

*Operating
Systems:
Internals
and Design
Principles*

Chapter 3 Process Description and Control

Ninth Edition
By William Stallings

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

What is a process?

- What is a process?
- We can have many definitions
- 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.

Process Elements

- Two essential elements of a process are:

Program code

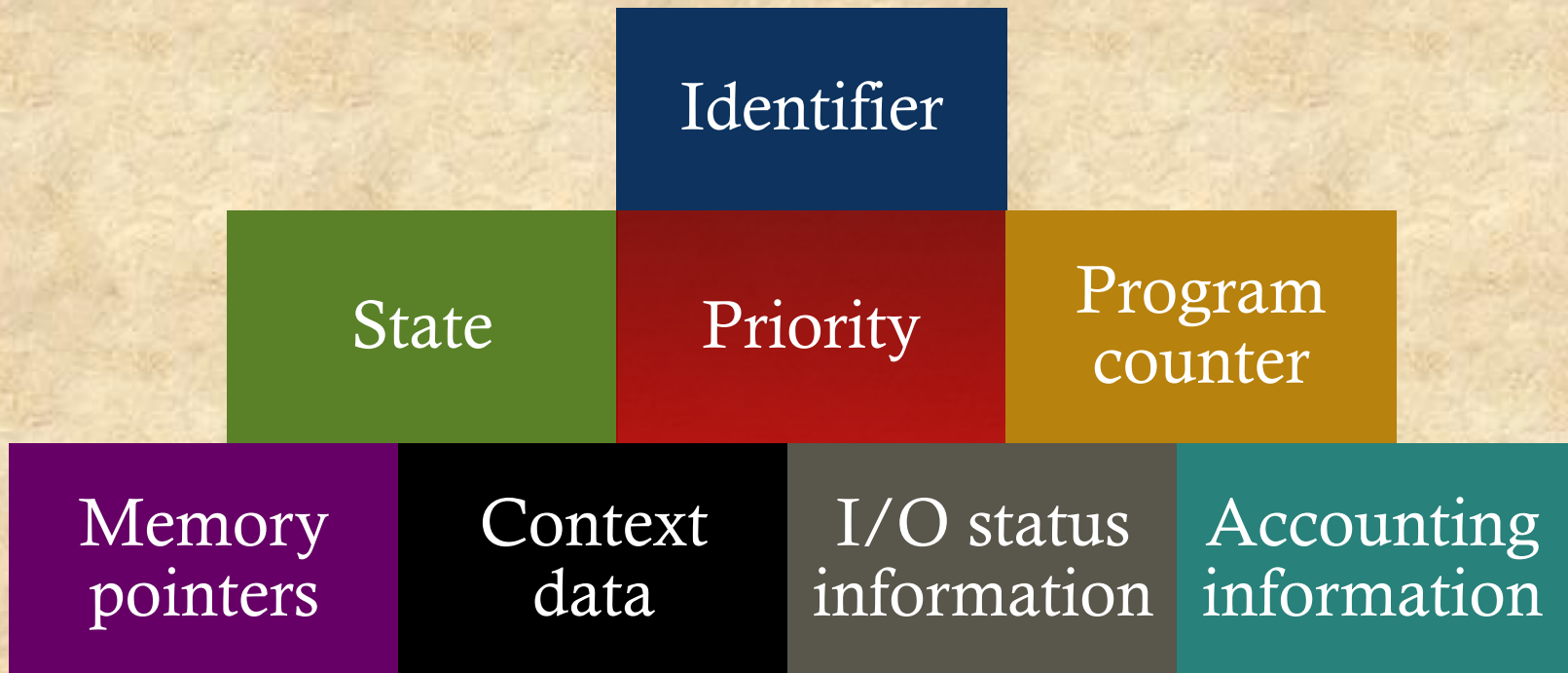
- which may be shared with other processes that are executing the same program

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:



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

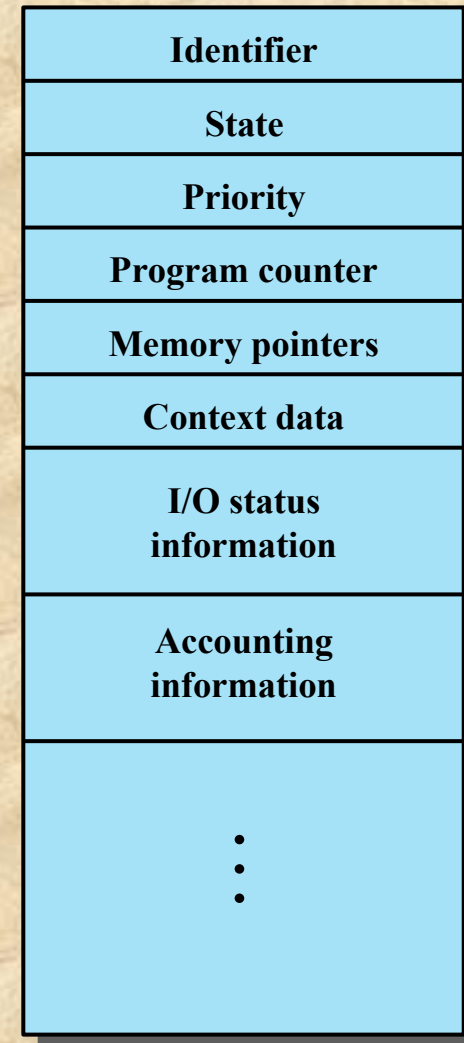


Figure 3.1 Simplified Process Control Block

What happens when a process is interrupted?

- The current values of the program counter and the processor registers (context data) are saved in the appropriate fields in PCB
- process control block, and the state of the process is changed to some other value, such as blocked or ready

