Operating System

Dr. GuoJun LIU

Harbin Institute of Technology

http://guojunos.hit.edu.cn

Outline

- What Is a Process?
- Process States
- Process Description
- Process Control
- Execution of the Operating System

Chapter 03

Process Description and Control

进程描述和控制

The concept of process is fundamental to the structure of modern computer operating systems. Its evolution in analyzing problems of synchronization, deadlock, and scheduling in operating systems has been a major intellectual contribution of computer science.

-- WHAT CAN BE AUTOMATED?: THE COMPUTER SCIENCE AND ENGINEERING RESEARCH STUDY, MIT Press, 1980

Learning Objectives

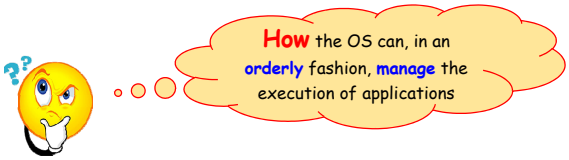
- Define the term process and explain the relationship between processes and process control blocks
- Explain the concept of a process state and discuss the state transitions the processes undergo
- List and describe the purpose of the data structures and data structure elements used by an OS to manage processes
- Assess the requirements for process control by the OS
- Understand the issues involved in the execution of OS code

Summary of Earlier Concepts

- A computer platform consists of a collection of hardware resources
- Computer applications are developed to perform some task
- It is inefficient for applications to be written directly for a given hardware platform
- The OS was developed to provide a convenient, feature-rich, secure, and consistent interface for applications to use
- We can think of the OS as providing a uniform, abstract representation of resources that can be requested and accessed by applications

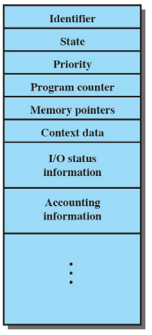
OS Management of Application Execution

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



Process Control Block (PCB)

- Contains the process elements
- It is possible to **interrupt** a running process and later **resume** execution
  - as if the interruption had not occurred
- Created and managed by OS
- **Key tool** (**data structure**) that allows support for multiple processes



Simplified Process Control Block

Process Elements

- **Recall the definition for the term process**
  - 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 a single sequential thread of execution, a current state, and an associated set of system resources
- **A process consists of two essential elements**
  - program code
    - may be **shared** with other processes that are executing the **same program**
  - a set of data associated with that code

Process States

Trace

characterize the behavior of an individual process by **listing the sequence of instructions** that execute for that process -- **trace**

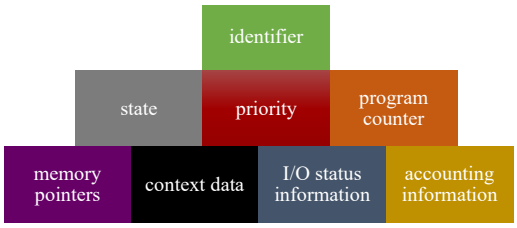
the behavior of the processor can be characterized by showing how the traces of the various processes are interleaved

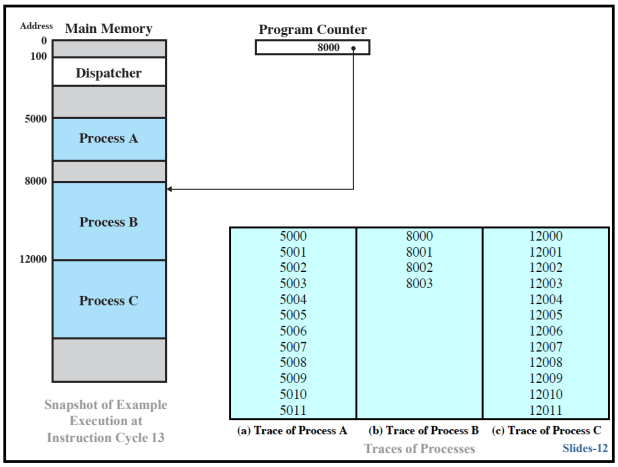
Dispatcher

small program that **switches** the processor from one process to another

Process Elements

- While the program is executing, this process can be uniquely characterized by a number of elements





