Operating Systems:
Internals and Design Principles, 6/E
William Stallings

Process Description and Control

Dr. Sanam Shahla Rizvi

Roadmap

- How are processes represented and controlled by the OS.
- *Process states* which characterize the behaviour of processes.
- *Data structures* used to manage processes.
- Ways in which the OS uses these data structures to control process execution.
- Discuss process management in UNIX SVR4.

Requirements of an Operating System

- *Fundamental Task: Process Management*
- The Operating System must
  - Interleave the execution of multiple processes
  - Allocate resources to processes, and protect the resources of each process from other processes,
  - Enable processes to share and exchange information,
  - Enable synchronization among processes.

What is a *"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 the execution of a sequence of instructions, a current state, and an associated set of system instructions

Process Elements

- A process is comprised of:
  - Program code
  - Set of data
  - Control block (Number of attributes describing the state of the process)

Process Elements

- While the process is running it has a number of elements including
  - Identifier
  - State
  - Priority
  - Program counter
  - Memory pointers
  - Context data
  - I/O status information
  - Accounting information

Process Control Block

- Contains the process elements
- Created and manage by the operating system
- Allows support for multiple processes

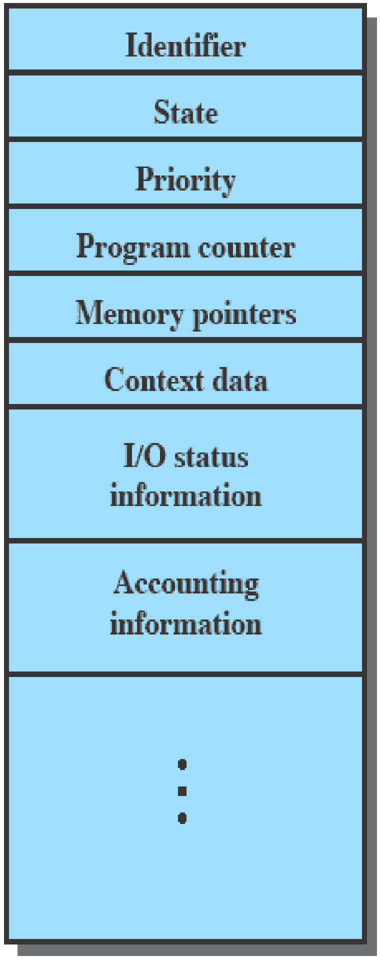
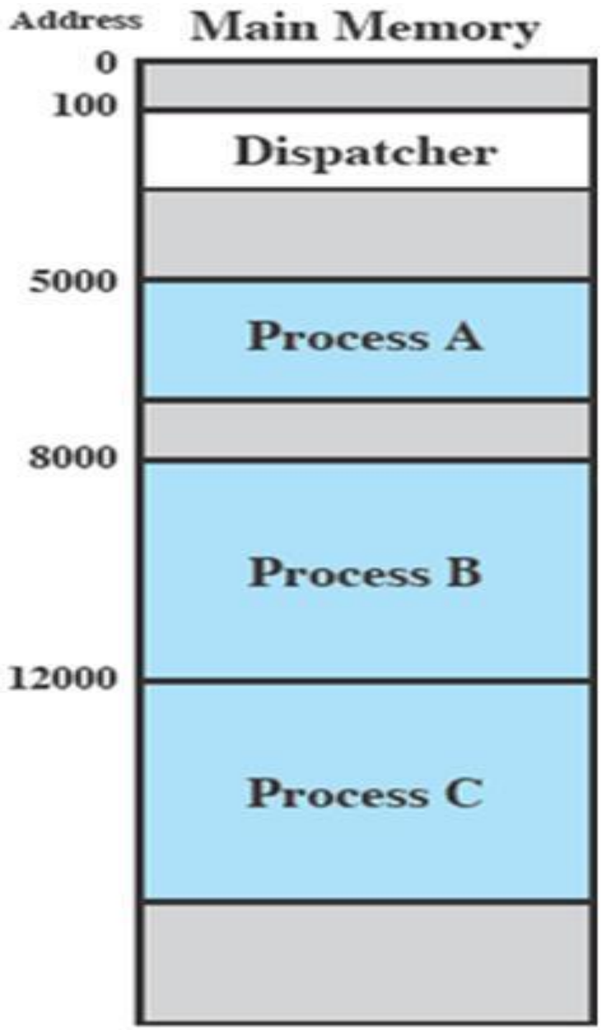


Figure 3.1 Simplified Process Control Block

Trace of the Process

- The behavior of an individual process is shown by listing the sequence of instructions that are executed
- This list is called a *Trace*
- *Dispatcher* is a small program which switches the processor from one process to another

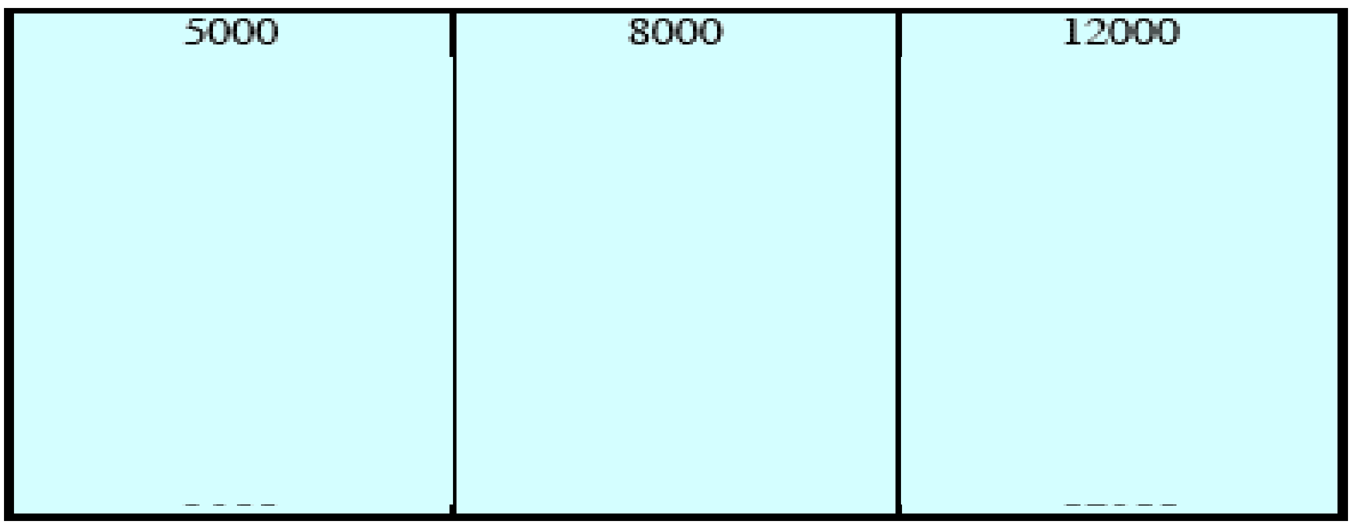
Process Execution



- Consider three processes being executed
- All are in memory (plus the dispatcher)
- Lets ignore virtual memory for this.