Process States

Trace

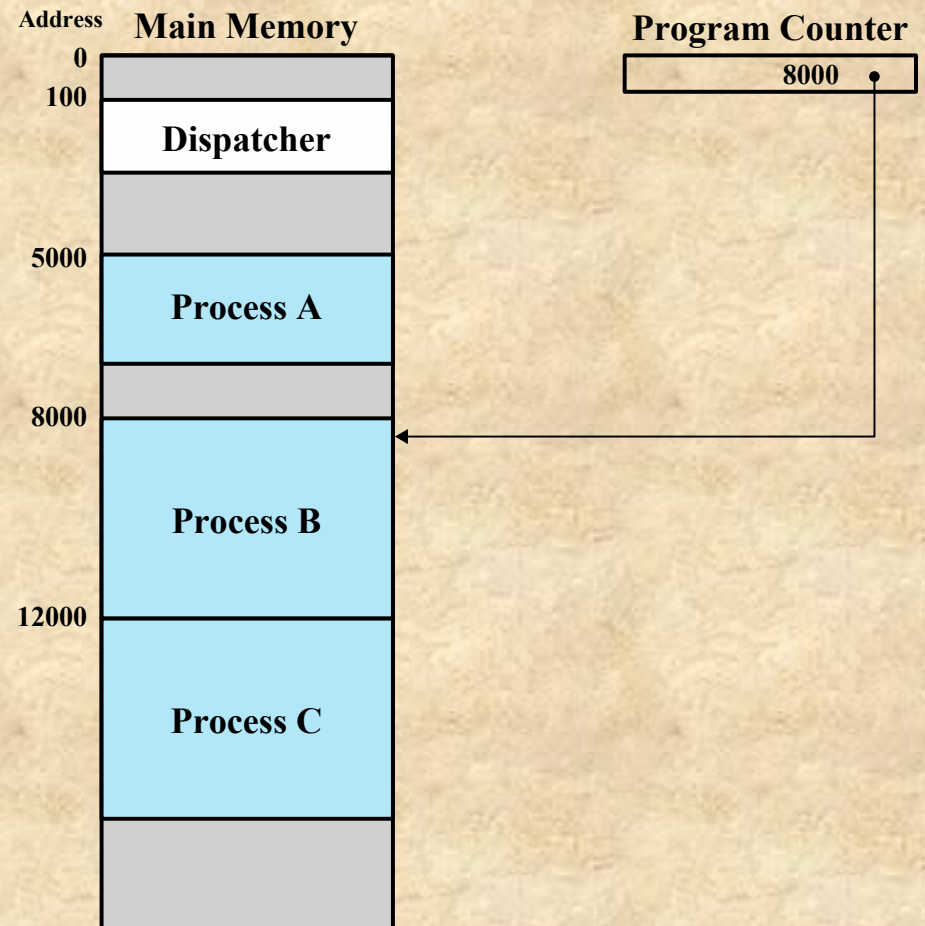
The behavior of an individual process by listing the sequence of instructions that execute for that process

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

Process Execution



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

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

(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

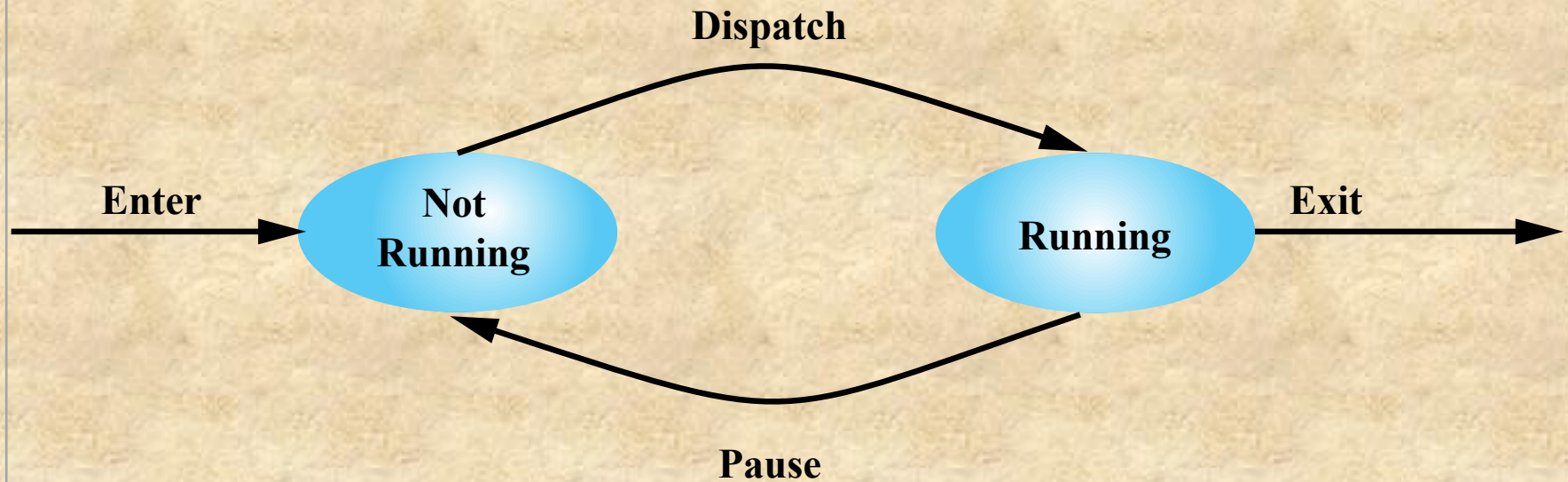
1	5000	27	12004
2	5001	28	12005
3	5002	----- Timeout	
4	5003	29	100
5	5004	30	101
6	5005	31	102
----- Timeout		32	103
7	100	33	104
8	101	34	105
9	102	35	5006
10	103	36	5007
11	104	37	5008
12	105	38	5009
13	8000	39	5010
14	8001	40	5011
15	8002	----- Timeout	
16	8003	41	100
----- I/O Request		42	101
17	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

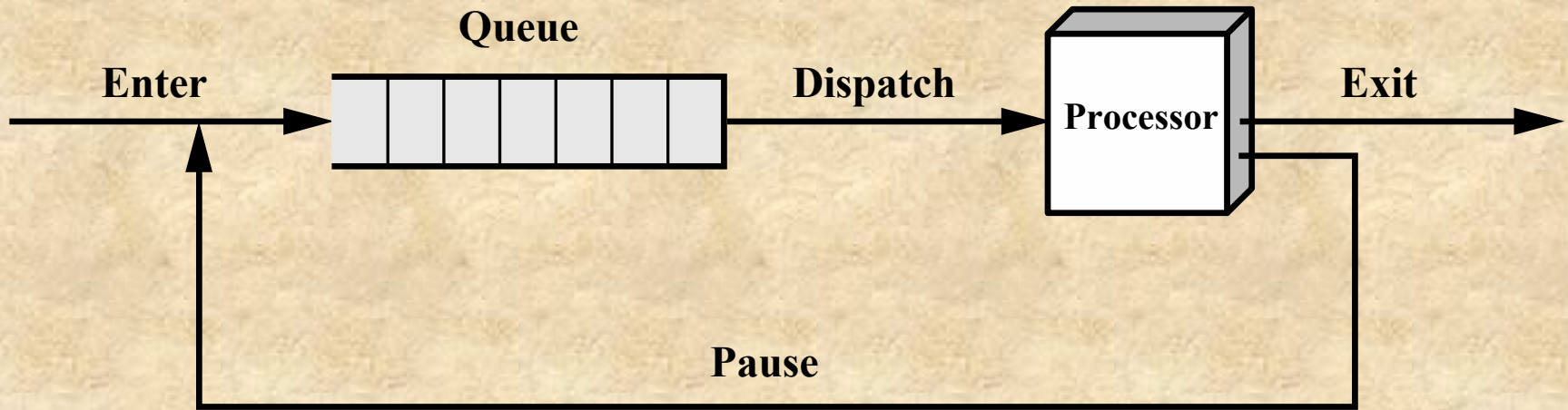
Shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Figure 3.4 Combined Trace of Processes of Figure 3.2

