Process Description and Control

Chapter 3

Major Requirements of an Operating System

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

Process

- Also called a task
- Execution of an individual program
- Can be traced
  - list the sequence of instructions that execute

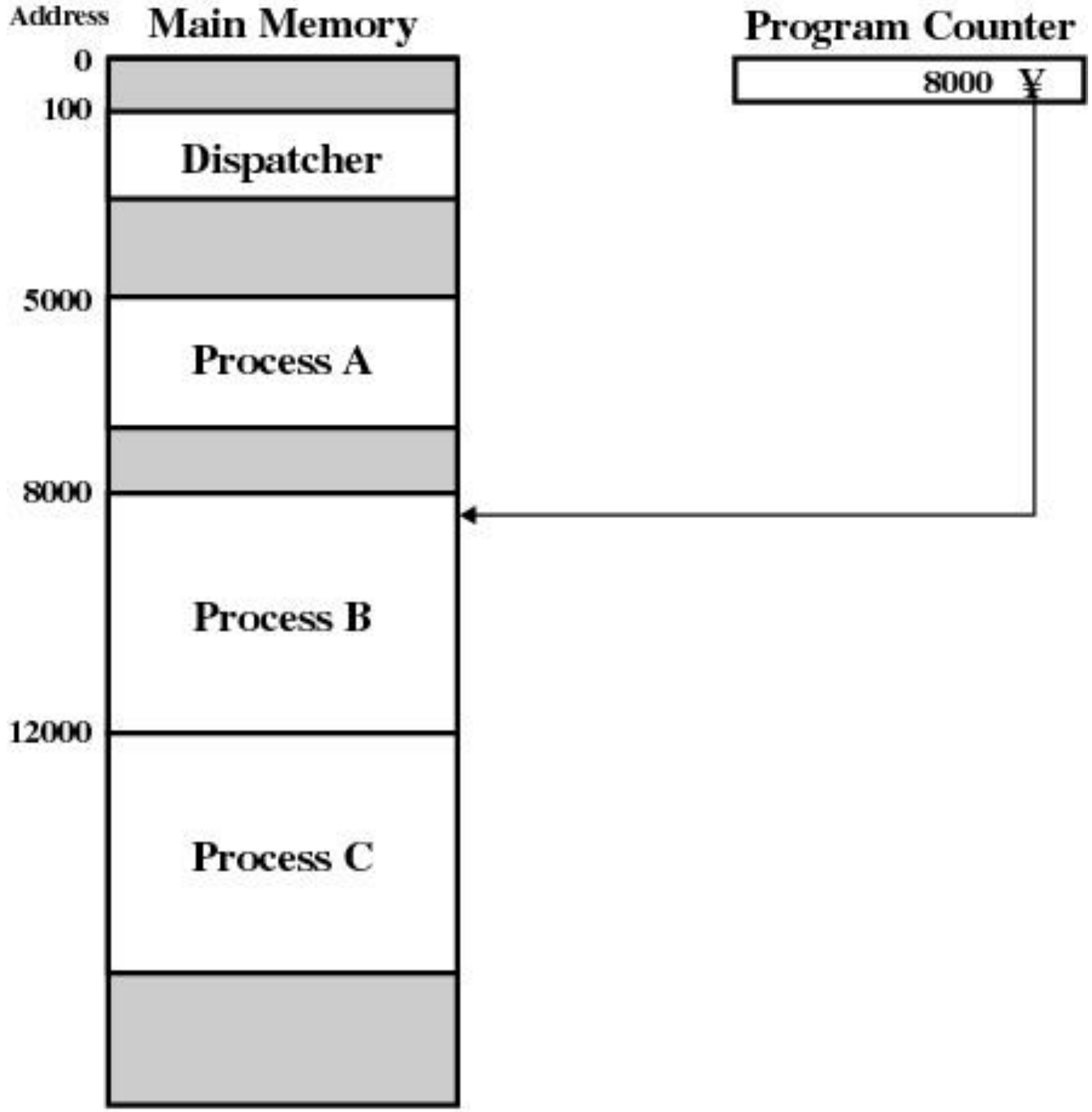


Figure 3.1 Snapshot of Example Execution (Figure 3.3) at Instruction Cycle 13

(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 3.2 Traces of Processes of Figure 3.1

1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
---------------------Time out	
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
---------------------Time out	
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
---------------------Time out	
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
---------------------Time out	

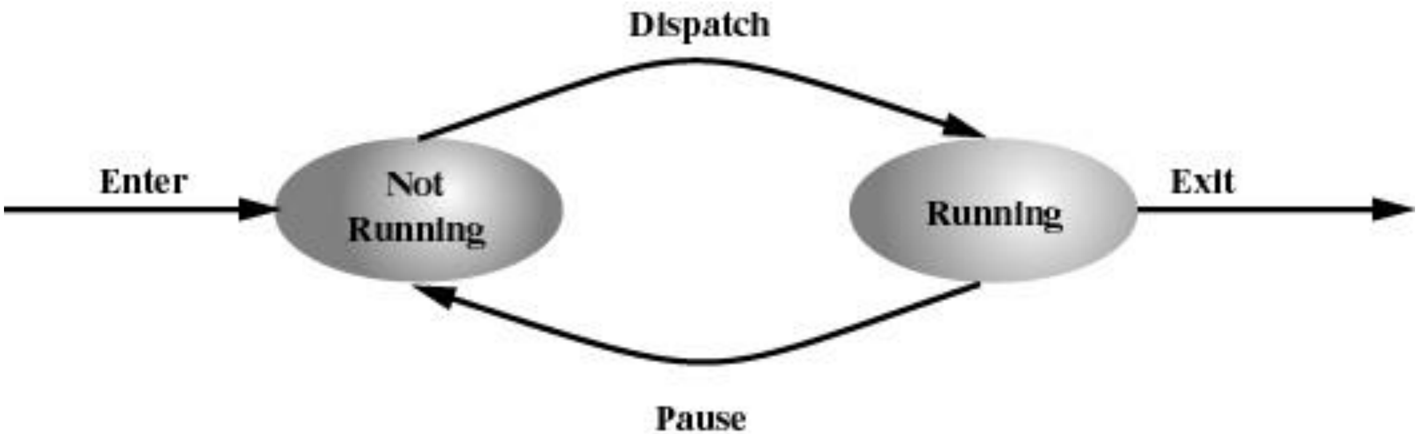
100 = Starting address of dispatcher program