Trace from the *processes* point of view:

- Each process runs to completion

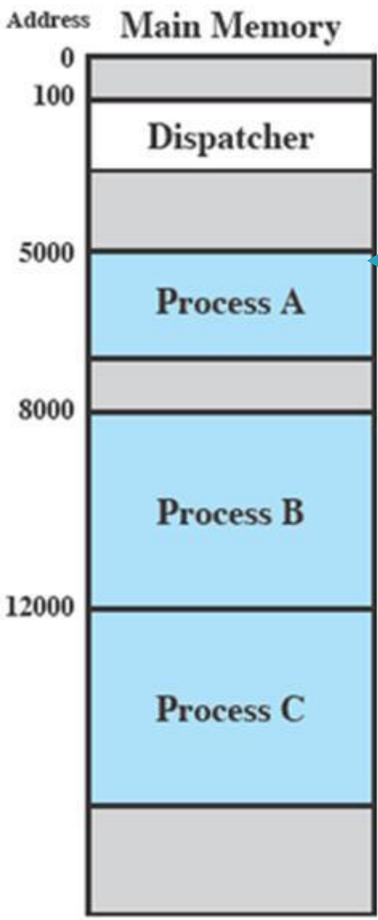


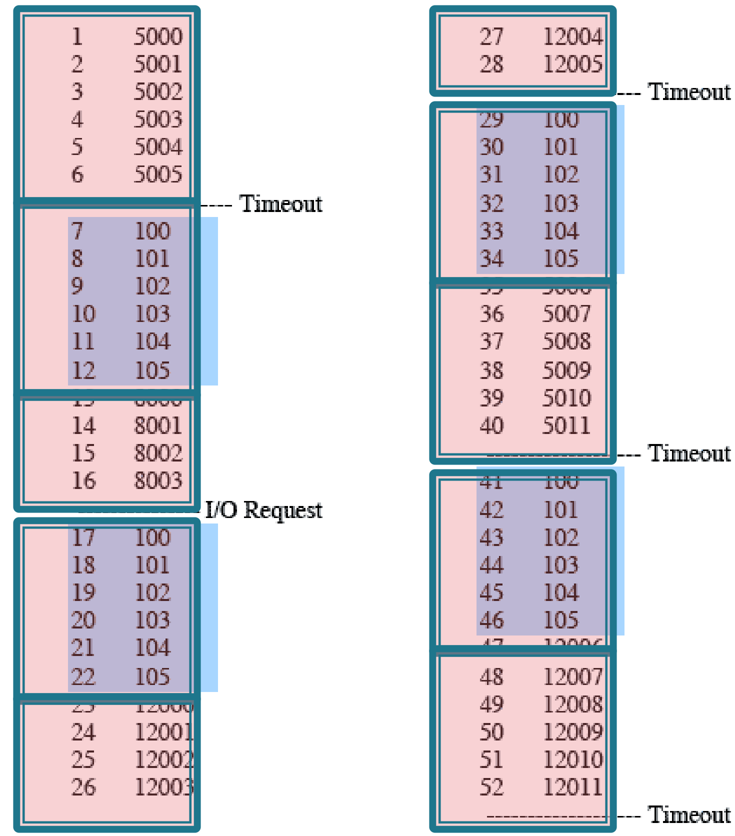
(a) Trace of Process A (b) Trace of Process B (c) Trace of Process C

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

Trace from Processors point of view





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

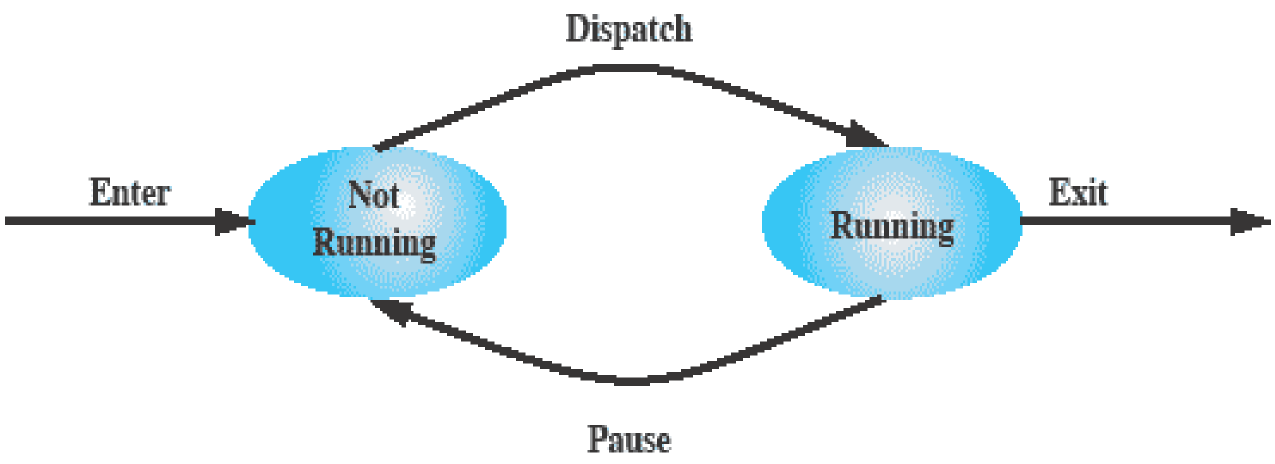
Figure 3.4 Combined Trace of Processes of Figure 3.2

Roadmap

- How are processes represented and controlled by the OS.
- *Process states* which characterize the behaviour of processes.
- *Data structures* used to manage processes.
- Ways in which the OS uses these data structures to control process execution.
- Discuss process management in UNIX SVR4.

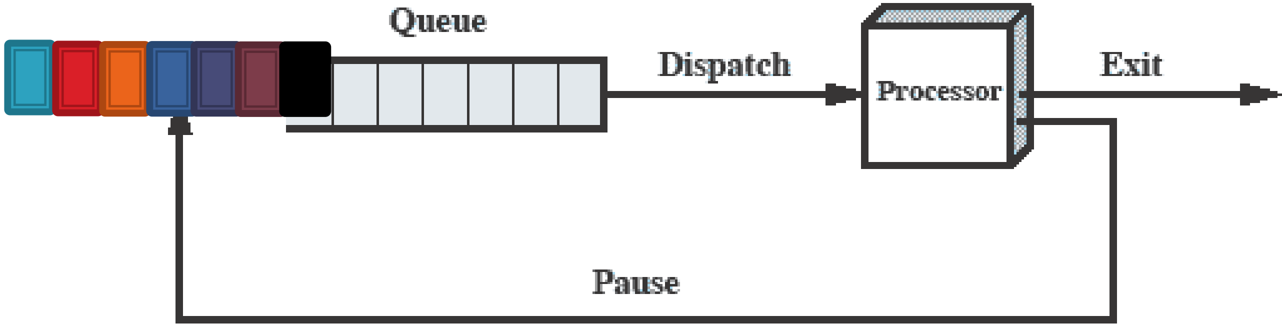
Two-State Process Model

- Process may be in one of two states
  - Running
  - Not-running



(a) State transition diagram

Queuing Diagram



(b) Queuing diagram

Processes moved by the dispatcher of the OS to the processor then back to the queue until the task is completed

Process Birth and Death

Creation	Termination
New batch job	Normal Completion
Interactive Login	Memory unavailable
Created by OS to provide a service	Protection error
Spawned by existing process	Operator or OS Intervention

Process Creation

- The OS builds a data structure to manage the process
- Traditionally, the OS created all processes
  - But it can be useful to let a running process create another
- This action is called ***process spawning***
  - *Parent Process* is the original, creating, process
  - *Child Process* is the new process

