Process Concept, Process State, Process Description, Process Control, Execution of Operating System

WS 3.1-3.2 (pg.131- 147)
WS 3.3 (pg.148- 156)
WS 3.4-3.5 (pg.156- 165)

Processes and Process Control Blocks

- 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.

Processes and Process Control Blocks

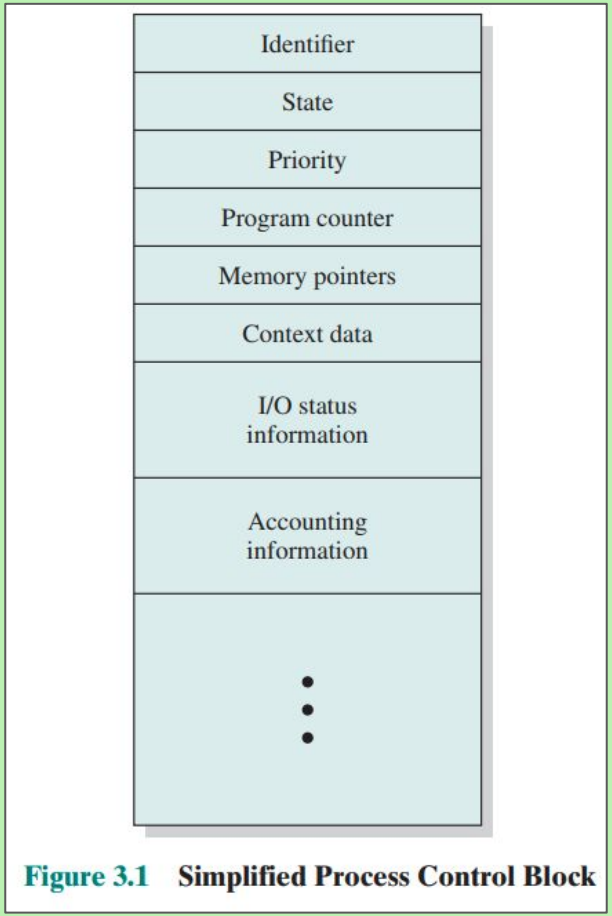


Figure 3.1 Simplified Process Control Block

A Two-State Process Model

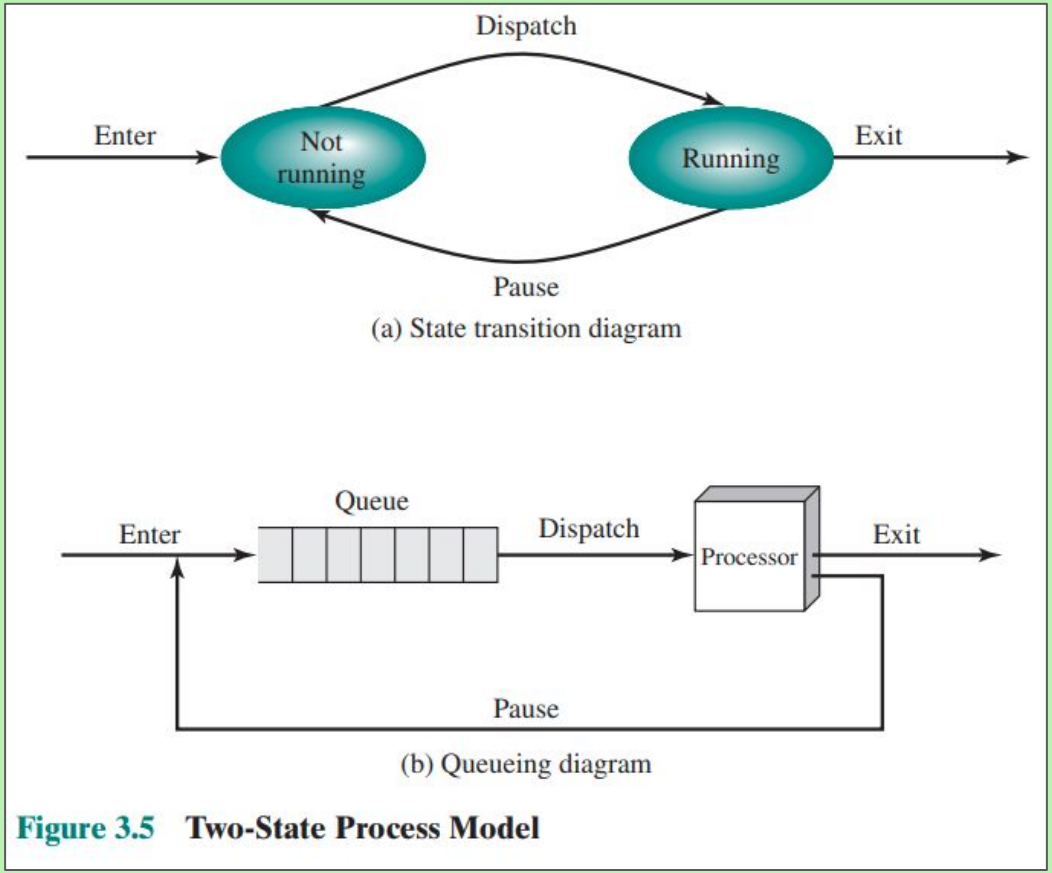