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 3.3 Combined Trace of Processes of Figure 3.1

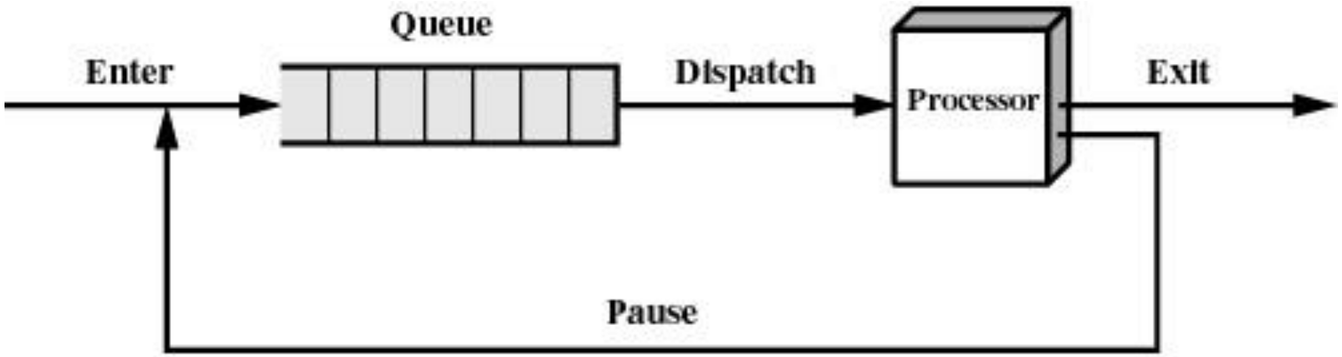
Two-State Process Model

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



(a) State transition diagram

Not-Running Process in a Queue



(b) Queuing diagram

Process Creation

- Submission of a batch job
- User logs on
- Created to provide a service such as printing
- Process creates another process

Process Termination

- Batch job issues *Halt* instruction
- User logs off
- Quit an application
- Error and fault conditions

Reasons for Process Termination

- Normal completion
- Time limit exceeded
- Memory unavailable
- Bounds violation
- Protection error
  - example write to read-only file
- Arithmetic error
- Time overrun
  - process waited longer than a specified maximum for an event

Reasons for Process Termination

- I/O failure
- Invalid instruction
  - happens when try to execute data
- Privileged instruction
- Data misuse
- Operating system intervention
  - such as when deadlock occurs
- Parent terminates so child processes terminate
- Parent request

Processes

- Not-running
  - ready to execute
- Blocked
  - waiting for I/O
- Dispatcher cannot just select the process that has been in the queue the longest because it may be blocked