Five-State Process Model

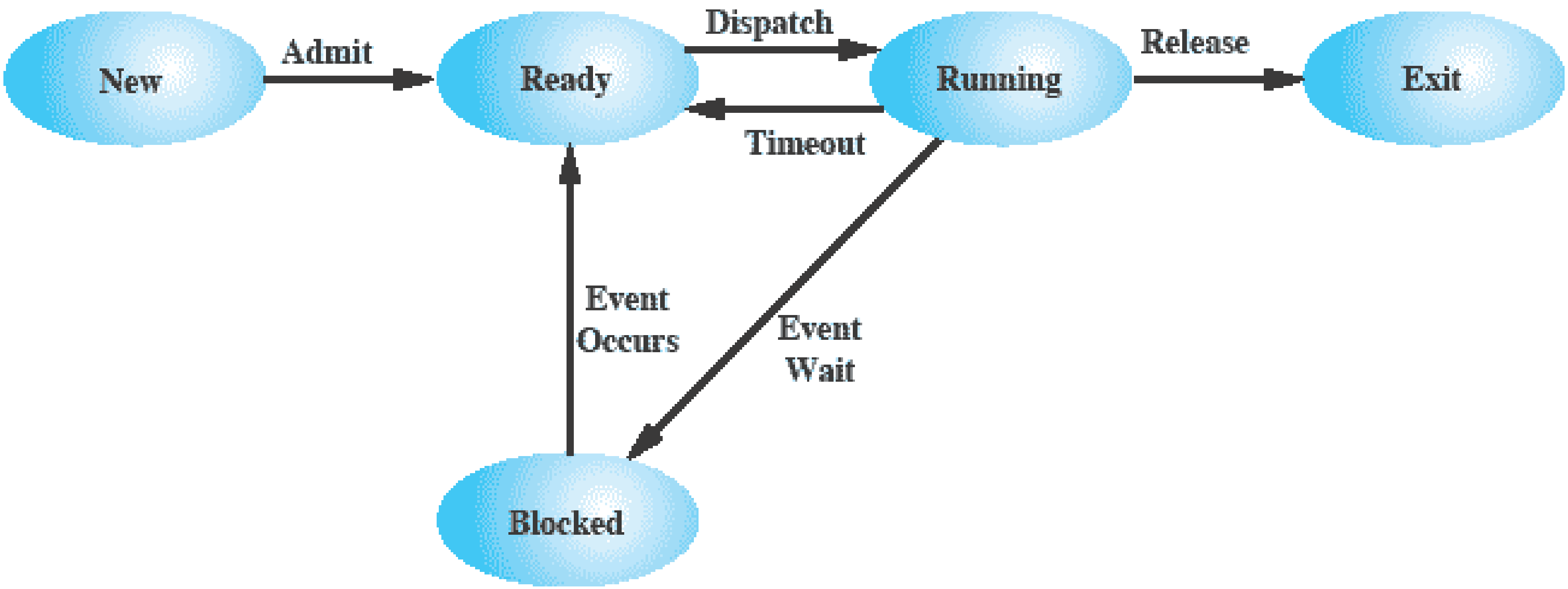
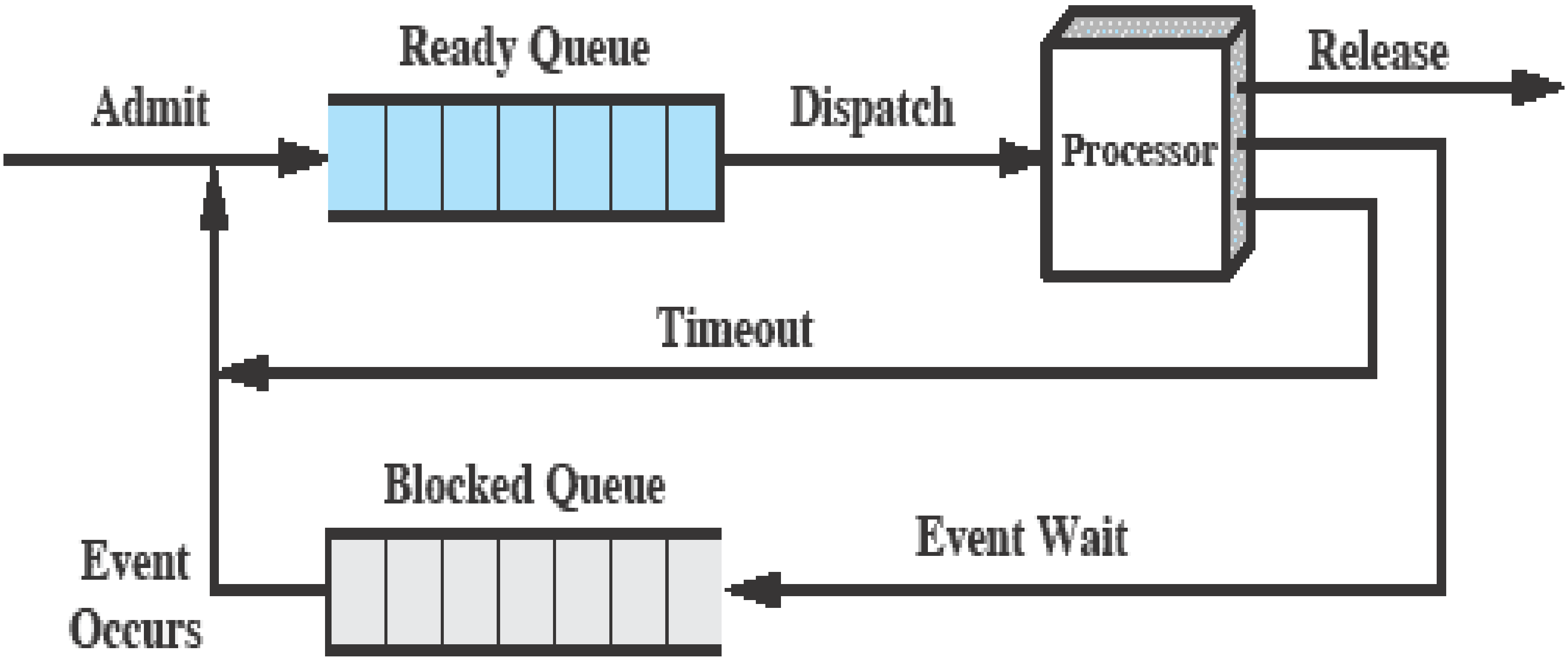