Figure 3.5 **Two-State Process Model**

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 log-on	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-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 (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 (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.

A Five-State Model

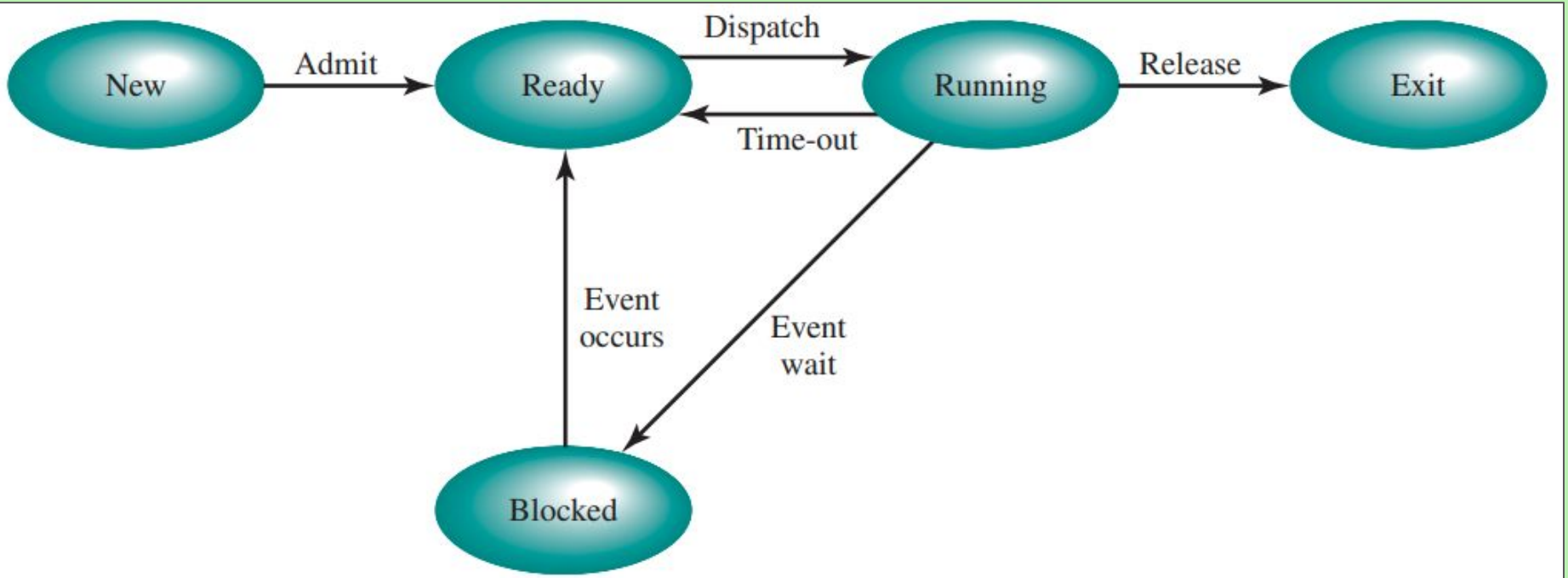
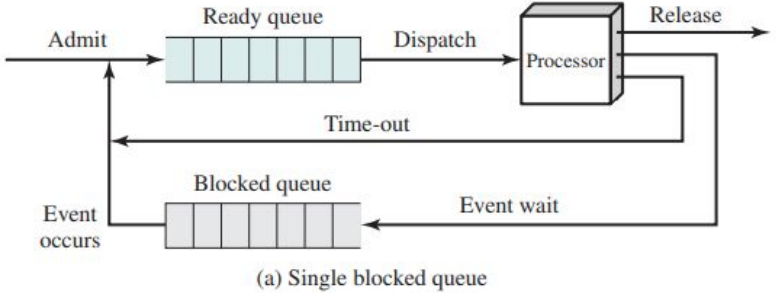
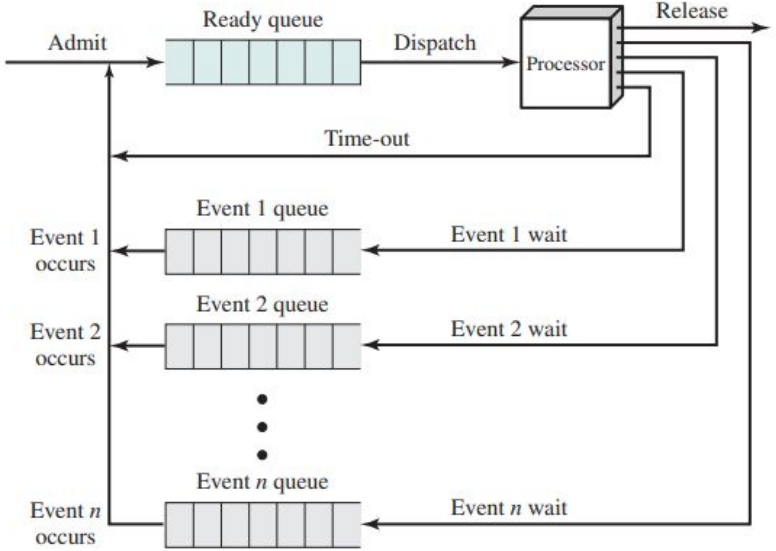


