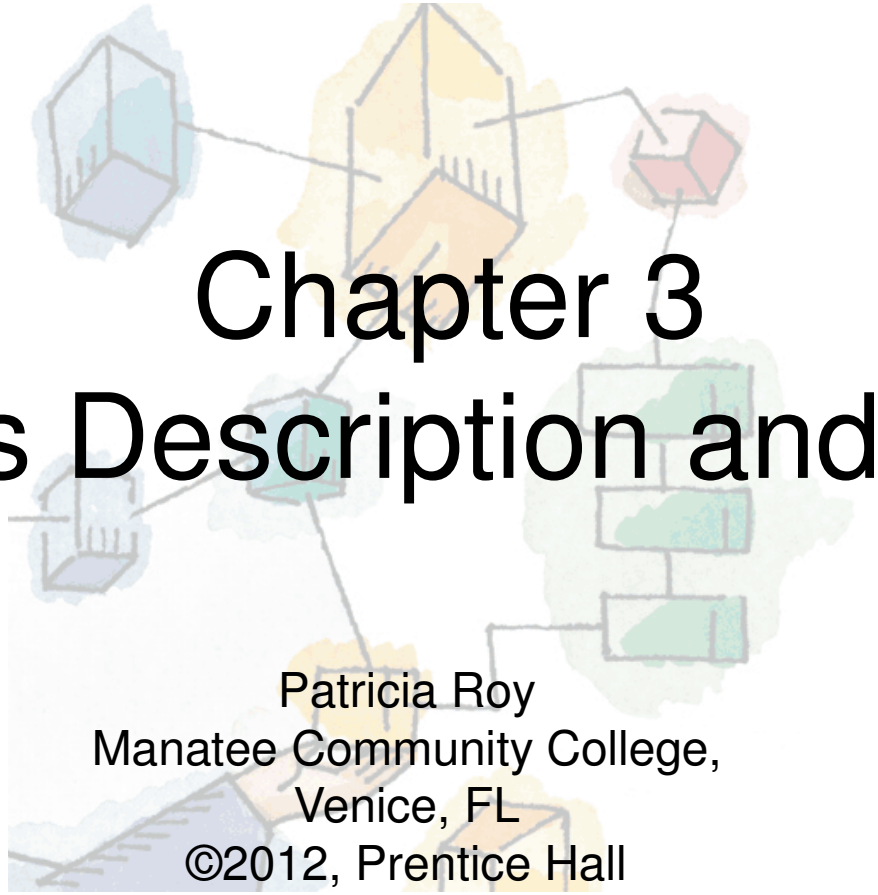
Operating Systems:
Internals and Design Principles, 8/E
William Stallings

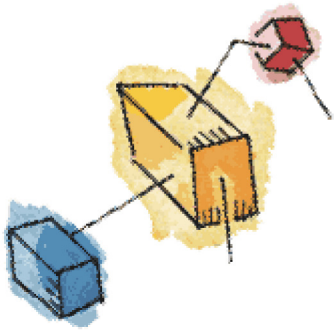


Chapter 3
Process Description and Control

Patricia Roy
Manatee Community College,
Venice, FL

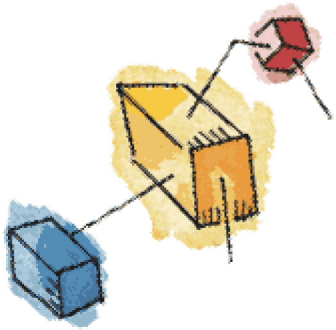
Outline

- 3.1 What is a Process?
- 3.2 Process State
- 3.3 Process Description
- 3.4 Process Control
- 3.5 Execution of the OS
- 3.6 Security Issues
- 3.7 UNIX SVR4 Process Management



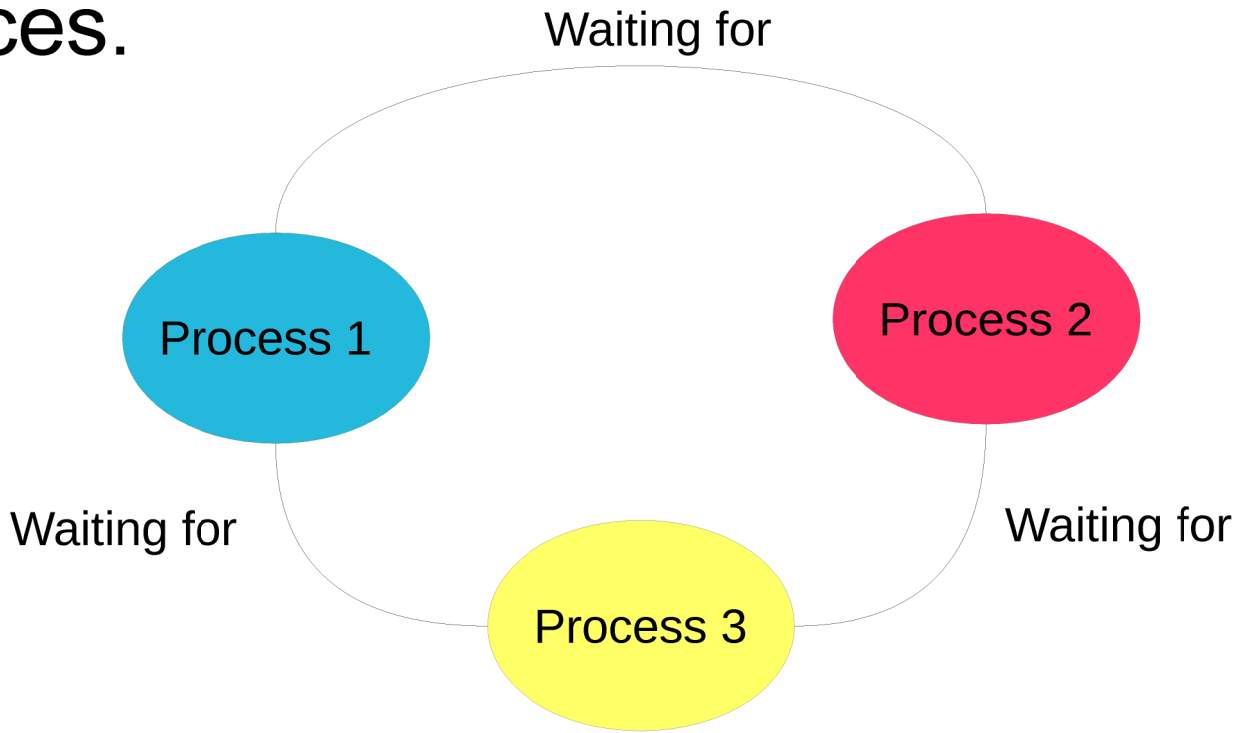
Requirements of an OS

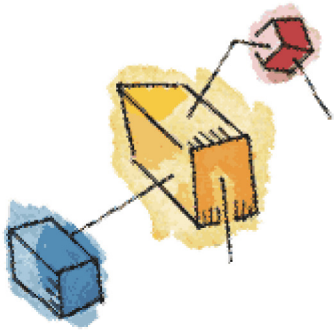
- Interleave the execution of multiple processes, to maximize processor utilization while providing reasonable response time
- Allocate resources to processes, while avoiding deadlock
- Support interprocess communication and user creation of processes



What is a deadlock??

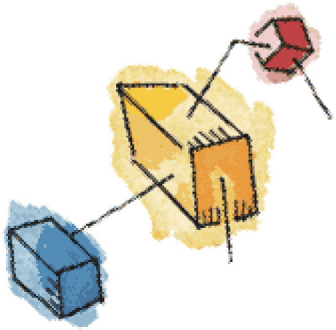
- A situation where 2 or more processes are waiting for each other to release resources.
-





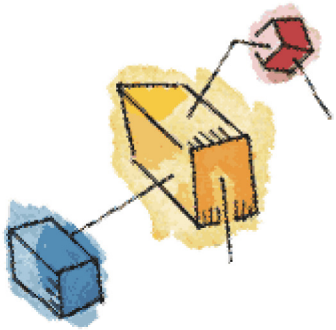
How the OS Manages the Execution of Applications

- Resources made available to multiple applications
- Processor is switched among multiple applications, so all appear to be progressing
- The processor and I/O devices can be used efficiently



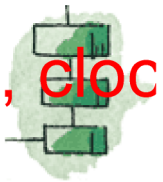
Process: The Definition

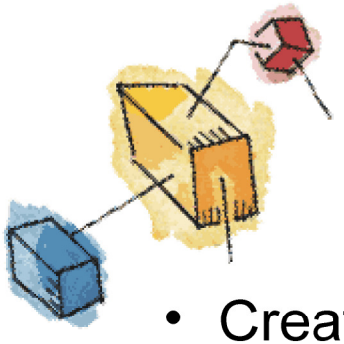
- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system resources



Process Elements

- Identifier
- State – executing → running state
- Priority
- Program counter (PC)
- Memory pointers – to program code & data
- Context data – data presents in registers
- I/O status information – I/O request, I/O devices, etc
- Accounting information – processor time, cloc time used, time limits, acc numbers





Process Control Block (PCB)

- Created and managed by OS
- Contains the process elements
  - Information in the preceding list stored in a data structure
- A key tool to support multiple processes
- Contains sufficient information possible for interrupt
- Process = Program code + data + PCB

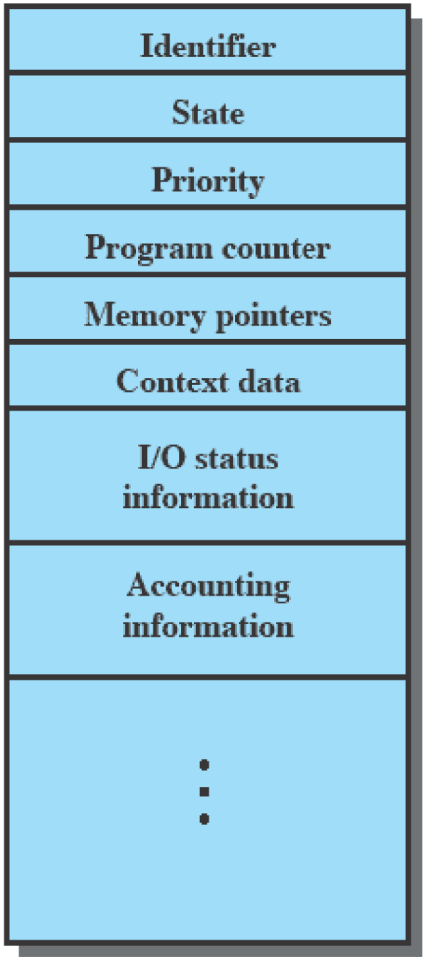
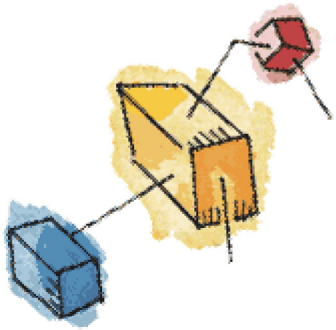