Two-State Process Model



(a) State transition diagram



(b) Queuing diagram

Figure 3.5 Two-State Process Model

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 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

- When the OS creates a process at the explicit request of another process

Parent process

- Is the original, creating, process

Child 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)

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.

(Table is located on page 111
in the textbook)

What are limitations of 2-state model?

- Some processes in not-running state are ready to execute while others are not.
 - Ex: Processes waiting for I/O event to occur.
 - How can OS select a process to execute?
- Need to go through the list of the not running processes and find the process that waited longest.
- This is not efficient. What can we do?
 - Divide the Not Running state into Ready and Blocked/waiting.
 - New and Exit states

Five-State Process Model

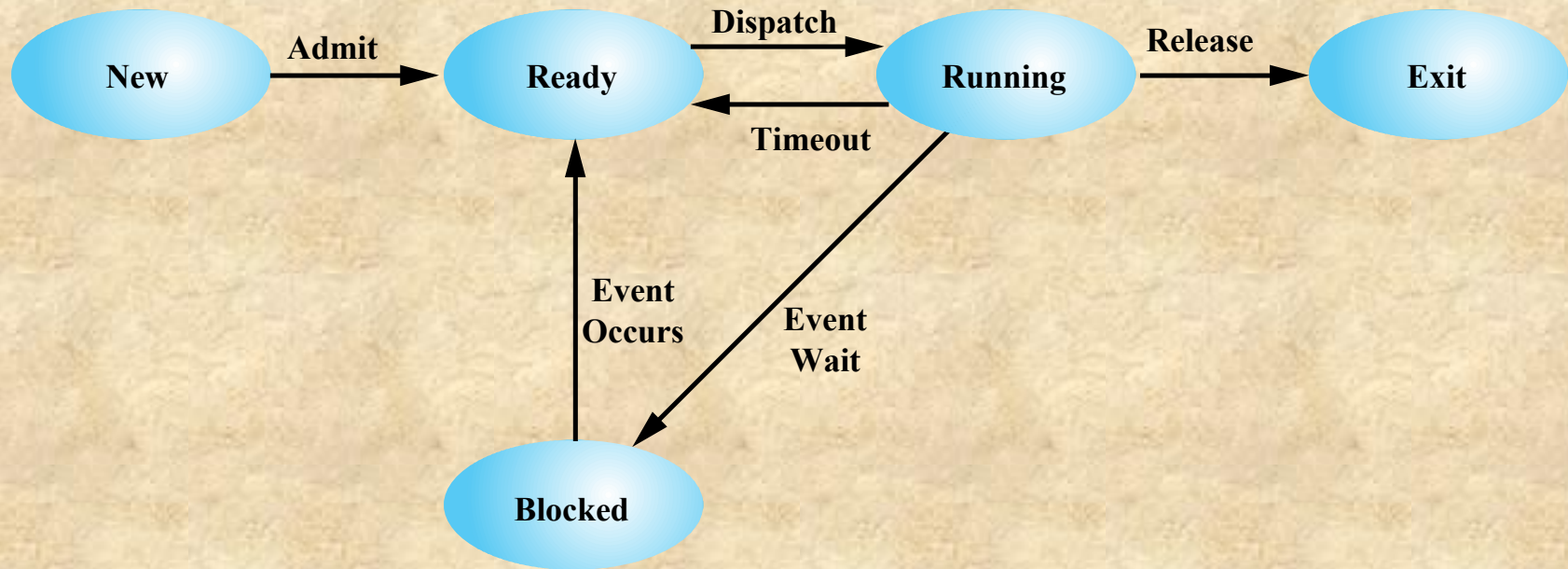


Figure 3.6 Five-State Process Model

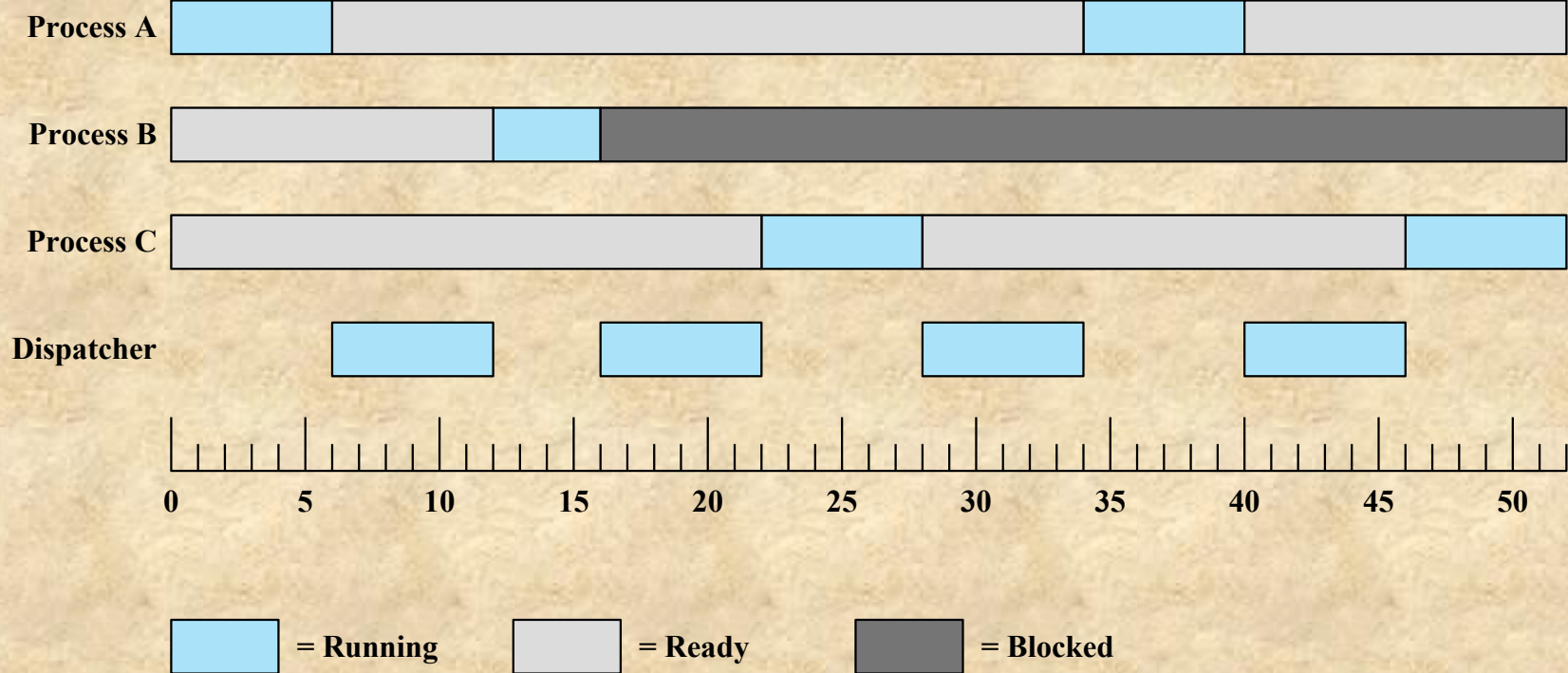
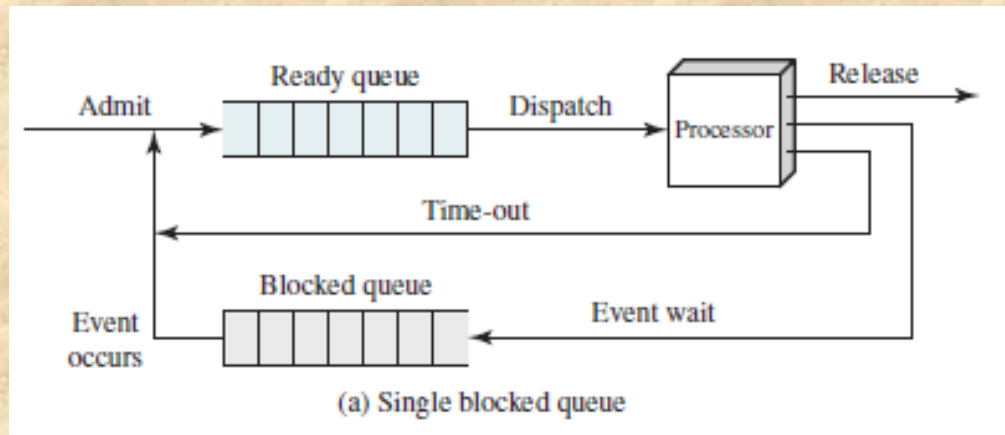


Figure 3.7 Process States for Trace of Figure 3.4

How can we implement this?

- Can we modify the earlier queue-based system for 5-state model?



- Any disadvantages?

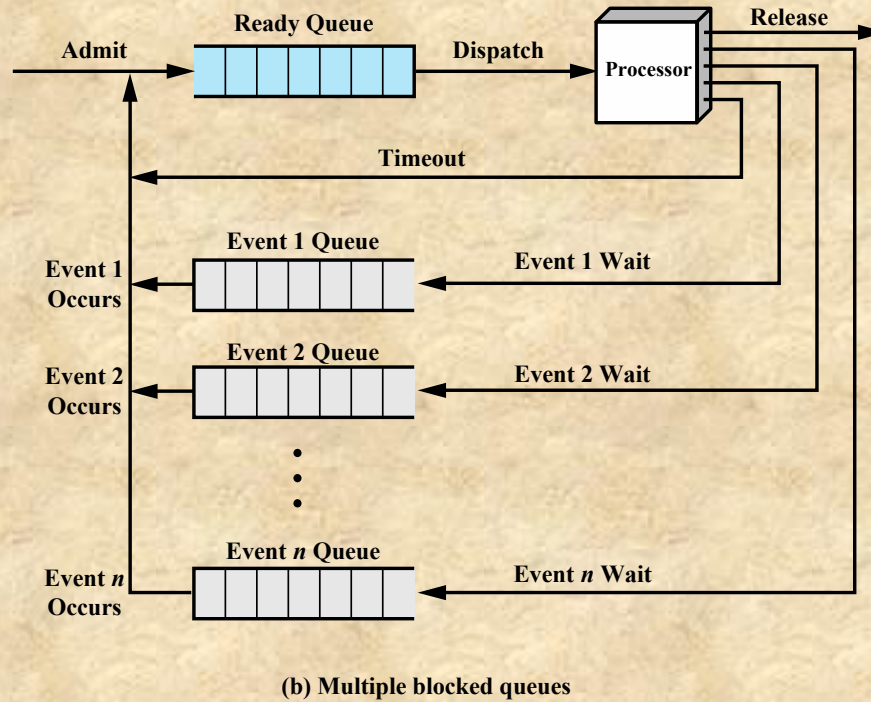
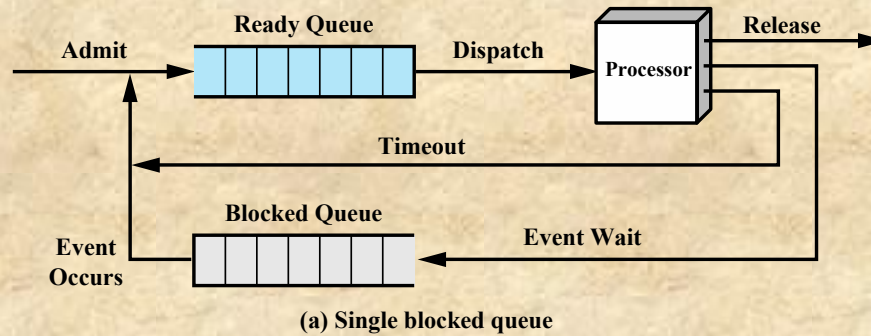


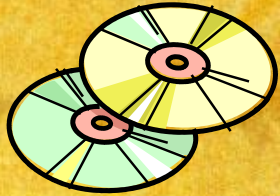
Figure 3.8 Queuing Model for Figure 3.6

Priority based scheduling

- What can we modify in earlier queuing model to support priority-based scheduling?
- Add a ready queue for each priority level

What if we employ swapping?