Figure 3.6 **Five-State Process Model**

Queues



(a) Single blocked queue



(b) Multiple blocked queues

Figure 3.8 **Queueing Model for Figure 3.6**

Suspended Processes
The Need for Swapping

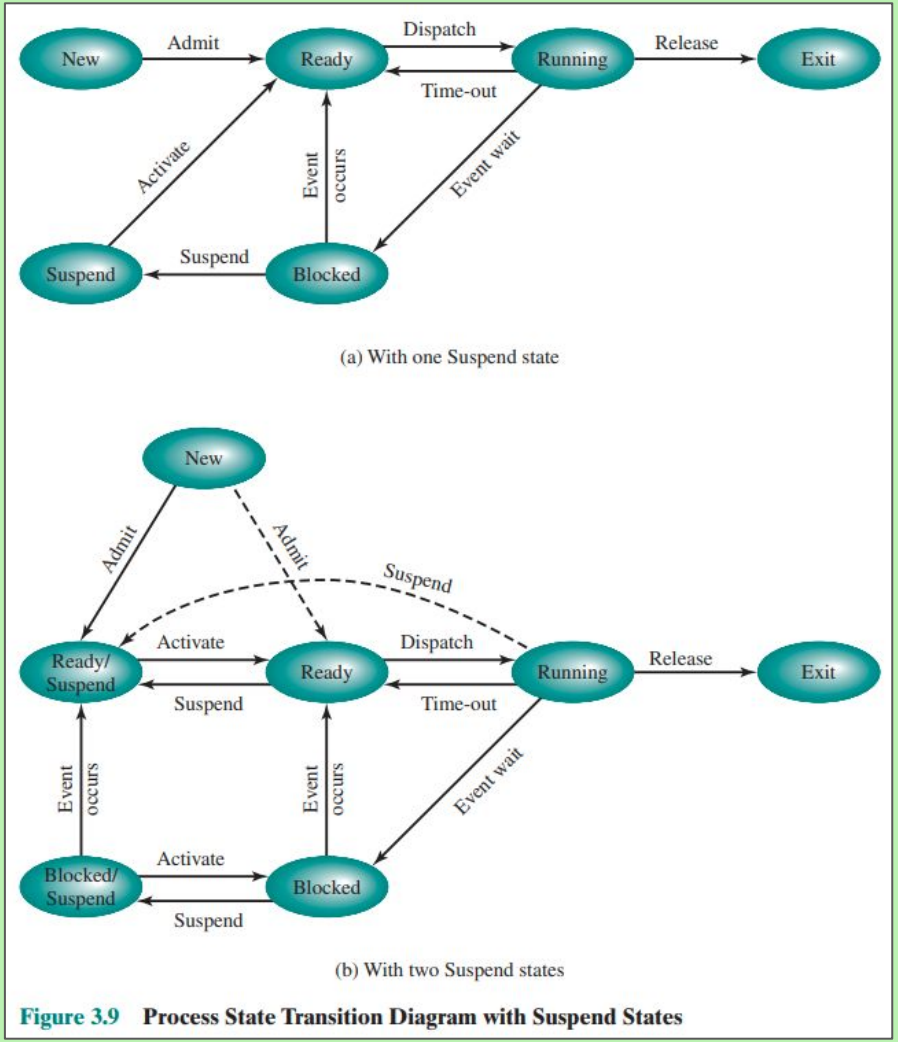


Figure 3.9 **Process State Transition Diagram with Suspend States**

Suspended Processes: The Need for Swapping

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.

Operating System Control Structures