Figure 3.1 Simplified Process Control Block

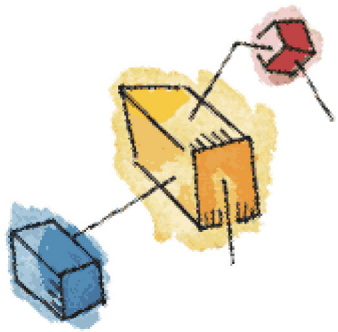


Process States

- A process is created for a program to be executed
- Processor executes instruction from its repertoire (list) in some sequence by changing the value in PC

Trace

- Characterize the behavior of an individual process by:
  - Listing the sequence of instructions that execute a process (called trace)
- Dispatcher switches the processor from one process to another



Example : Dispatcher

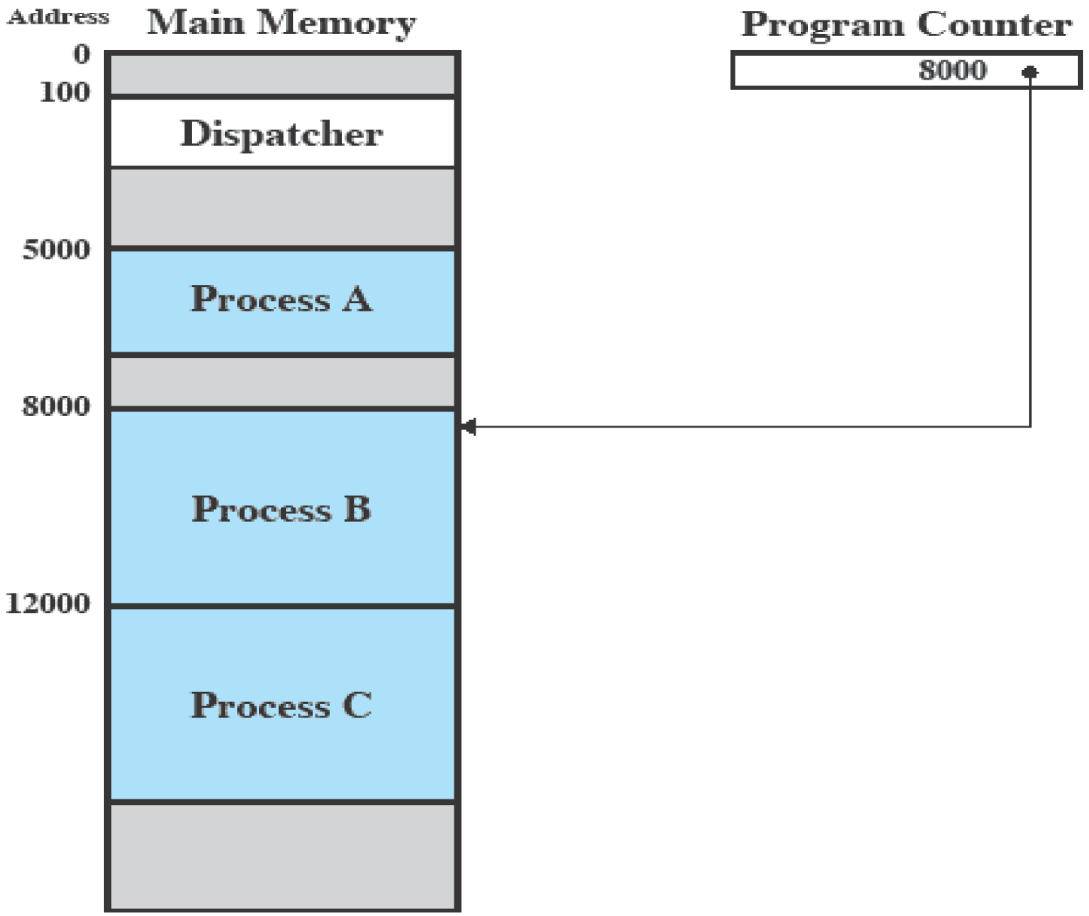
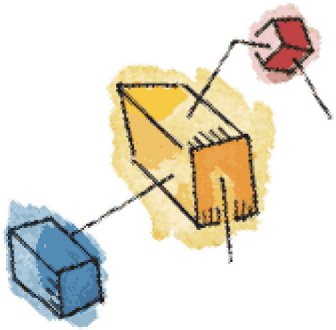


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13



Traces of Processes

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Figure 3.3 Traces of Processes of Figure 3.2

Combined Trace of Processes

Code executed by dispatcher

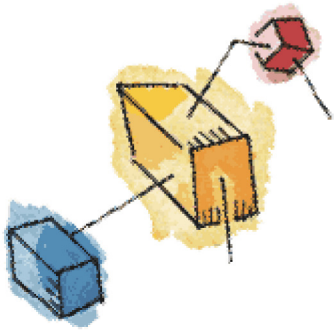
1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
----------	Timeout
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
----------	I/O Request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003

27	12004
28	12005
----------	Timeout
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
----------	Timeout
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
----------	Timeout

100 = Starting address of dispatcher program

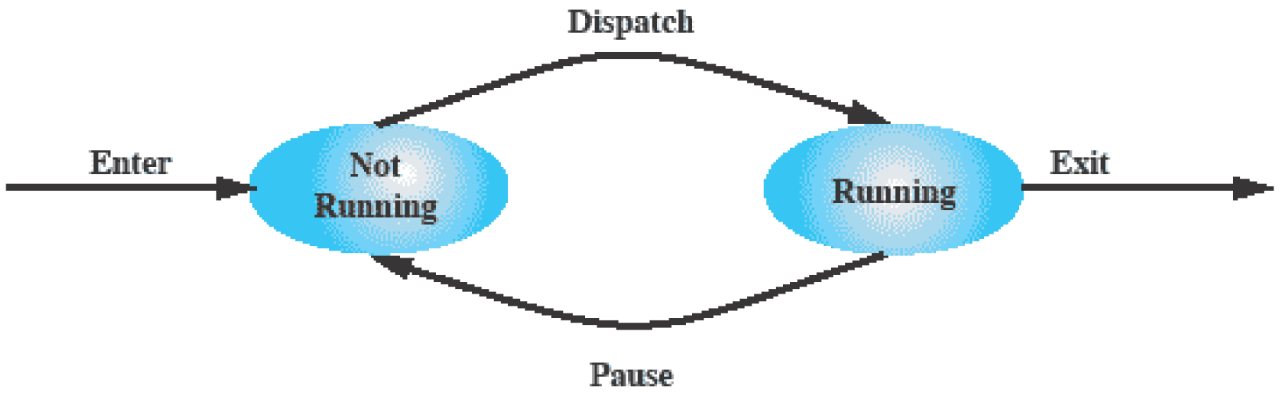
Shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Figure 3.4 Combined Trace of Processes of Figure 3.2

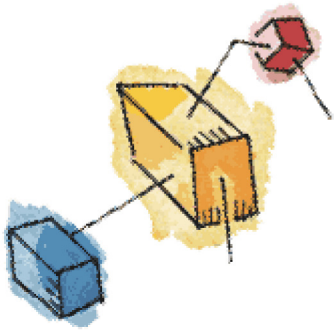


Two-State Process Model

- Process may be in one of two states
  - Running
  - Not-running – waiting for an opportunity to execute



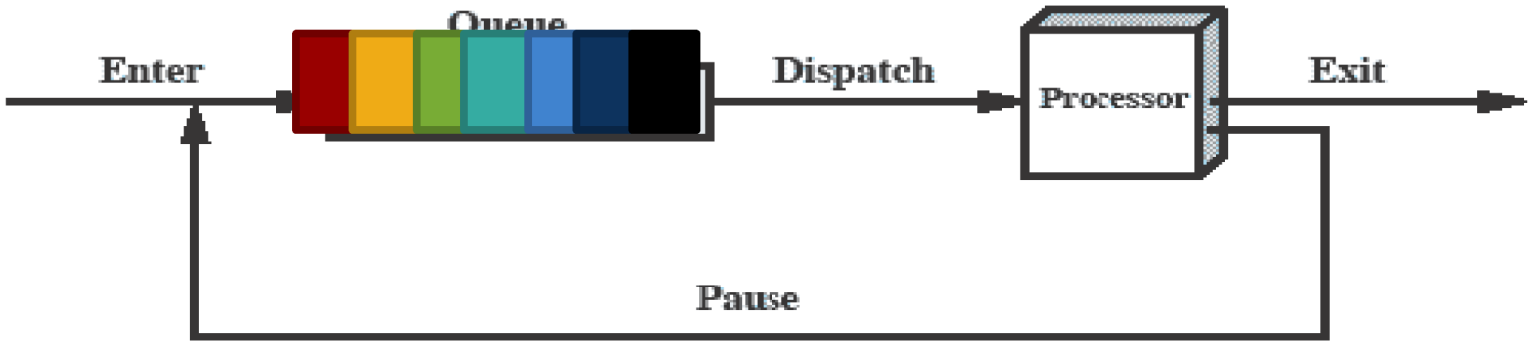
(a) State transition diagram



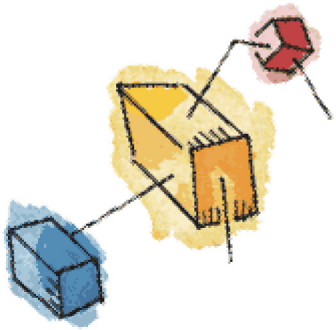
Queuing Diagram

(Round-Robin Fashion)

- For Not-Running State
- Waiting for turn to execute



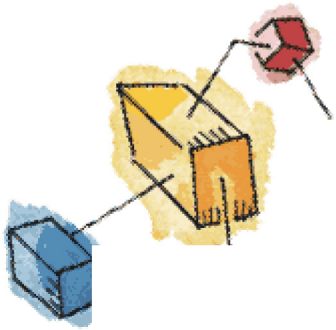
(b) Queuing diagram



Process Creation

Table 3.1 Reasons for Process Creation

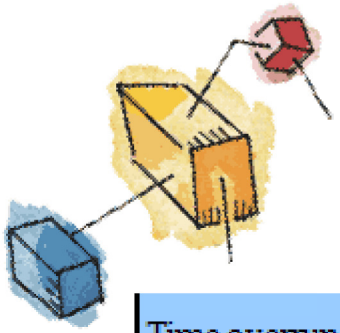
New batch job	The OS is provided with a batch job control stream, usually on tape or disk. When the OS is prepared to take on new work, it will read the next sequence of job control commands.
Interactive logon	A user at a terminal logs on to the system.
Created by OS to provide a service	The OS can create a process to perform a function on behalf of a user program, without the user having to wait (e.g., a process to control printing).
Spawned by existing process	For purposes of modularity or to exploit parallelism, a user program can dictate the creation of a number of processes.



Process Termination

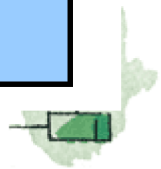
Table 3.2 Reasons for Process Termination

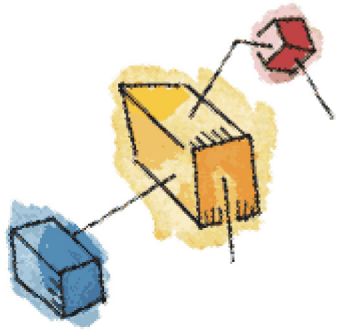
Normal completion	The process executes an OS service call to indicate that it has completed running.
Time limit exceeded	The process has run longer than the specified total time limit. There are a number of possibilities for the type of time that is measured. These include total elapsed time ("wall clock time"), amount of time spent executing, and, in the case of an interactive process, the amount of time since the user last provided any input.
Memory unavailable	The process requires more memory than the system can provide.
Bounds violation	The process tries to access a memory location that it is not allowed to access.
Protection error	The process attempts to use a resource such as a file that it is not allowed to use, or it tries to use it in an improper fashion, such as writing to a read-orly file.
Arithmetic error	The process tries a prohibited computation, such as division by zero, or tries to store numbers larger than the hardware can accommodate.



