IT 540 Operating Systems ECE519 Advanced Operating Systems

Prof. Dr. Hasan Hüseyin BALIK

(3rd Week)

(Advanced) Operating Systems

3. Process Description and Control

3. Outline

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

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 Elements

- Two essential elements of a process are:

Program code

- which may be shared with other processes that are executing the same program

A set of data associated with that code

- when the processor begins to execute the program code, we refer to this executing entity as a ***process***

Process Elements

- While the program is executing, this process can be uniquely characterized by 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
- It is possible to interrupt a running process and later resume execution as if the interruption had not occurred
- Created and managed by the operating system
- Key tool that allows support for multiple processes

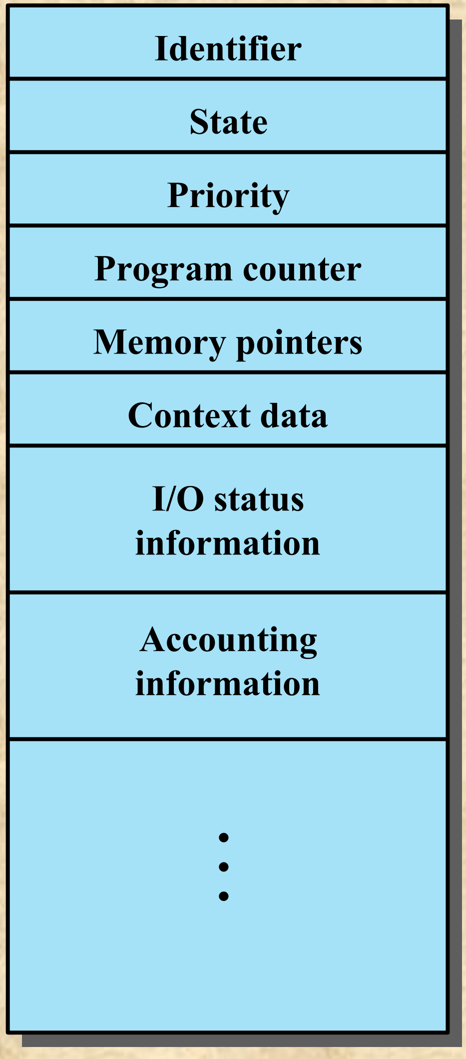


Figure 3.1 Simplified Process Control Block

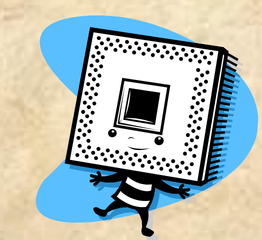
Process States

Trace

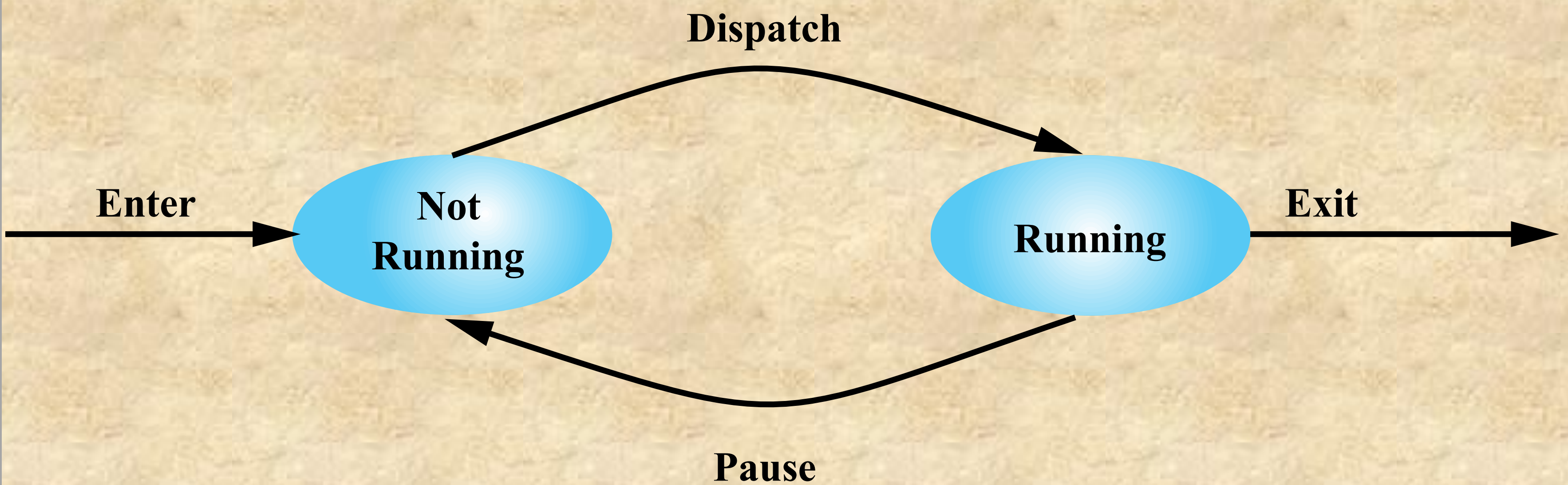
- the behavior of an individual process by listing the sequence of instructions that execute for that process
- the behavior of the processor can be characterized by showing how the traces of the various processes are interleaved