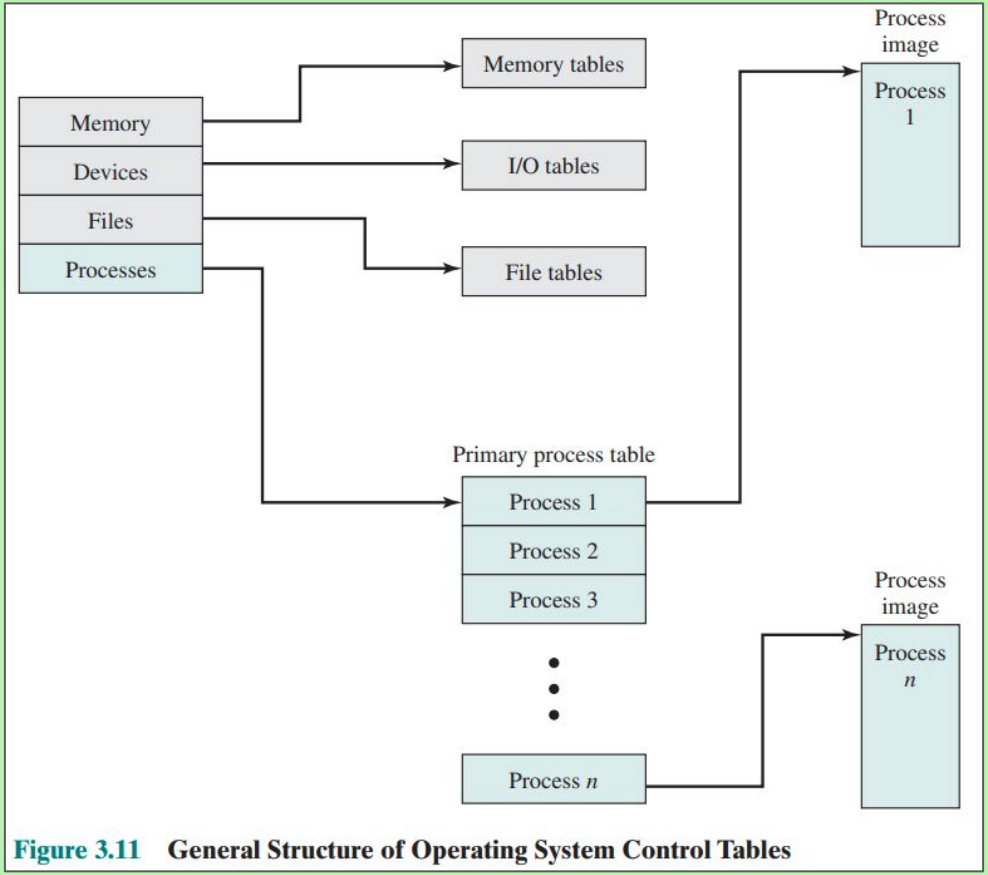


Figure 3.11 General Structure of Operating System Control Tables

Process Control Structures

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).

Process Control Structures

Table 3.5 Typical Elements of a Process Control Block

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.

Process Control Structures

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 include:

- **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.