Process Termination

Time overrun	The process has waited longer than a specified maximum for a certain event to occur.
I/O failure	An error occurs during input or output, such as inability to find a file, failure to read or write after a specified maximum number of tries (when, for example, a defective area is encountered on a tape), or invalid operation (such as reading from the line printer).
Invalid instruction	The process attempts to execute a nonexistent instruction (often a result of branching into a data area and attempting to execute the data).
Privileged instruction	The process attempts to use an instruction reserved for the operating system.
Data misuse	A piece of data is of the wrong type or is not initialized.
Operator or OS intervention	For some reason, the operator or the operating system has terminated the process (for example, if a deadlock exists).
Parent termination	When a parent terminates, the operating system may automatically terminate all of the offspring of that parent.
Parent request	A parent process typically has the authority to terminate any of its offspring.





Five-State Model

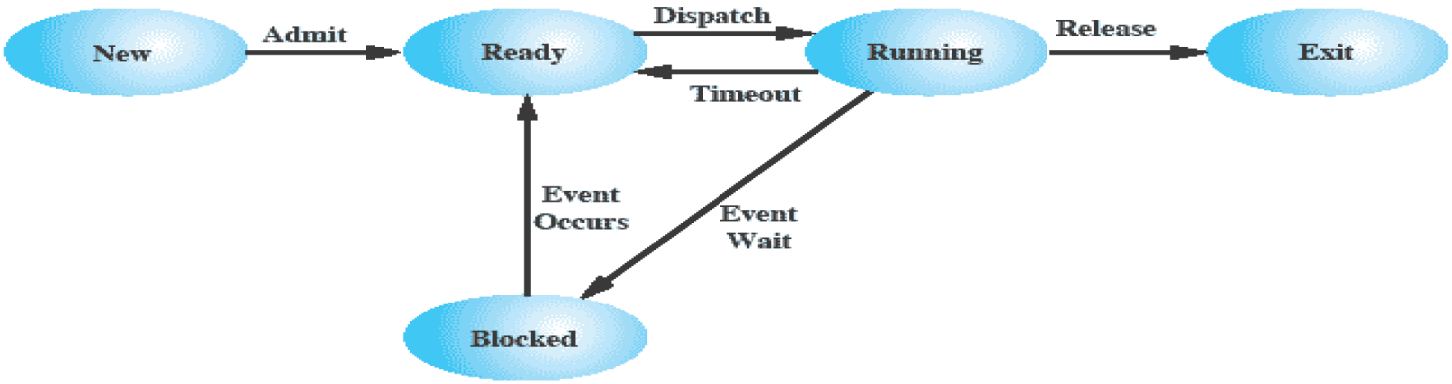
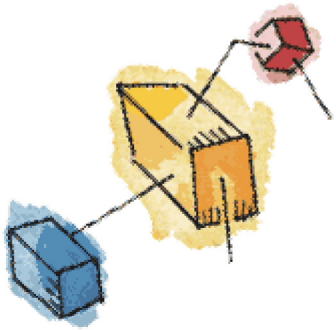


Figure 3.6 Five-State Process Model

If all processes are always ready to execute → Round-Robin (RR) is suitable

If some are ready, while some are blocked → RR is inadequate:

- Some processes in the Not-Running state is ready to be executed, while others are blocked waiting for the I/O to finish
- Dispatcher has to scan the list looking for the unblocked process

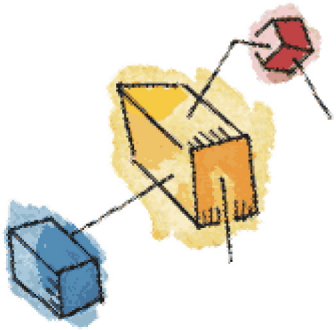


Related Commands

Following tables most commonly used command(s) with process:

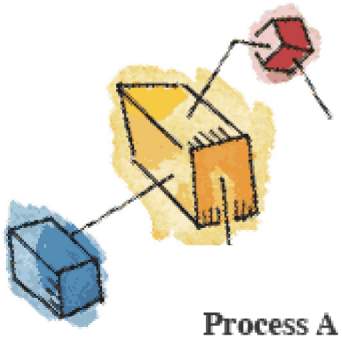
For this purpose	Use this Command	Examples*
To see currently running process	ps	**$ ps**
To stop any process by PID i.e. to kill process	kill {PID}	**$ kill 1012**
To stop processes by name i.e. to kill process	killall {Process-name}	**$ killall httpd**
To get information about all running process	ps -ag	**$ ps -ag**
To stop all process except your shell	kill 0	**$ kill 0**
For background processing (With &, use to put particular command and program in background)	linux-command &	**$ ls / -R | wc -l &**
To display the owner of the processes along with the processes	ps aux	**$ ps aux**
To see if a particular process is running or not. For this purpose you have to use ps command in combination with the grep command	ps ax | grep process-U-want-to see	For e.g. you want to see whether Apache web server process is running or not then give command **$ ps ax | grep httpd**
To see currently running processes and other information like memory and CPU usage with real time updates.	top See the output of top command.	**$ top** **Note** that to exit from top command press q.
To display a tree of processes	pstree	**$ pstree**

***** To run some of this command you need to be root or equivalnt user.



Condition for Five-State Model

- Processes
  - Not-running ready to execute
  - Not-running blocked
  - Dispatcher must scan list to find process not-running, ready, and in queue the longest



Process States

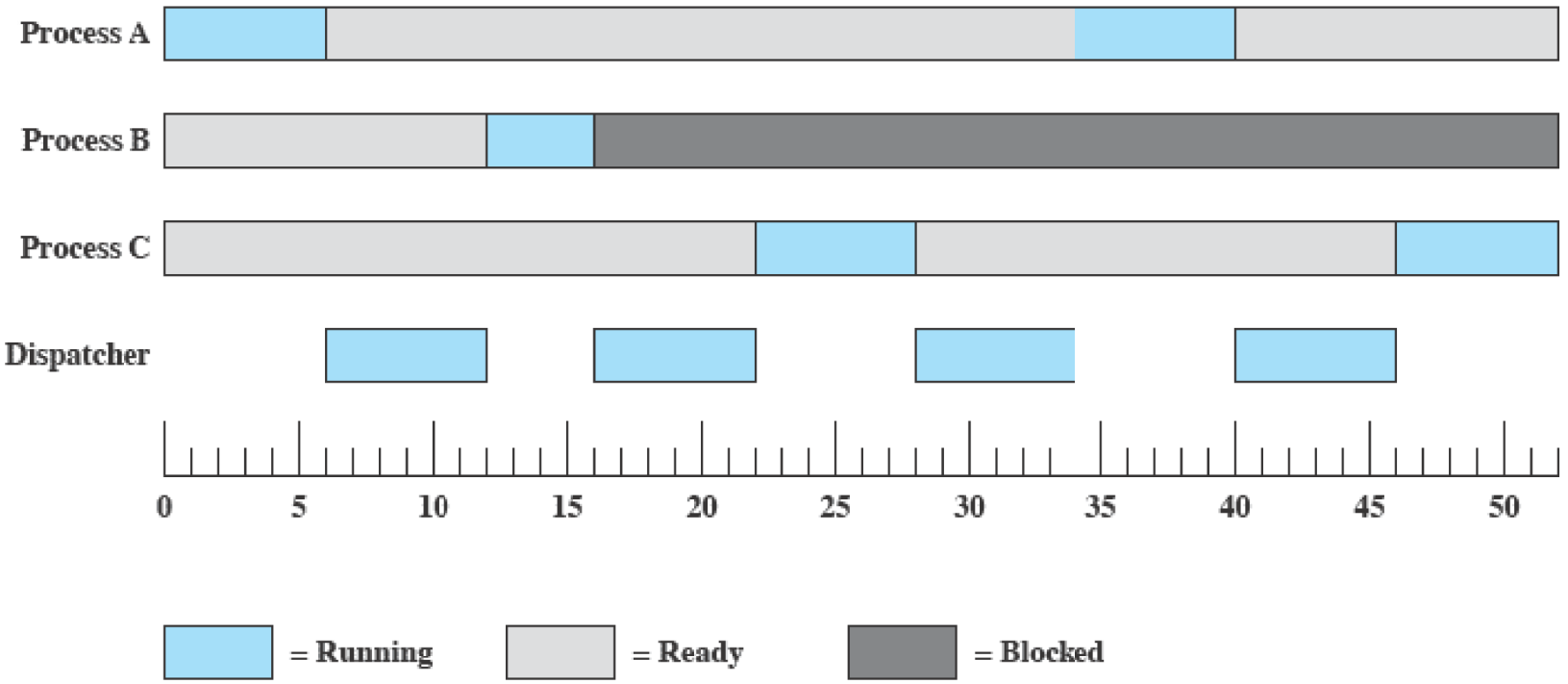
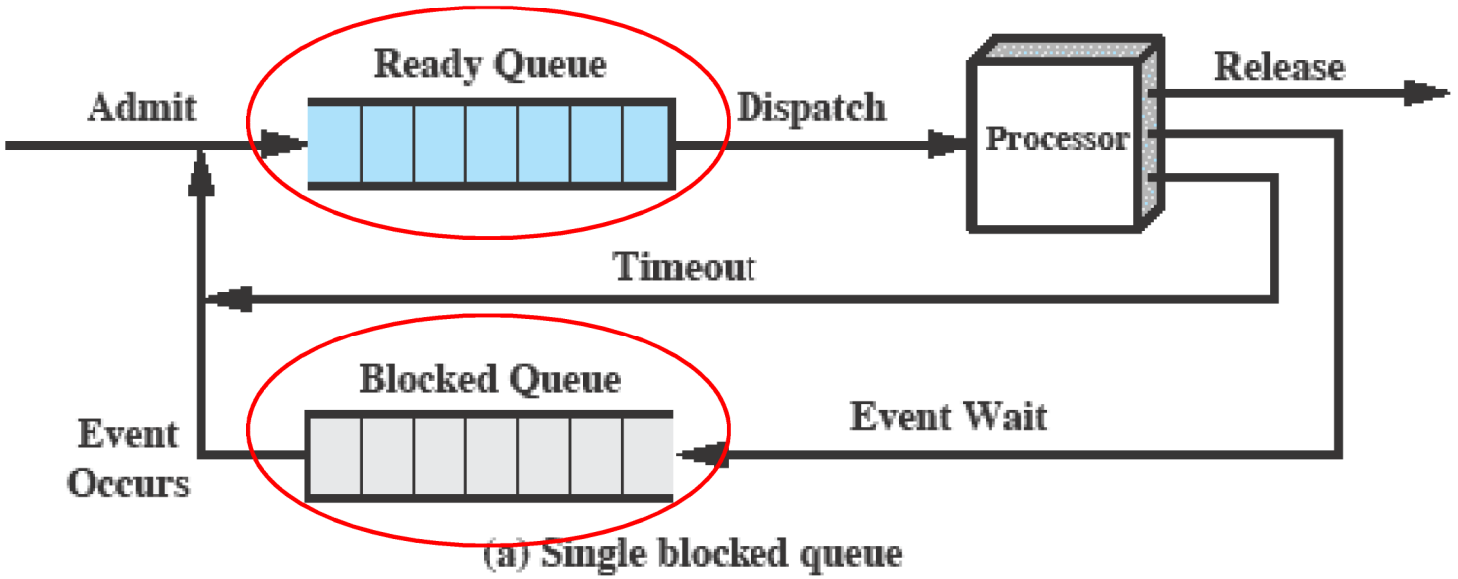