Snapshot of Example Execution at Instruction Cycle 13

Traces of Processes

Combined Trace of Processes

1	5000		27	12004	
2	5001		28	12005	
3	5002		---------- Timeout		
4	5003		29	100	
5	5004		30	101	
6	5005		31	102	
---------- Timeout			32	103	
7	100		33	104	
8	101		34	105	
9	102		35	5006	
10	103		36	5007	
11	104		37	5008	
12	105		38	5009	
13	8000		39	5010	
14	8001		40	5011	
15	8002		---------- Timeout		
16	8003		41	100	
---------- I/O Request			42	101	
17	100		43	102	
18	101		44	103	
19	102		45	104	
20	103		46	105	
21	104		47	12006	
22	105		48	12007	
23	12000		49	12008	
24	12001		50	12009	
25	12002		51	12010	
26	12003		52	12011	
			---------- Timeout		

100 = Starting address of dispatcher program

Shaded areas indicate execution of dispatcher process

Reasons for Process Creation

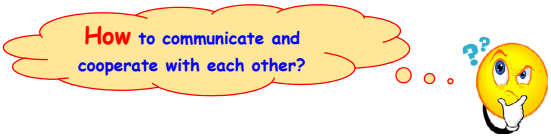
New batch job	The OS is provided with a batch job control stream, usually on tape or disk. When the OS is prepared to take on new work, it will read the next sequence of job control commands
Interactive logon	A user at a terminal logs on to the system
Created by OS to provide a service	The OS can create 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

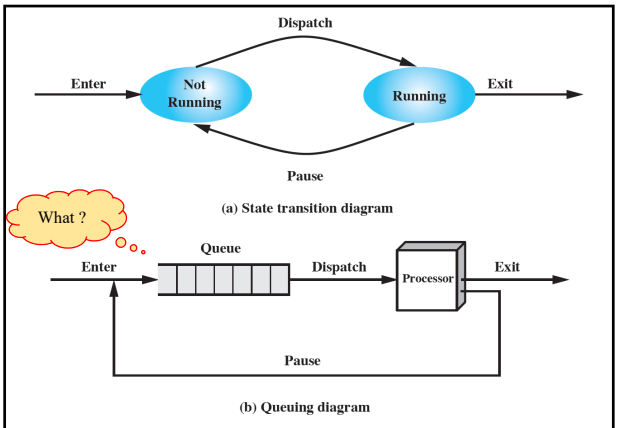
Two-State Process Model

- **OS principal responsibility is controlling the execution of processes**
  - determining the interleaving pattern for execution
  - allocating resources to processes
- **The first step in designing an OS to control processes is to describe the behavior**
  - running
  - not-running

Process Creation

Process spawning	*Parent process*	*Child process*
• when the OS creates a process at the explicit request of another process	• is the original creating process	• is the new process





Two-State Process Model

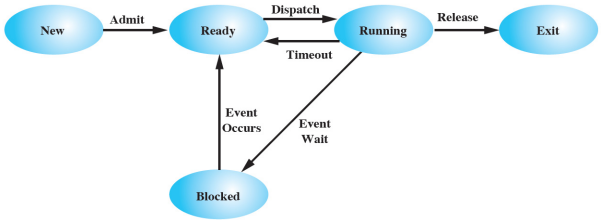
Process Termination

- **There must be a means for a process to indicate its completion**
- **A batch job call for termination**
  - a HALT instruction
    - generate an **interrupt** to alert the OS that a process has completed
  - an explicit OS service call for termination
- **For an interactive application, the action of the user will indicate when the process is completed**
  - log off
  - quitting an application

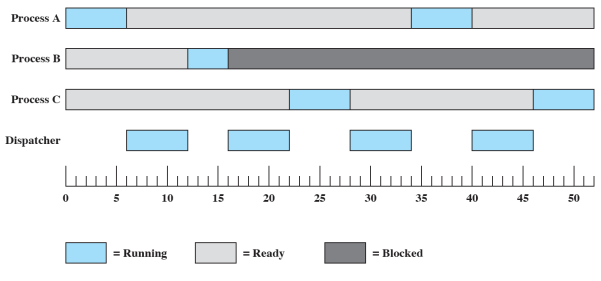
Reasons for Process Termination

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
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-only 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
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
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 (e.g., 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