A Five-State Model

- Running
- Ready
- Blocked
- New
- Exit

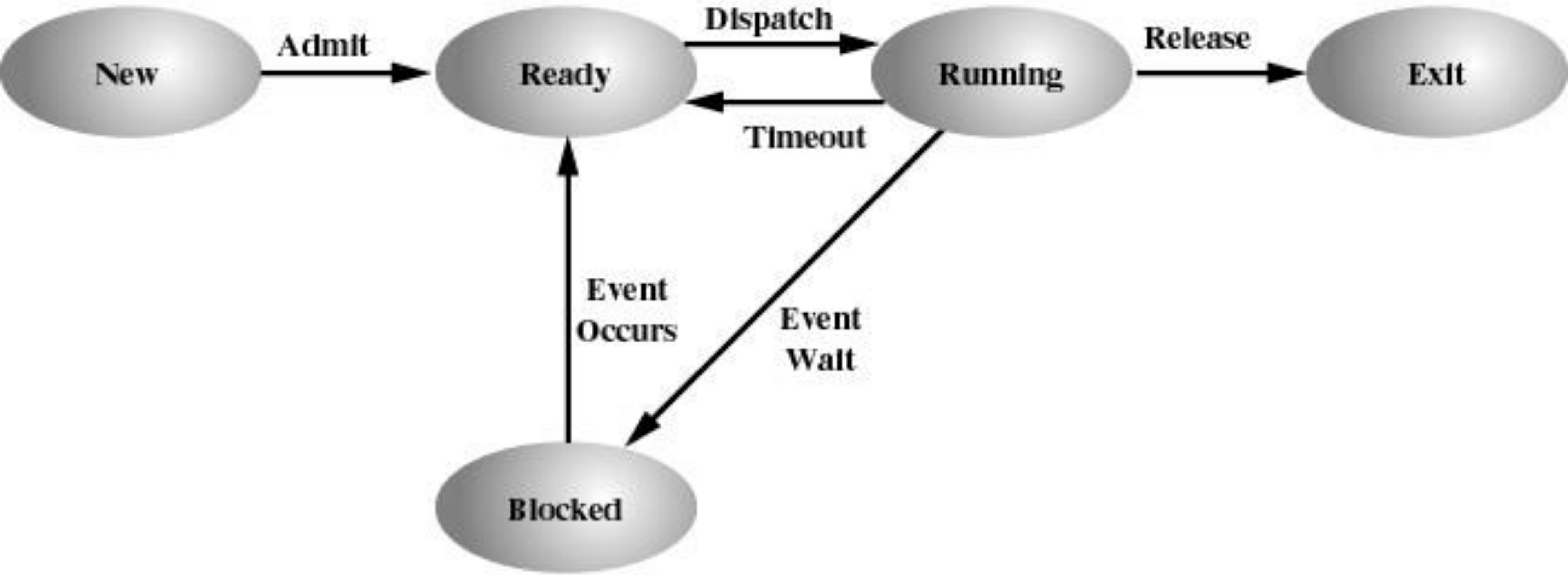


Figure 3.5 Five-State Process Model