Dispatcher

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



Two-State Process Model



(a) State transition diagram

Process Execution

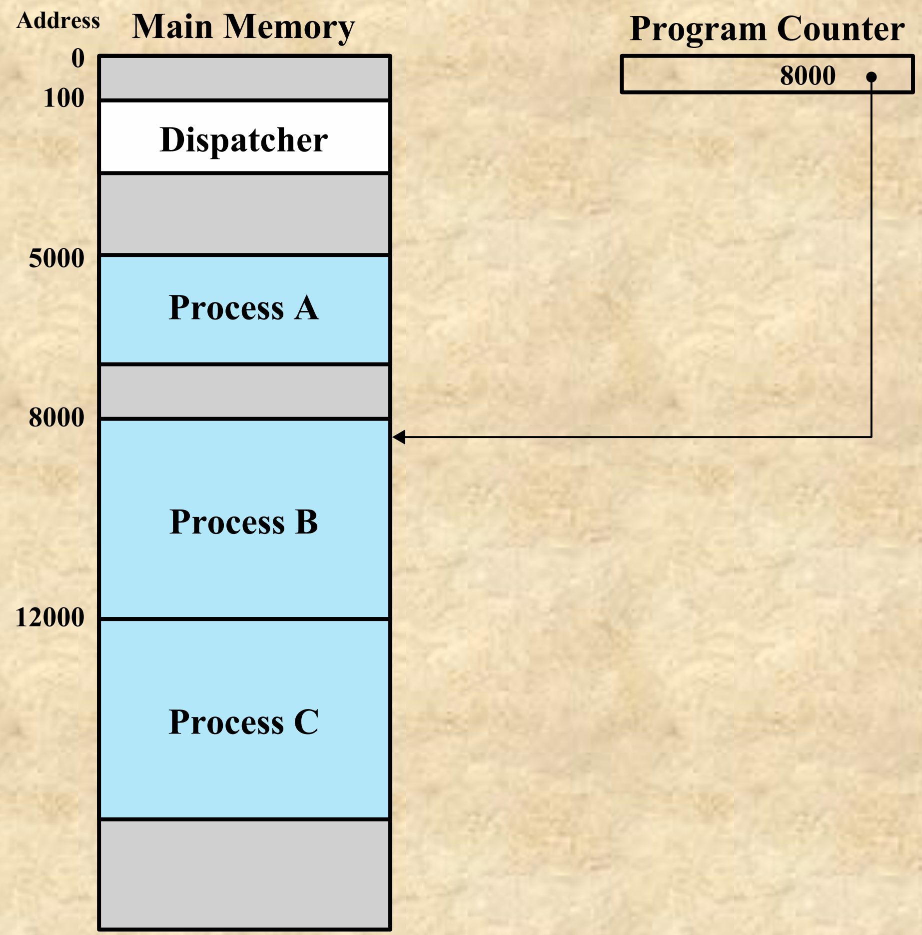


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13

(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

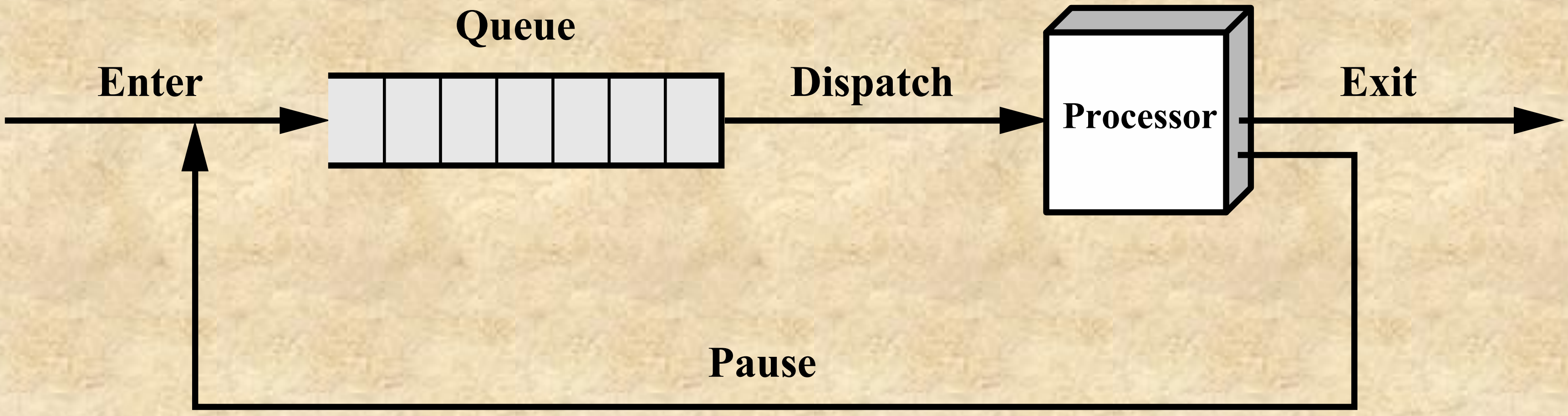
Figure 3.3 Traces of Processes of Figure 3.2

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;
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



(b) Queuing diagram

Figure 3.5 Two-State Process Model

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.

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

Process Termination

- There must be a means for a process to indicate its completion
- A batch job should include a HALT instruction or an explicit OS service call for termination
- For an interactive application, the action of the user will indicate when the process is completed (e.g. log off, quitting an application)