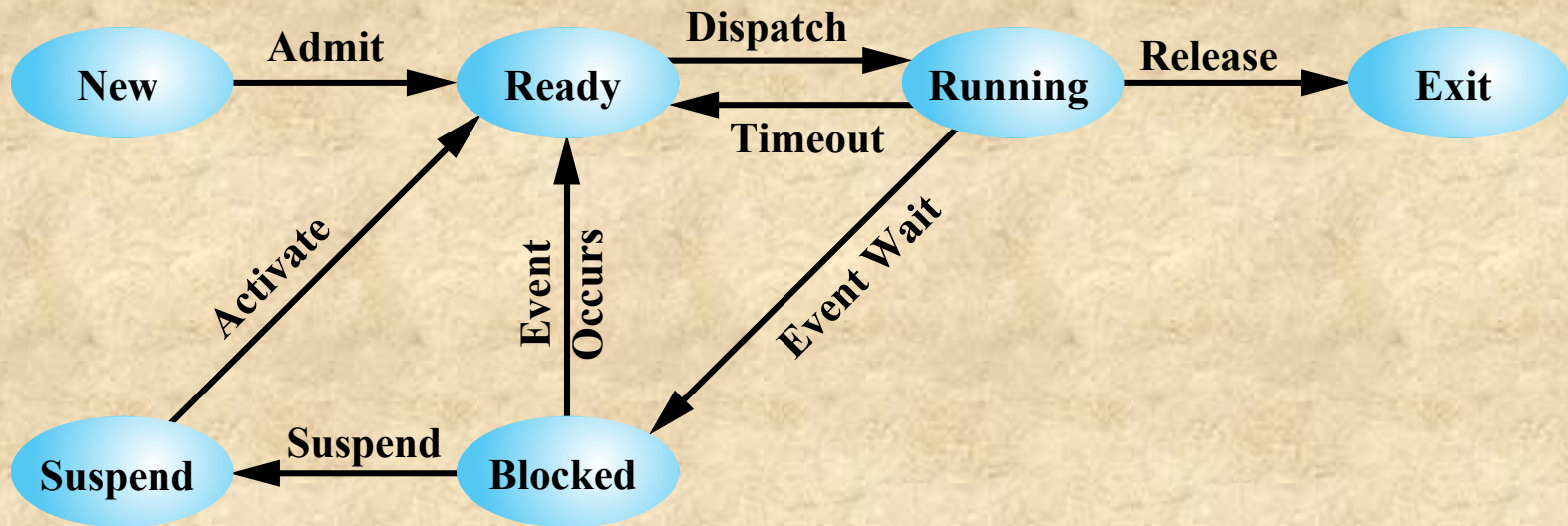
- What if all processes in MM are blocked and has no empty space (or low on space?)



Suspended Processes

■ Swapping

- Involves moving part of all of a process from main memory to disk.
- We talked about this idea Virtual Memory.
- 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
 - This is a queue of existing processes that have been temporarily kicked out of main memory, or suspended
 - The OS then brings in another process from the suspend queue or it honors a new-process request
 - Execution then continues with the newly arrived process
- We can add Suspend state to support this.

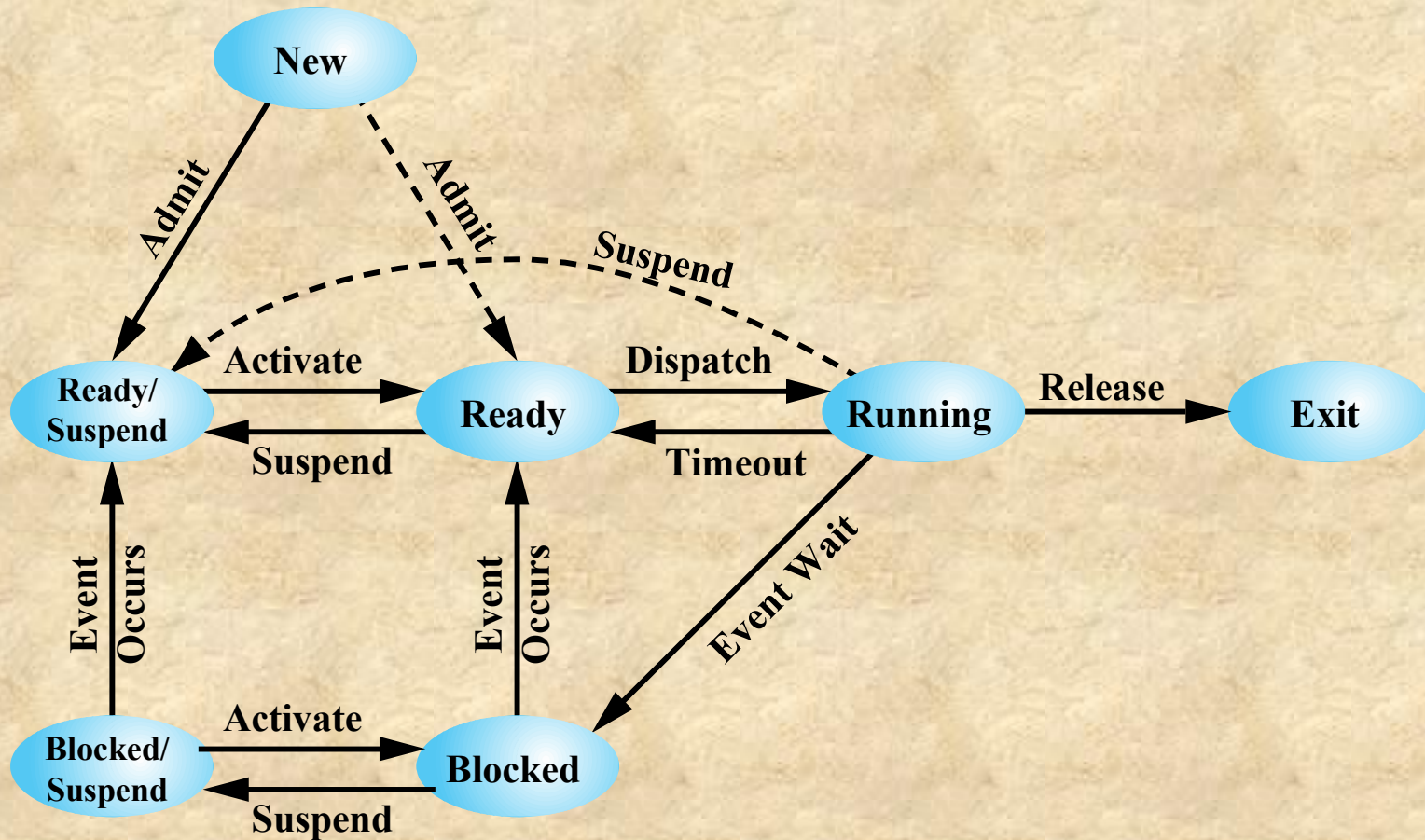


(a) With One Suspend State

Figure 3.9 Process State Transition Diagram with Suspend States

We need to improve this

- Suspended process may still be waiting for event to occur.
- Event that the suspended process was waiting may have occurred already.
- Note that if we use virtual memory, we can execute a process which is partially in the MM.
- But even with VM, OS may have to swap the whole process.