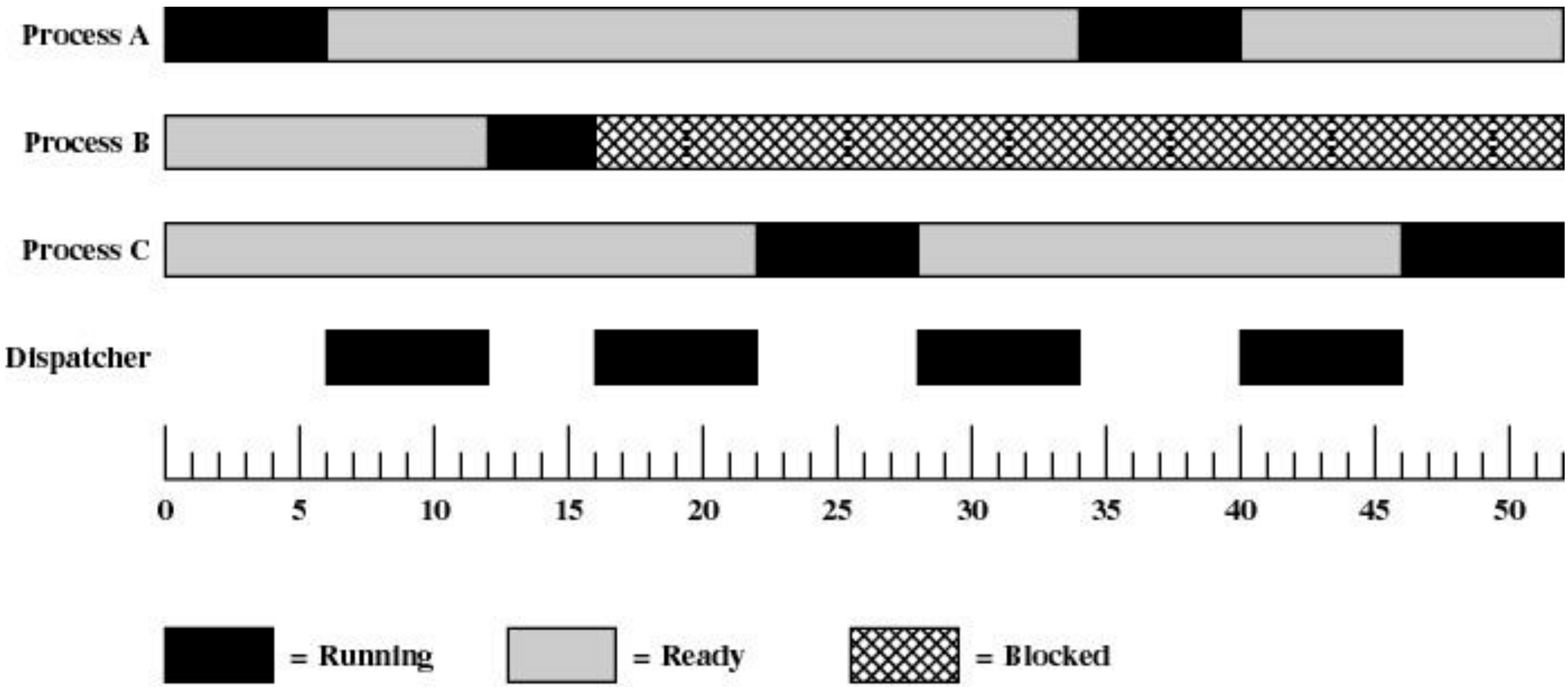
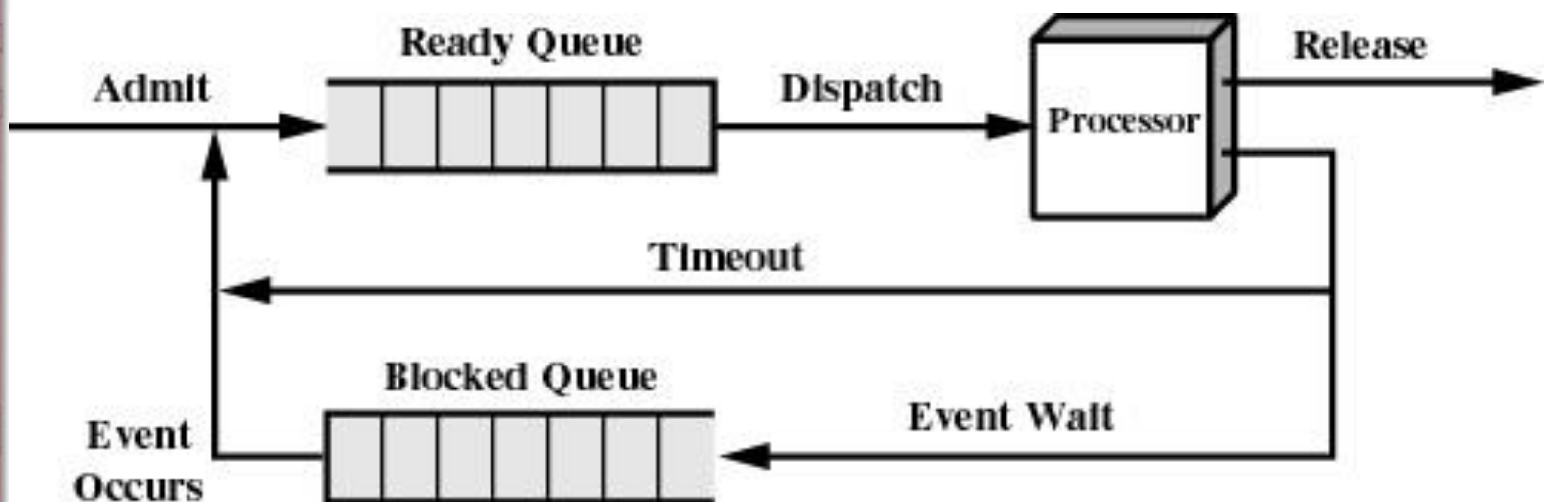
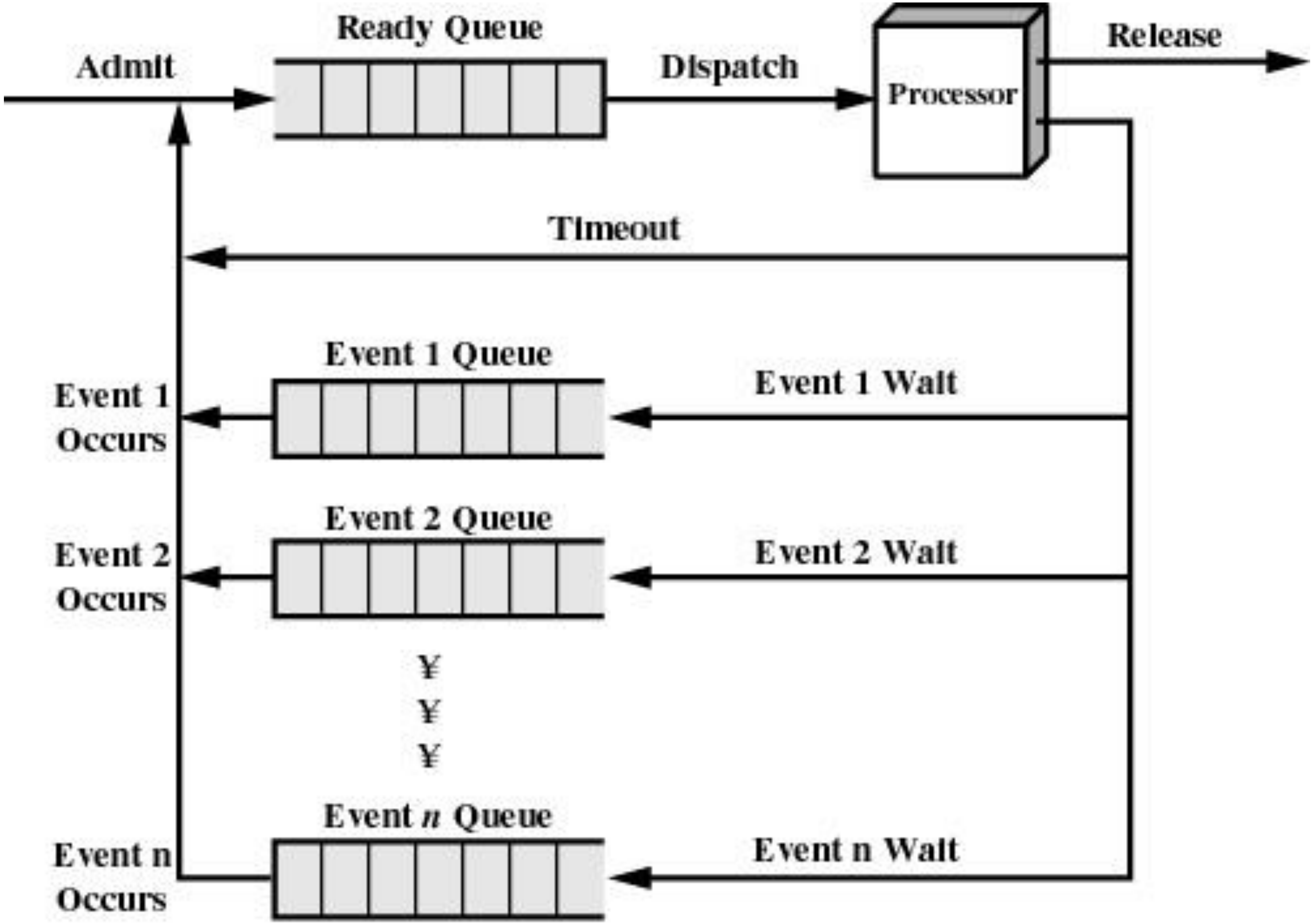


