Process Description and Control

summary

- basic concepts
  - process control block
  - process trace
  - process dispatching
  - process states
- process description
- process control

Process

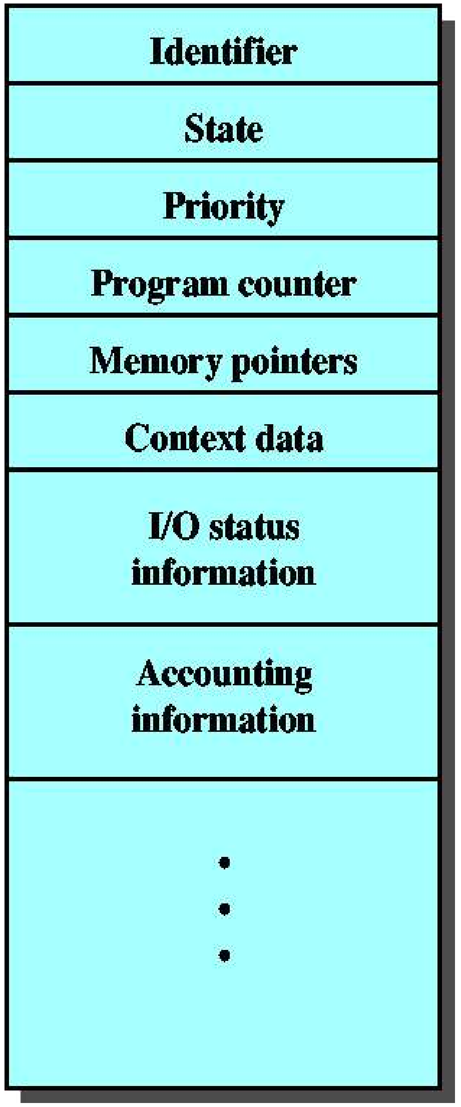
- A program in execution (running) on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by
  - a at least one sequential thread
  - an associated set of system resources
  - a current state of CPU (and other resources)

Operating System and Processes

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

Process Control Block (PCB)

- contains data about **one** process
  - one instance for each process
- contains all the information we need to...
  - ...interrupt a running process
  - ...resume execution
- created and managed by the operating system
- allows support for multiple processes



Process Elements in PCB

they largely depend on the OS

- Process Identifier (PID)
- State
  - running, blocked, etc.
- Priority (for the scheduler)
- Program counter (saved from CPU)
- Memory pointers
  - program, data, stack, etc.
- Context data (CPU registers)
- I/O status information
  - files, outstanding I/O requests, etc
- Accounting information
  - CPU time used, limits, etc.

PCB synonyms

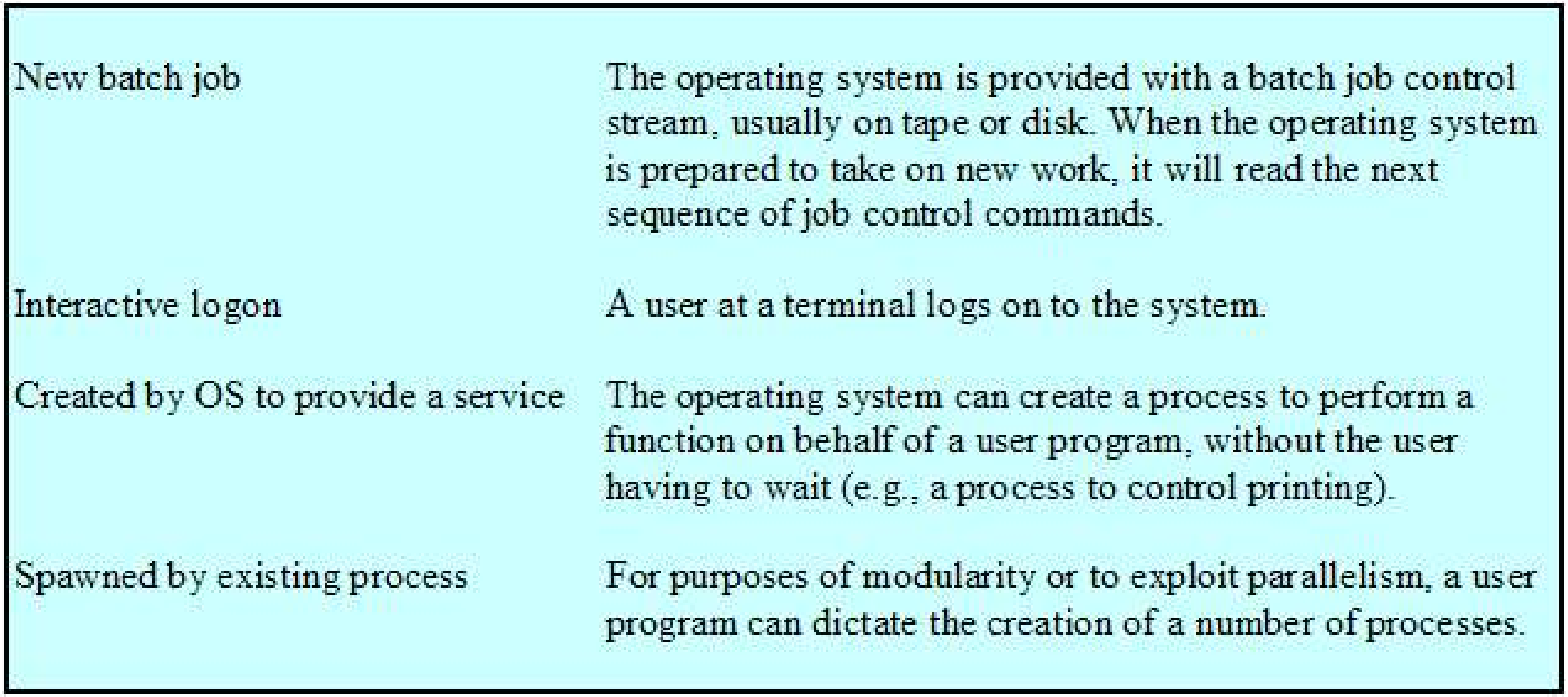
- process descriptor
- task control block
- task descriptor