Figure 3.7 Process States for Trace of Figure 3.4

Using Two Queues

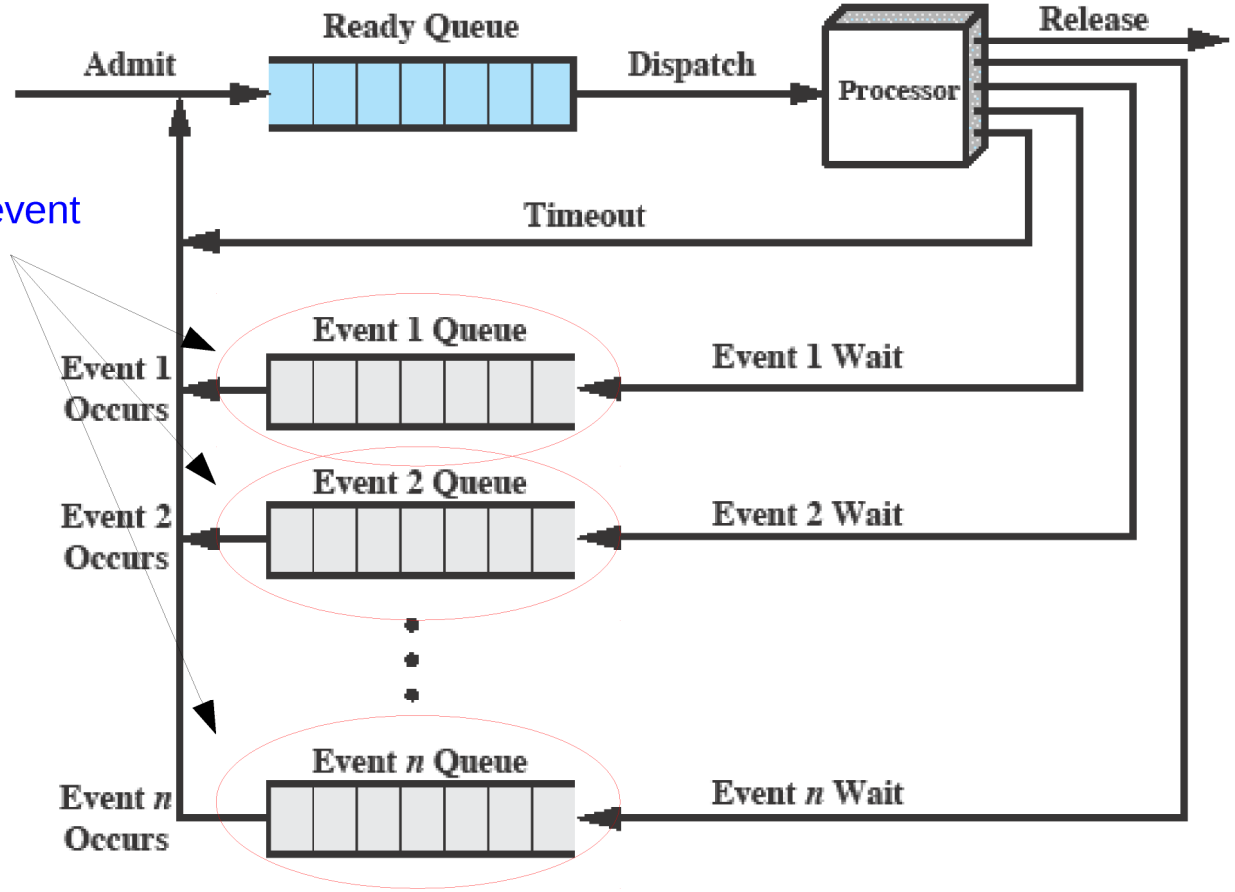


(a) Single blocked queue

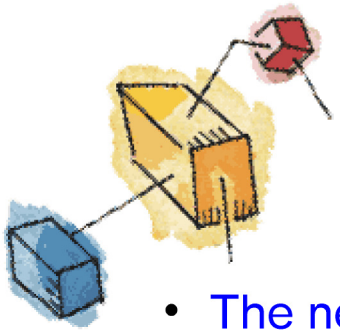
- If no piority → FIFO
- Single blocked queue = When an event occurs, the OS must scan an entire blocked queue searching for those processes waiting for that event
- Not efficient for large OS (hundreds to thousands processes)

Multiple Blocked Queues

One for each event

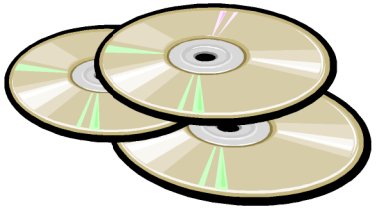


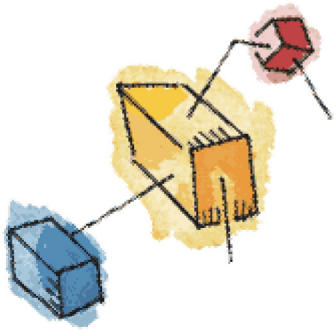
(b) Multiple blocked queues



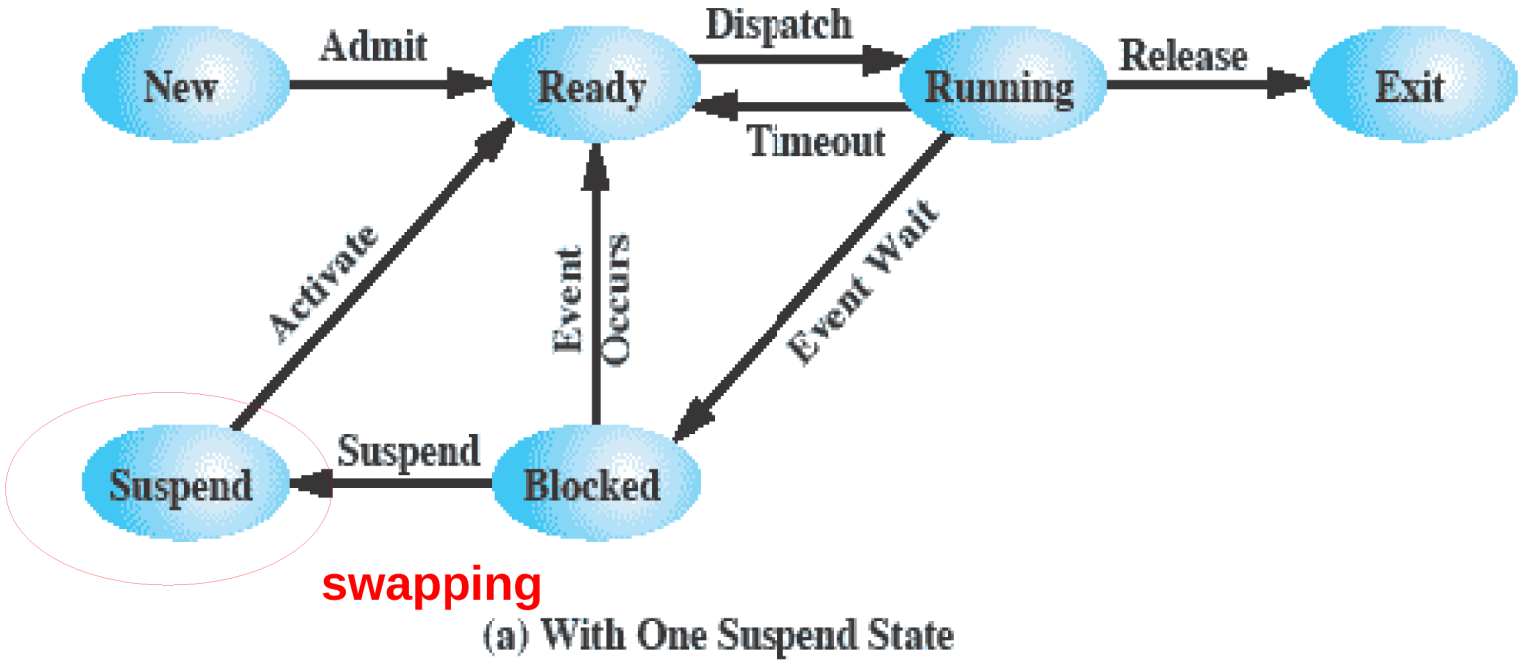
Suspended Processes

- The need for Swapping:
  - Processor is faster than I/O so all processes could be waiting for I/O
    - Could be idle most of the time
    - Possible solution – expand the main memory???
      - Cost
      - Larger memory, larger processes, NOT more processes
  - Swapping → moving part or all processes from main memory to disk to free up more memory
  - When none of processes in main memory is in Ready state, Blocked state becomes Suspend state when swapped to disk
    - The OS then bring a new process
  - Two new states
    - Blocked/Suspend
    - Ready/Suspend