Figure 3.6 Process States for Trace of Figure 3.3

Using Two Queues



(a) Single blocked queue

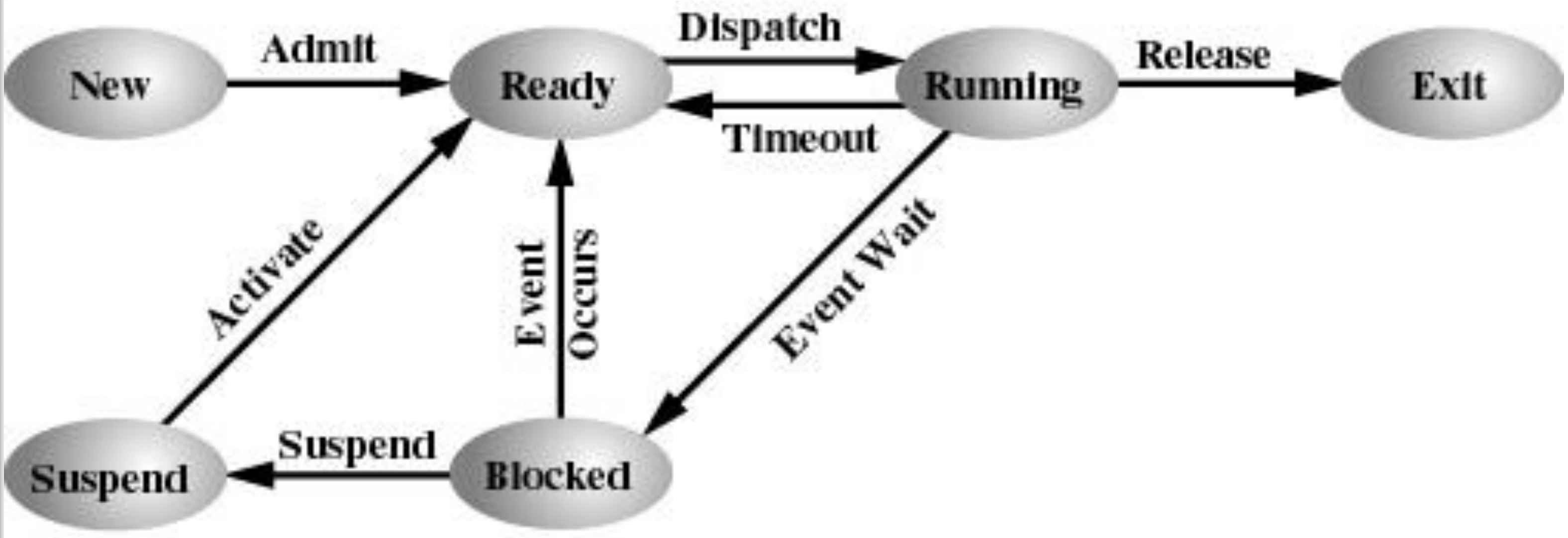


(b) Multiple blocked queues

Suspended Processes

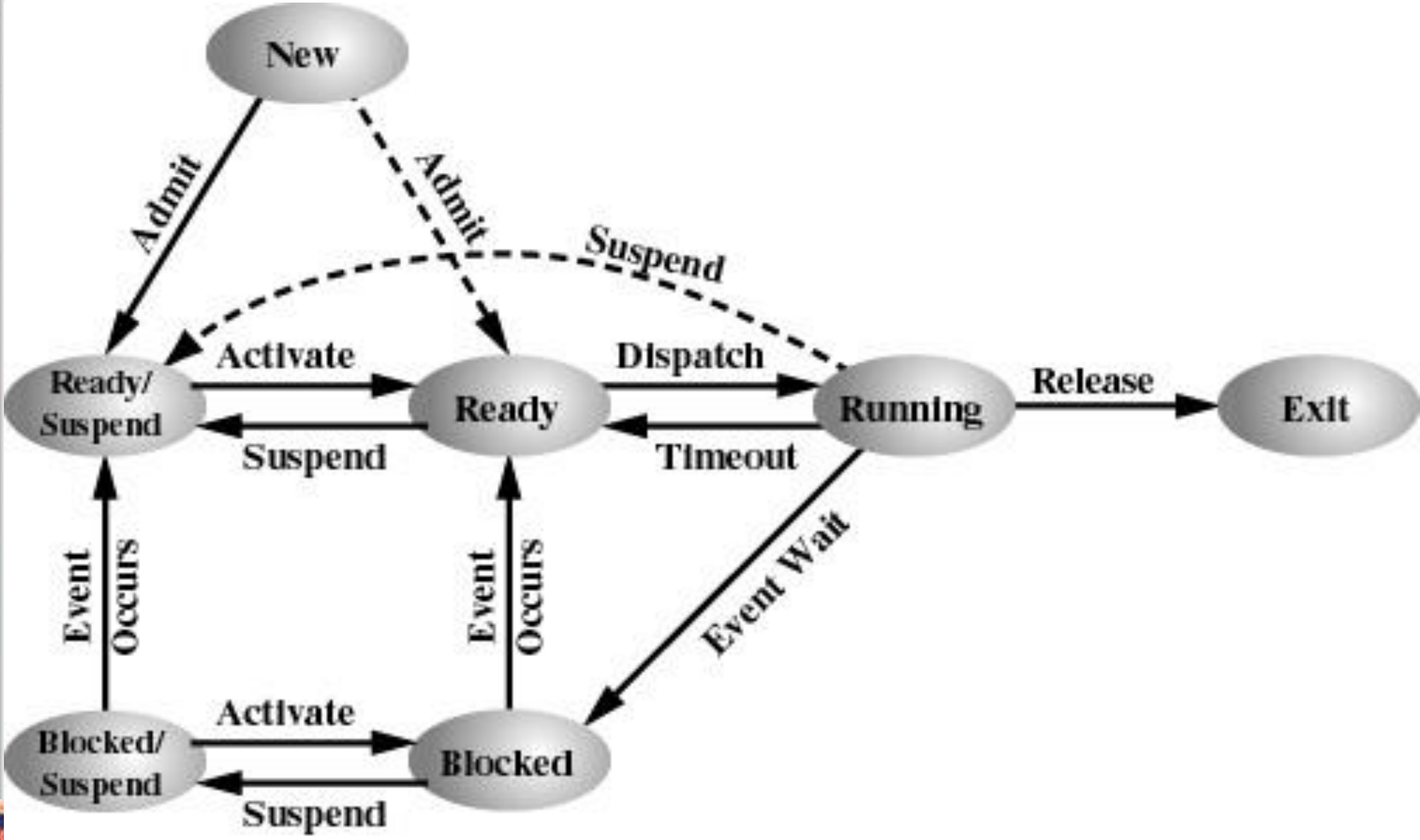
- Processor is faster than I/O so all processes could be waiting for I/O
- Swap these processes to disk to free up more memory
- Blocked state becomes suspend state when swapped to disk
- Two new states
  - Blocked, suspend
  - Ready, suspend

One Suspend State



(a) With One Suspend State

Two Suspend States



(b) With Two Suspend States

Reasons for Process Suspension

Swapping	The operating system needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS reason	The operating system 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 descendents.

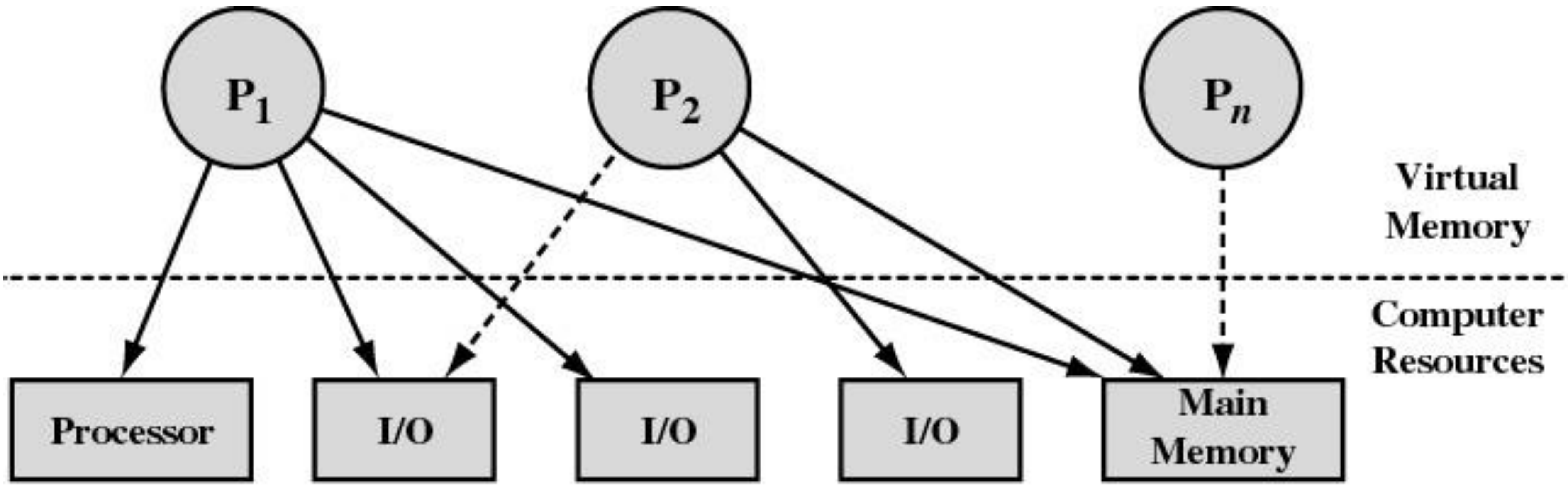


Figure 3.9 Processes and Resources (resource allocation at one snapshot in time)