(b) With Two 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 may or may not be waiting on an event
- 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 not be removed from this state until the agent explicitly orders the removal

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.

Process Description

- Operating system Control Structures
- What information does the OS need to control processes and manage resources for them?

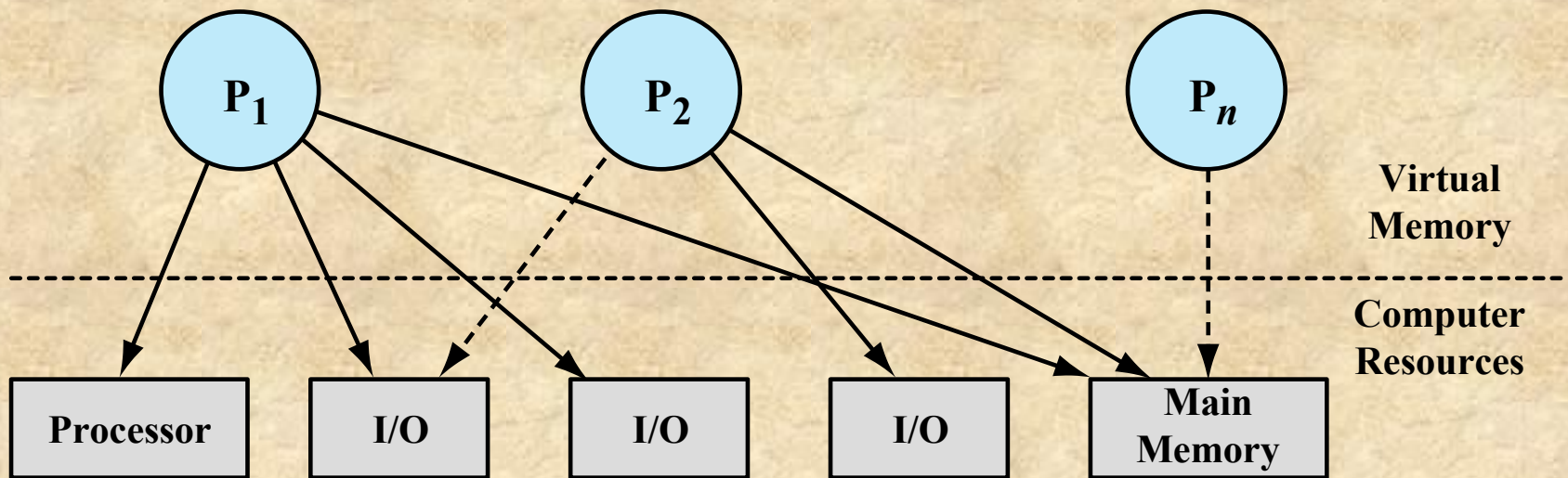


Figure 3.10 Processes and Resources (resource allocation at one snapshot in time)

