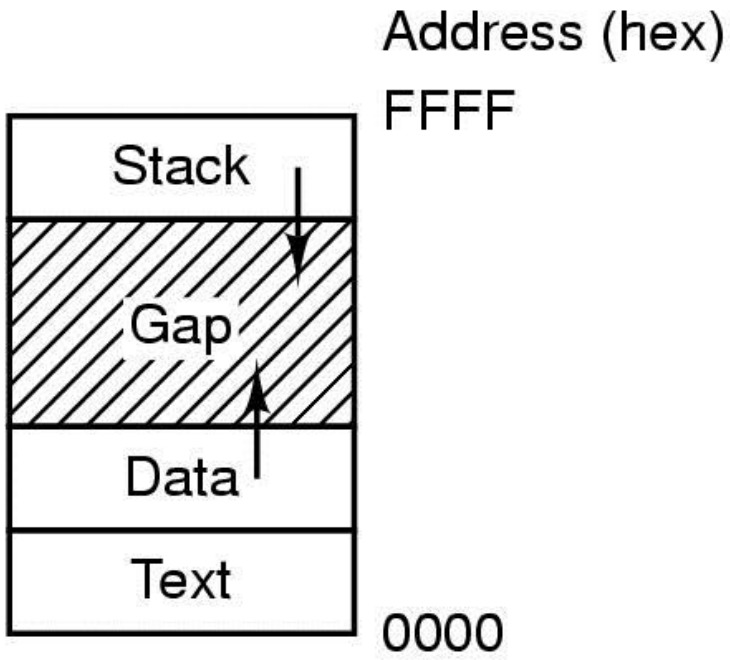
- A stripped down shell using process management system calls

```
while (TRUE) {                                          /* repeat forever */
    type_prompt( );                                     /* display prompt */
    read_command (command, parameters)                  /* input from terminal */

if (fork() != 0) {                                      /* fork off child process */
    /* Parent code */
    waitpid( -1, &status, 0);                           /* wait for child to exit */
} else {
    /* Child code */
    execve (command, parameters, 0);                    /* execute command */
 }
}
```

System Calls for Process Management

- brk system call specifies the new address where the data segment is to end.
  - malloc library procedure dynamically allocates storage



Handling of Interrupt

- When an interrupt occurs:

1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.