One Suspend State



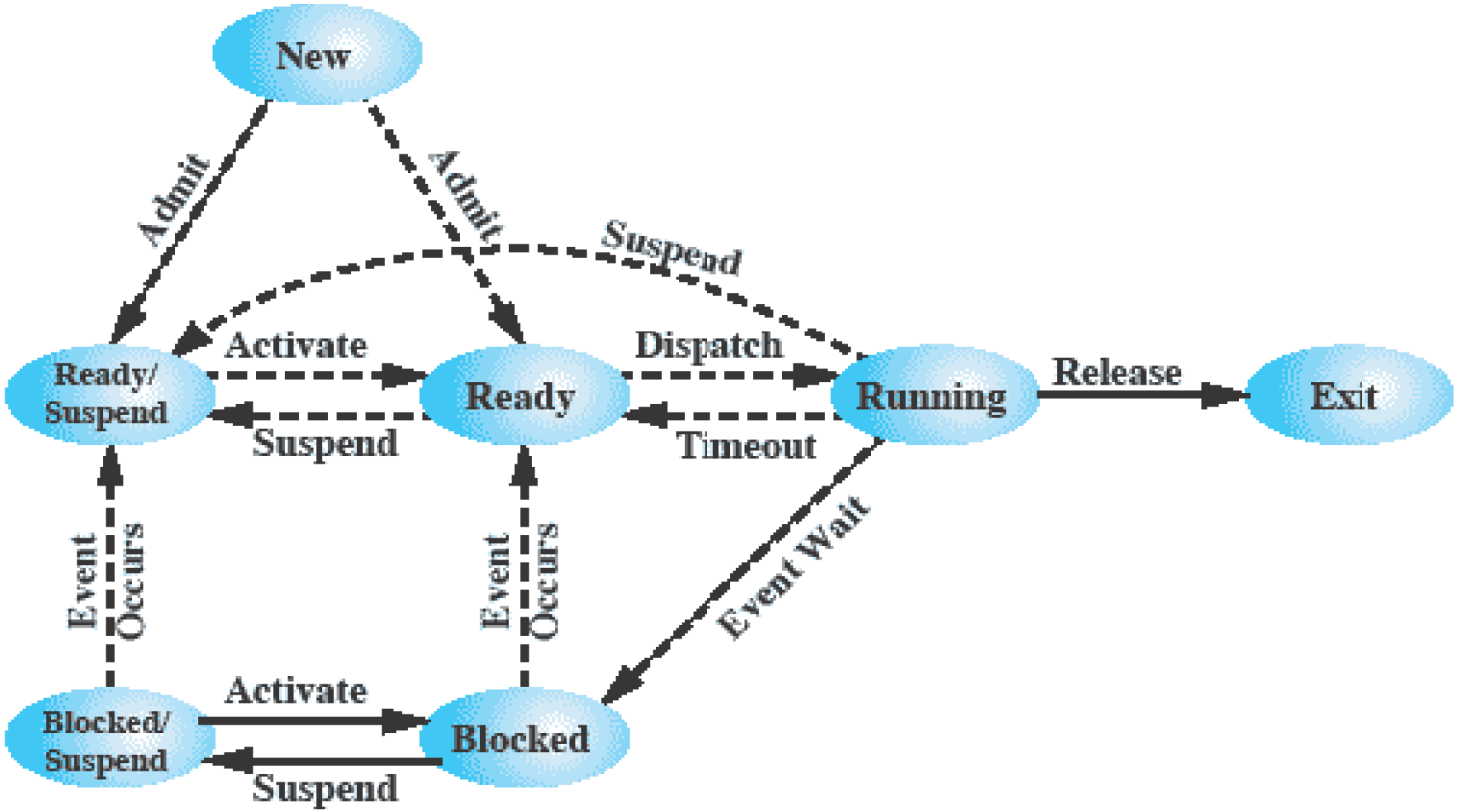
swapping

(a) With One Suspend State

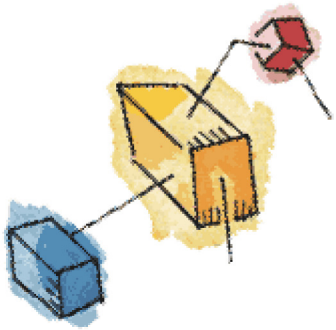
When all the processes in main memory are in the Blocked State.

- OS suspends one process, put in Suspend State, transfer to disk
- the free space is use to bring another process (new or previously suspended).

Two Suspend States



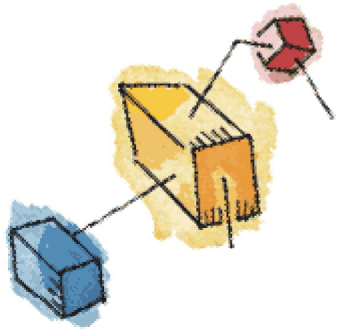
(b) With Two Suspend States



Reason for Process Suspension

Table 3.3 Reasons for Process Suspension

Swapping	The OS needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS reason	The OS may suspend a background or utility process or a process that is suspected of causing a problem.
Interactive user request	A user may wish to suspend execution of a program for purposes of debugging or in connection with the use of a resource.
Timing	A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time interval.
Parent process request	A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendants.



Processes and Resources

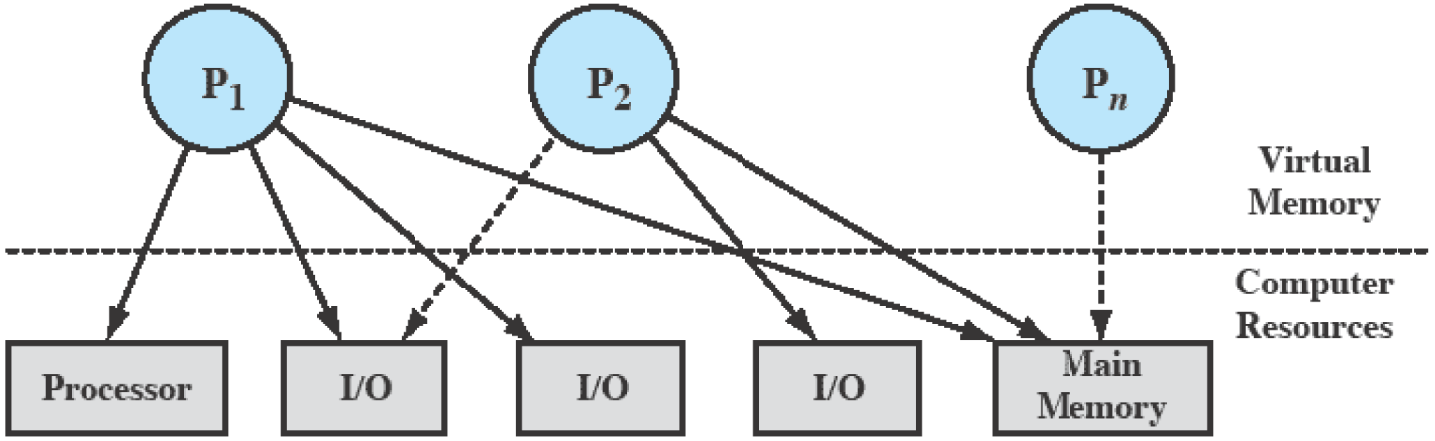
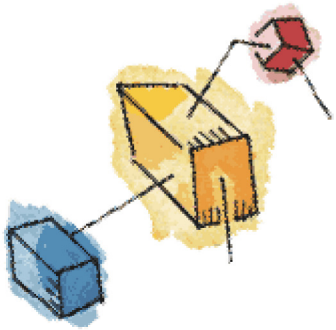


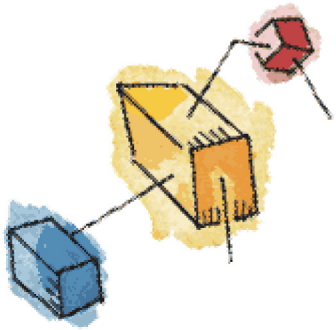
Figure 3.10 Processes and Resources (resource allocation at one snapshot in time)

P1 is running in main memory, has control of two I/O devices
P2 is in main memory, but is blocked, waiting for an I/O allocated to P1
Pn has been swapped out and is therefore suspended



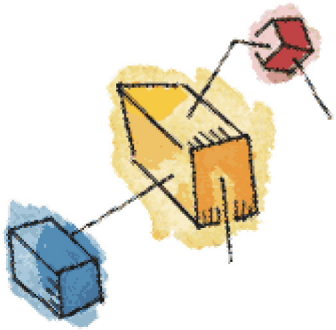
Operating System Control Structures

- Hold the information about the current status of each process and resource
- Tables are constructed for each entity the operating system manages
  - Memory Tables
  - I/O Tables
  - File Tables
  - Process Tables



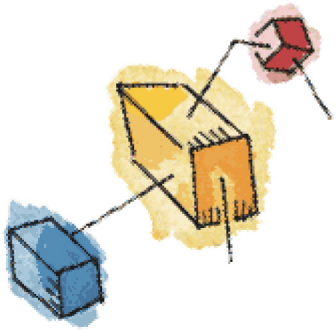
Memory Tables

- Includes the following information:
  - Allocation of main memory to processes
  - Allocation of secondary memory to processes
  - Protection attributes for access to shared memory regions
  - Information needed to manage virtual memory



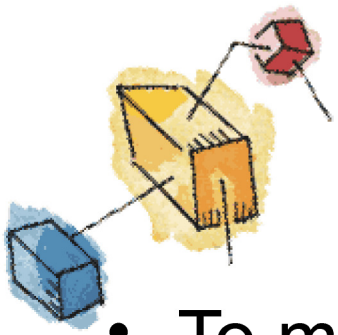
I/O Tables

- Used by the OS to manage the I/O devices and channels of the computer system
- At any given time, I/O device may be available or assigned to a particular process
- If I/O operation is in progress, the OS needs to know:
  - Status of I/O operation
  - Location in main memory being used as the source or destination of the I/O transfer



File Tables

- Provide information of the:
  - Existence of files
  - Location on secondary memory
  - Current Status
  - Attributes
  - Sometimes this information is maintained by a file management system



Process Tables

- To manage processes
- Process control block
  - Process image is the collection of program, data, stack, and attributes

Table 3.4 Typical Elements of a Process Image

User Data The modifiable part of the user space. May include program data, a user stack area, and programs that may be modified.
User Program The program to be executed.
Stack Each process has one or more last-in-first-out (LIFO) stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls.
Process Control Block Data needed by the OS to control the process (see Table 3.5).

OS Control Tables

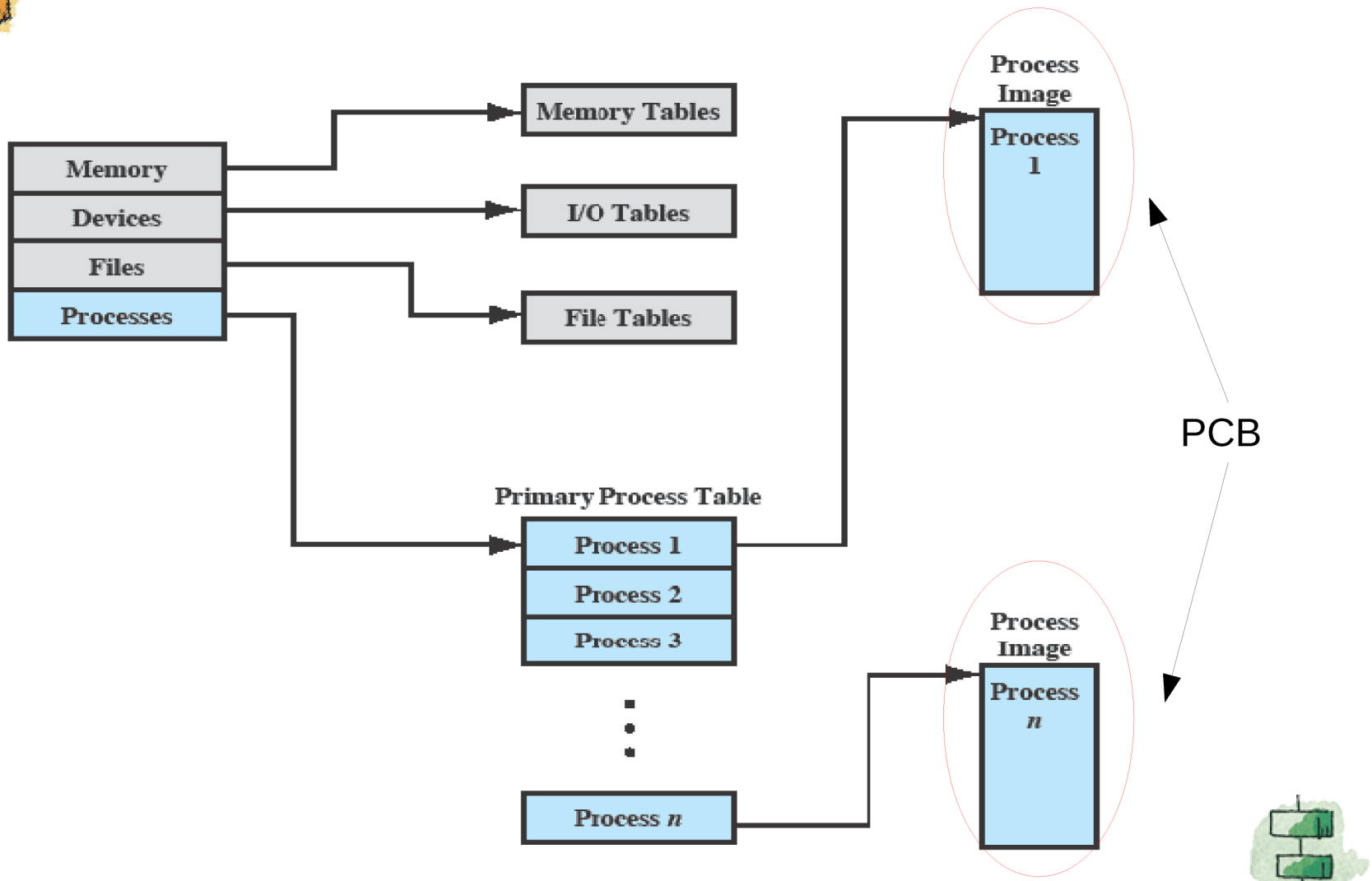
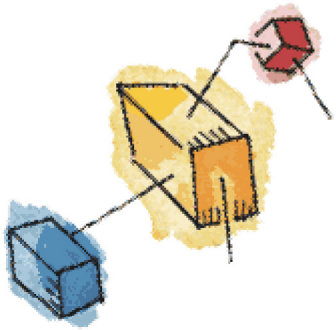