Five-State Process Model

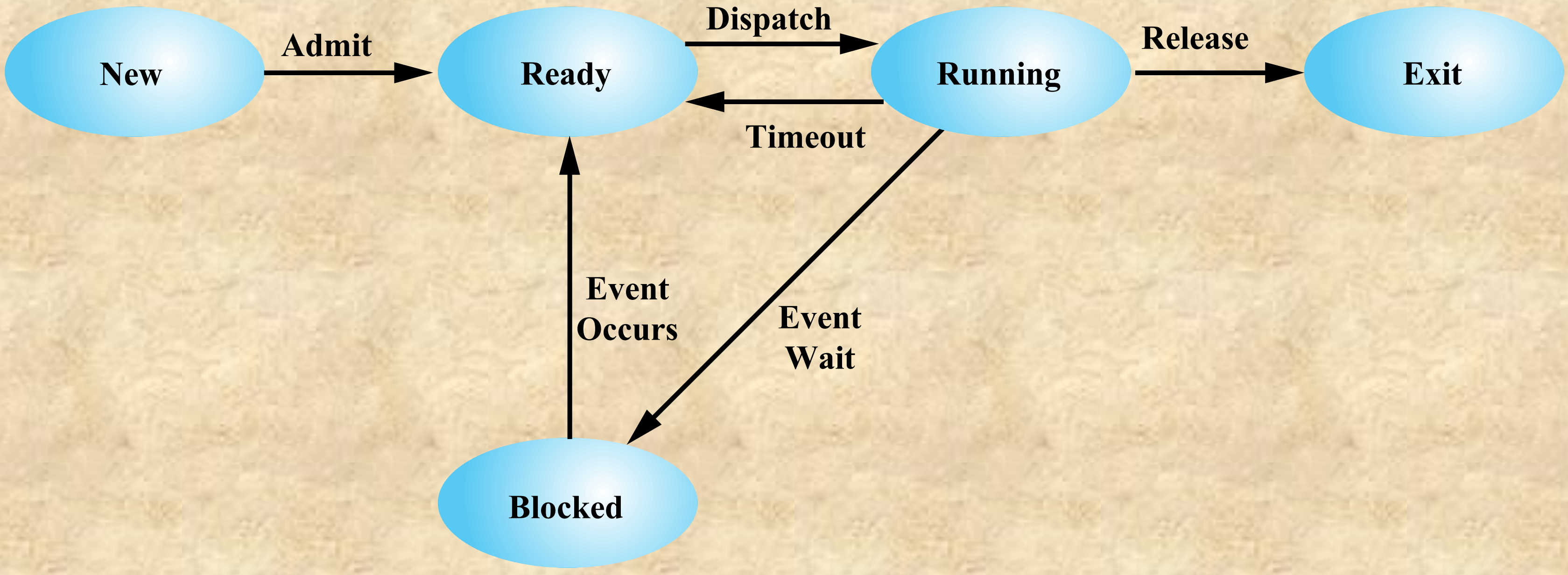


Figure 3.6 Five-State Process Model

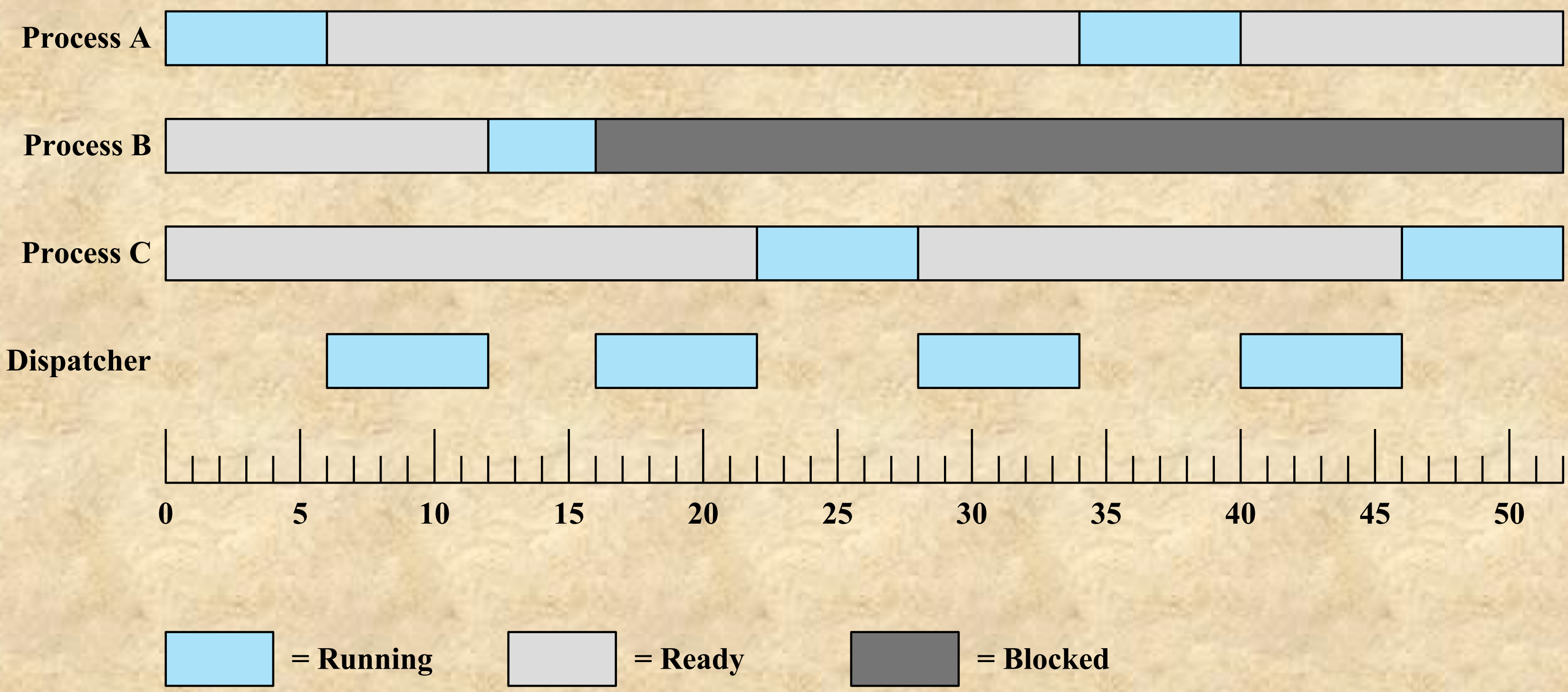