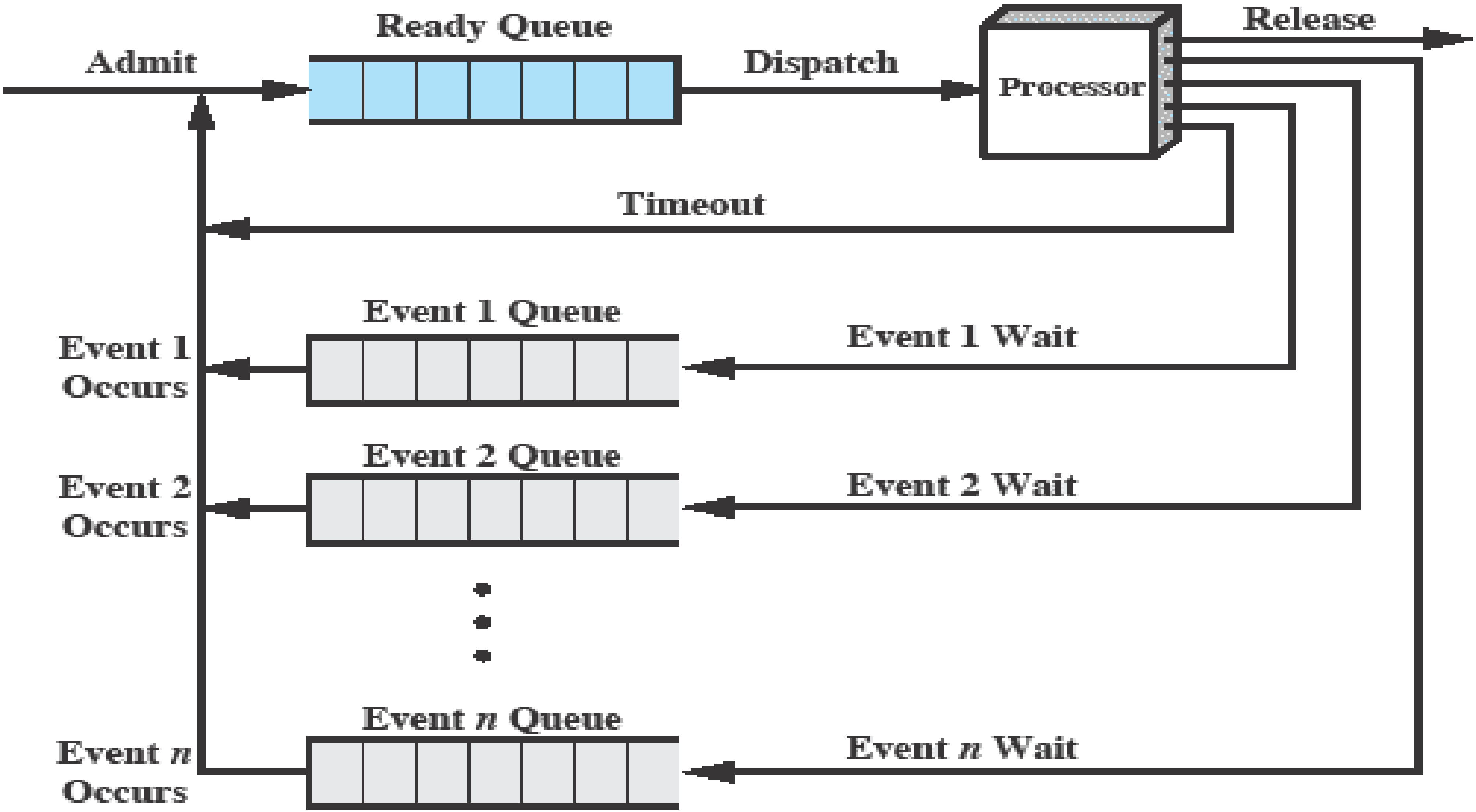
Figure 3.6 Five-State Process Model

Using Two Queues



(a) Single blocked queue

Multiple Blocked Queues



(b) Multiple blocked queues

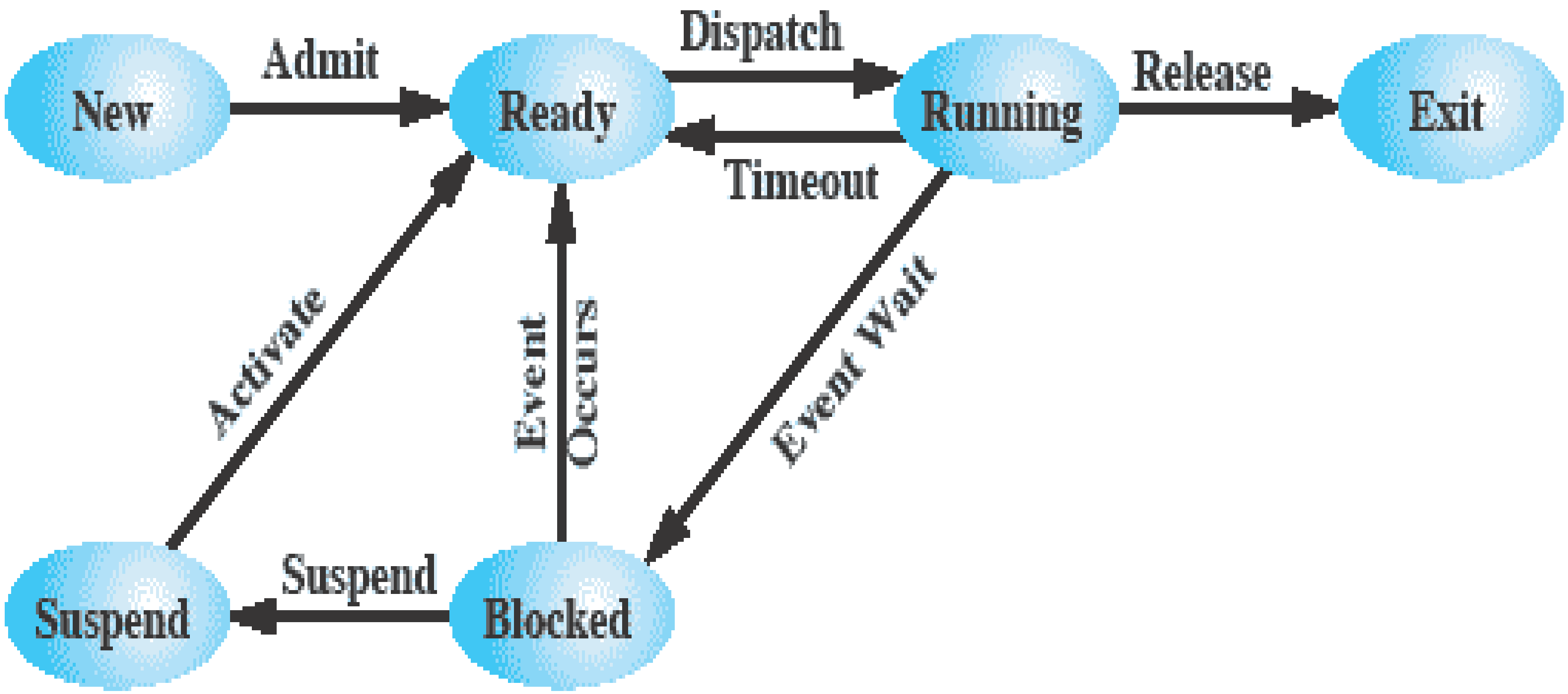
Suspended Processes

- Processor is faster than I/O so all processes could be waiting for I/O
  - Swap these processes to disk to free up more memory and use processor on more processes
- Blocked state becomes *suspend* state when swapped to disk
- Two new states
  - Blocked/Suspend
  - Ready/Suspend