Figure 3.11 General Structure of Operating System Control Tables



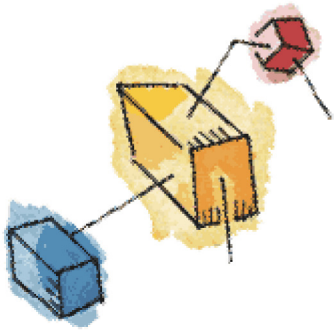
Elements of a Process Control Block (details)

Process Identification

Identifiers

Numeric identifiers that may be stored with the process control block include

- Identifier of this process
- Identifier of the process that created this process (parent process)
- User identifier



Elements of a Process Control Block (details)

Processor State Information

User-Visible Registers

A user-visible register is one that may be referenced by means of the machine language that the processor executes while in user mode. Typically, there are from 8 to 32 of these registers, although some RISC implementations have over 100.

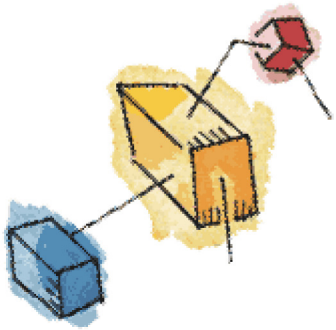
Control and Status Registers

These are a variety of processor registers that are employed to control the operation of the processor. These include

- *Program counter:* Contains the address of the next instruction to be fetched
- *Condition codes:* Result of the most recent arithmetic or logical operation (e.g., sign, zero, carry, equal, overflow)
- *Status information:* Includes interrupt enabled/disabled flags, execution mode

Stack Pointers

Each process has one or more last-in-first-out (LIFO) system stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls. The stack pointer points to the top of the stack.



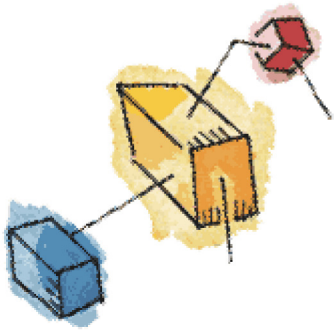
Elements of a Process Control Block

Process Control Information

Scheduling and State Information

This is information that is needed by the operating system to perform its scheduling function. Typical items of information:

- *Process state:* Defines the readiness of the process to be scheduled for execution (e.g., running, ready, waiting, halted).
- *Priority:* One or more fields may be used to describe the scheduling priority of the process. In some systems, several values are required (e.g., default, current, highest-allowable)
- *Scheduling-related information:* This will depend on the scheduling algorithm used. Examples are the amount of time that the process has been waiting and the amount of time that the process executed the last time it was running.
- *Event:* Identity of event the process is awaiting before it can be resumed.



Elements of a Process Control Block (details)

Process Privileges

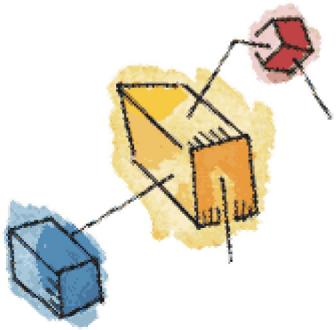
Processes are granted privileges in terms of the memory that may be accessed and the types of instructions that may be executed. In addition, privileges may apply to the use of system utilities and services.

Memory Management

This section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.

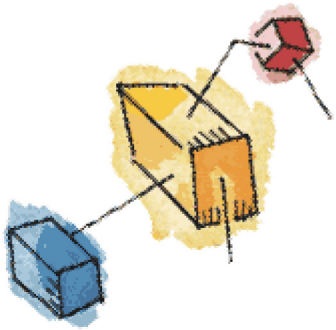
Resource Ownership and Utilization

Resources controlled by the process may be indicated, such as opened files. A history of utilization of the processor or other resources may also be included; this information may be needed by the scheduler.



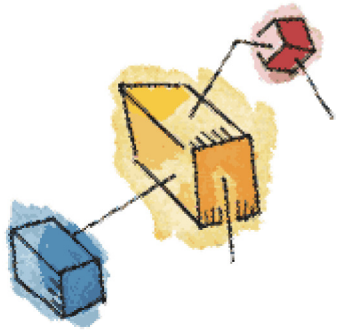
Process Identification

- Each process is assigned a unique number of ID
  - Otherwise there must be a mapping that locate the appropriate table based on the process ID
- Useful in several ways:
  - Many of the other tables controlled by the OS may use process ID to cross-reference process table
    - e.g. indication of which process is assigned to each region. Similar reference in I/O & files table
  - When processes communicate with each other, the process ID informs the OS the destination of a particular communication

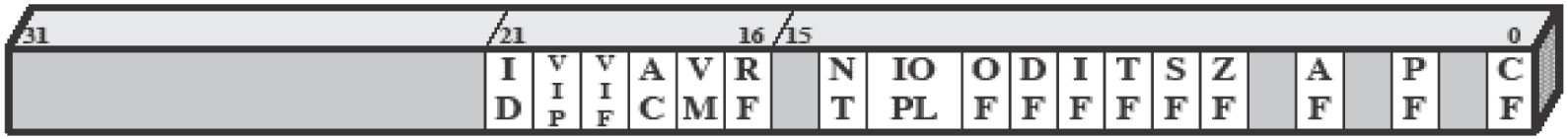


Processor State Information

- Consists of the contents of processor registers
  - User-visible registers
  - Control and status registers
  - Stack pointers
- Program status word (PSW)
  - contains condition code + status information
  - Example: the EFLAGS register on Pentium processors



Pentium II EFLAGS Register

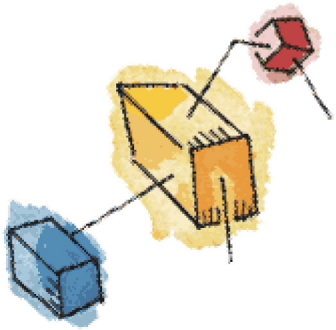


ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag

DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

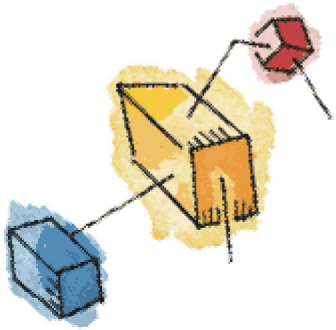
Figure 3.12 Pentium II EFLAGS Register

Condition code: OF, CF, AF, PF, SF, ZF
Control bits: AC, ID, RF, IOPL, DF, IF, TF
Operating Mode bits: NT, VM, VIP, VIF



Process Control Information

The additional information needed by the OS to control and coordinate the various active processes



User Processes in VM

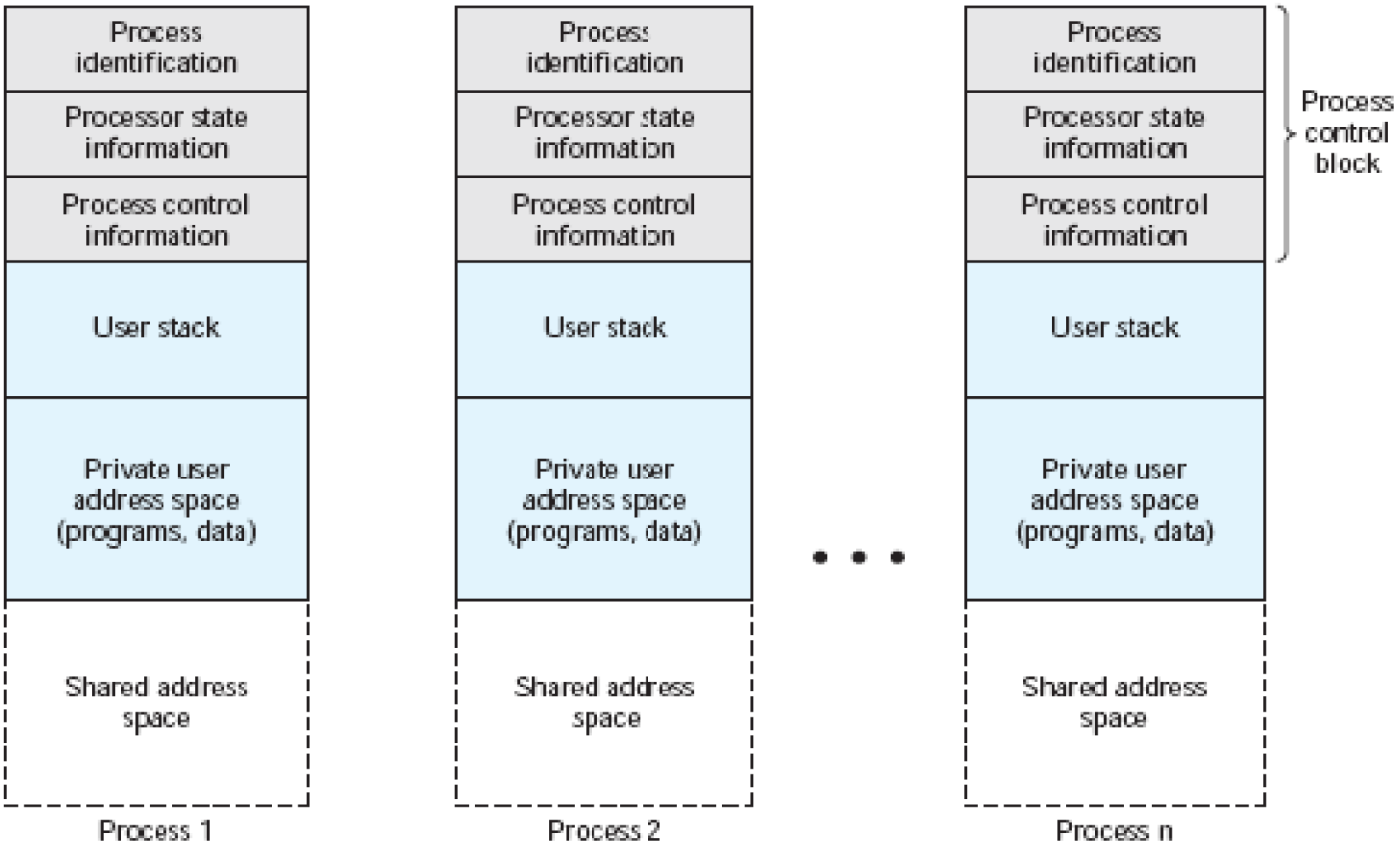
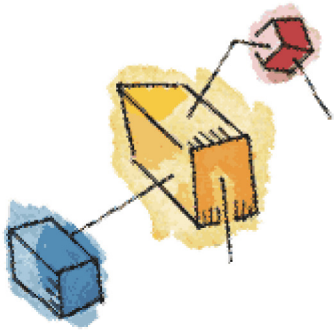