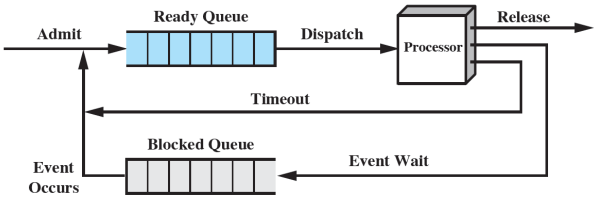
Five-State Process Model



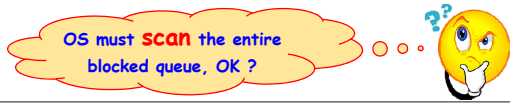
Process States for Trace



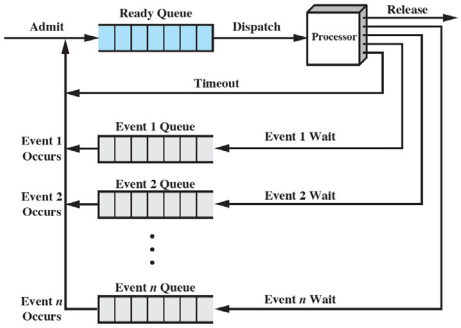
Using Two Queues



(a) Single blocked queue

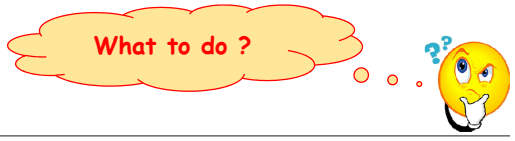


Multiple Blocked Queues



Suspended Processes

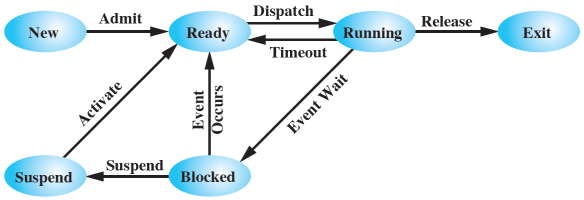
- **New question**
  - all of the processes in all of the queues must be resident in main memory
  - the processor is so much faster than I/O that it will be common for all of the processes in memory to be waiting for I/O
  - even with multiprogramming, a processor could be idle most of the time



Suspended Processes

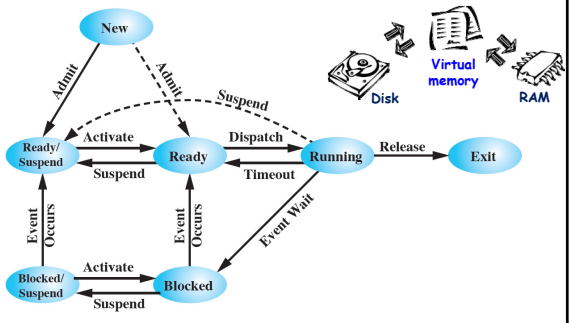
- **Main memory could be expanded**
  - to accommodate more processes
  - two flaws
    - cost
    - larger memory results in larger processes, **not more processes**
- **Swapping**
  - move part of all of a process from main memory to disk
  - when none of the processes in main memory is in the Ready state, the OS **swaps** one of the blocked processes out on to disk into a suspend queue
  - **disk I/O** is generally the fastest I/O on a system (printer I/O)