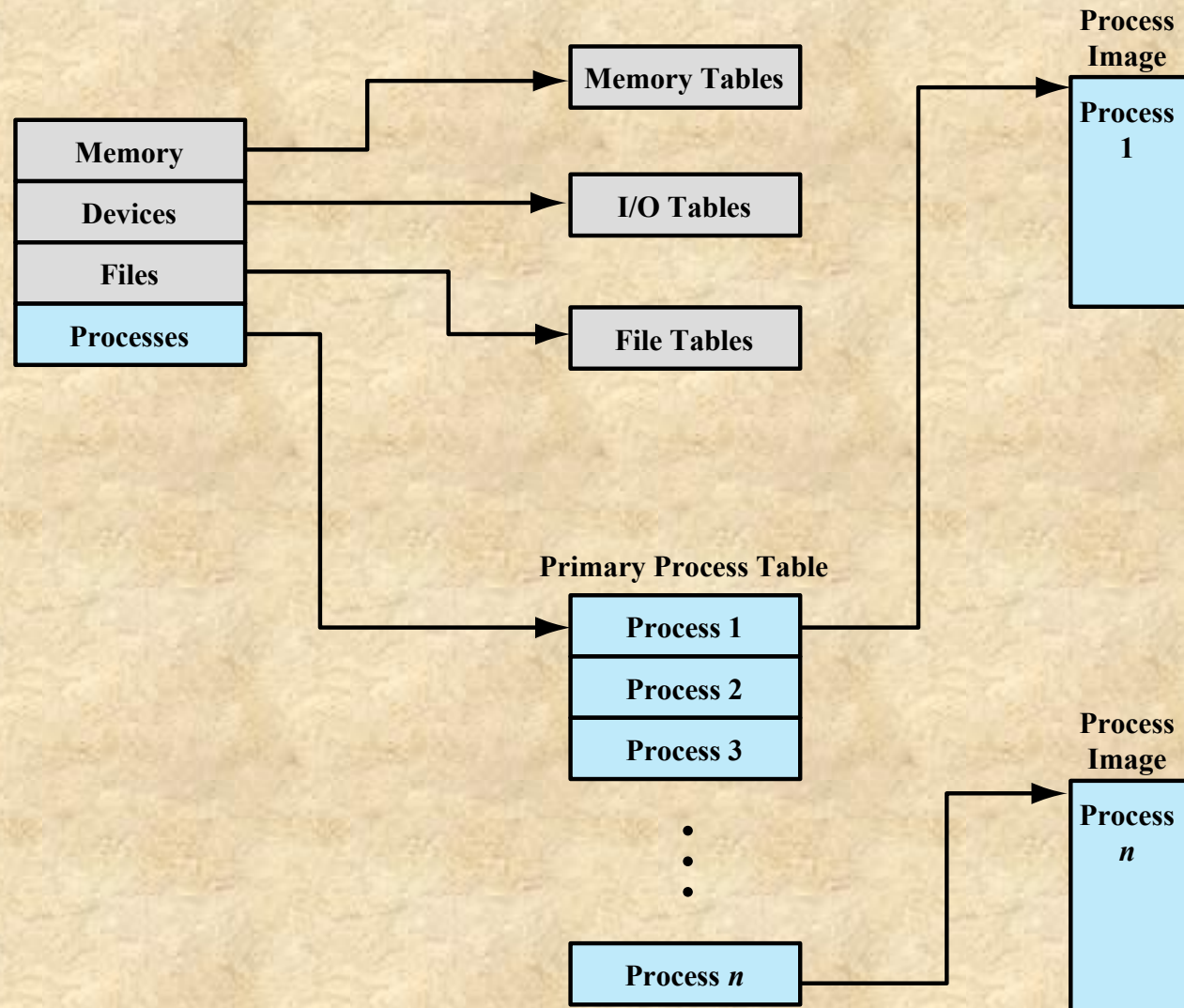


Figure 3.11 General Structure of Operating System Control Tables

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
- Page table, segment table, swap table, etc.

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
- Device status table, I/O buffer table, etc.

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
- File descriptor table

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

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 Block

- We can group process control block information into 3 categories
 - Process Identification
 - Process State Information
 - Process Control Information

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.

Table 3.5
Typical
Elements
of a
Process
Control
Block
(page 1 of 2)

(Table is located
on page 125 in the
textbook)

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.

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.

Table 3.5

Typical

Elements of a

Process

Control Block

(page 2 of 2)

(Table is located
on page 125 in the textbook)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	I D	V I P	V I F	A C	V M	R F	0	N T	I O P L	O F	D F	I F	T F	S F	Z F	0	A F	0	P F	1	C F	

X ID = Identification flag
 X VIP = Virtual interrupt pending
 X VIF = Virtual interrupt flag
 X AC = Alignment check
 X VM = Virtual 8086 mode
 X RF = Resume flag
 X NT = Nested task flag
 X IOPL = I/O privilege level
 S OF = Overflow flag

C DF = Direction flag
 X IF = Interrupt enable flag
 X TF = Trap flag
 S SF = Sign flag
 S ZF = Zero flag
 S AF = Auxiliary carry flag
 S PF = Parity flag
 S CF = Carry flag

S Indicates a Status Flag
 C Indicates a Control Flag
 X Indicates a System Flag
 Shaded bits are reserved

Figure 3.12 x86 EFLAGS Register

Table 3.6

x86

EFLAGS

Register

Bits

(Table is located on page 127 in the textbook)

Status Flags (condition codes)

AF (Auxiliary carry flag)

Represents carrying or borrowing between half-bytes of an 8-bit arithmetic or logic operation using the AL register.

CF (Carry flag)

Indicates carrying out or borrowing into the leftmost bit position following an arithmetic operation. Also modified by some of the shift and rotate operations.

OF (Overflow flag)

Indicates an arithmetic overflow after an addition or subtraction.

PF (Parity flag)

Parity of the result of an arithmetic or logic operation. 1 indicates even parity; 0 indicates odd parity.

SF (Sign flag)

Indicates the sign of the result of an arithmetic or logic operation.

ZF (Zero flag)

Indicates that the result of an arithmetic or logic operation is 0.

Control Flag

DF (Direction flag)

Determines whether string processing instructions increment or decrement the 16-bit half-registers SI and DI (for 16-bit operations) or the 32-bit registers ESI and EDI (for 32-bit operations).

System Flags (should not be modified by application programs)

AC (Alignment check)

Set if a word or doubleword is addressed on a nonword or nondoubleword boundary.

ID (Identification flag)

If this bit can be set and cleared, this processor supports the CPUID instruction. This instruction provides information about the vendor, family, and model.

RF (Resume flag)

Allows the programmer to disable debug exceptions so that the instruction can be restarted after a debug exception without immediately causing another debug exception.

IOPL (I/O privilege level)

When set, causes the processor to generate an exception on all accesses to I/O devices during protected mode operation.

IF (Interrupt enable flag)

When set, the processor will recognize external interrupts.

TF (Trap flag)

When set, causes an interrupt after the execution of each instruction. This is used for debugging.

NT (Nested task flag)

Indicates that the current task is nested within another task in protected mode operation.

VM (Virtual 8086 mode)

Allows the programmer to enable or disable virtual 8086 mode, which determines whether the processor runs as an 8086 machine.

VIP (Virtual interrupt pending)

Used in virtual 8086 mode to indicate that one or more interrupts are awaiting service.

VIF (Virtual interrupt flag)

Used in virtual 8086 mode instead of IF.