Reason for Process Suspension

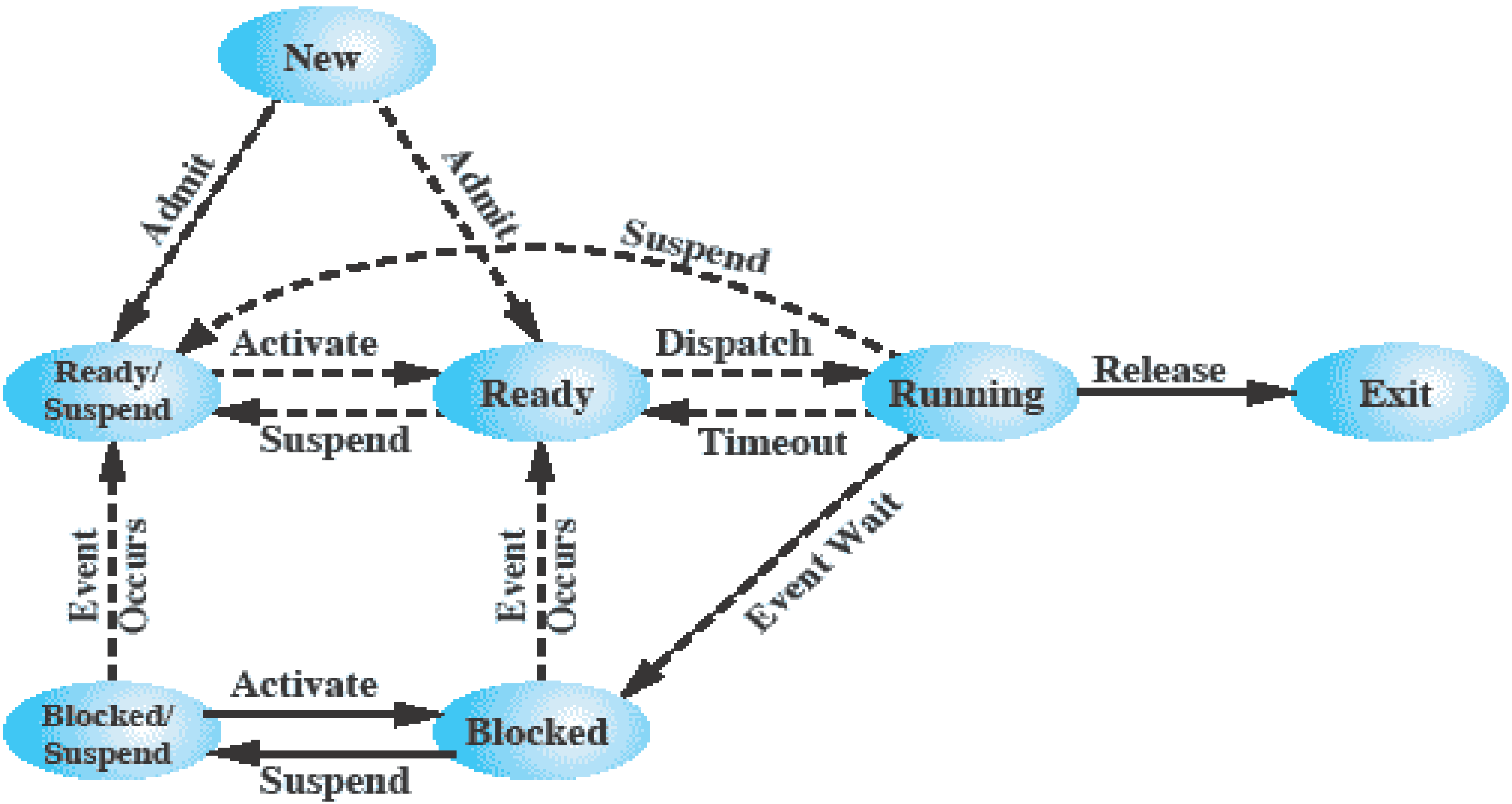
Reason	Comment
Swapping	The OS needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS Reason	OS suspects process of causing a problem.
Interactive User Request	e.g. debugging or in connection with the use of a resource.
Timing	A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time.
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.

One Suspend State



(a) With One Suspend State

Two Suspend States



(b) With Two Suspend States

Roadmap

- How are processes represented and controlled by the OS.
- *Process states* which characterize the behaviour of processes.
- *Data structures* used to manage processes.
- Ways in which the OS uses these data structures to control process execution.
- Discuss process management in UNIX SVR4.

Processes and Resources

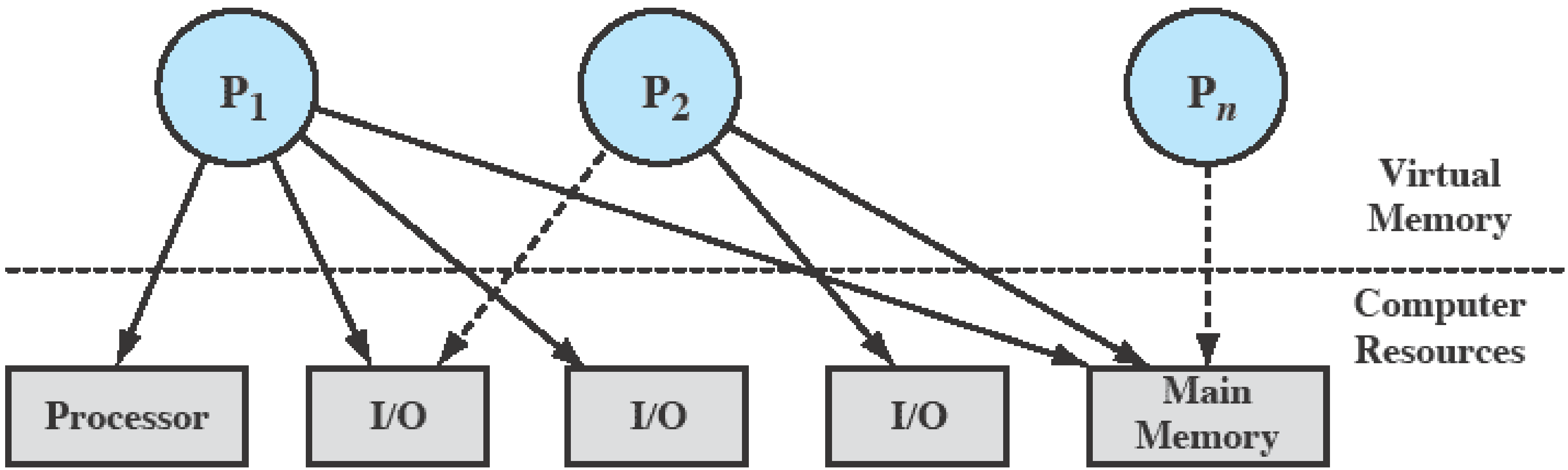


Figure 3.10 Processes and Resources (resource allocation at one snapshot in time)

Operating System Control Structures

- For the OS, to manage processes and resources, it must have information about the current status of each process and resource.
- Tables are constructed for each entity the operating system manages.