linux

- task_struct

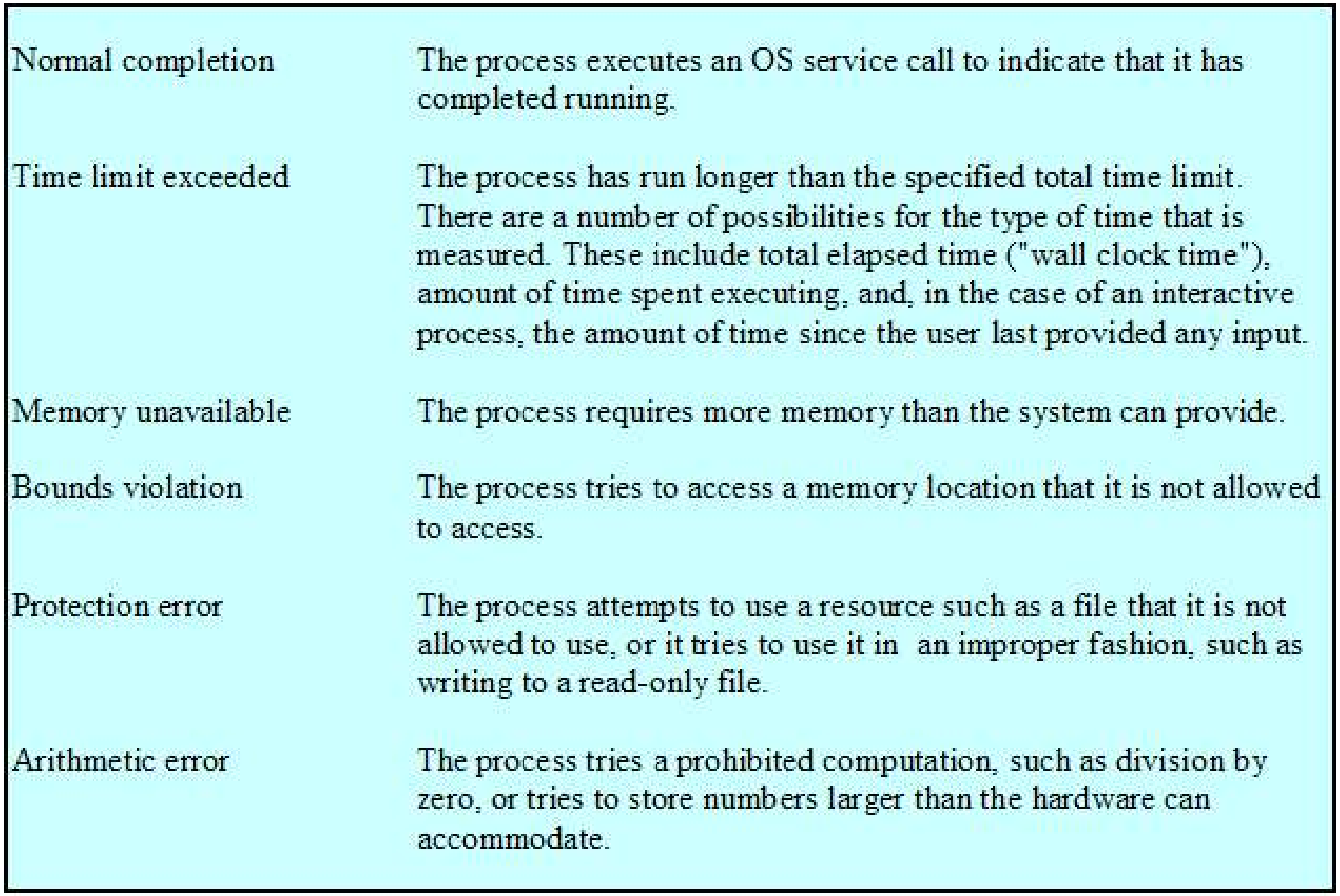
Process Creation

Table 3.1 Reasons for Process Creation



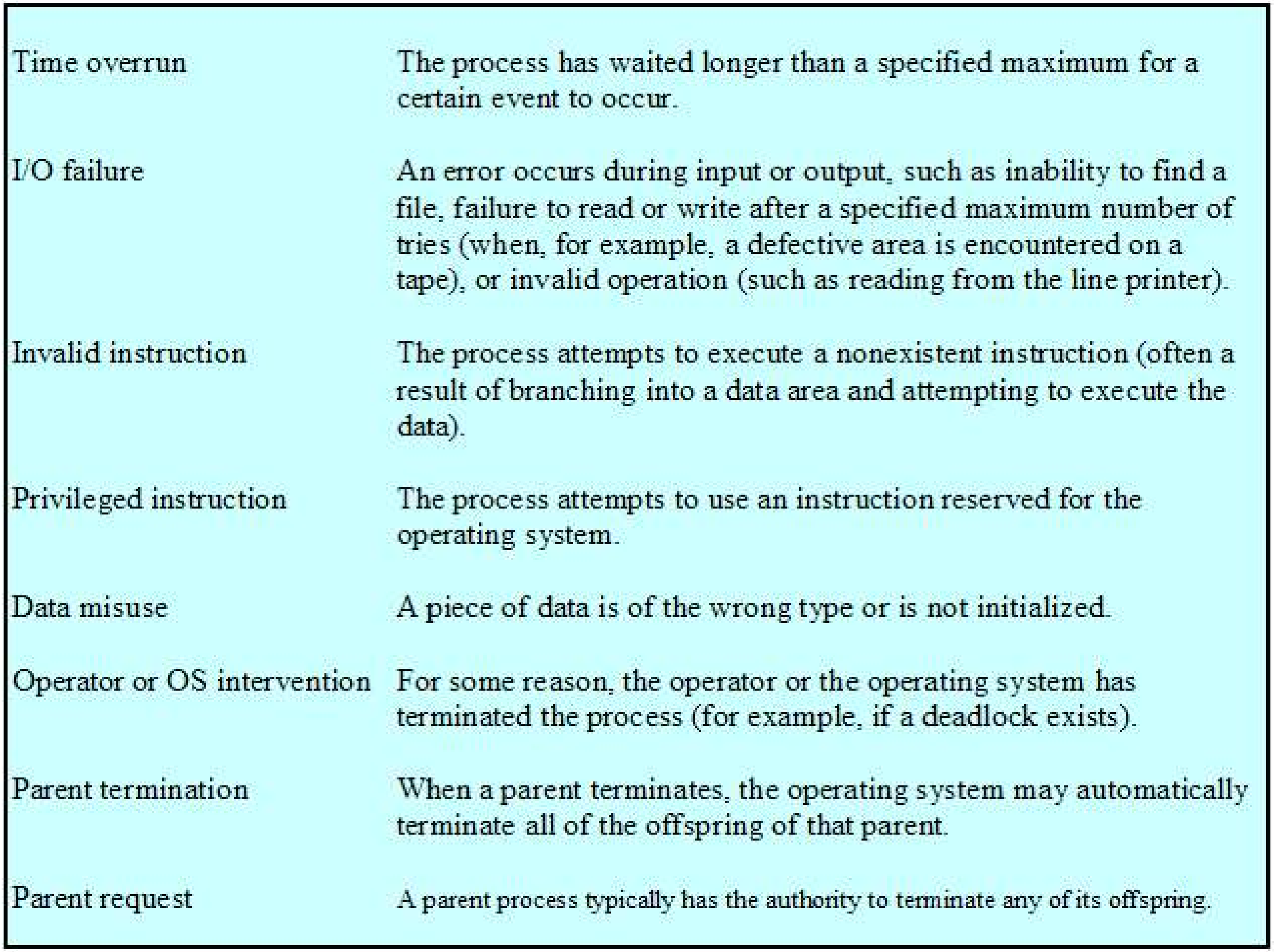
New batch job	The operating system is provided with a batch job control stream, usually on tape or disk. When the operating system is prepared to take on new work, it will read the next sequence of job control commands.
Interactive logon	A user at a terminal logs on to the system.
Created by OS to provide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



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.

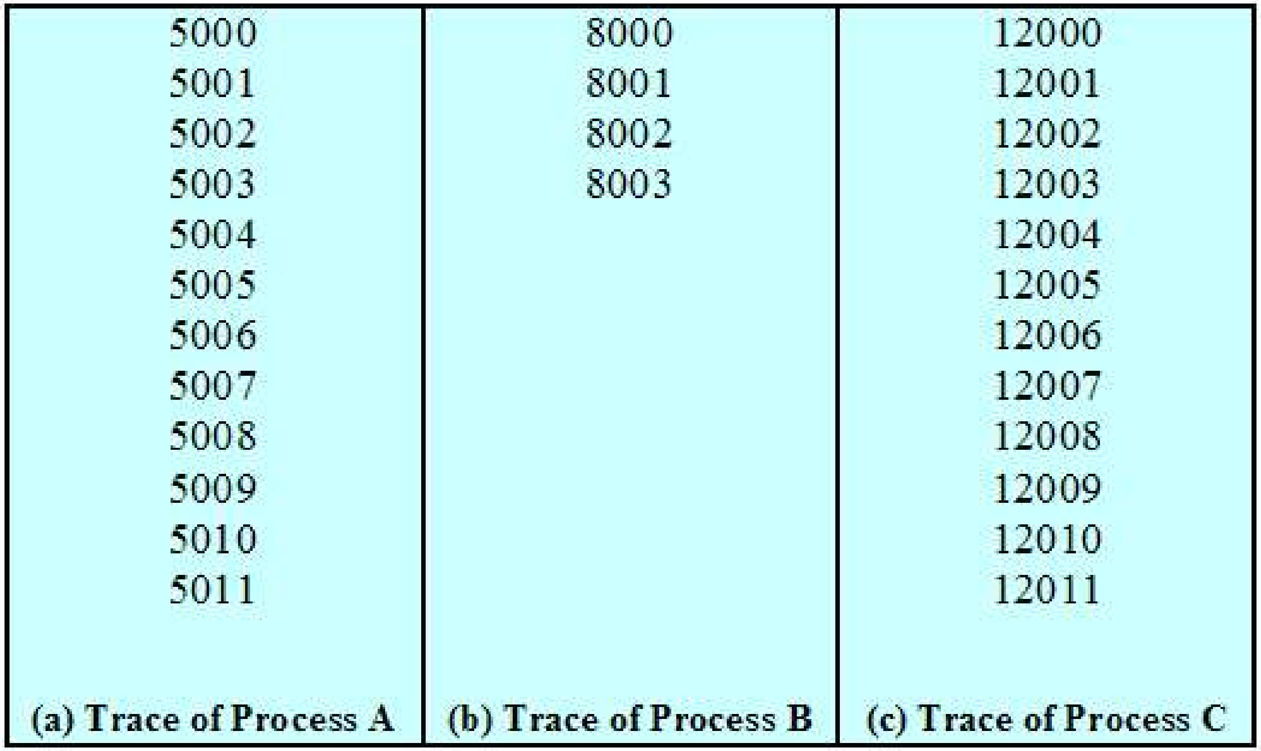
Process Termination



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.

Trace of Process

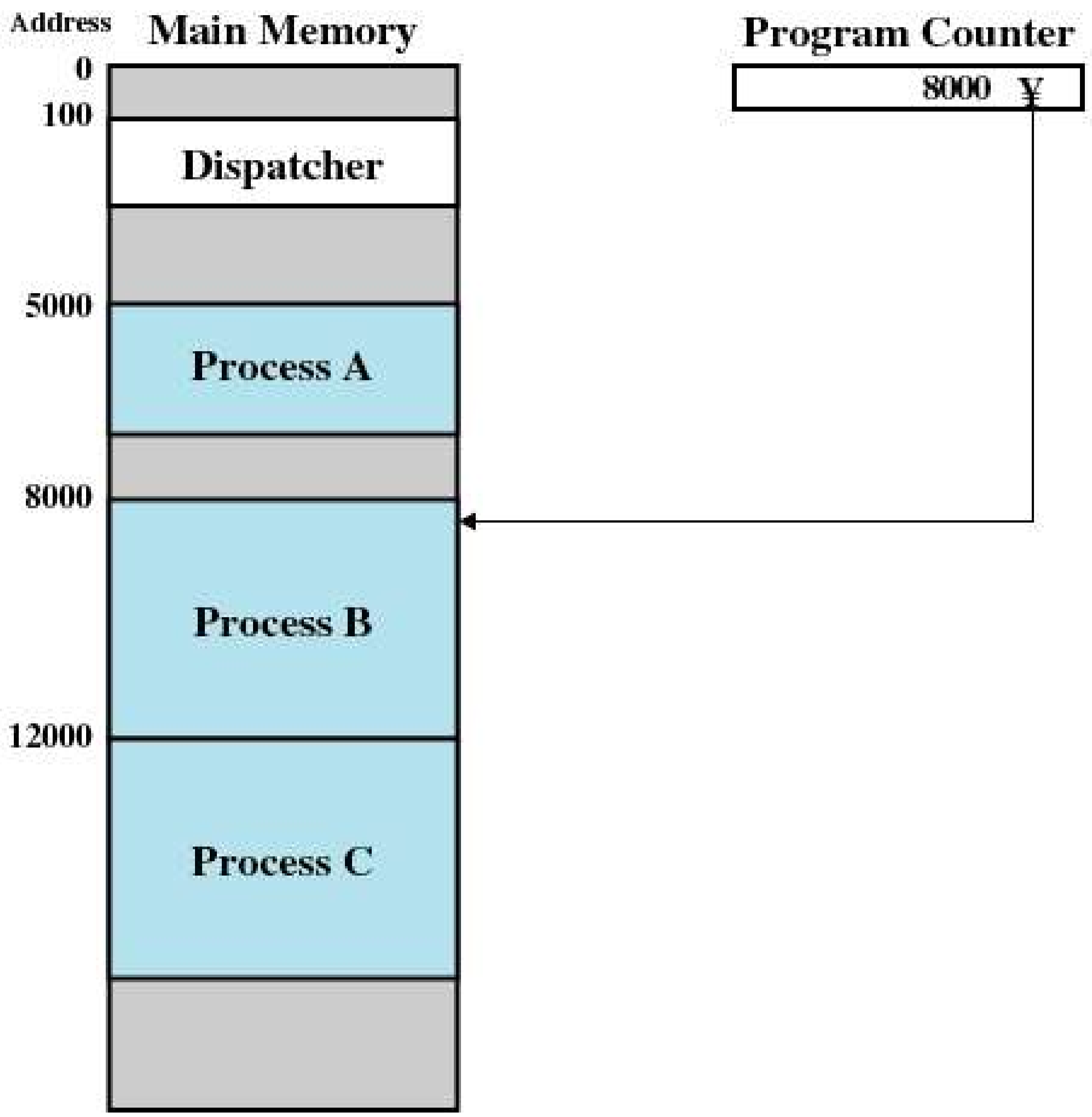
- Sequence of instruction (addresses) for each process