Process Control Structures

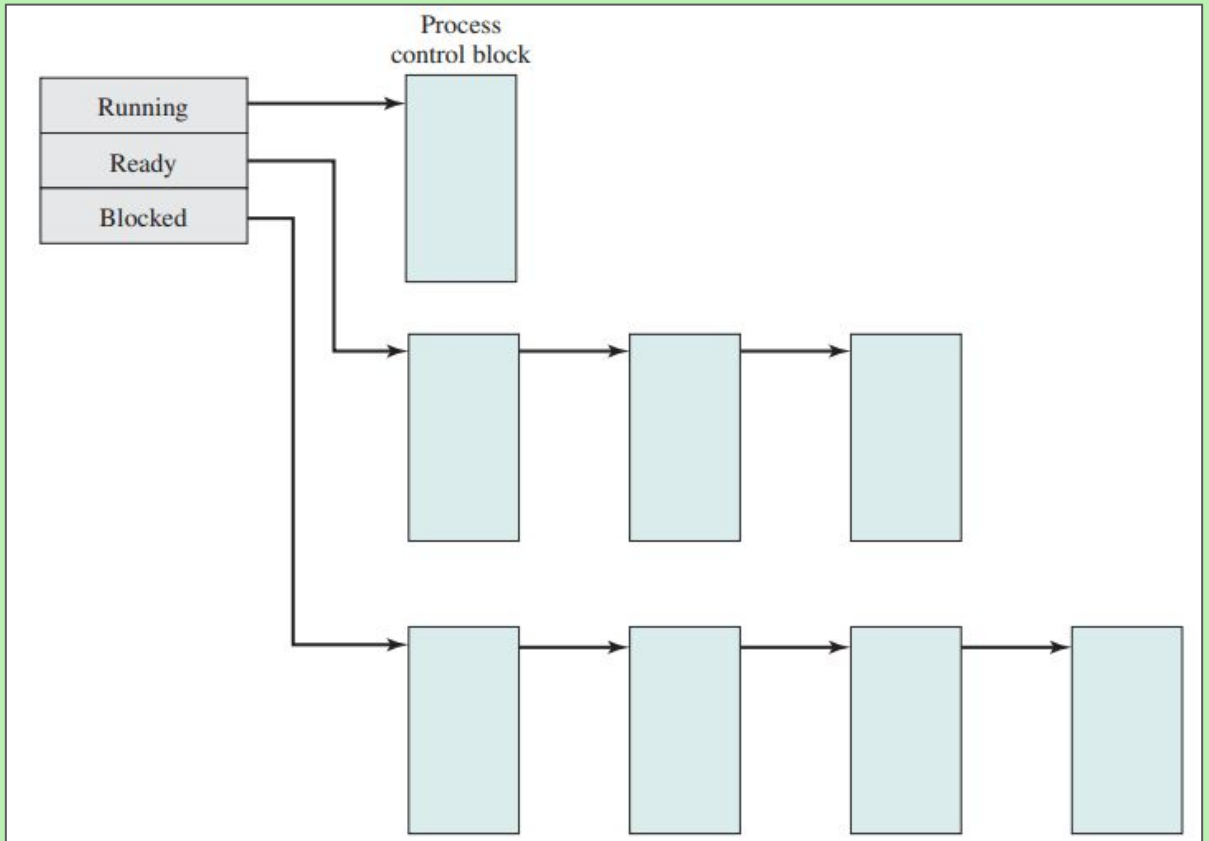


Figure 3.14 Process List Structures

PROCESS CONTROL: Modes of Execution

Table 3.7 Typical Functions of an Operating System 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

When to Switch Processes

Table 3.8 Mechanisms for Interrupting the 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