OS Control Tables

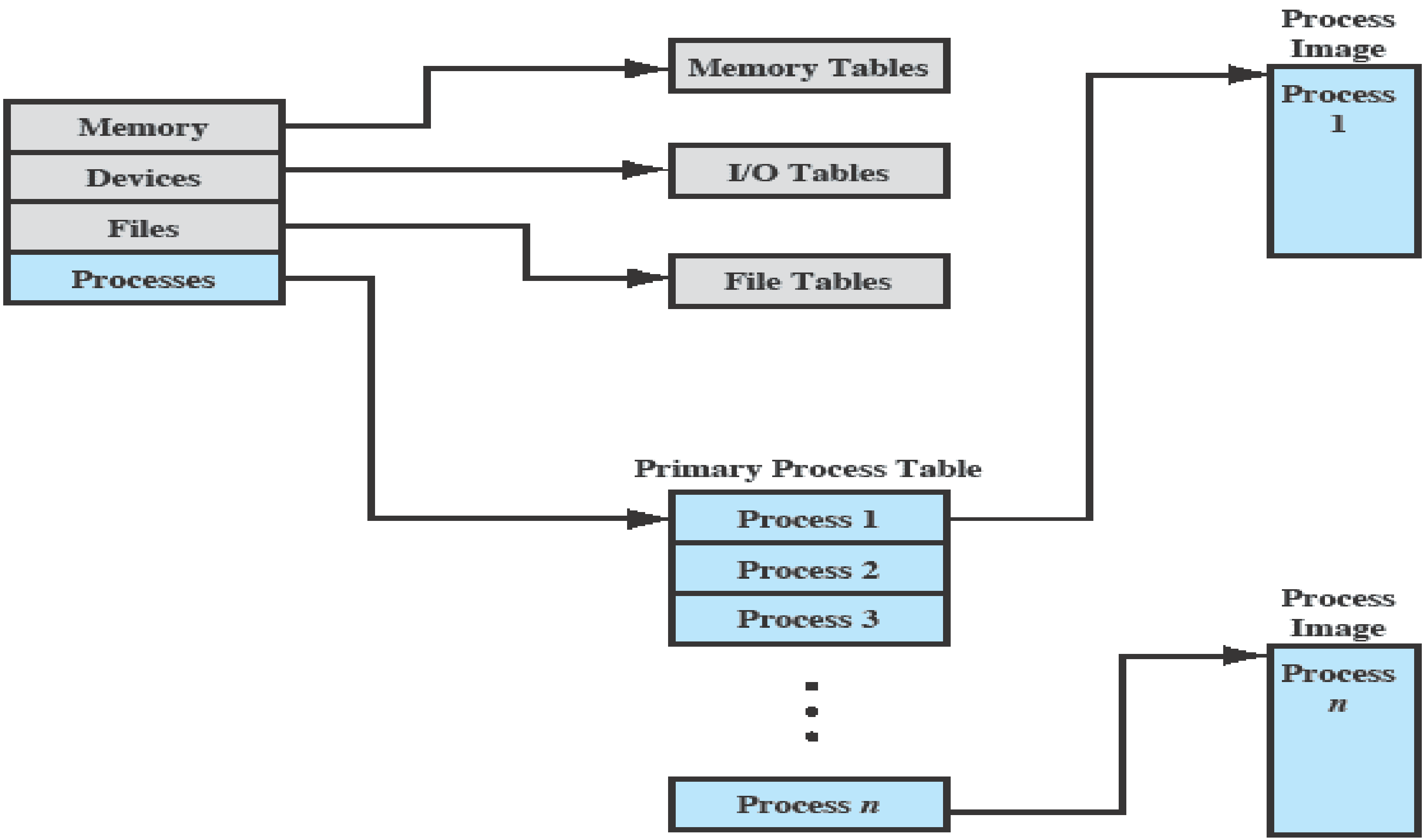


Figure 3.11 General Structure of Operating System Control Tables

Memory Tables

- Memory tables are used to keep track of both main and secondary memory.
- Must include this 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.
- The OS needs to know
  - Whether the I/O device is available or assigned
  - The status of I/O operation
  - The location in main memory being used as the source or destination of the I/O transfer

File Tables

- These tables provide information about:
  - Existence of files
  - Location on secondary memory
  - Current Status
  - Other attributes.
- Sometimes this information is maintained by a file management system

Process Tables

- To manage processes the OS needs to know details of the processes
  - Process ID
  - Current state
  - Location in memory
  - etc
- Process Image
  - *C*ollection of program, data, stack and attributes (process control block).

Process Attributes

- We can group the process control block information into three general categories:
  - Process identification
  - Processor state information
  - Process control information

Process Identification

- Each process is assigned a unique numeric identifier.
- Many of the other tables controlled by the OS may use process identifiers to cross-reference process tables.

Processor State Information

- This consists of the contents of processor registers.
  - User-visible registers
    - Data
    - Address
  - Control and status registers
    - PC
    - IR
    - PSW
  - Stack pointers

Process Control Information

- This is the additional information needed by the OS to control and coordinate the various active processes.
  - Scheduling & state information
    - State (running, ready, waiting, …)
    - Priority (default, current, highest allowable)
  - Inter-Process Communication
  - Process Privileges
  - Memory Management

Structure of Process Images in Virtual Memory

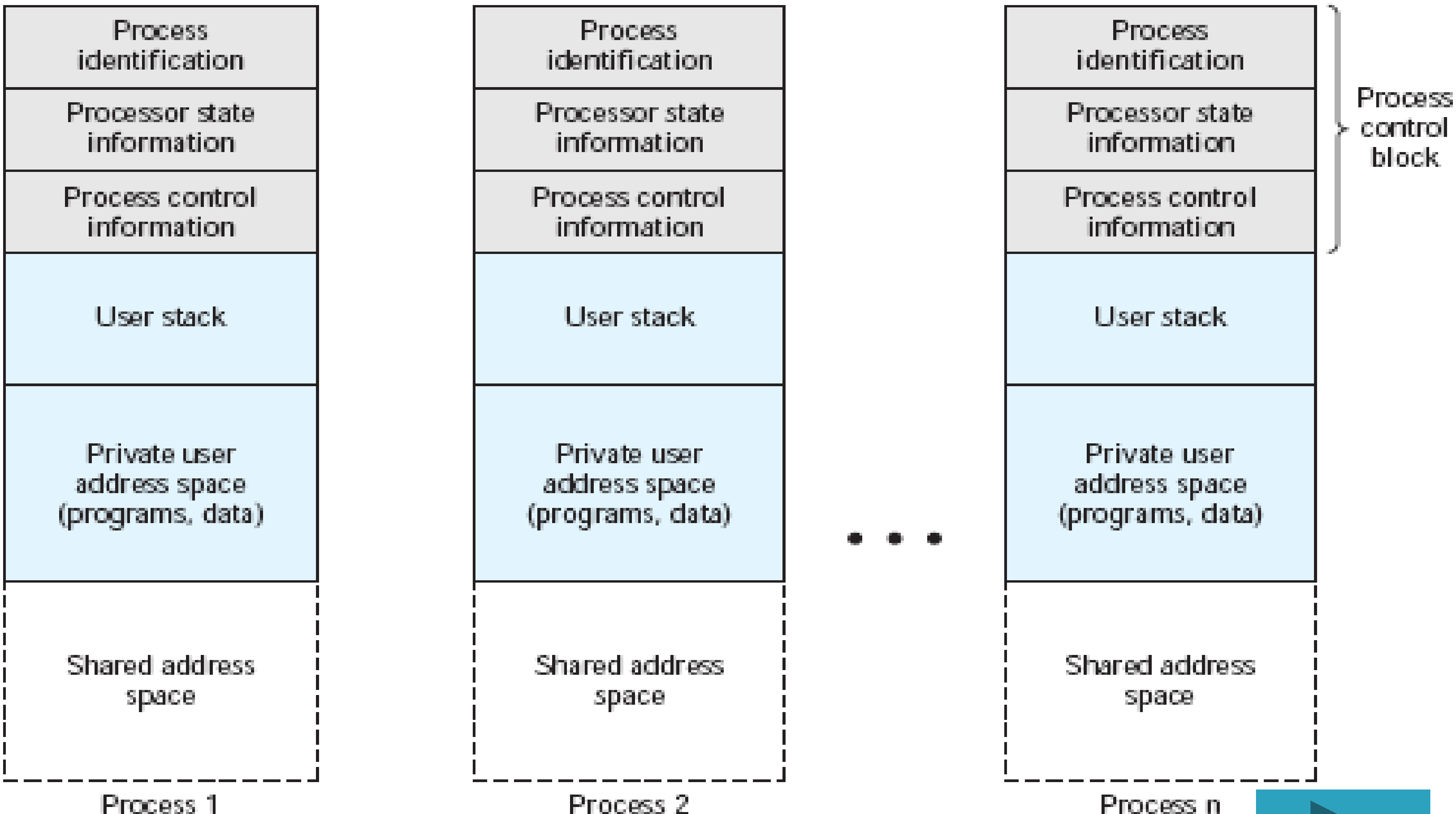


Figure 3.13 User Processes in Virtual Memory

Role of the Process Control Block

- The most important data structure in an OS
  - It defines the state of the OS
- Process Control Block requires protection
  - A faulty routine could cause damage to the block destroying the OS's ability to manage the process
  - Any design change to the block could affect many modules of the OS

Roadmap

- How are processes represented and controlled by the OS.
- *Process states* which characterize the behaviour of processes.
- *Data structures* used to manage processes.
- Ways in which the OS uses these data structures to control process execution.
- Discuss process management in UNIX SVR4.