(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

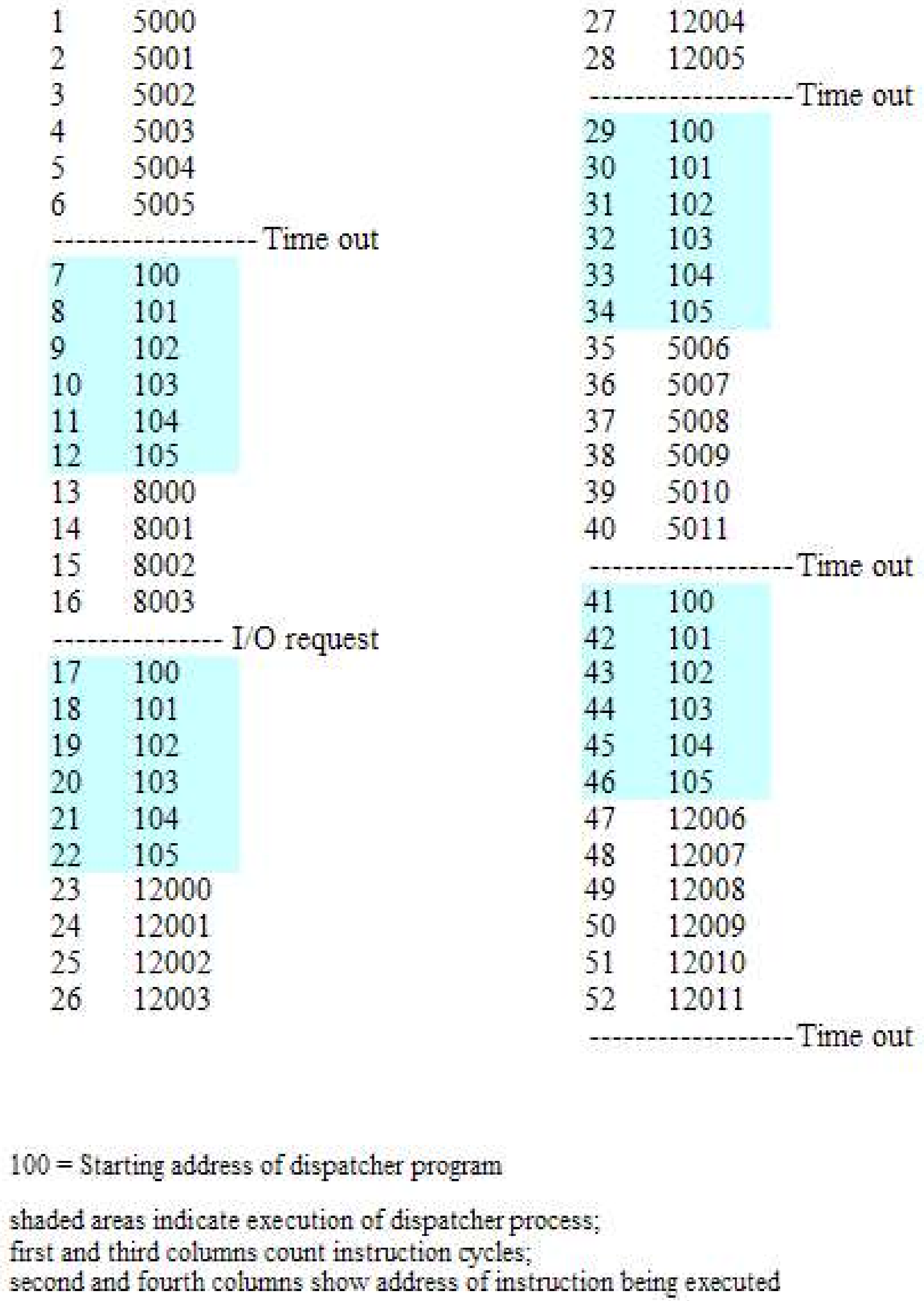
5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Processes and Memory



Dispatcher

- The *dispatcher* switches the processor from one process to another (**process switch**)

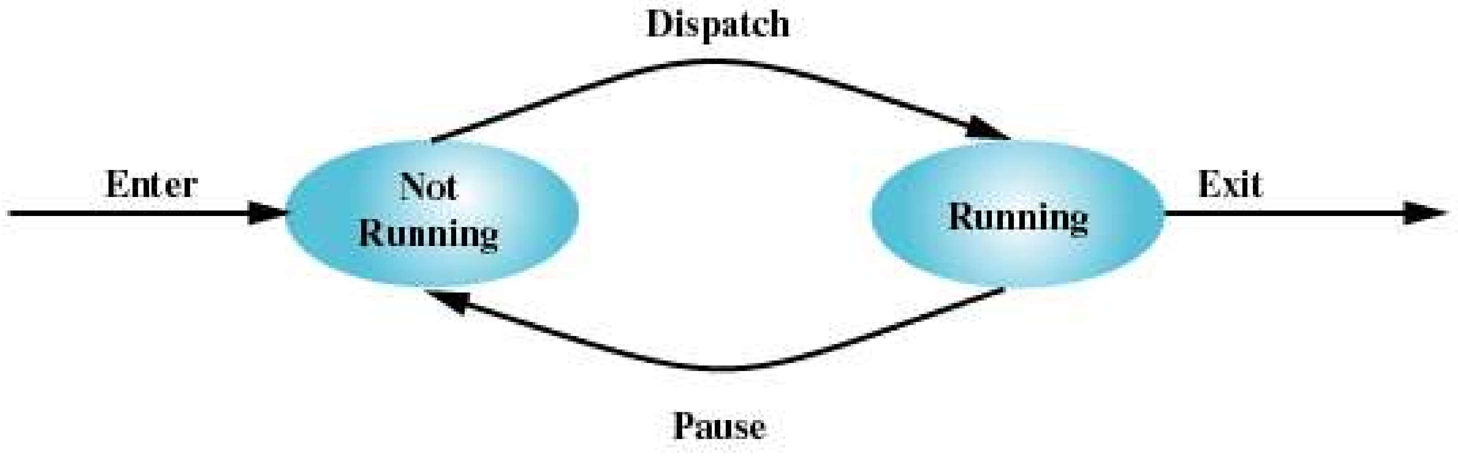


1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
------------------ Time out	
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
--------------- I/O request	
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003
27	12004
28	12005
------------------ Time out	
29	100
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	

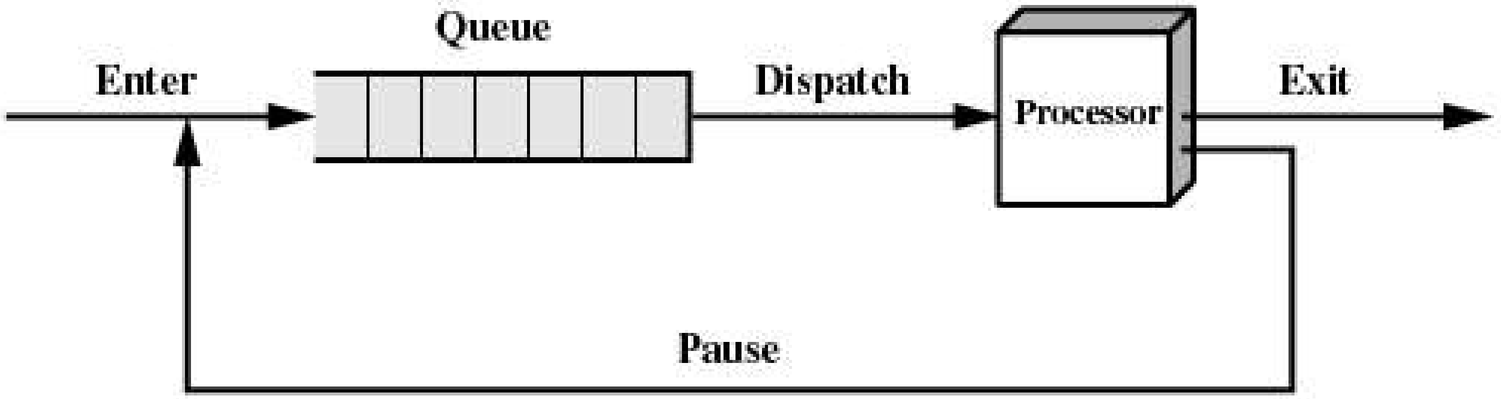
100 = Starting address of dispatcher program