Mode Switching

EXECUTION OF THE OPERATING SYSTEM

Non-process Kernel

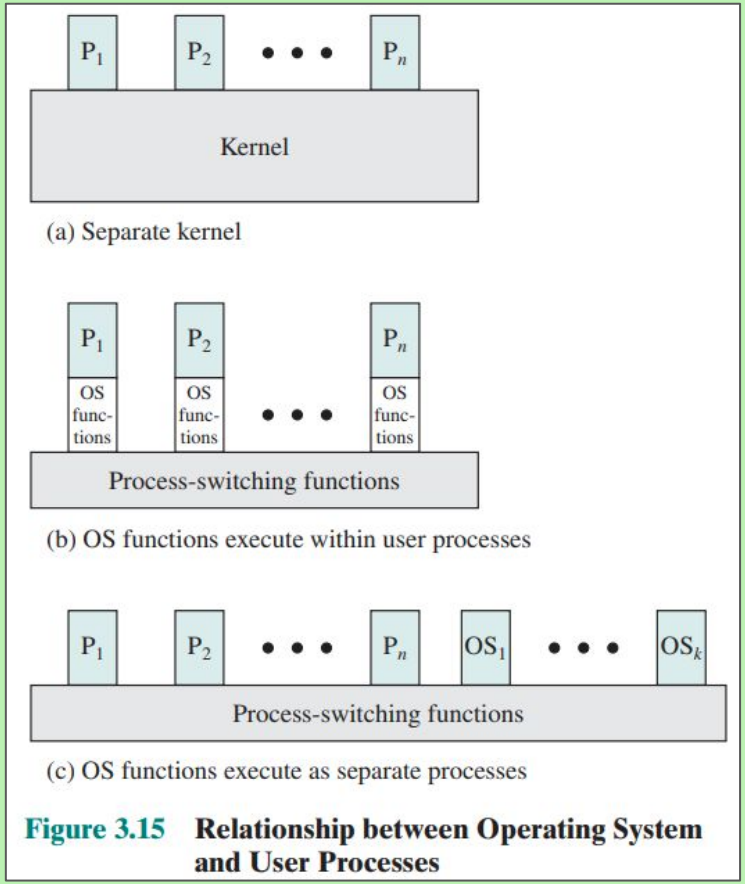


Figure 3.15 **Relationship between Operating System and User Processes**

EXECUTION OF THE OPERATING SYSTEM

Execution within User Processes

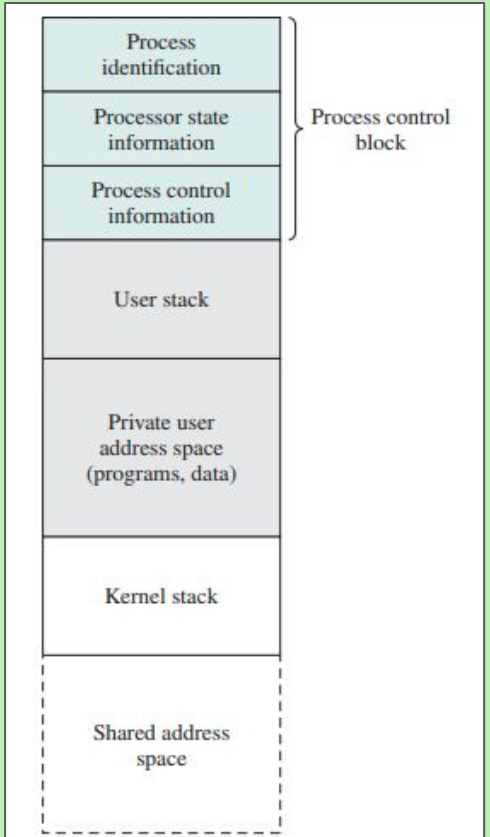


Figure 3.16 **Process Image: Operating System Executes within User Space**

EXECUTION OF THE OPERATING SYSTEM

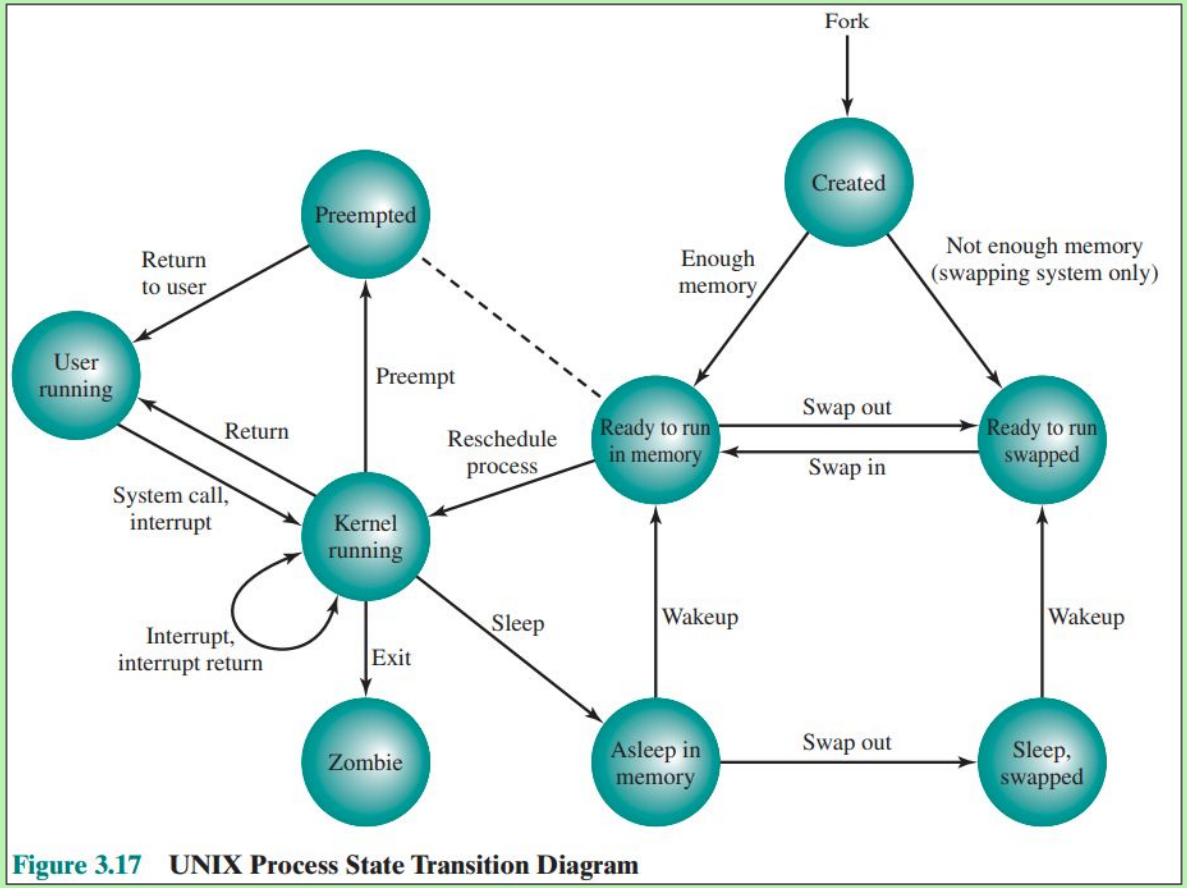


Figure 3.17 UNIX Process State Transition Diagram

EXECUTION OF THE OPERATING SYSTEM

Table 3.9 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.

EXECUTION OF THE OPERATING SYSTEM

Table 3.10 UNIX Process Image

User-Level Context	
Process text	Executable machine instructions of the program
Process data	Data accessible by the program of this process
User stack	Contains the arguments, local variables, and pointers for functions executing in user mode
Shared memory	Memory shared with other processes, used for interprocess communication
Register Context	
Program counter	Address of next instruction to be executed; may be in kernel or user memory space of this process
Processor status register	Contains the hardware status at the time of preemption; contents and format are hardware dependent
Stack pointer	Points to the top of the kernel or user stack, depending on the mode of operation at the time or preemption
General-purpose registers	Hardware dependent
System-Level Context	
Process table entry	Defines state of a process; this information is always accessible to the operating system
U (user) area	Process control information that needs to be accessed only in the context of the process