Figure 3.7 Process States for Trace of Figure 3.4

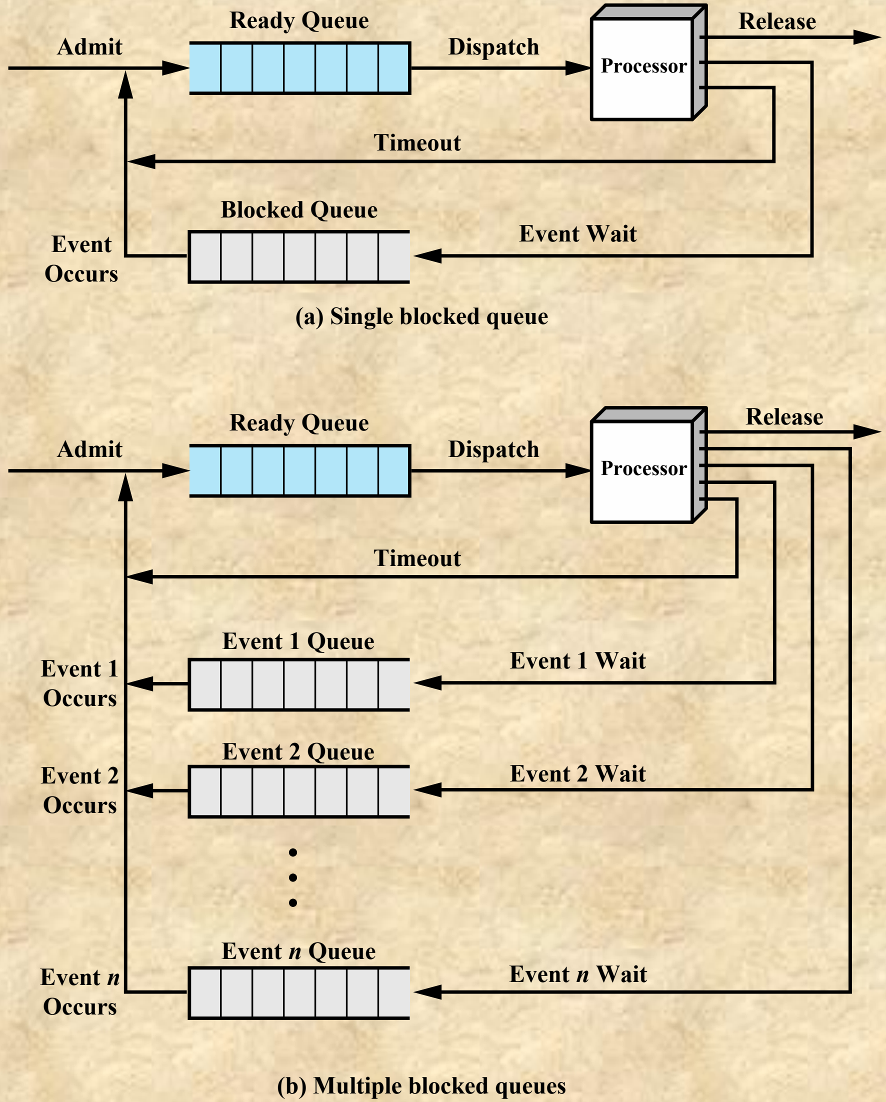
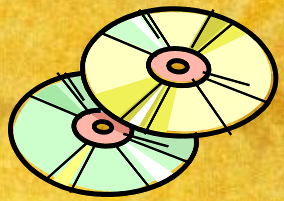
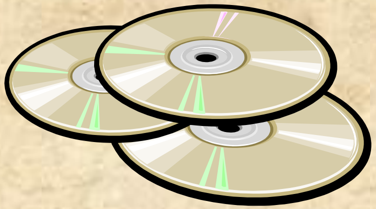