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

Two-State Process Model



(a) State transition diagram



(b) Queuing diagram

Five-State Process Model

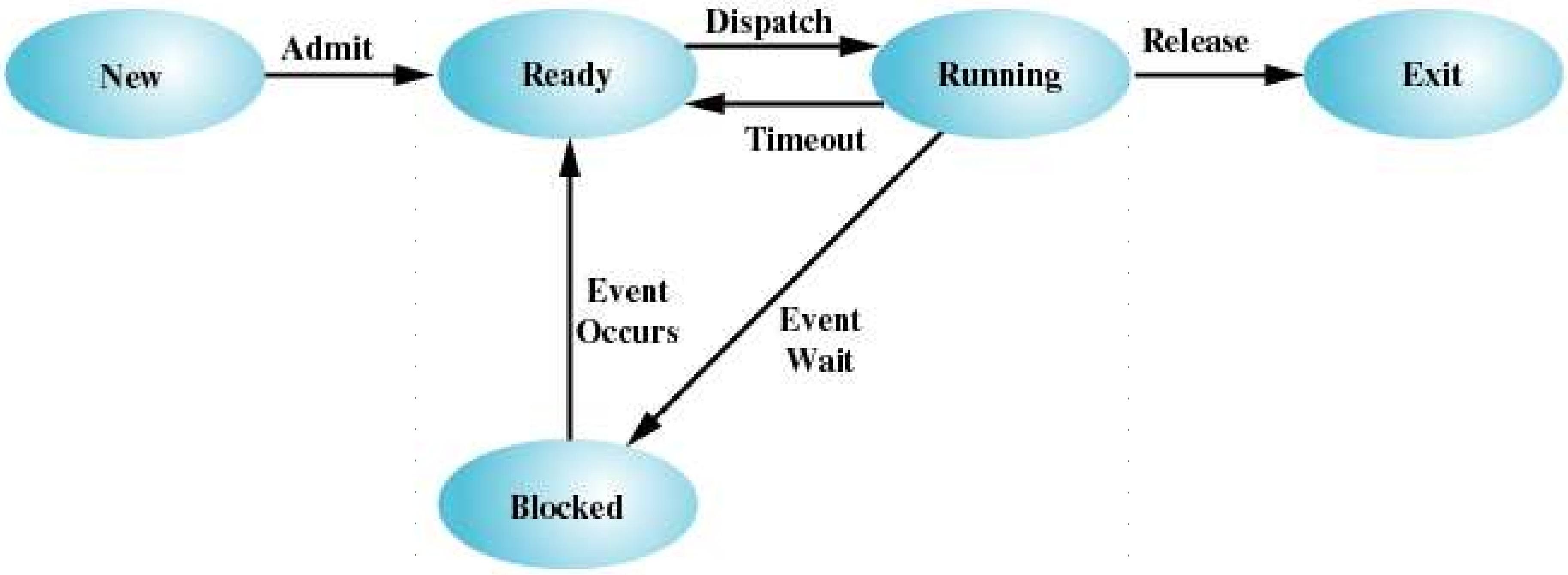


Figure 3.6 Five-State Process Model

Process States

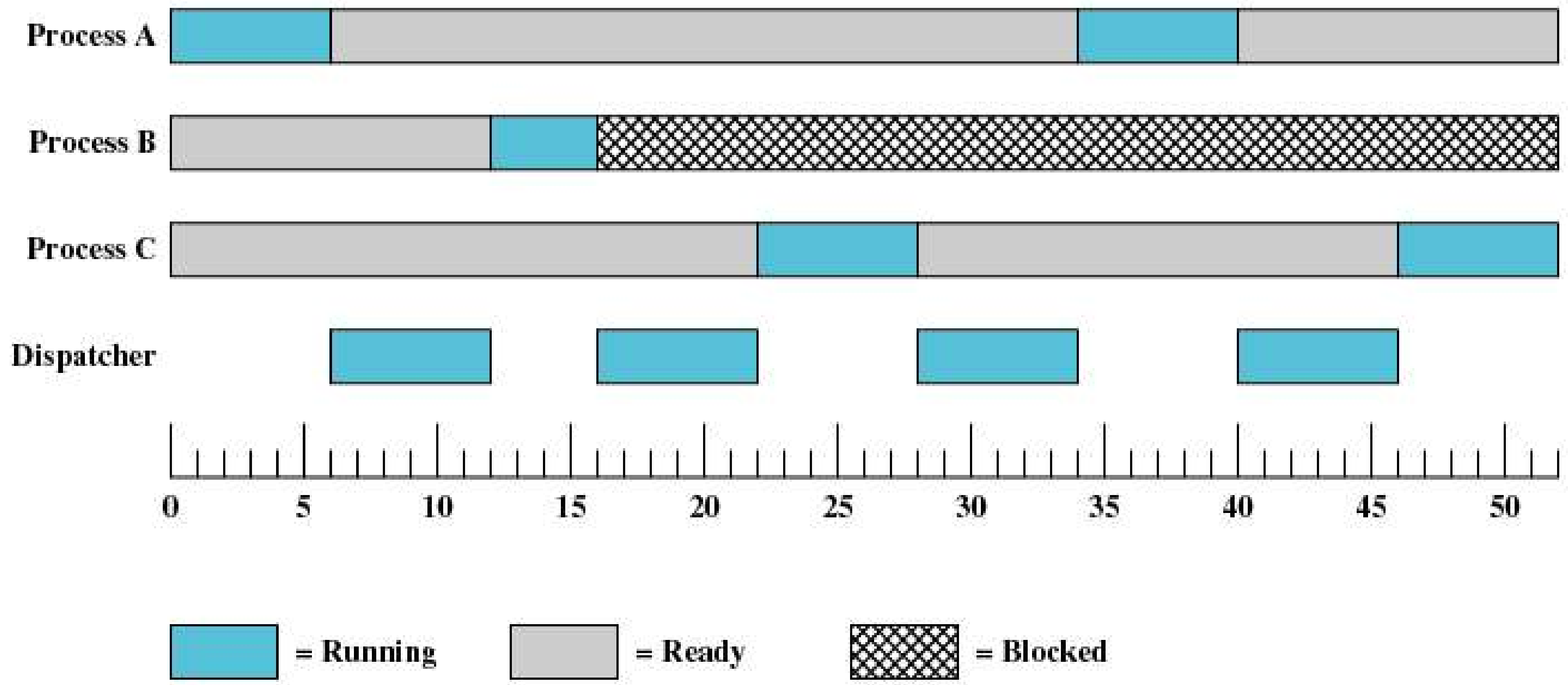
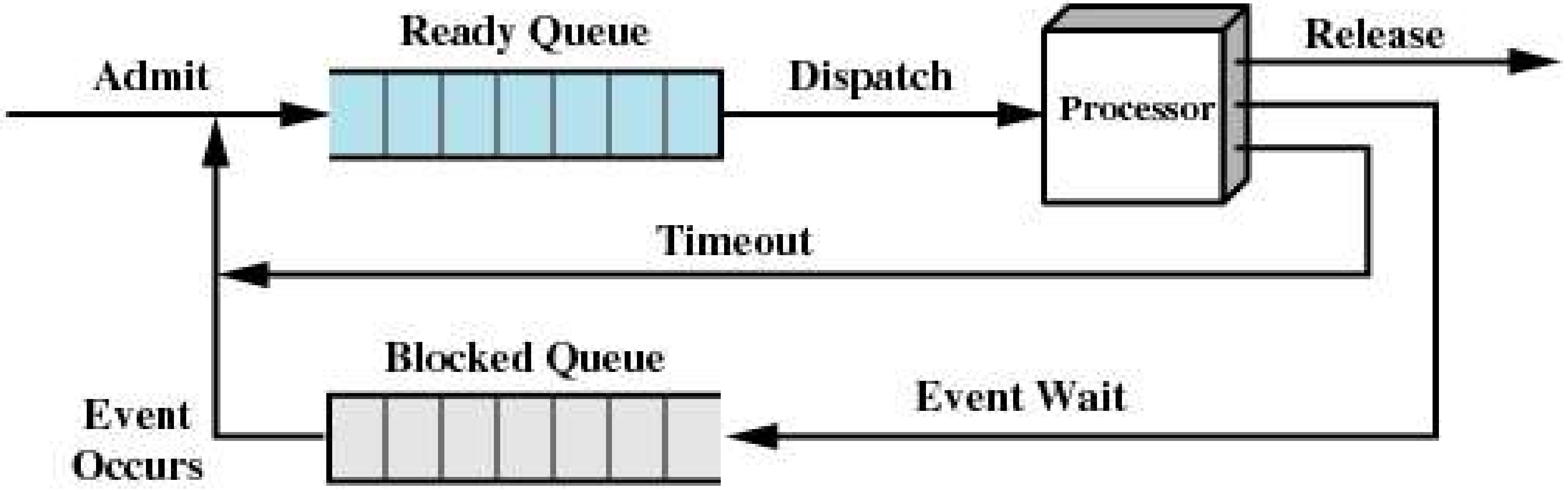
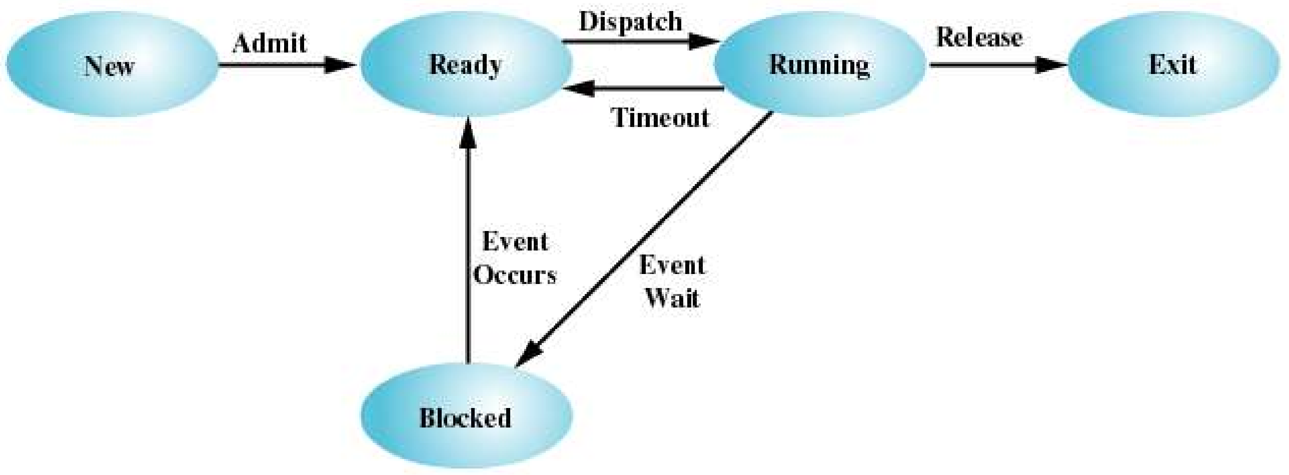