One Suspend State



(a) With One Suspend State

Two Suspend States



(b) With Two Suspend States

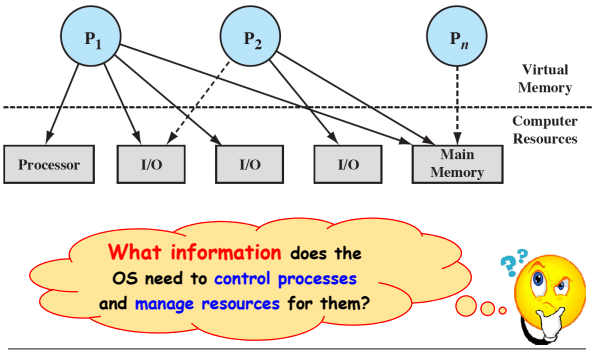
Characteristics of Suspended Process

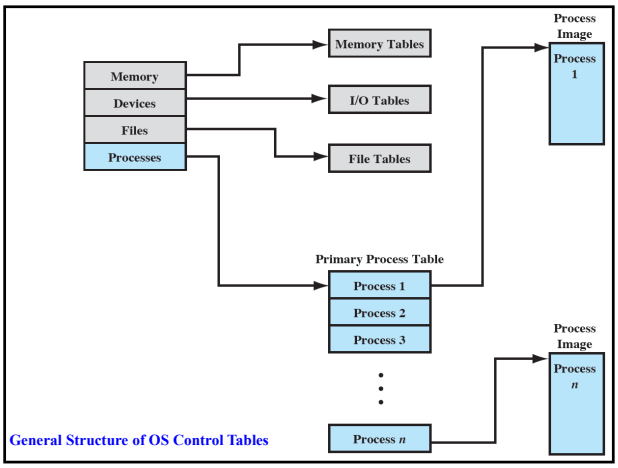
- **The process is not immediately available for execution**
- **The process was placed in a suspended state by an agent for the purpose of preventing its execution**
  - itself
  - a parent process
  - the OS
- **The process may or may not be waiting on an event**
- **The process may not be removed from this state until the agent explicitly orders the removal**

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





General Structure of OS Control Tables

File Tables

- **Information may be maintained and used by a file management system**
  - in which case the OS has little or no knowledge of files
- **In other OS**
  - much of the detail of file management is managed by the **OS itself**
- **These tables provide information about**
  - existence of files
  - location on secondary memory
  - current status
  - other attributes

Memory Tables

- **Used to keep track of both main (real) and secondary (virtual) memory**
- **Processes are maintained on**
  - secondary memory using some sort of virtual memory
  - simple swapping mechanism
- **Must include**
  - allocation of main memory to processes
  - allocation of secondary memory to processes
  - **protection attributes of blocks** of main or virtual memory
    - which processes may **access** certain **shared memory regions**
  - information needed to manage virtual memory

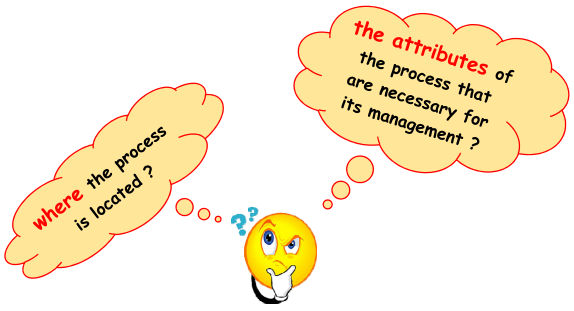
Process Tables

- **Must be maintained to manage processes**
- **There must be some reference directly or indirectly**
  - memory
  - I/O
  - files
- **The tables themselves must be accessible by the OS**
  - they **are subject to memory management**

I/O Tables

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

What OS must know



Process Control Structures

- **Process Location**
  - A process must include a program or set of programs to be executed
  - A process will consist of at least **sufficient memory** to hold the programs and data of that process
  - The execution of a program typically involves **a stack** that is used to keep track of procedure calls and parameter passing between procedures
- **Process Attributes**
  - Each process has associated with it a number of attributes that are used by the OS for process control
  - The collection of program, data, stack, and attributes is referred to as the process image
  - **Process image location** will depend on the memory management scheme being used

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

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.