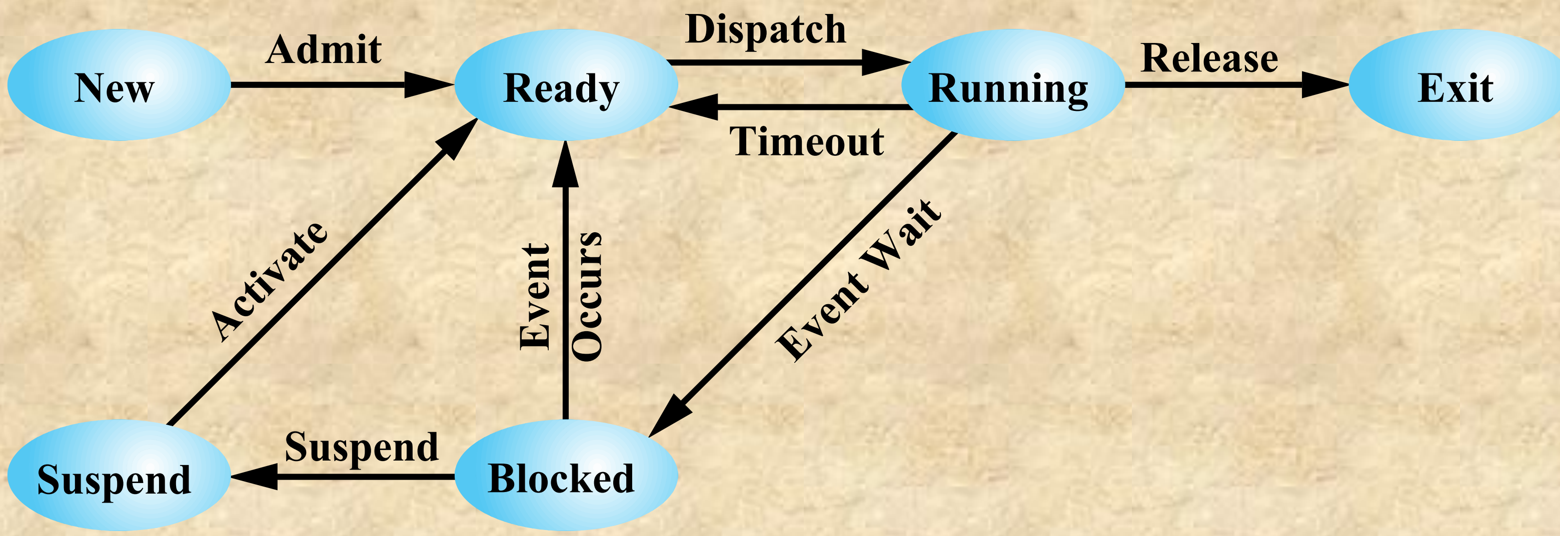
Figure 3.8 Queuing Model for Figure 3.6



Suspended Processes

- Swapping
  - involves moving 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





(a) With One Suspend State

Figure 3.9 Process State Transition Diagram with Suspend States

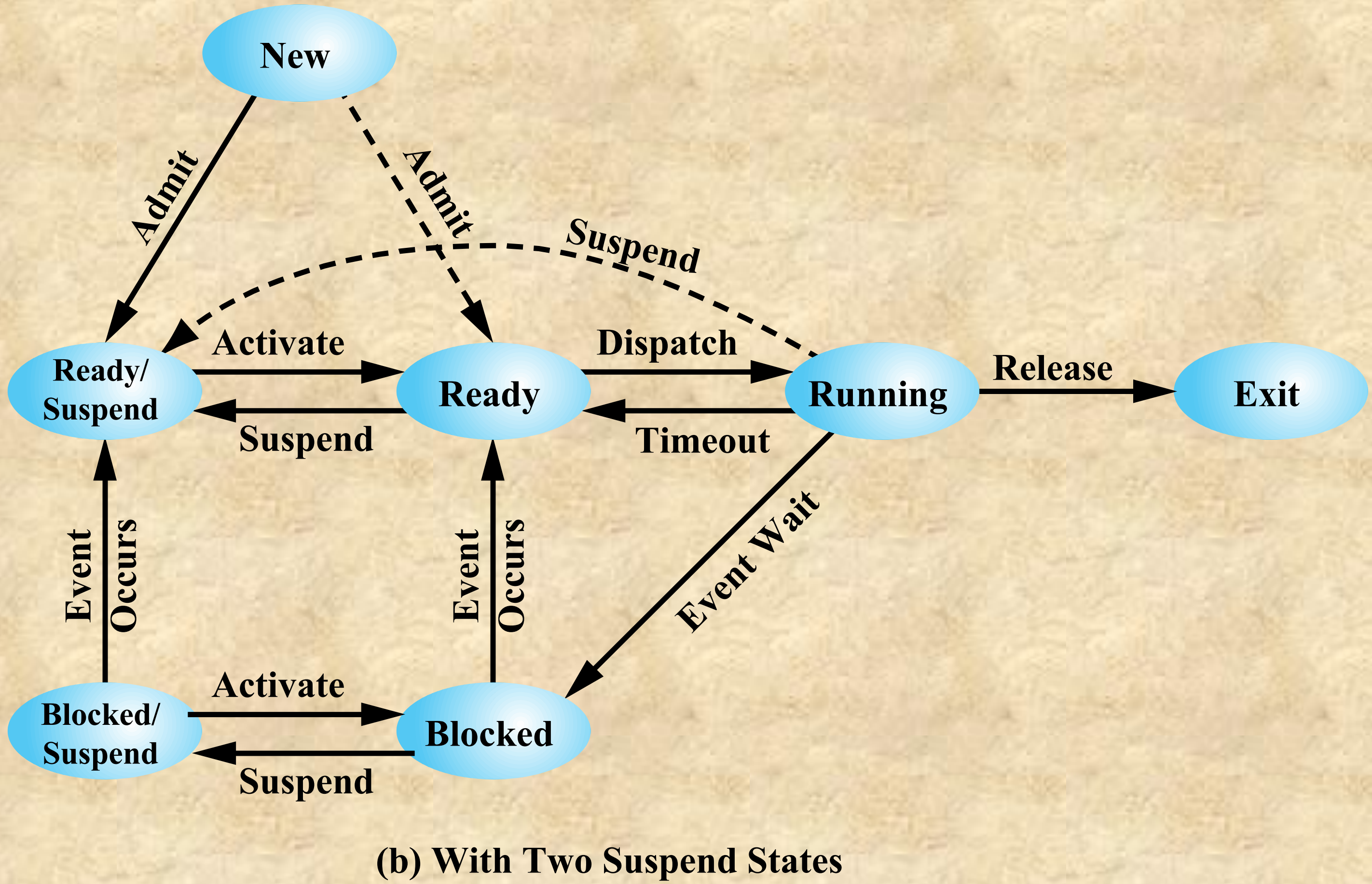


Figure 3.9 Process State Transition Diagram with Suspend States

Characteristics of a Suspended Process

- The process is not immediately available for execution
- The process was placed in a suspended state by an agent: either itself, a parent process, or the OS, for the purpose of preventing its execution
- 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.