Per process region table	Defines the mapping from virtual to physical addresses; also contains a permission field that indicates the type of access allowed the process: read-only, read-write, or read-execute
Kernel stack	Contains the stack frame of kernel procedures as the process executes in kernel mode

EXECUTION OF THE OPERATING SYSTEM

Table 3.11 UNIX Process Table Entry

Process status	Current state of process.
Pointers	To U area and process memory area (text, data, stack).
Process size	Enables the operating system to know how much space to allocate the process.
User identifiers	The **real user ID** identifies the user who is responsible for the running process. The **effective user ID** may be used by a process to gain temporary privileges associated with a particular program; while that program is being executed as part of the process, the process operates with the effective user ID.
Process identifiers	ID of this process; ID of parent process. These are set up when the process enters the Created state during the fork system call.
Event descriptor	Valid when a process is in a sleeping state; when the event occurs, the process is transferred to a ready-to-run state.
Priority	Used for process scheduling.
Signal	Enumerates signals sent to a process but not yet handled.
Timers	Include process execution time, kernel resource utilization, and user-set timer used to send alarm signal to a process.
P_link	Pointer to the next link in the ready queue (valid if process is ready to execute).
Memory status	Indicates whether process image is in main memory or swapped out. If it is in memory, this field also indicates whether it may be swapped out or is temporarily locked into main memory.