Typical Elements of a Process Image

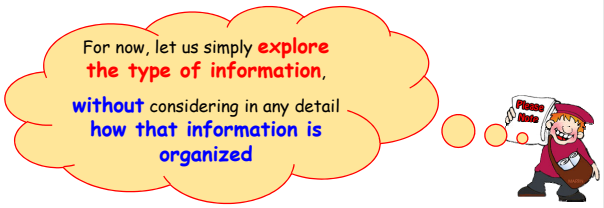
- **User Data**
  - The modifiable part of the user space
    - program data
    - a user stack area
    - programs that may be modified
- **User Program**
  - The program to be executed
- **Stack**
  - Each process has **one or more** last-in-first-out (LIFO) stacks associated with it.
  - A stack is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

Process Identification

- **Each process is assigned a unique numeric identifier**
  - otherwise there must be a mapping that allows the OS to **locate** the appropriate tables based on the process identifier
- **Many of the tables controlled by the OS may use process identifiers to cross-reference process tables**
  - Memory tables may be organized to provide a map of main memory with an indication of **which process is assigned to each region**
  - similar references will appear in I/O and file tables
- **When processes communicate with one another, the process identifier informs the OS of the destination of a particular communication**
- **When processes are allowed to create other processes, identifiers indicate the parent and descendents of each process**
- **a user identifier**
  - that indicates the user responsible for the job

PCB information

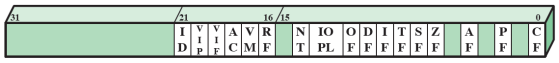
- **Group PCB information into three general categories**
  - Process identification
  - Processor state information
  - Process control information



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 codes plus other status information
  - **EFLAGS register** is an example of a PSW used by any OS running on an **x86 processor**

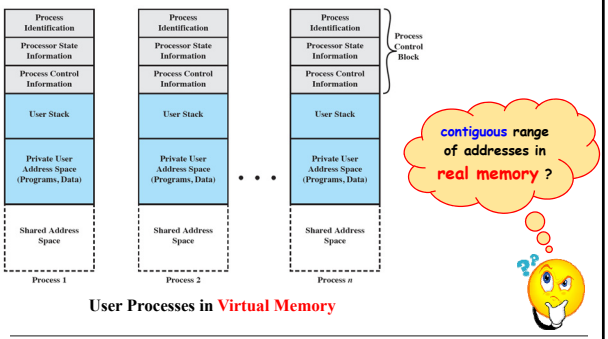
x86 EFLAGS Register



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

Structure of Process Images



User Processes in Virtual Memory

Pentium EFLAGS Register Bits

Control Bits

AC (Alignment check)
Set if a word or doubleword is addressed on a nonword or nondoubleword boundary.

ID (Identification flag)
If this bit can be set and cleared, this processor supports the CPUID instruction. This instruction provides information about the vendor, family, and model.

RF (Resume flag)
Allows the programmer to disable debug exceptions so that the instruction can be restarted after a debug exception without immediately causing another debug exception.

IOPL (I/O privilege level)
When set, causes the processor to generate an exception on all accesses to I/O devices during protected mode operation.

DF (Direction flag)
Determines whether string processing instructions increment or decrement the 16-bit half-registers SI and DI (for 16-bit operations) or the 32-bit registers ESI and EDI (for 32-bit operations).

IF (Interrupt enable flag)
When set, the processor will recognize external interrupts.

TF (Trap flag)
When set, causes an interrupt after the execution of each instruction. This is used for debugging.

Operating Mode Bits

NT (Nested task flag)
Indicates that the current task is nested within another task in protected mode operation.

VM (Virtual 8086 mode)
Allows the programmer to enable or disable virtual 8086 mode, which determines whether the processor runs as an 8086 machine.

VIP (Virtual interrupt pending)
Used in virtual 8086 mode to indicate that one or more interrupts are awaiting service.

VIF (Virtual interrupt flag)
Used in virtual 8086 mode instead of IF.

Condition Codes

AF (Auxiliary carry flag)
Represents carrying or borrowing between half-bytes of an 8-bit arithmetic or logic operation using the AL register.

CF (Carry flag)
Indicates carrying out or borrowing into the leftmost bit position following an arithmetic operation. Also modified by some of the shift and rotate operations.