Figure 3.13 **User Processes in Virtual Memory**



Process List Structure

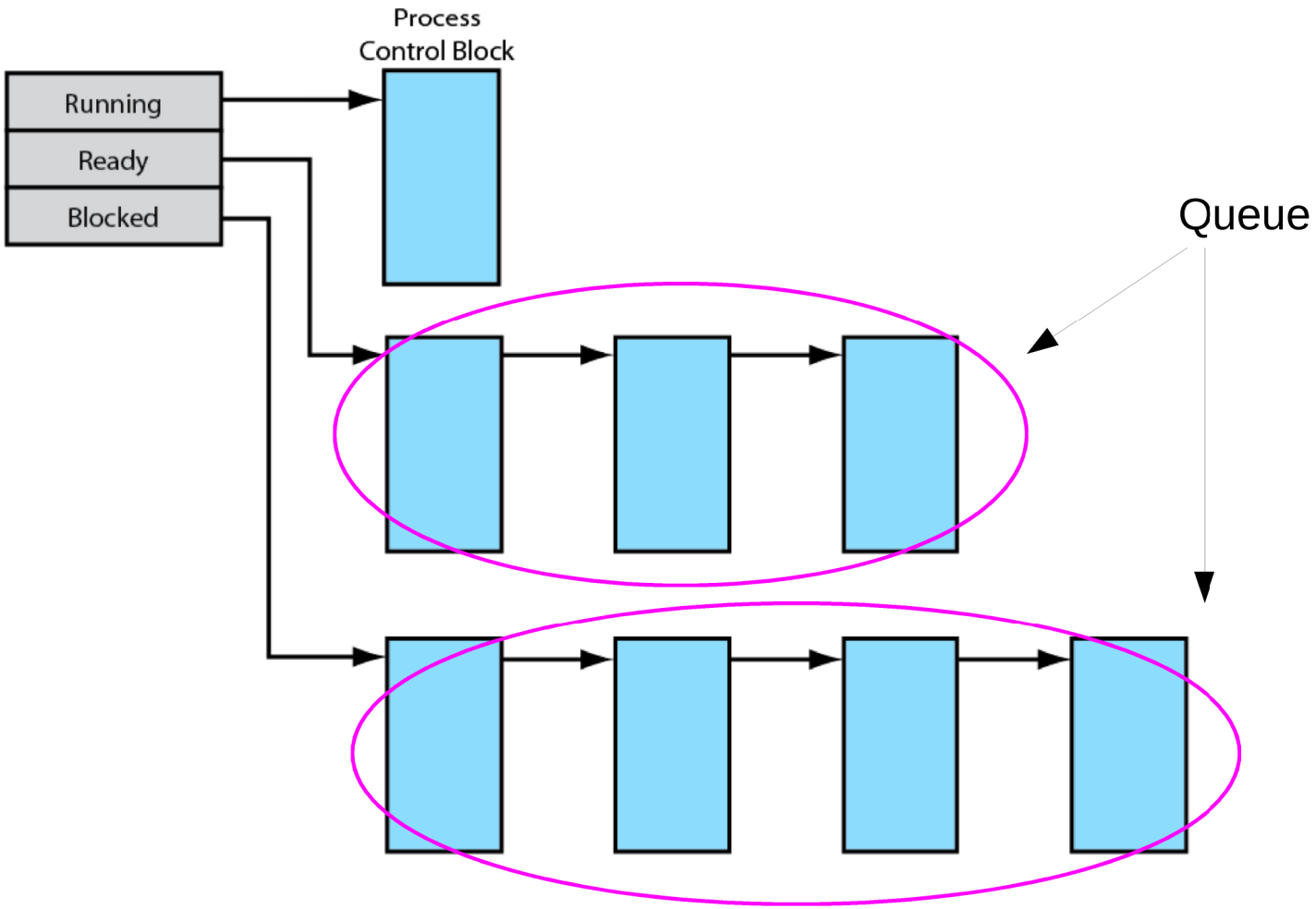
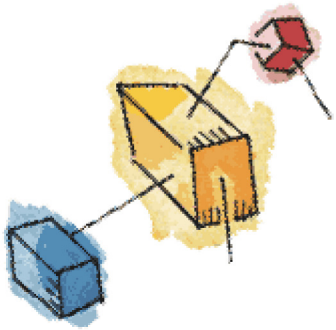
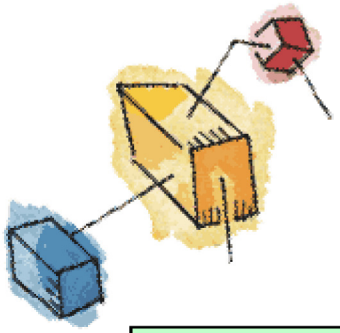


Figure 3.14 Process List Structures



Process Control:
Modes of Execution

- Most processors support at least two modes:
  - User mode
    - Less-privileged mode
    - User programs typically execute in this mode
  - System mode, control mode, or kernel mode
    - More-privileged mode
    - Kernel of the operating system
      - Encompass important system functions
  - Reason:
    - To protect OS and its tables from interfering with user programs
    -



Typical Functions of OS Kernel

Process Management

- Process creation and termination
- Process scheduling and dispatching
- Process switching
- Process synchronization and support for interprocess communication
- Management of process control blocks

Memory Management

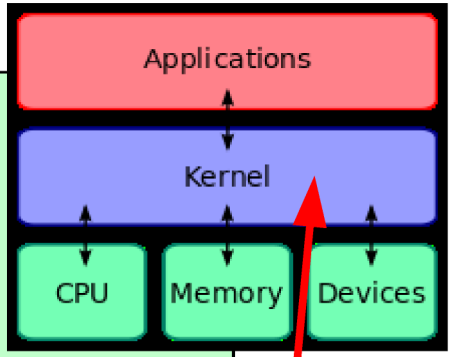
- Allocation of address space to processes
- Swapping
- Page and segment management

I/O Management

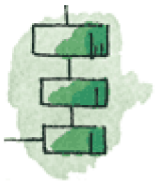
- Buffer management
- Allocation of I/O channels and devices to processes

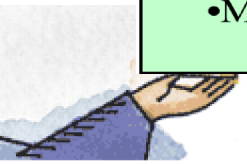
Support Functions

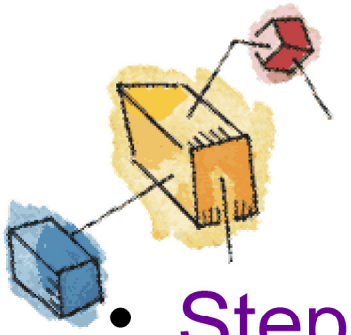
- Interrupt handling
- Accounting
- Monitoring



Central part of the OS

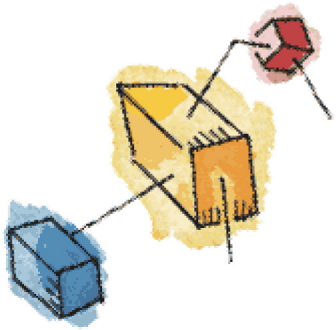






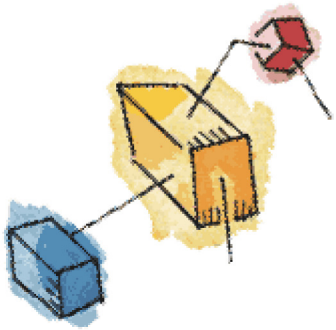
Process Creation

- Steps involved:
  - Assign a unique process identifier
    - A new entry is added to process table
  - Allocate space for the process
    - Includes all elements of the process image
  - Initialize process control block
  - Set up appropriate linkages
    - Put in Ready OR Ready/Suspend list
  - Create or expand other data structures
    - Accounting file for assessment



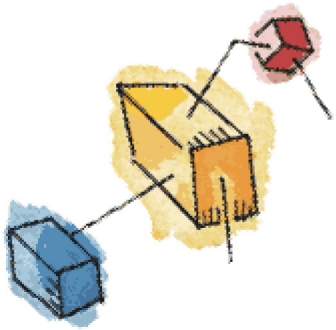
Process Switching

- WHEN to Switch a process?
- Clock interrupt
  - process has executed for the maximum allowable unit of time
- I/O interrupt
  - The OS determine what I/O action has occurred (Blocked to Ready or Ready/Suspend)
- Memory fault
  - memory address is in virtual memory so it must be brought into main memory



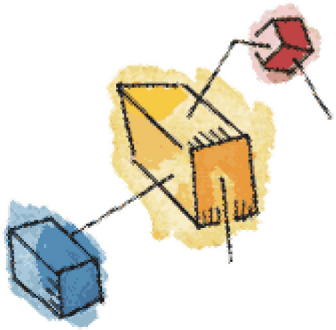
Mechanisms for Interrupting execution of a Process

Mechanism	Cause	Use
Interrupt	External to the execution of the current instruction	Reaction to an asynchronous external event
Trap	Associated with the execution of the current instruction	Handling of an error or an exception condition
Supervisor call	Explicit request	Call to an operating system function



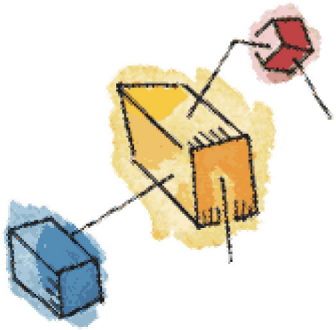
Change of Process State

- Save context of processor including program counter and other registers
- Update the process control block of the process that is currently in the Running state
  - Includes changing the state to one of the other states (Ready, Blocked, Ready/suspend, Exit)
- Move process control block to appropriate queue – ready; blocked; ready/suspend