Operating System Control Structures

- Information about the current status of each process and resource
- Tables are constructed for each entity the operating system manages

Memory Tables

- 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

- I/O device is available or assigned
- Status of I/O operation
- Location in main memory being used as the source or destination of the I/O transfer

File Tables

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

Process Table

- Where process is located
- Attributes necessary for its management
  - Process ID
  - Process state
  - Location in memory

Process Location

- Process includes set of programs to be executed
  - Data locations for local and global variables
  - Any defined constants
  - Stack
- Process control block
  - Collection of attributes
- Process image
  - Collection of program, data, stack, and attributes

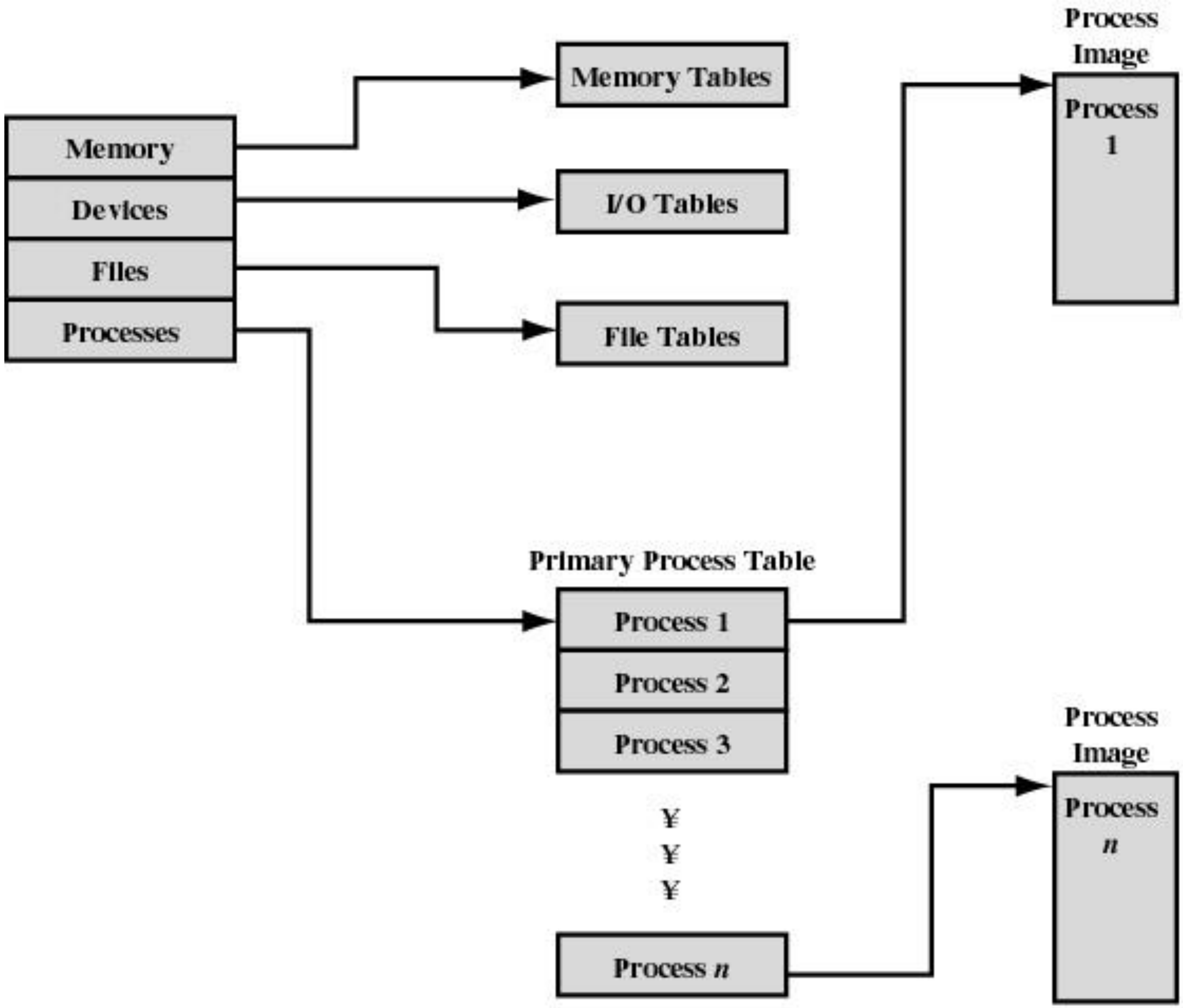


Figure 3.10 General Structure of Operating System Control Tables

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

Process Control Block

- 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. Typically, there are from 8 to 32 of these registers, although some RISC implementations have over 100.

Process Control Block

- Processor State Information
  - 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

Process Control Block

- Processor State Information
  - 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 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

Process Control Block

- Process Control Information
  - 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.

Process Control Block

- Process Control Information
  - 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.

Process Control Block

- Process Control Information
  - 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.