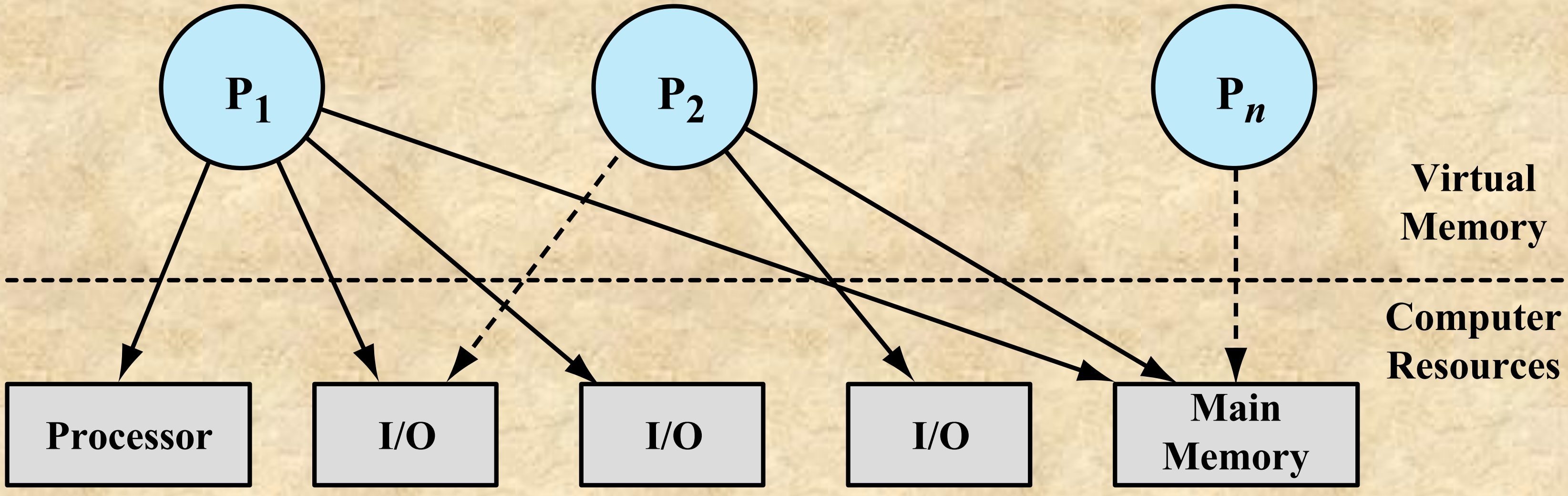


Figure 3.10 Processes and Resources (resource allocation at one snapshot in time)

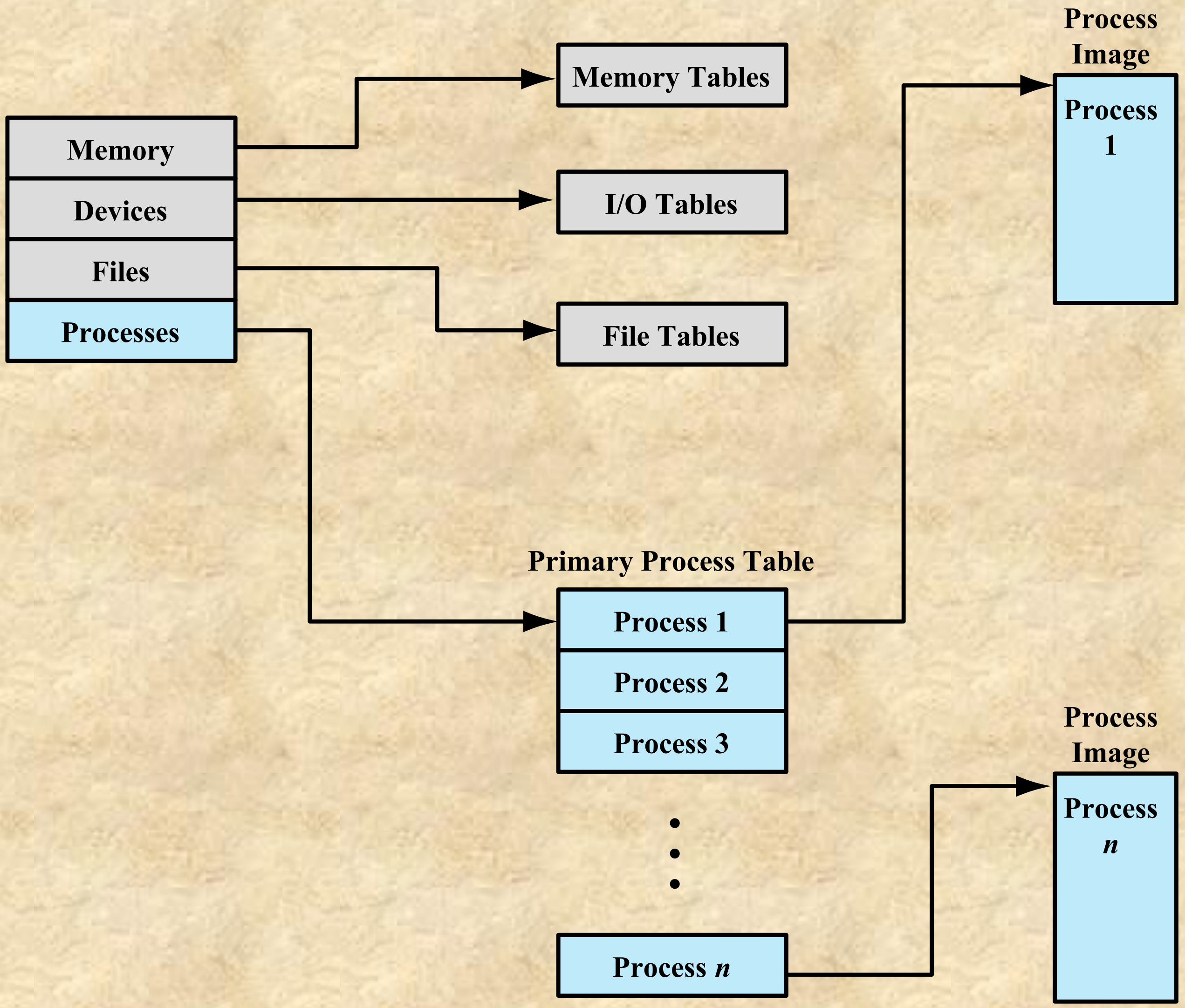


Figure 3.11 General Structure of Operating System Control Tables

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 or 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
- information needed to manage virtual memory