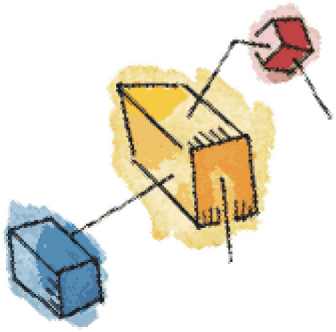
Change of Process State

- Select another process for execution
- Update the process control block of the process selected
- Update memory-management data structures
- Restore context of the processor which exist at the time the selected process was last switched

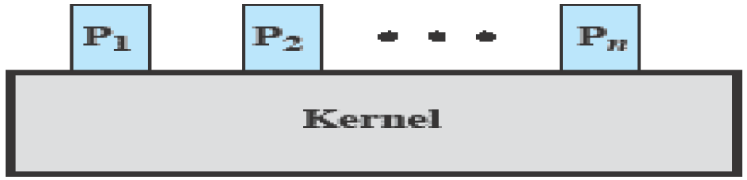


Execution of the Operating System

- Non-process Kernel
  - Execute kernel outside of any process
  - Operating system code is executed as a separate entity that operates in privileged mode
- Execution Within User Processes
  - Common on smaller computers
  - Execute virtually all OS software within context of a user process
- Process-Based OS
  - Major kernel functions are organized as separate processes



Execution of the Operating System



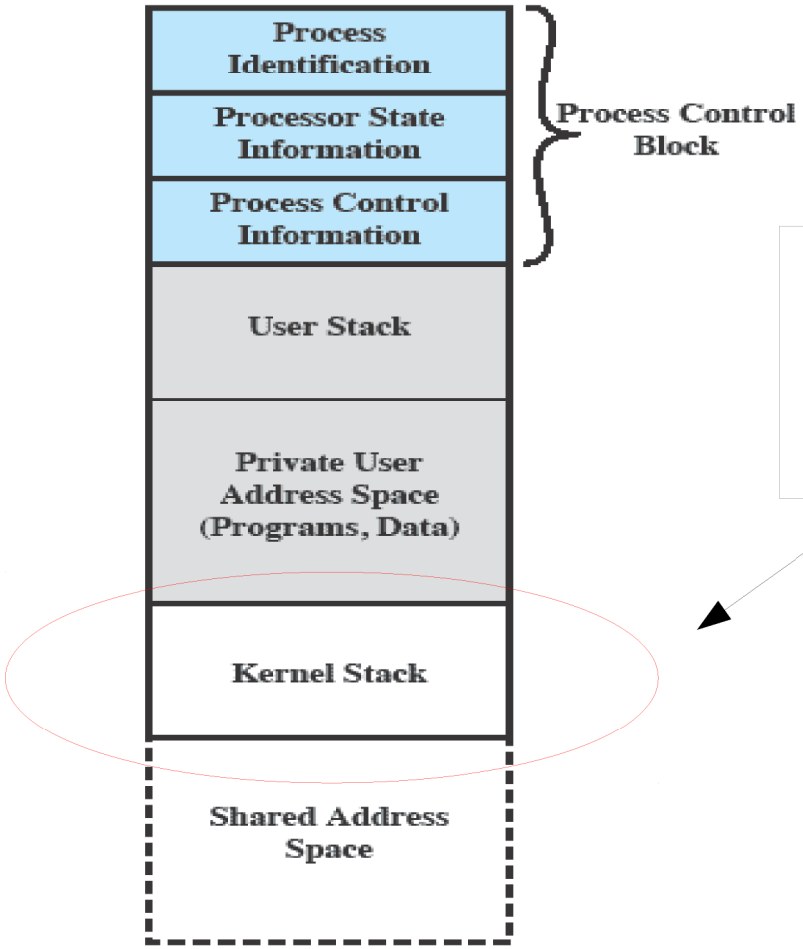
(a) Separate kernel

(b) OS functions execute within user processes

(c) OS functions execute as separate processes

Figure 3.15 Relationship Between Operating System and User Processes

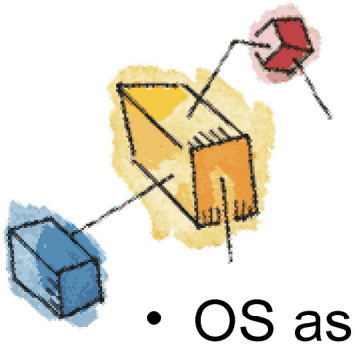
OS Executes in User Space



Manage calls/returns while the process is in kernel mode

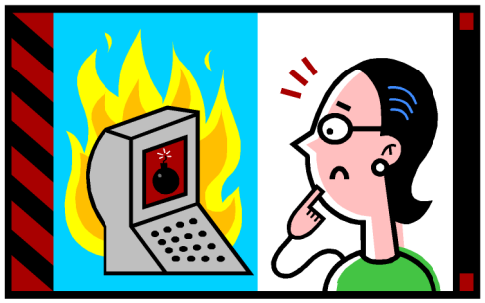
e.g: interrupt, trap, supervisor calls

Figure 3.16 Process Image: Operating System Executes Within User Space

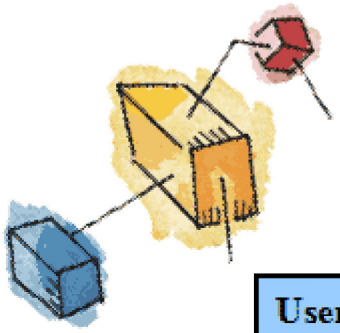


Security Issues

- OS associates a set of privileges with each process
- To prevent or detect attempts by a user or malicious from gaining unauthorized privileges on the system
- System access threats
  - Intruders (hackers)
    - Masquerader - outsider
    - Misfeasor - insider
    - Clandestine User – both insider or outsider
  - Malicious software



- Countermeasures
  - Intrusion detection
  - Authentication
  - Access control
  - Firewalls



UNIX Process States

User Running	Executing in user mode.
Kernel Running	Executing in kernel mode.
Ready to Run, in Memory	Ready to run as soon as the kernel schedules it.
Asleep in Memory	Unable to execute until an event occurs; process is in main memory (a blocked state).
Ready to Run, Swapped	Process is ready to run, but the swapper must swap the process into main memory before the kernel can schedule it to execute.
Sleeping, Swapped	The process is awaiting an event and has been swapped to secondary storage (a blocked state).
Preempted	Process is returning from kernel to user mode, but the kernel preempts it and does a process switch to schedule another process.
Created	Process is newly created and not yet ready to run.
Zombie	Process no longer exists, but it leaves a record for its parent process to collect.

UNIX Process State Transition Diagram

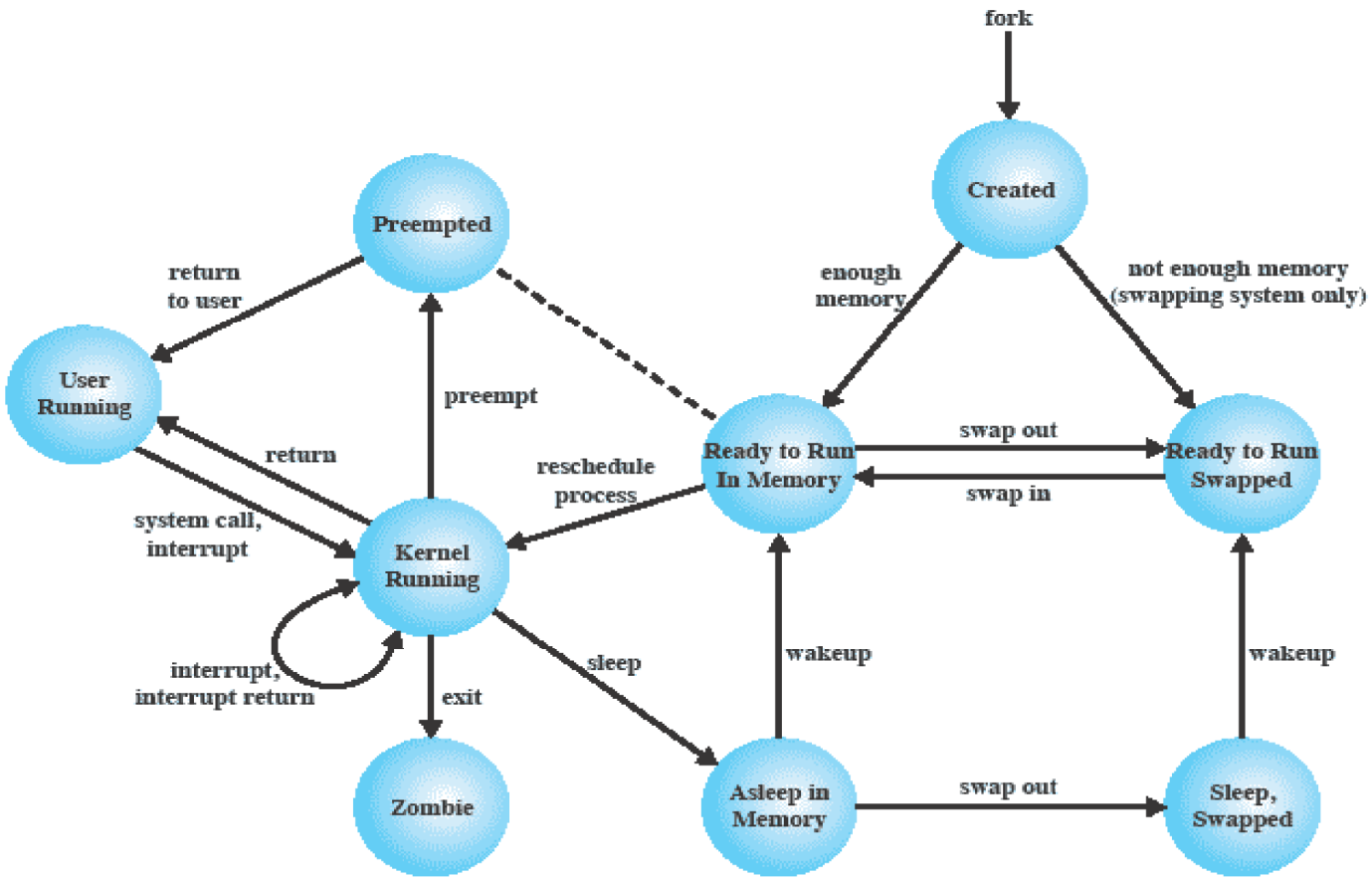
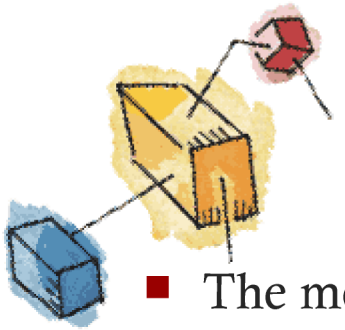


Figure 3.17 UNIX Process State Transition Diagram



Summary

- The most fundamental concept in a modern OS is the process
- The principal function of the OS is to create, manage, and terminate processes
- Process control block contains all of the information that is required for the OS to manage the process, including its current state, resources allocated to it, priority, and other relevant data
- The most important states are Ready, Running and Blocked
- The running process is the one that is currently being executed by the processor
- A blocked process is waiting for the completion of some event
- A running process is interrupted either by an interrupt or by executing a supervisor call to the OS