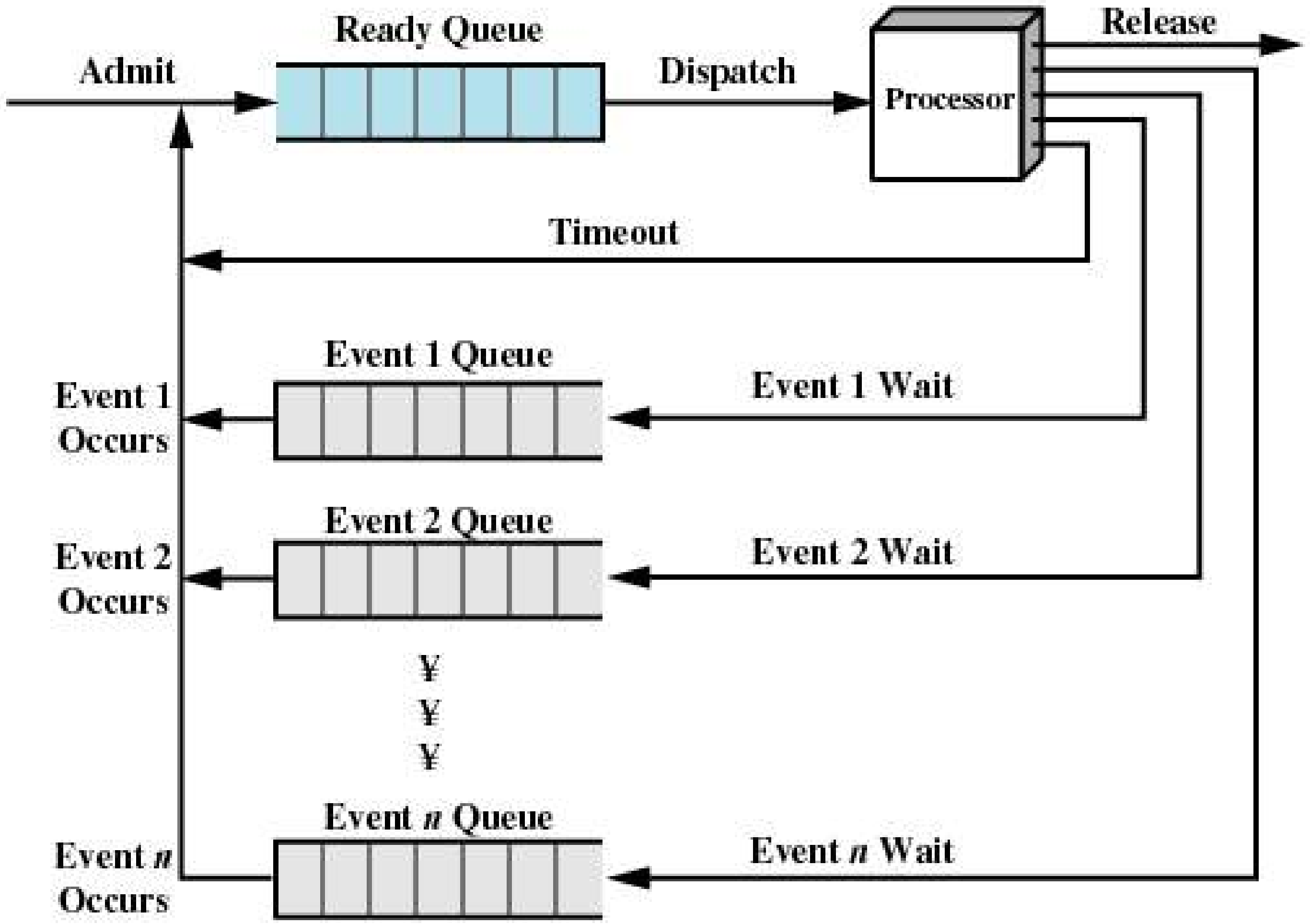
Figure 3.7 Process States for Trace of Figure 3.4