OS, Data Structure & Process Execution

- Modes of Execution
- Process Creation
- Process Switching

- Is the OS a Process?
- Execution of OS

Modes of Execution

- Most processors support at least two modes of execution
- User mode
  - Less-privileged mode
  - User programs typically execute in this mode
- System mode
  - More-privileged mode
  - Kernel of the operating system

Process Creation

- Once the OS decides to create a new process, it:
  - Assigns a unique process identifier
  - Allocates space for the process
  - Initializes process control block
  - Sets up appropriate linkages
  - Creates or expand other data structures

Process Switching

- Several design issues are raised regarding process switching:
  - What events trigger a process switch?
  - We must distinguish 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

A process switch may occur any time that the OS has gained control from the currently running process. 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

Change of Process State

- The steps in a process switch are:
  1. Save context of processor including program counter and other registers
  2. Update the process control block of the process that is currently in the Running state
  3. Move process control block to appropriate queue – ready; blocked; ready/suspend
  4. Select another process for execution
  5. Update the process control block of the process selected
  6. Update memory-management data structures
  7. Restore context of the processor

Is the OS a Process?

- If the OS is just a collection of programs and if it is executed by the processor just like any other program, is the OS a process?
- If so, how is it controlled?
  - Who (what) controls it?

Execution of the Operating System

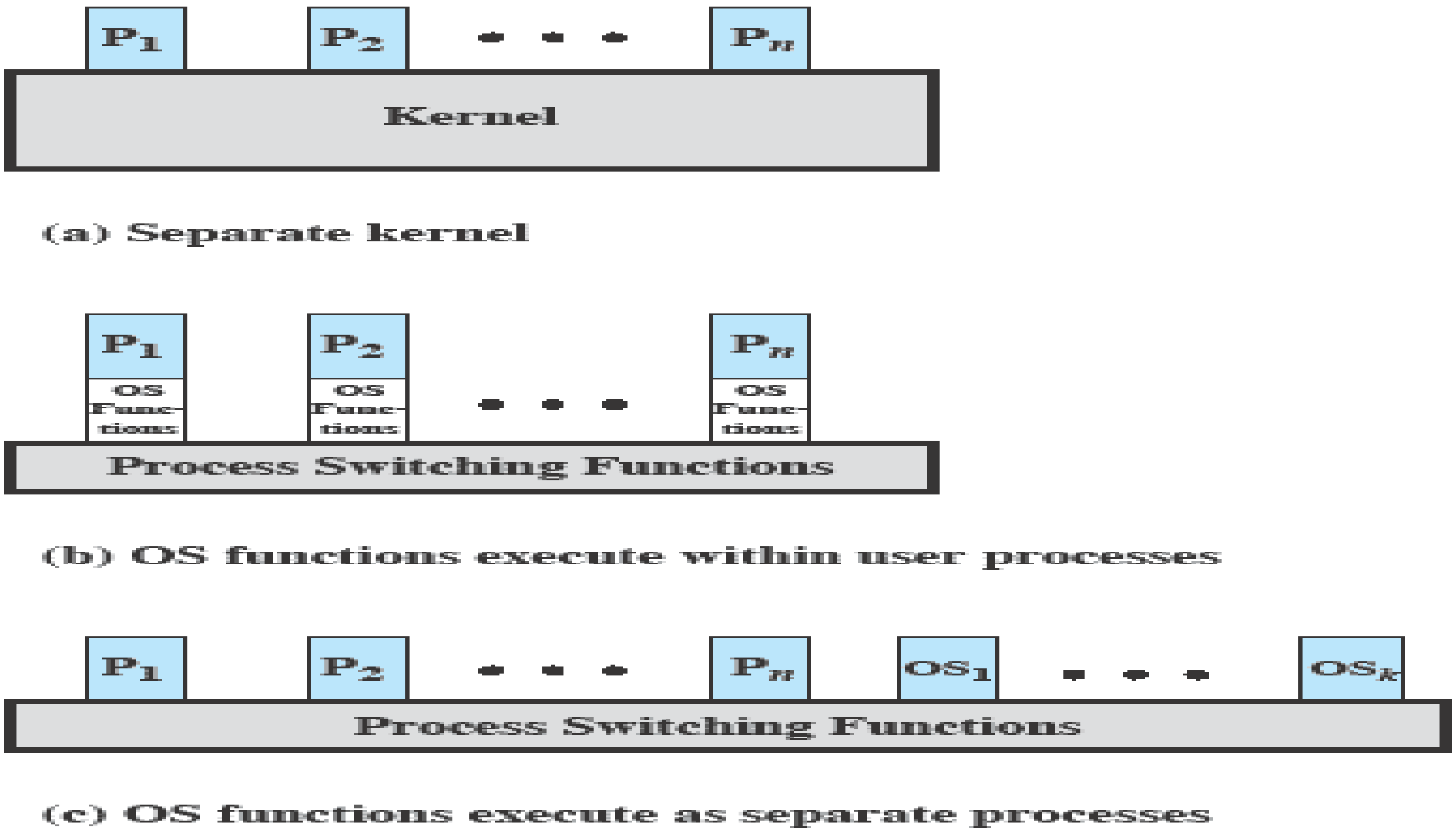
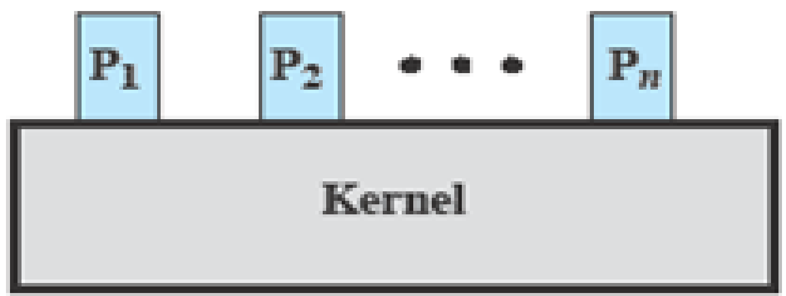


Figure 3.15 Relationship Between Operating System and User Processes

Non-process Kernel

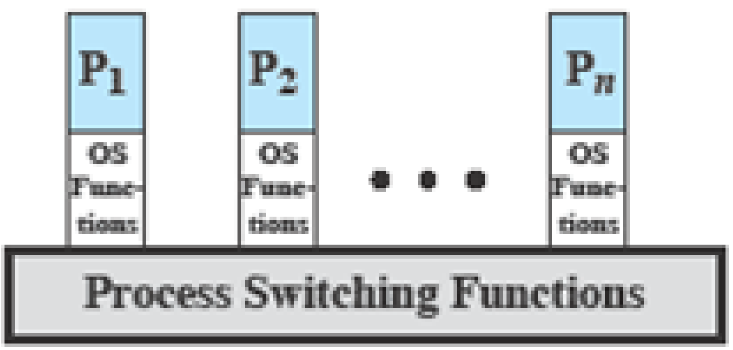
- Execute kernel outside of any process
- The concept of process is considered to apply only to user programs
  - Operating system code is executed as a separate entity that operates in privileged mode



(a) Separate kernel

Execution *Within* User Processes

- Execution Within User Processes
  - Operating system software within context of a user process
  - No need for Process Switch to run OS routine