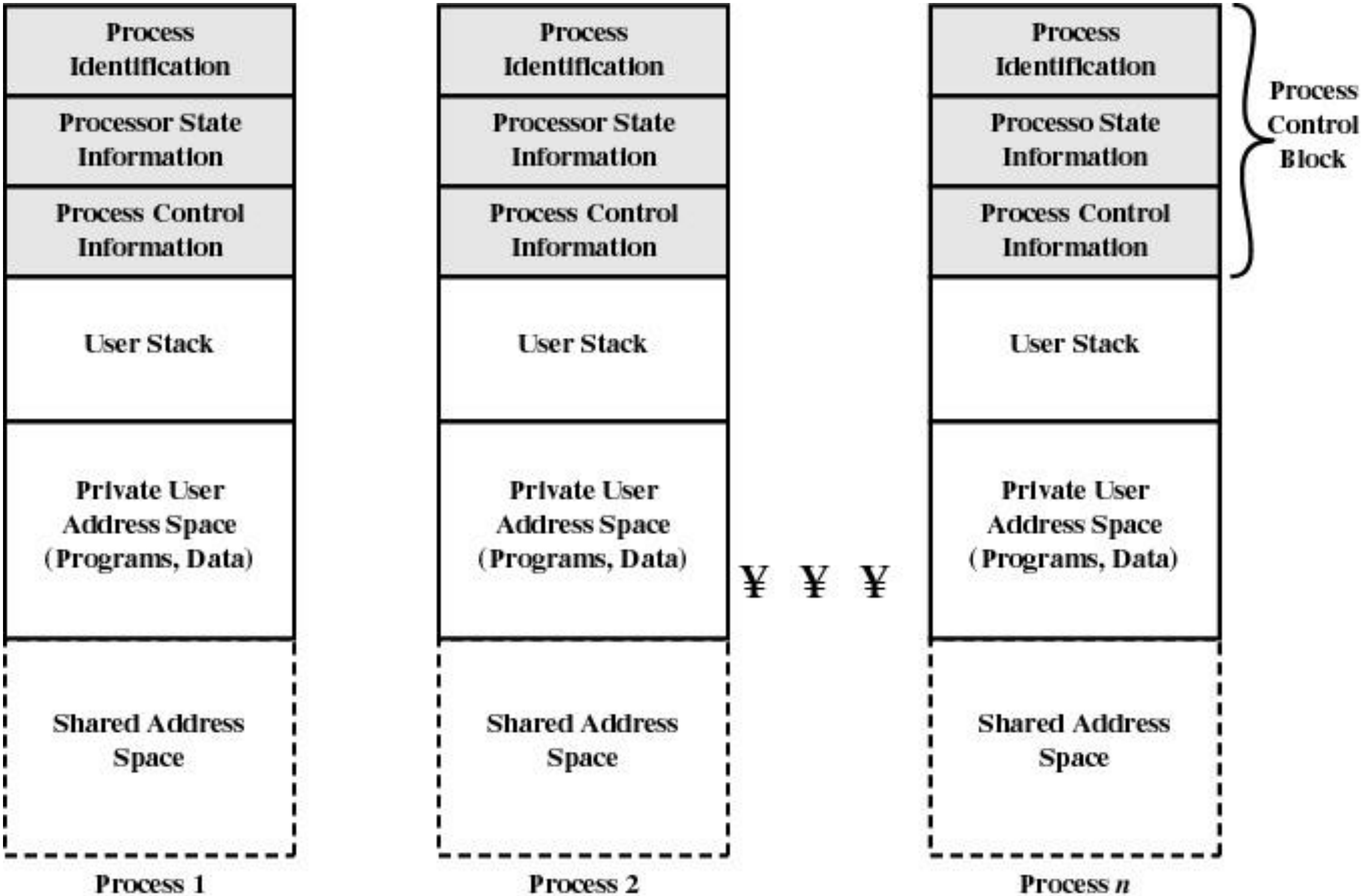
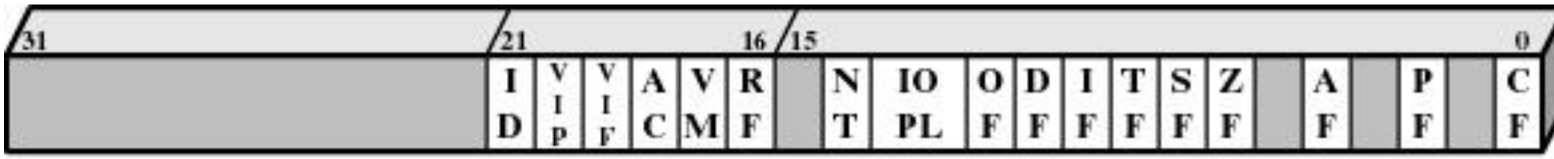


Figure 3.12 User Processes in Virtual Memory

Processor State Information

- Contents of processor registers
  - User-visible registers
  - Control and status registers
  - Stack pointers
- Program status word (PSW)
  - contains status information
  - Example: the EFLAGS register on Pentium machines

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 3.11 Pentium II EFLAGS Register

Modes of Execution

- 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

Process Creation

- Assign a unique process identifier
- Allocate space for the process
- Initialize process control block
- Set up appropriate linkages
  - Ex: add new process to linked list used for scheduling queue
- Create of expand other data structures
  - Ex: maintain an accounting file

When to Switch a Process

- Clock interrupt
  - process has executed for the maximum allowable time slice
- I/O interrupt
- Memory fault
  - memory address is in virtual memory so it must be brought into main memory

When to Switch a Process

- Trap
  - error occurred
  - may cause process to be moved to Exit state
- Supervisor call
  - such as file open

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 running
- Move process control block to appropriate queue - ready, blocked
- Select another process for execution

Change of Process State

- Update the process control block of the process selected
- Update memory-management data structures
- Restore context of the selected process

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
  - operating system software within context of a user process
  - process executes in privileged mode when executing operating system code