One sequential I/O device





Many sequential I/O devices

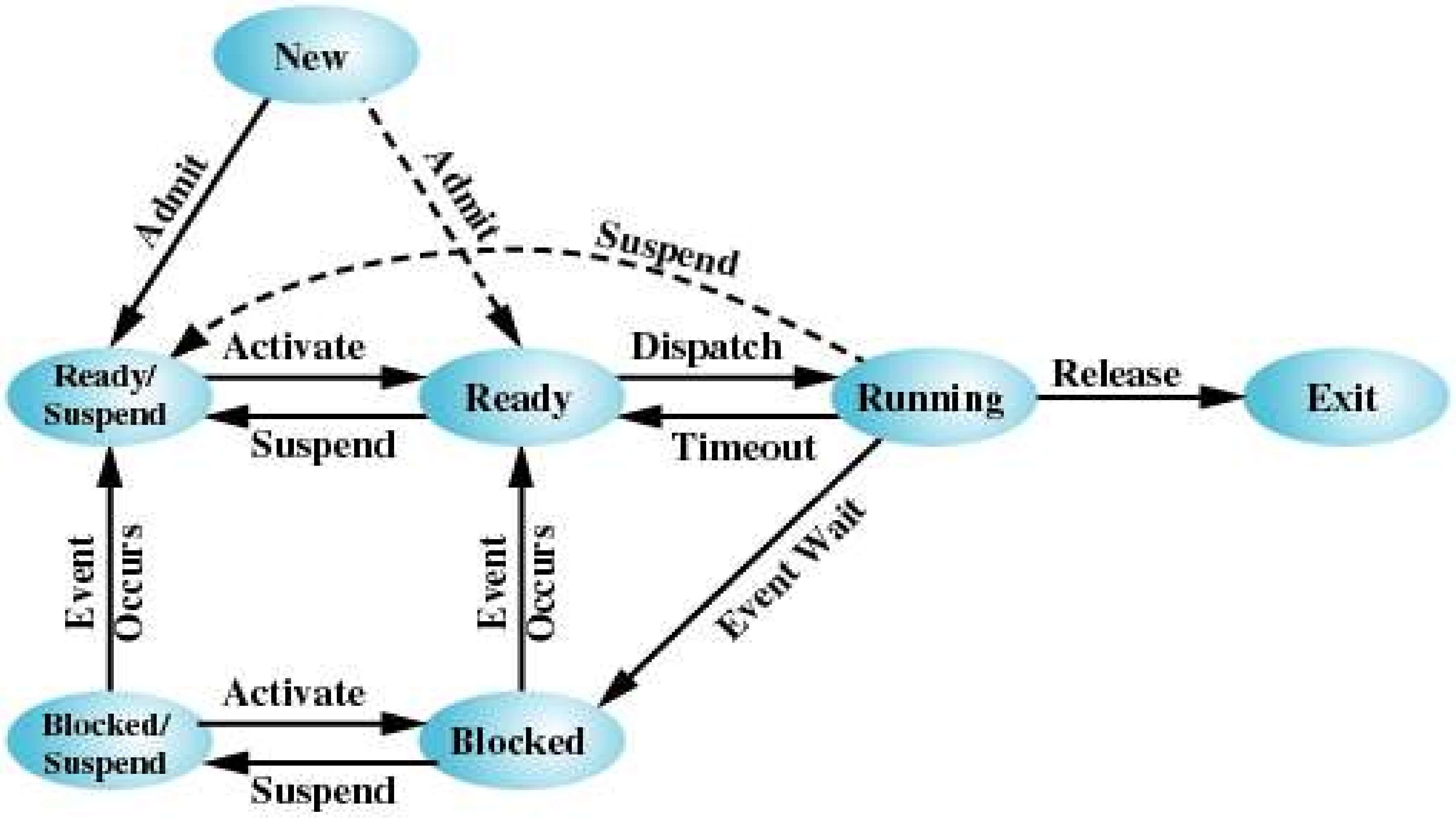


(b) Multiple blocked queues

Suspended Processes

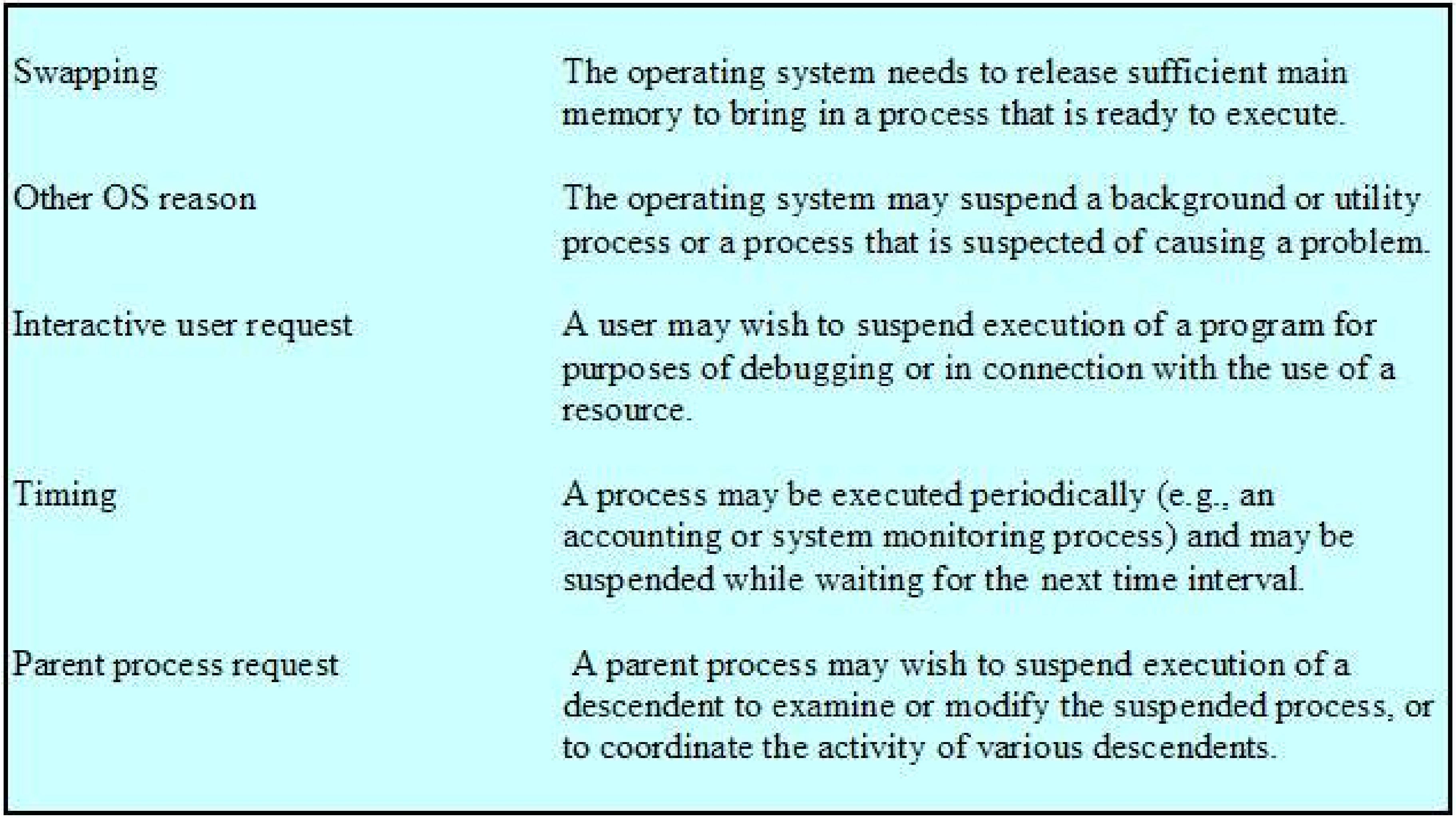
- Processor is faster than I/O so many processes could be waiting for I/O
- Swap these processes to disk to free up memory
- Blocked state becomes suspend state when swapped to disk
- Two new states
  - Blocked/Suspend
  - Ready/Suspend

Two New States



(b) With Two Suspend States

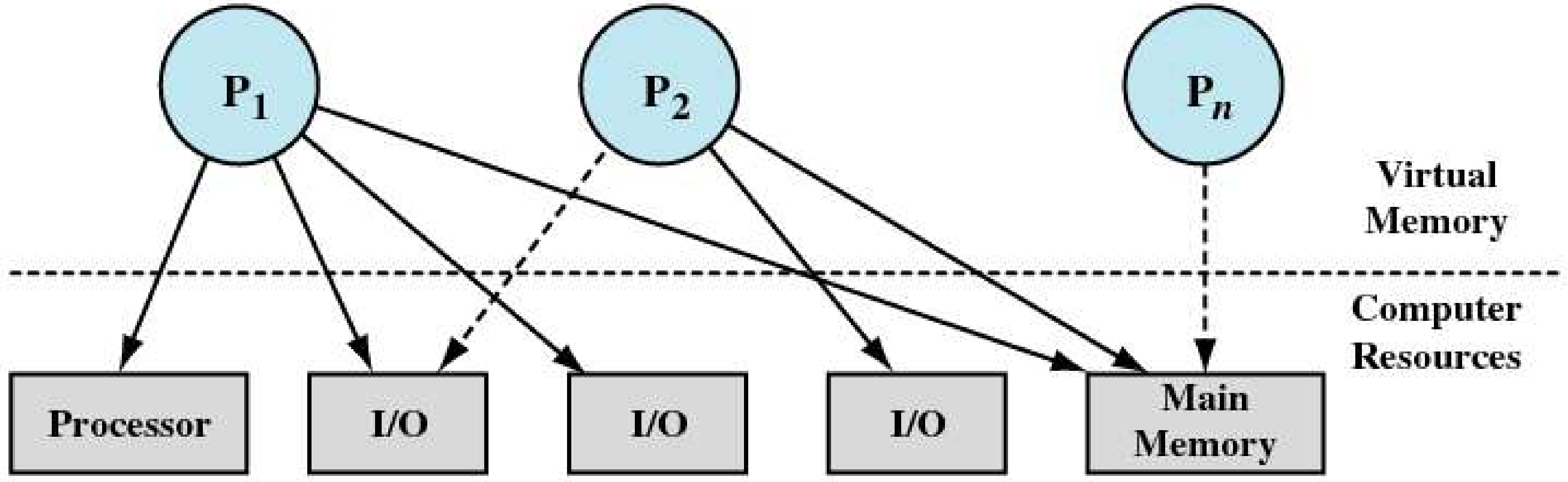
Several Reasons for Process Suspension



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.