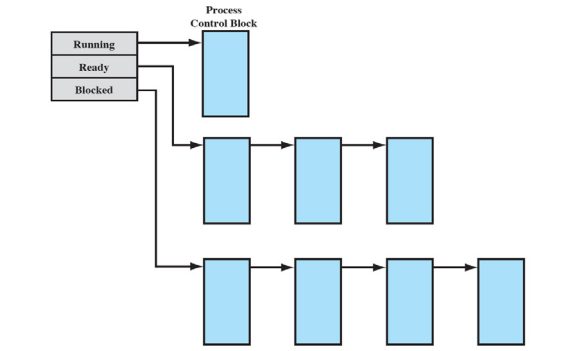
OF (Overflow flag)
Indicates an arithmetic overflow after an addition or subtraction.

PF (Parity flag)
Parity of the result of an arithmetic or logic operation. 1 indicates even parity; 0 indicates odd parity.

SF (Sign flag)
Indicates the sign of the result of an arithmetic or logic operation.

ZF (Zero flag)
Indicates that the result of an arithmetic or logic operation is 0.

Process List Structures



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

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.

Data Structuring
A process may be linked to other process in a queue, ring, or some other structure. For example, all processes in a waiting state for a particular priority level may be linked in a queue. A process may exhibit a parent-child (creator-created) relationship with another process. The process control block may contain pointers to other processes to support these structures.

Interprocess Communication
Various flags, signals, and messages may be associated with communication between two independent processes. Some or all of this information may be maintained in the process control block.

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.

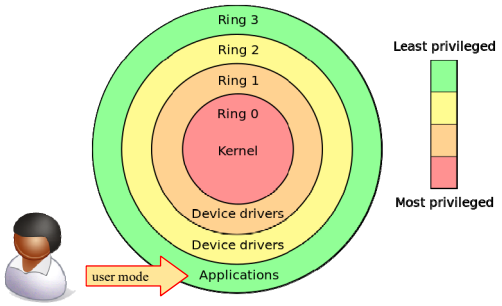
Memory Management
This section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.

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.

Role of the Process Control Block

- **The most important data structure in an OS**
  - contains all of the information about a process
  - blocks are **read and/or modified by virtually every module** in the OS
  - defines the state of the OS
- **Difficulty is not access, but protection**
  - a bug in a single routine could damage process control blocks, which could destroy the system's ability to manage the affected processes
  - a design change in the structure or semantics of the process control block could affect a number of modules in the OS

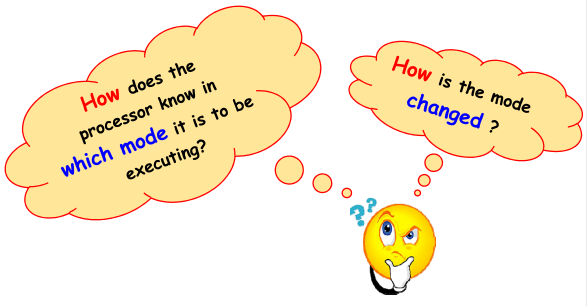
Modes of Execution



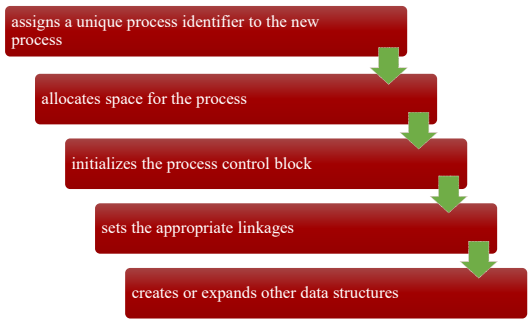
Typical Functions of an OS Kernel

- **Process Management**
  - Process creation and termination
  - Process scheduling and dispatching
  - Process switching
  - Process synchronization and support for interprocess communication
  - Management of process control blocks
- **I/O Management**
  - Buffer management
  - Allocation of I/O channels and devices to processes
- **Memory Management**
  - Allocation of address space to processes
  - Swapping
  - Page and segment management
- **Support Functions**
  - Interrupt handling
  - Accounting
  - Monitoring

Two questions



Process Creation



Process Switching

- **A process switch may occur any time that the OS has gained control from the currently running process**
- **Several design issues are raised**
  - what events trigger a process switch?
  - must recognize the distinction between **mode switching** and **process switching**
  - what must the OS do to the various data structures under its control to achieve a process switch?