I/O Tables

- Used by the OS to manage the I/O devices and 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

File Tables

These tables provide information about:

- existence of files
- location on secondary memory
- current status
- other attributes

- 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 operating systems, much of the detail of file management is managed by the OS itself

Process Tables

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

Process Control Structures

To manage and control a process the OS must know:

- where the process is located
- the attributes of the process that are necessary for its management

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

- 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

Processor State Information

Consists of the contents of processor registers

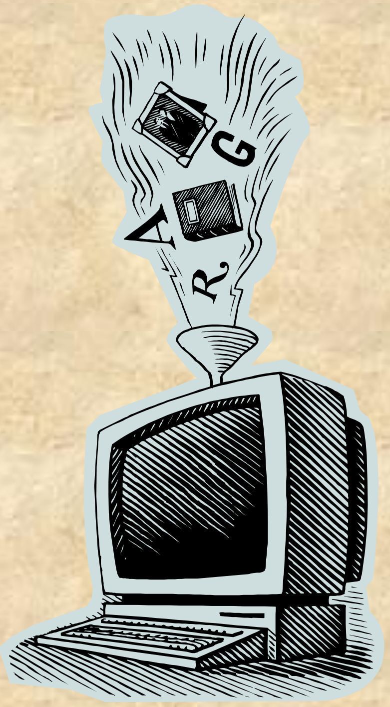
- user-visible registers
- control and status registers
- stack pointers