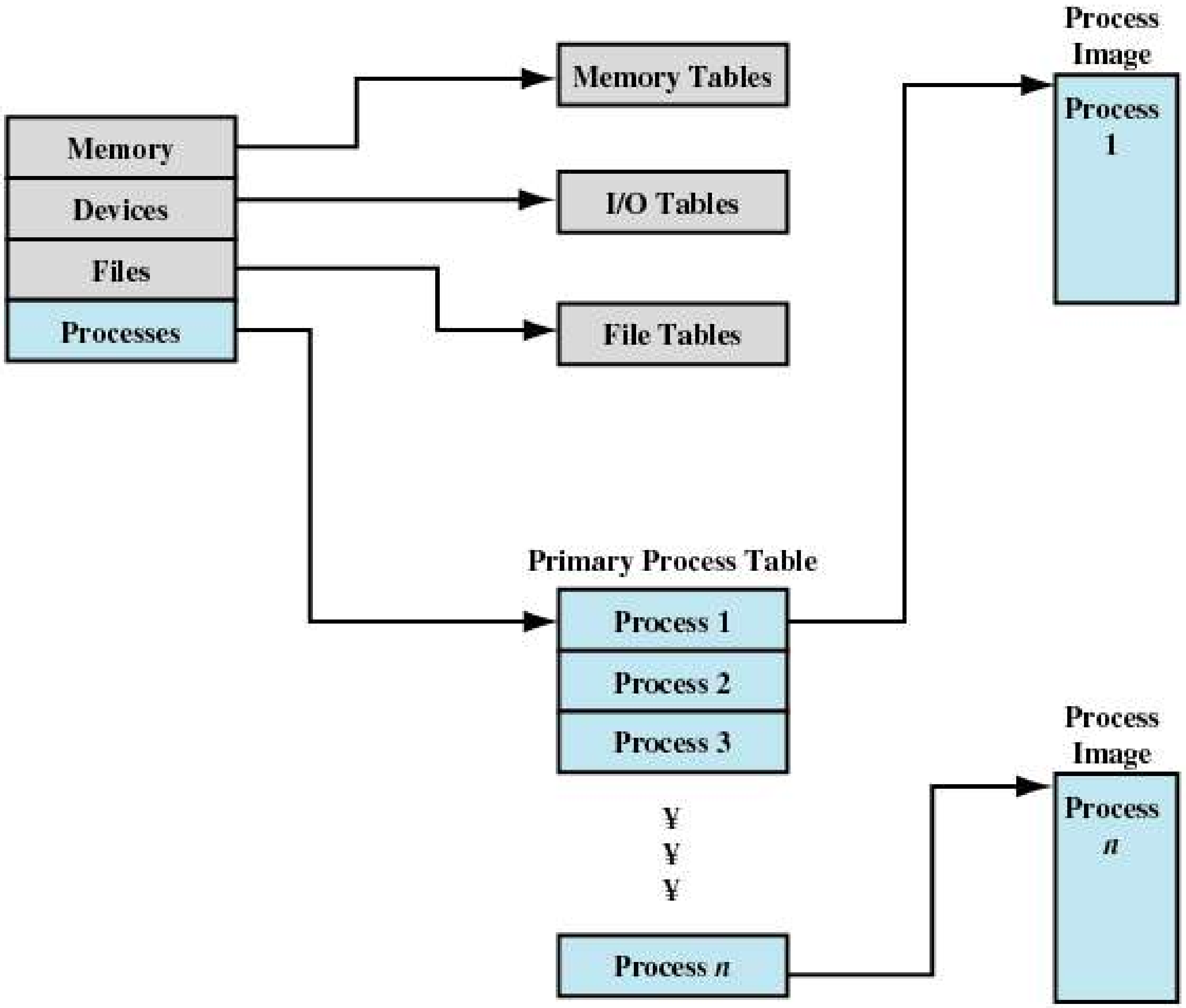
process description

OS controls assignment of resources to processes



- the OS maintains many data structures for this purpose

example of OS data structure



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

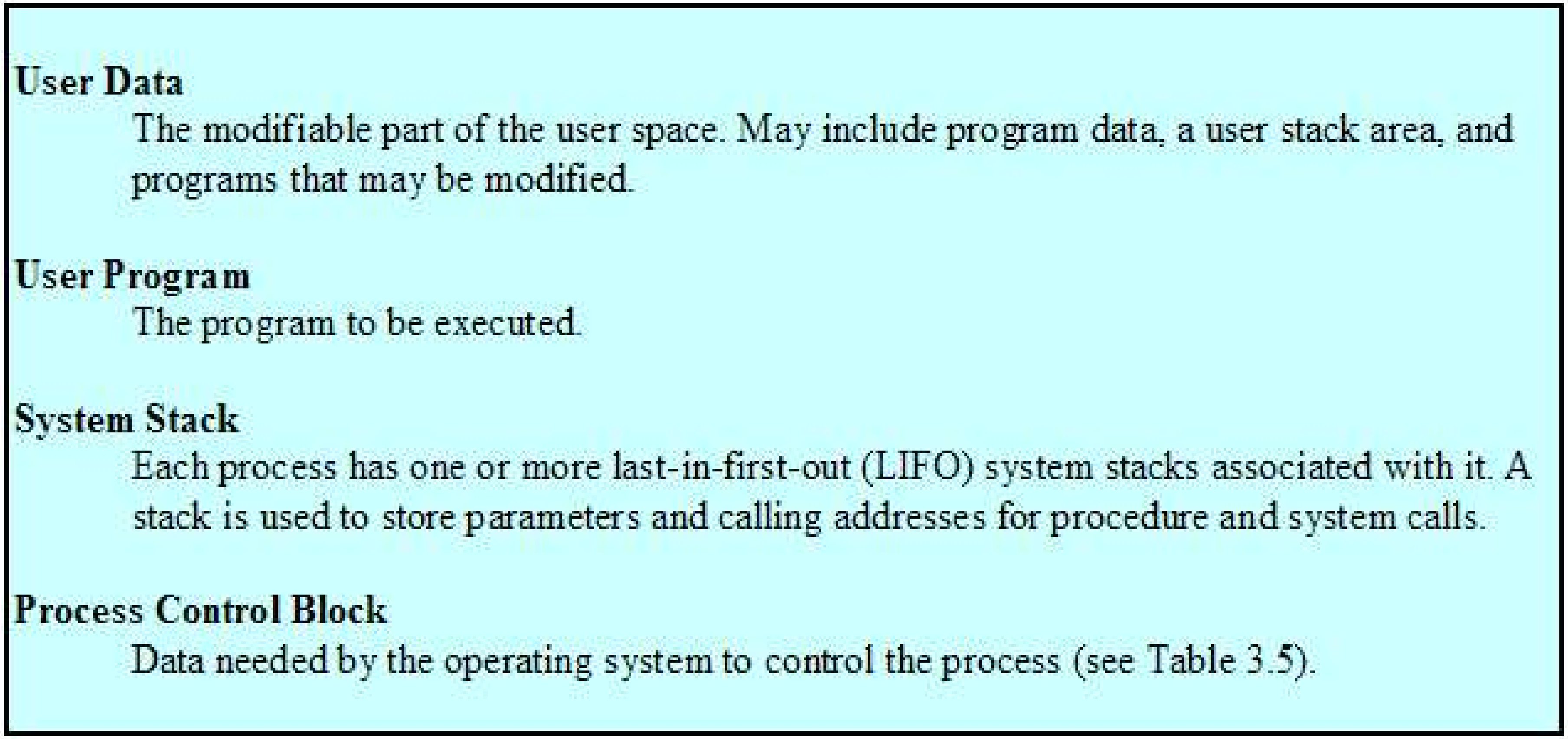
- Existence of files
- Location on secondary memory
- Current Status
- Attributes
- Sometimes this information is maintained by a file management system

Process Table

- one entry for each process
- contains a minimal amount of information needed to activate the process
  - a “pointer” to the image

Process Image

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.

System Stack

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.

Process Control Block

Data needed by the operating system to control the process (see Table 3.5).

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 (context)
  - 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.

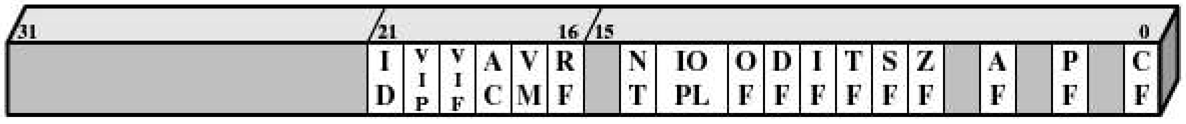
Process Control Block

- Processor State Information (context)
  - 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

  - Example: the EFLAGS register on Pentium machines



ID	=	Identification flag	DF	=	Direction flag
VIP	=	Virtual interrupt pending	IF	=	Interrupt enable flag
VIF	=	Virtual interrupt flag	TF	=	Trap flag
AC	=	Alignment check	SF	=	Sign flag
VM	=	Virtual 8086 mode	ZF	=	Zero flag
RF	=	Resume flag	AF	=	Auxiliary carry flag
NT	=	Nested task flag	PF	=	Parity flag
IOPL	=	I/O privilege level	CF	=	Carry flag
OF	=	Overflow flag			

Process Control Block

- Processor State Information (context)
  - 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

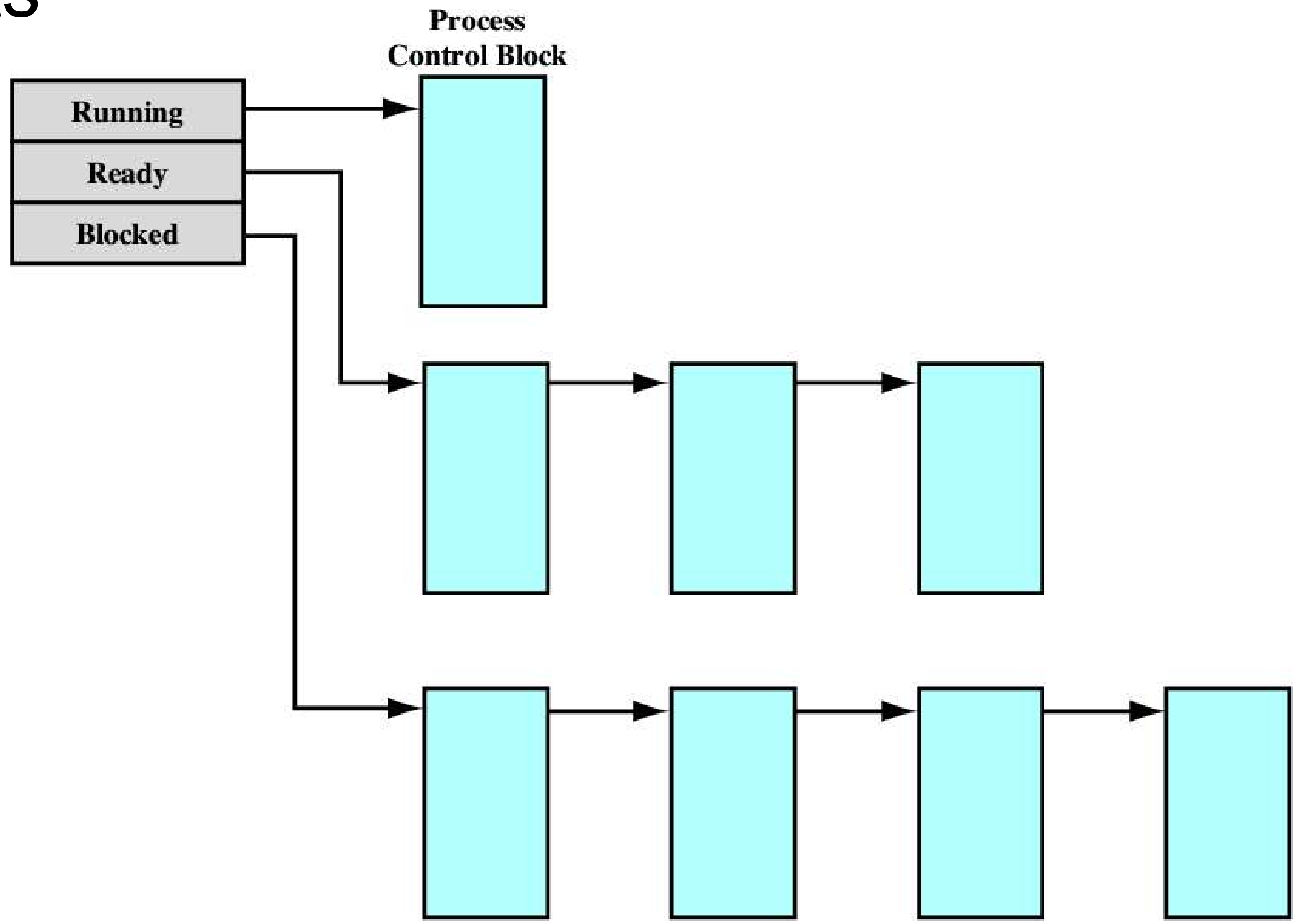
- Process Control Information
  - Scheduling and State Information
    - This is information that is needed by the operating system to perform its scheduling function. Typical items of information:
    - *Process state:* defines the readiness of the process to be scheduled for execution (e.g., running, 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.

Data Structuring

- example: queues can be realized as linked lists



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.