Process Control Information

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

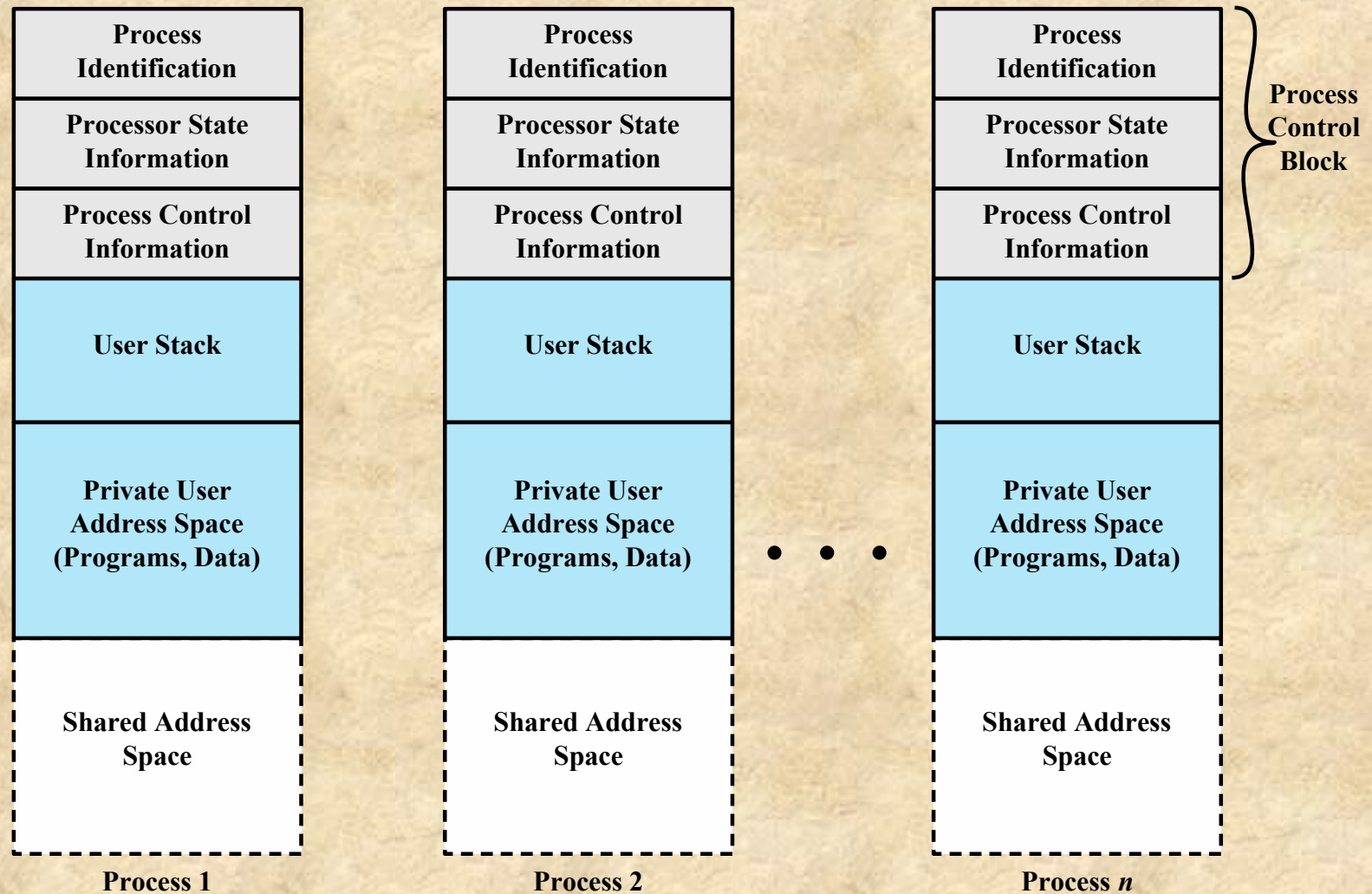


Figure 3.13 User Processes in Virtual Memory

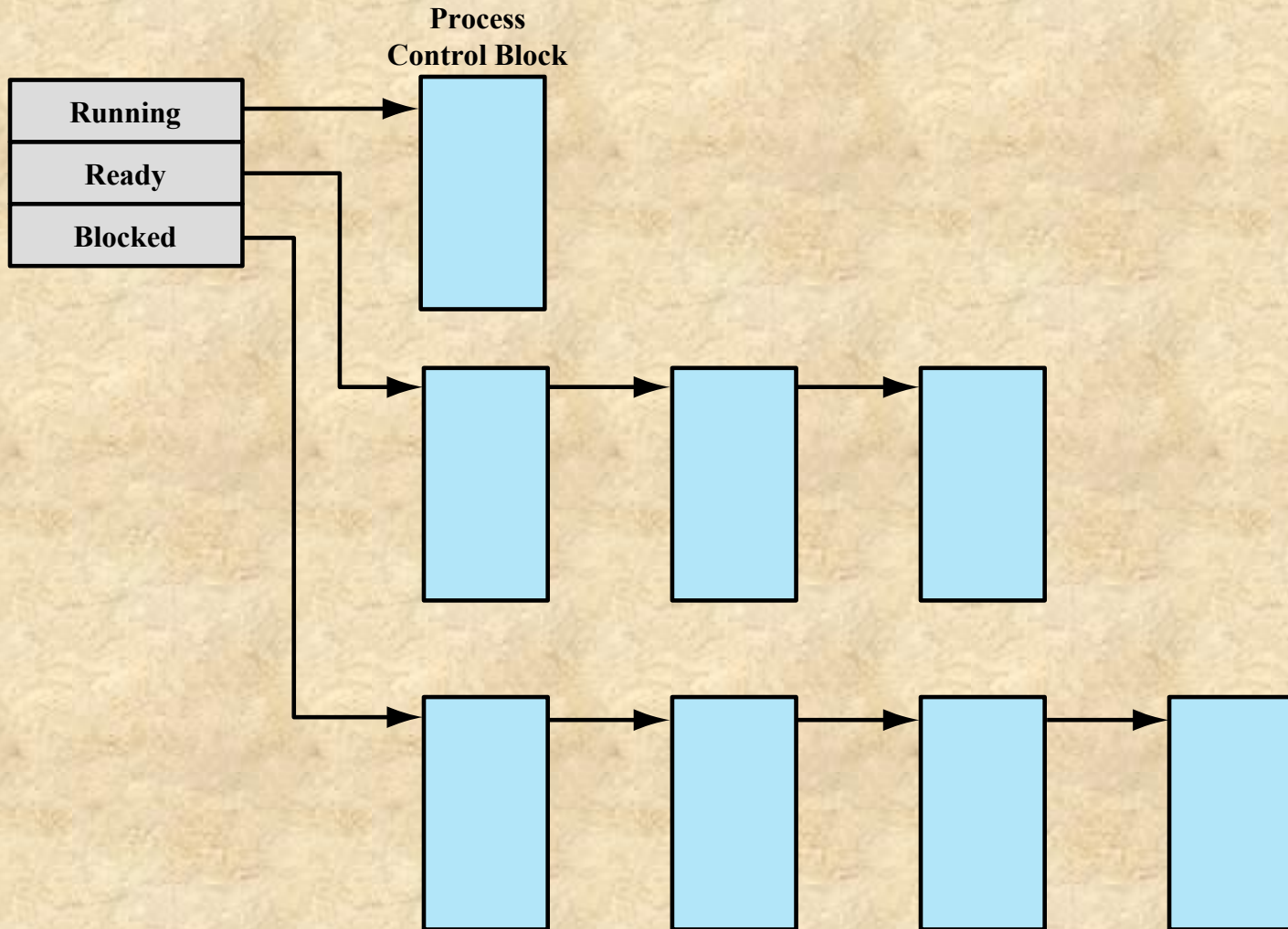


Figure 3.14 Process List Structures

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
- A Handler can be used which takes care of writing and reading PCBs

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
- Kernel mode/control mode

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

Table 3.7

Typical Functions of an Operating System Kernel

Process Creation

- Once the OS decides to create a new process it:

Assigns a unique process identifier to the new process

Allocates space for the process

Initializes the process control block

Sets the appropriate linkages

Creates or expands other data structures

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

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 the design of the OS

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:

Save the context of the processor



Update the process control block of the process currently in the Running state



Move the process control block of this process to the appropriate queue



Select another process for execution



Update the process control block of the process selected



Update memory management data structures



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