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

Process Control Information

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



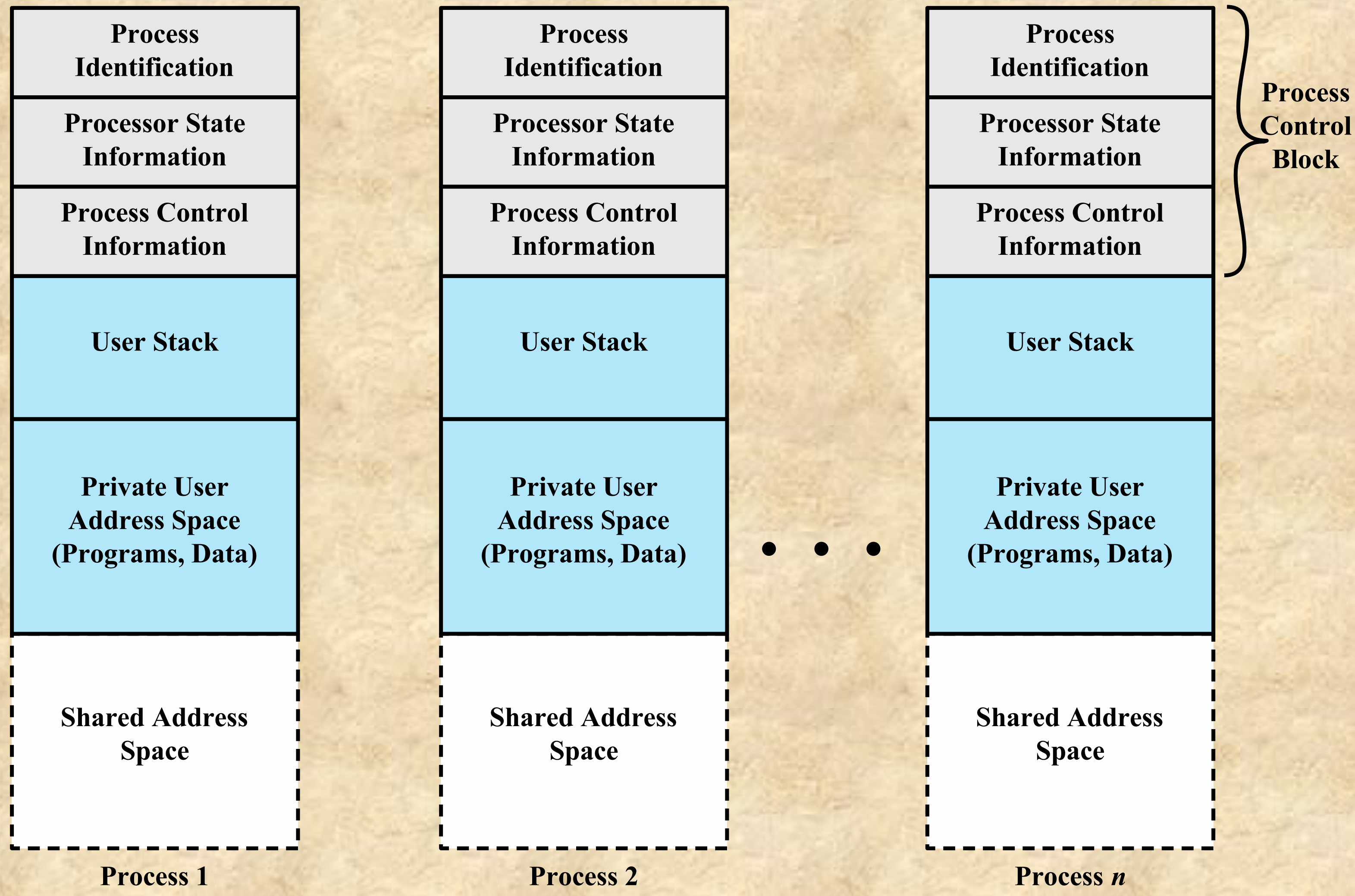


Figure 3.13 User Processes in Virtual Memory

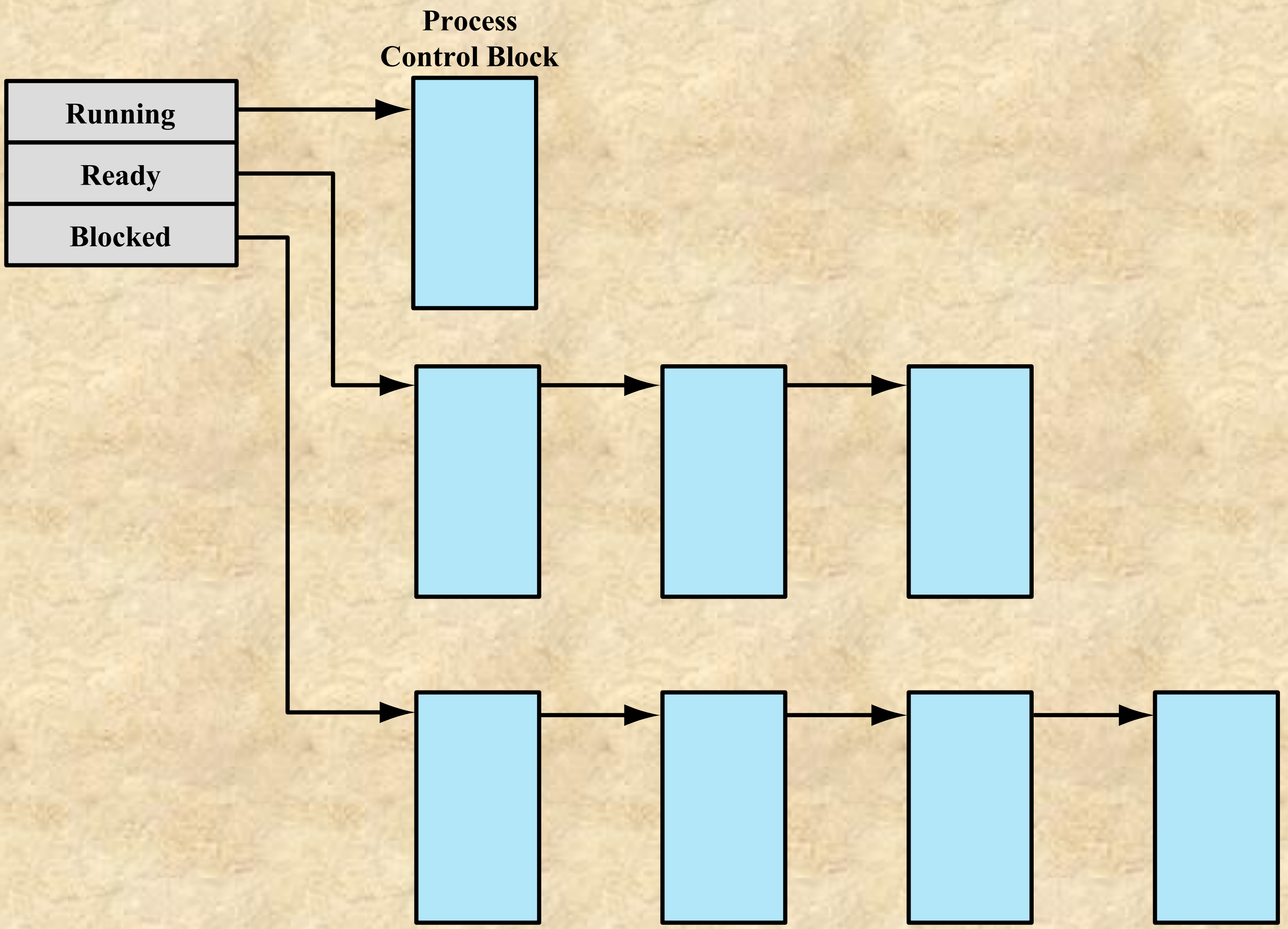


Figure 3.14 Process List Structures

Role of the Process Control Block

- The most important data structure in an OS
  - contains all of the information about a process that is needed by the OS
  - 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

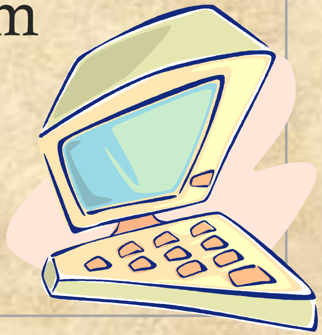
Modes of Execution

User Mode

- less-privileged mode
- user programs typically execute in this mode

System Mode

- more-privileged mode
- also referred to as control mode or kernel mode
- kernel of the operating system



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

Process Creation

- Once the OS decides to create a new process it:
  1. assigns a unique process identifier to the new process
  2. allocates space for the process
  3. initializes the process control block
  4. sets the appropriate linkages
  5. creates or expands other data structures

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

- The steps in a full process switch are:

1. save the context of the processor
2. update the process control block of the process currently in the Running state
3. move the process control block of this process to the appropriate queue
4. select another process for execution
5. update the process control block of the process selected
6. update memory management data structures
7. restore the context of the processor to that which existed at the time the selected process was last switched out

If the currently running process is to be moved to another state (Ready, Blocked, etc.), then the OS must make substantial changes in its environment