Process Creation

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

process control

mode switch

- two cases
  - user-mode → kernel-mode
    - triggered by an interrupt, a trap or a system call
    - set cpu in priviledged mode
    - may save the cpu state
  - kernel-mode → user-mode
    - may restore all or part of the cpu state
    - set cpu in unpriviledged mode
    - when the kernel “decides” to resume process execution

mode switch

- it is not a process switch
- not necessarily implies a process switch
- many system calls require a mode switch

process switch

- a process switch assigns the cpu to a different process
  - before: P_1 running, P_2 ready
  - after: P_1 not running, P_2 running
- it is performed in kernel-mode
  - it requires two mode switches
    1. user-mode → kernel-mode before the process switch
       - triggered by interrupt, trap or system call
       - kernel possibly fulfill a request (e.g. I/O)
    2. kernel-mode → user-mode after the process switch
       - into the process chosen by the kernel

process switch

- it modifies OS data structures
  - set proper state in PCB of P_1 and P_2
  - update queues
    - move P_1 into the appropriate queue
    - move P_2 out of the ready queue
- the next mode switch (kernel-mode → user-mode) will restore the cpu state of P_2

typical situation for switching mode and/or process

- Clock interrupt
  - process has executed for the maximum allowable time slice
  - always switch process
- system call
  - process switch when it is a blocking I/O request
  - if a process is in charge for fulfilling the request a process switch occours
  - OS may check if other processes have greater priority and possibly switch process

typical situation for switching mode and/or process

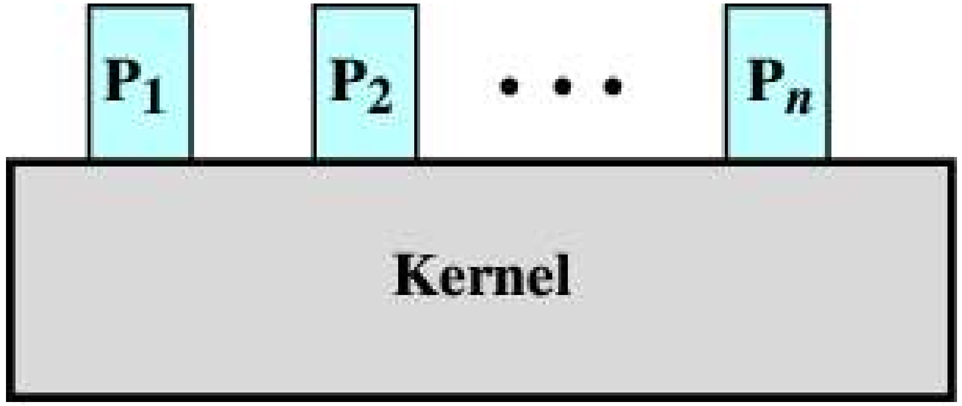
- I/O interrupt
  - a blocked process may become ready
  - process switch depends on OS policies and priorities
- Trap
  - error or exception
    - current process usually die and process is switched
  - memory fault (virtual memory)
    - current process becomes blocked (waiting for the page) and process is switched

execution of the OS

- the OS is executed by the cpu
- is the OS a process?
- several architectures are possible
  - non-process kernel
  - kernel execution within user processes
  - process-based operating system

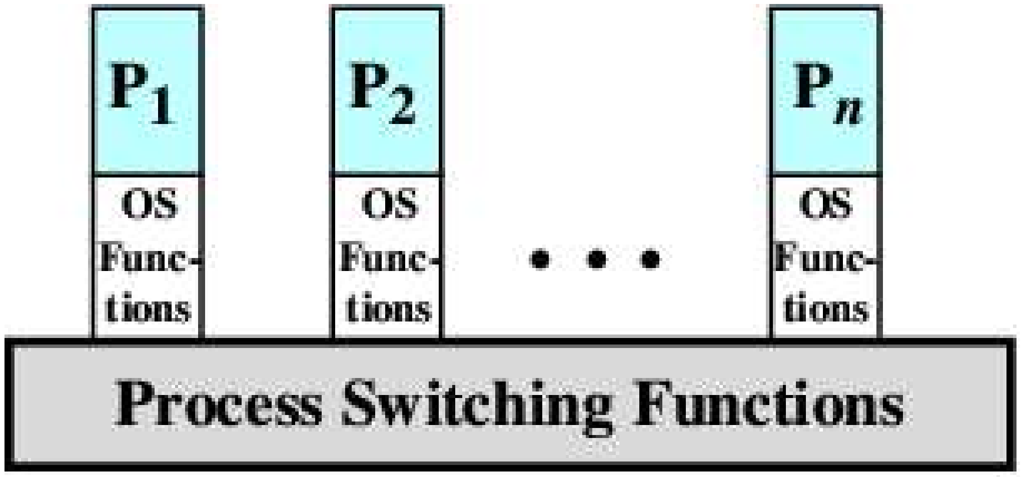
non-process kernel

- kernel is executed outside of any process
  - when it is executed cpu is in privileged mode
  - kernel implements tricks to access the images of processes
- the concept of process applies only to user programs



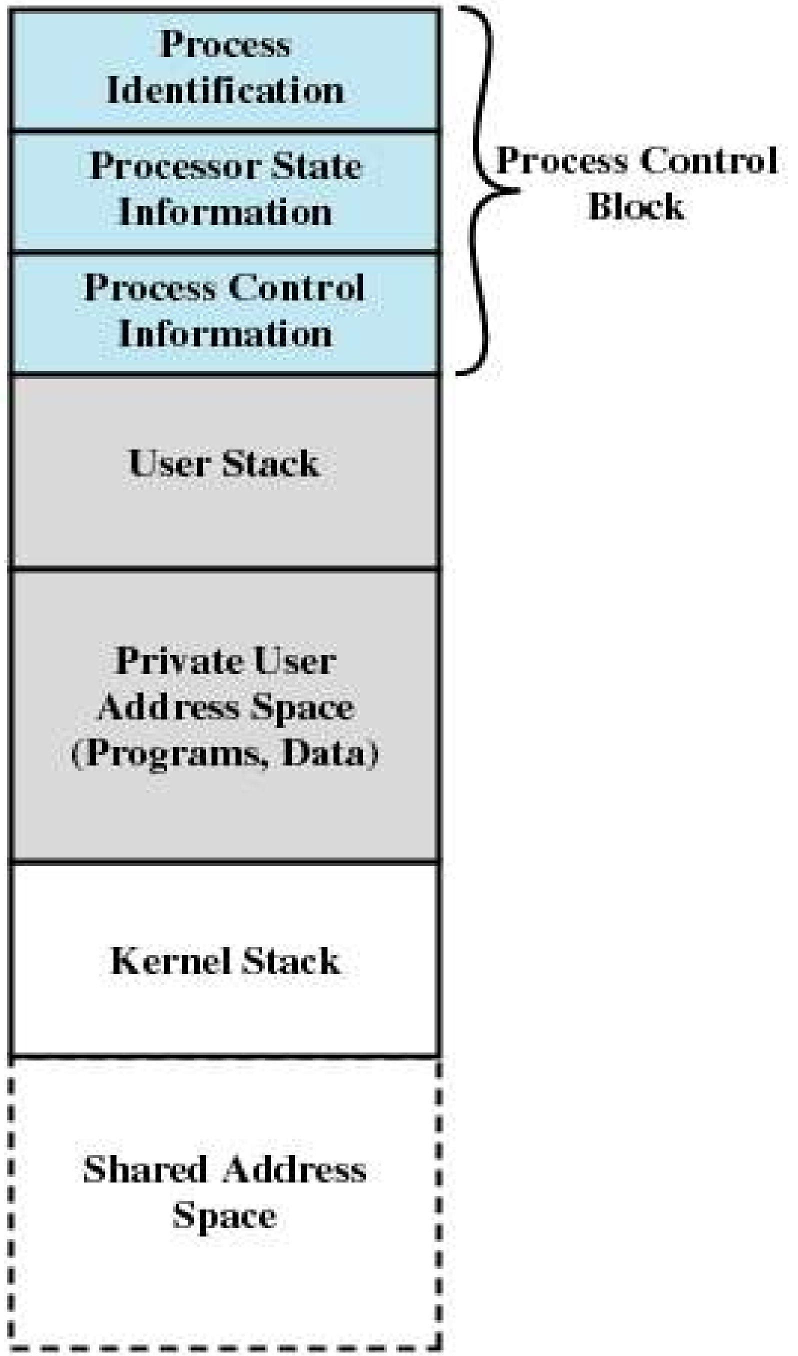
Execution Within User Processes

- part of the kernel executes within the context of a user process
- Process executes in privileged mode when executing operating system code



Execution Within User Processes

- each process has its own **image**
- image contains also
  - kernel stack
  - kernel program
  - kernel data
- kernel program and data are shared by all images

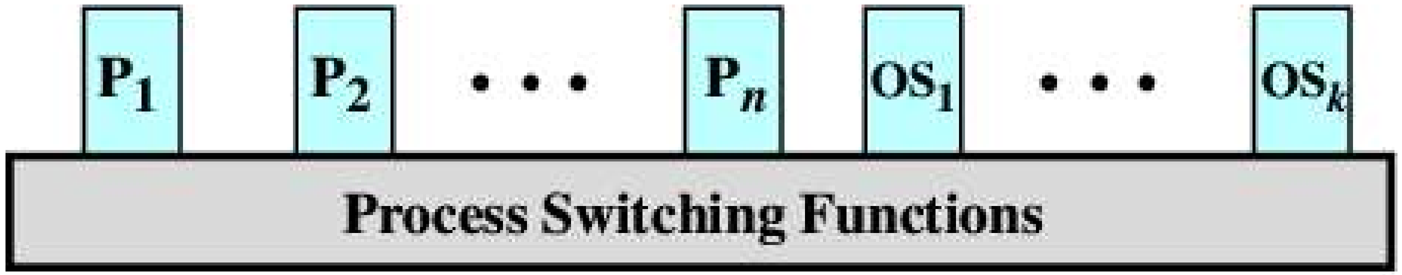

Process Identification
Processor State Information
Process Control Information
Process Control Block
User Stack
Private User Address Space (Programs, Data)
Kernel Stack
Shared Address Space

Execution Within User Processes

- to fulfill a system call, interrupt or trap
  - mode is switched
  - process is not switched
  - current usable image remain the same
  - both kernel data and current process data can be accessed
- a process switch occours if and only if a new process should be dispatched
  - process switch is the only activity that can be considerd outside of any process

process-based OS

- Implement the os as a collection of system processes
- each system call pays a process switch penalty
- modular and flexible
  – can take advantage of multi-processor or multi-computer environment
- process switch is the only activity that can be considerd outside of any process


P_1
P_2
• • •
P_n
OS_1
• • •
OS_k
Process Switching Functions