When to Switch Processes

- **Possible events giving OS control are**

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

System Interrupts

- **Interrupt**
  - Due to some sort of event that is **external** to and independent of the currently running process
    - clock interrupt
    - I/O interrupt
    - memory fault
  - Time slice
    - the maximum amount of time that a process can execute before being interrupted
- **Trap**
  - An error or exception condition generated **within** the currently running process
  - OS determines if the condition is fatal
    - moved to the Exit state and a **process switch** occurs
    - action will depend on the nature of the error

Mode Switching

If no interrupts are pending the processor:

proceeds to the fetch stage and fetches the next instruction of the current program in the current process

If an interrupt is pending the processor:

sets the program counter to the **starting address** of an interrupt handler program

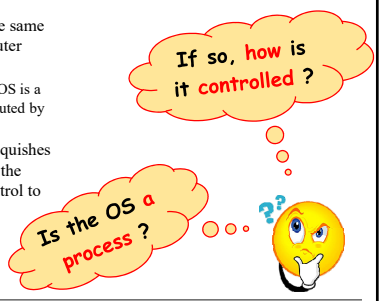
switches from user mode to kernel mode so that the interrupt processing code may include privileged instructions

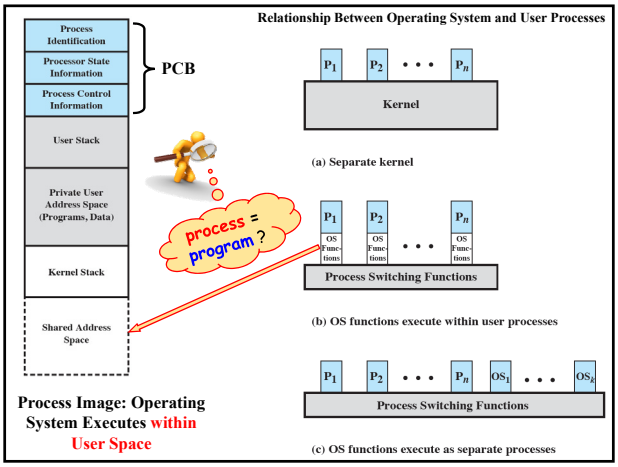
Change of Process State

- **Save the context of the processor**
  - program counter and other registers
- **Update PCB of the process that is currently in the Running state**
  - the **state** of the process to one of the other states
  - the reason for leaving
  - accounting information
- **Move PCB of this process to the appropriate queue**
  - Ready; Blocked on Event i ; Ready/Suspend
- **Select another process for execution**
- **Update PCB of the process selected**
  - changing the state of this process to **Running**
- **Update memory management data structures**
- **Restore the context of the processor**
  - to that which existed at the time the selected process was last switched out of the Running state
  - by loading in the previous values of the program counter and other registers

Execution of the Operating System

- **two intriguing facts about operating systems**
  - The **OS functions** in the same way as **ordinary** computer software
    - in the sense that the OS is a set of programs executed by the processor
  - The OS frequently relinquishes control and depends on the processor to restore control to the OS





Process Image: Operating System Executes within User Space

Summary

- **What Is a Process?**
  - Background
  - Processes and PCB
- **Process States**
  - A Two-State Process Model
  - The Creation and Termination of Processes
  - A Five-State Model
  - Suspended Processes
- **Process Description**
  - Operating System Control Structures
  - Process Control Structures
- **Process Control**
  - Modes of Execution
  - Process Creation
  - Process Switching
    - what events trigger a process switch ?
    - the distinction between mode switching and process switching
    - what must the OS do ?
- **Execution of the Operating System**
  - Nonprocess Kernel
  - Execution within User Processes
  - Process-Based Operating System