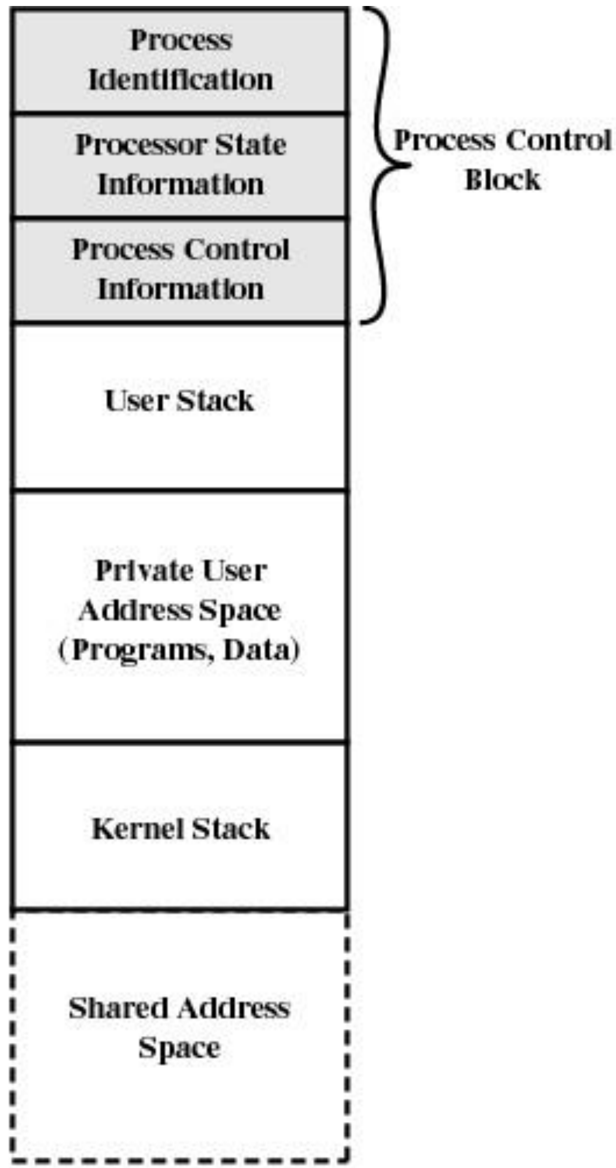


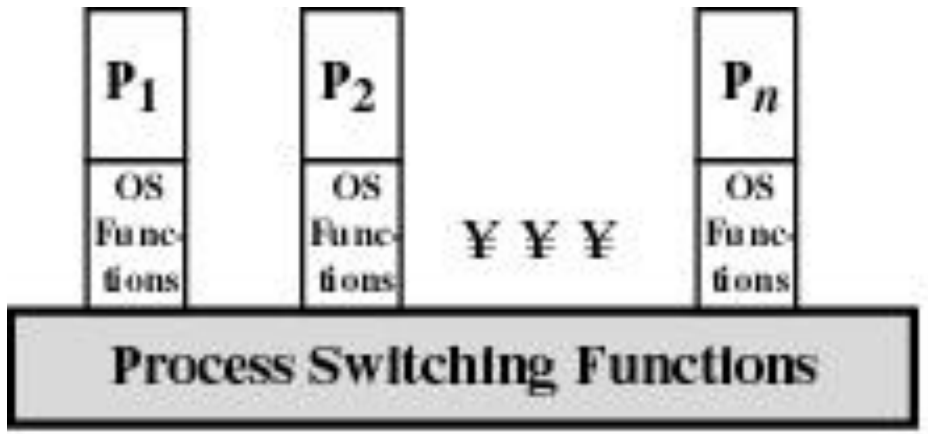
Figure 3.15 Process Image: Operating System Executes Within User Space

Execution of the Operating System

- Process-Based Operating System
  - major kernel functions are separate processes
  - Useful in multi-processor or multi-computer environment

UNIX SVR4 Process Management

- Most of the operating system executes within the environment of a user process



(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.

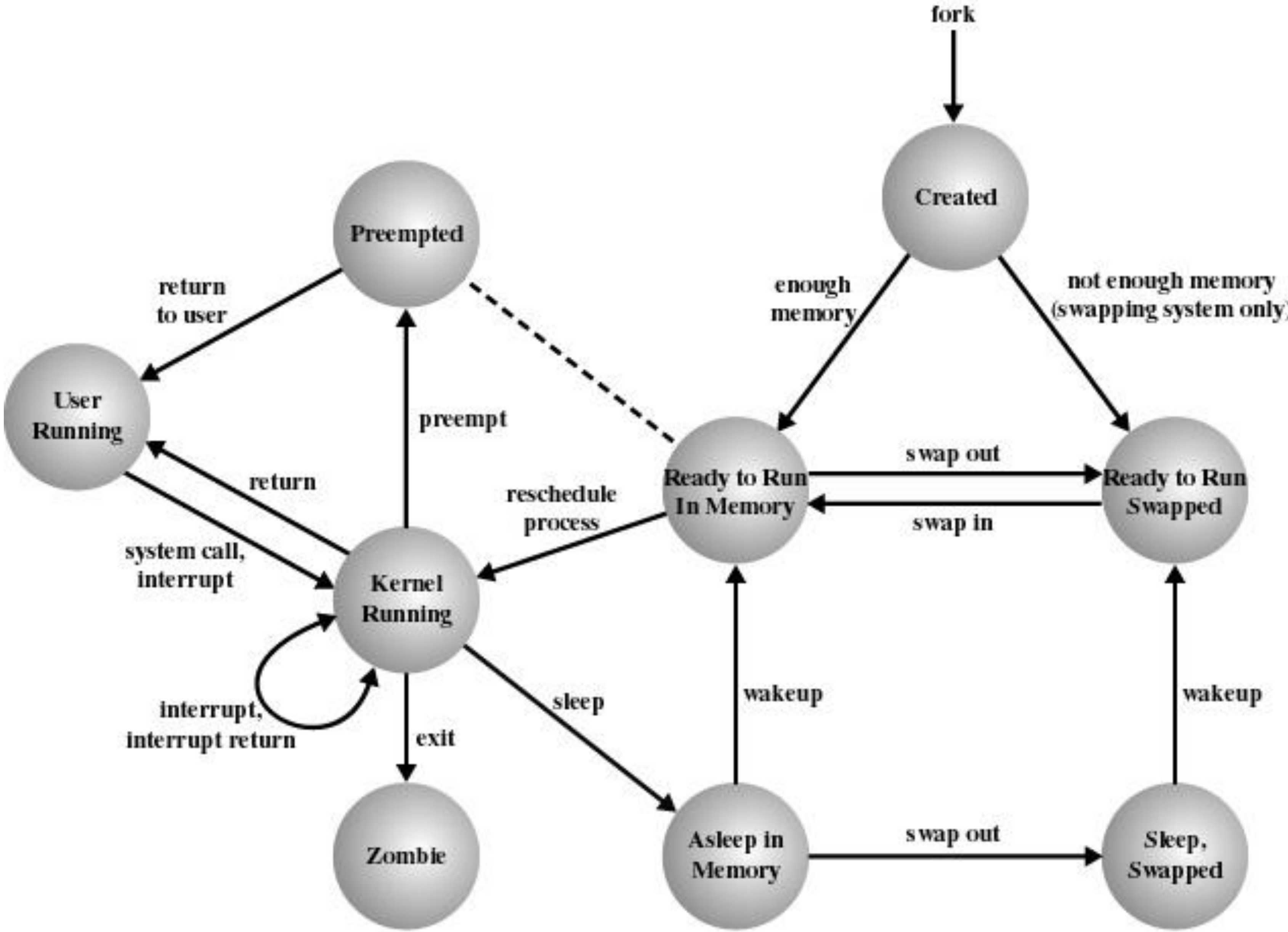


Figure 3.16 UNIX Process State Transition Diagram