(b) OS functions execute within user processes

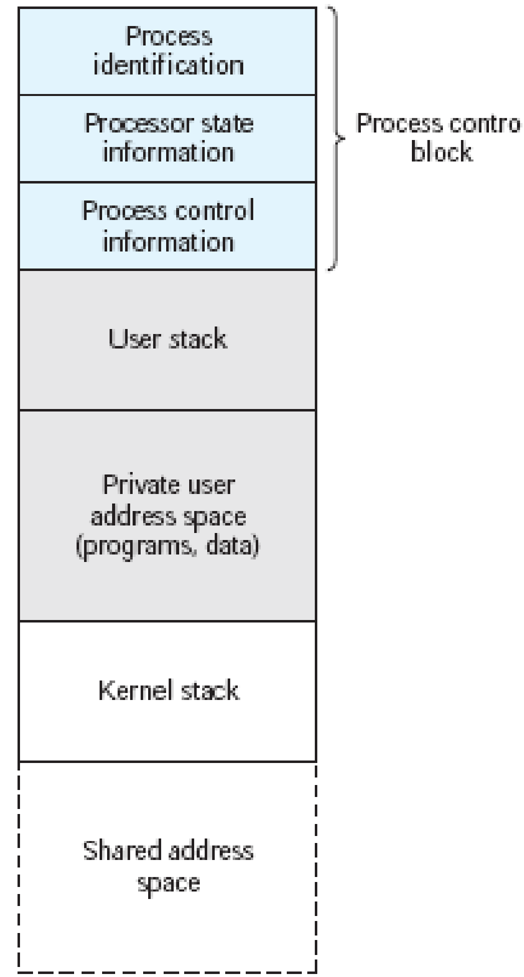
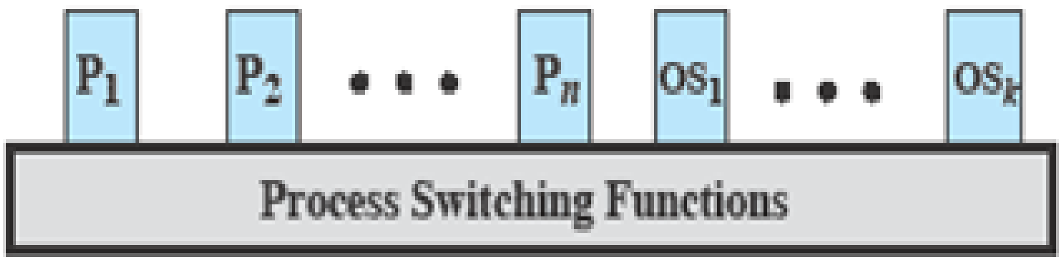


Figure 3.16 **Process Image: Operating System Executes within User Space**

Process-based Operating System

- Process-based operating system
  - Implement the OS as a collection of system processes.



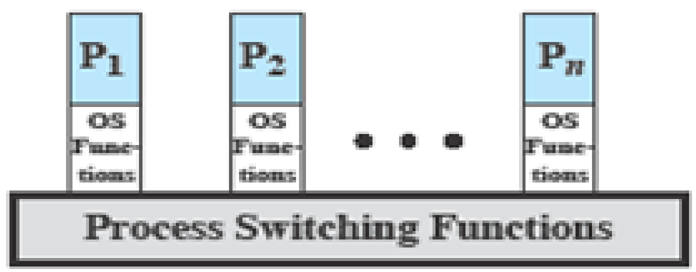
(c) OS functions execute as separate processes

Roadmap

- How are processes represented and controlled by the OS.
- *Process states* which characterize the behaviour of processes.
- *Data structures* used to manage processes.
- Ways in which the OS uses these data structures to control process execution.
- Discuss process management in UNIX SVR4.

Unix SVR4 (System V Release 4)

- Most of the OS executes in the user process
- System Processes – Kernel mode only
- User Processes
  - User mode to execute user programs and utilities
  - Kernel mode to execute instructions that belong to the kernel.



(b) OS functions execute within user processes

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

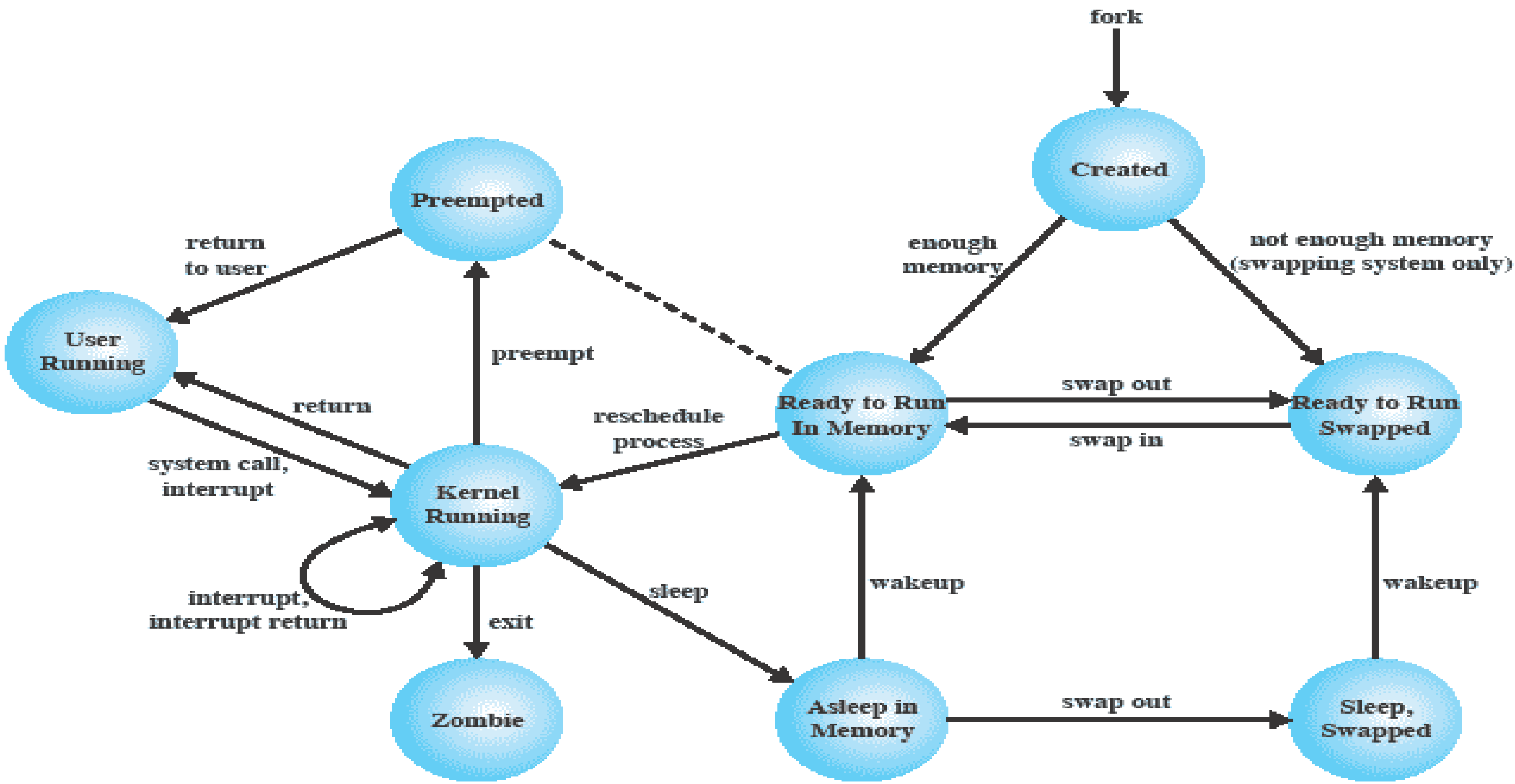


Figure 3.17 UNIX Process State Transition Diagram

Process Creation

- Process creation is by means of the kernel system call,fork( ).
- This causes the OS, in Kernel Mode, to:
    1. Allocate a slot in the process table for the new process.
    2. Assign a unique process ID to the child process.
    3. Copy of process image of the parent, with the exception of any shared memory.

Process Creation cont…

4. Increment the counters for any files owned by the parent, to reflect that an additional process now also owns those files.
5. Assign the child process to the Ready to Run state.
6. Returns the ID number of the child to the parent process, and a 0 value to the child process.

After Creation

- After creating the process the Kernel can do one of the following, as part of the dispatcher routine:
  - Stay in the parent process.
  - Transfer control to the child process
  - Transfer control to another process.