

# Process Description and Control

## Chapter 3

1

# 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

2

# Process

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

3

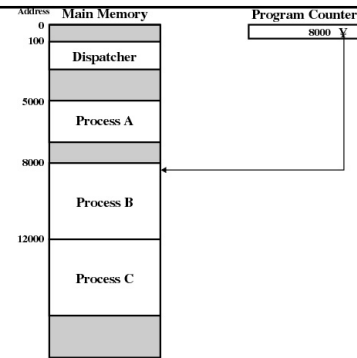


Figure 3.1 Snapshot of Example Execution (Figure 3.3) at Instruction Cycle 13

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 3.2 Traces of Processes of Figure 3.1

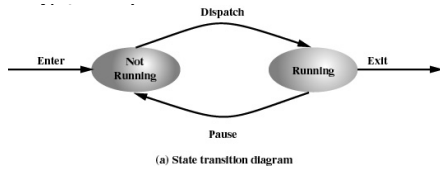
1	5000	27	12004
2	5001	28	12005
3	5002		Time out
4	5003	29	100
5	5004	30	101
6	5005	31	102
			Time out
7	100	32	103
8	101	33	104
9	102	34	105
10	103	35	5006
11	104	36	5007
12	105	37	5008
13	8000	38	5009
14	8001	39	5010
15	8002	40	5011
16	8003		Time out
		41	100
		42	101
17	100	43	102
18	101	44	103
19	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
			Time out

100 = Starting address of dispatcher program  
 Shaded areas indicate execution of dispatcher process.  
 First and third columns record instruction cycles.  
 Second and fourth columns show address of instruction being executed

Figure 3.3 Combined Trace of Processes of Figure 3.1

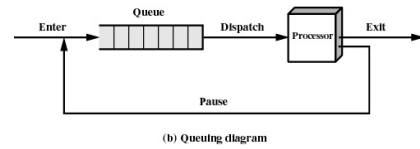
## Two-State Process Model

- Process may be in one of two states
  - Running



7

## Not-Running Process in a Queue



8

## Process Creation

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

9

## Process Termination

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

10

## 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

11

## 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

12

## 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

13

## A Five-State Model

- Running
- Ready
- Blocked
- New
- Exit

14

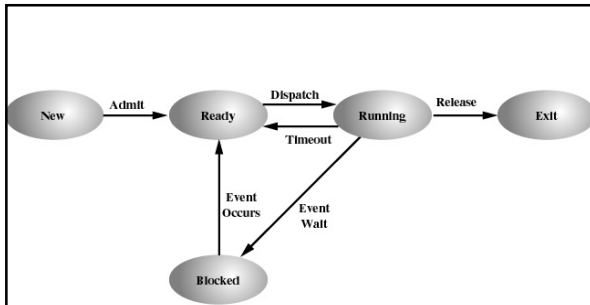


Figure 3.5 Five-State Process Model

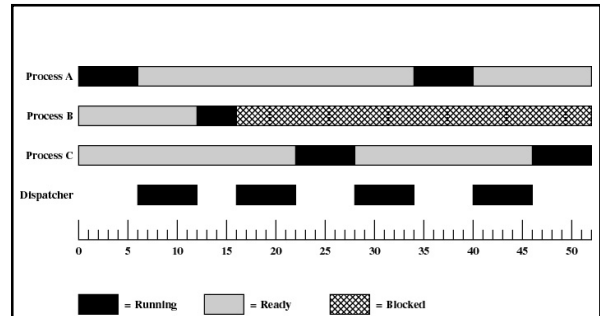
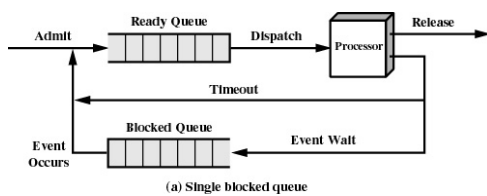
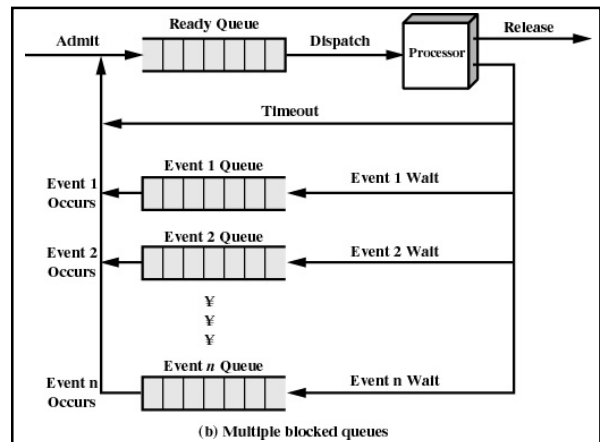


Figure 3.6 Process States for Trace of Figure 3.3

## Using Two Queues



17

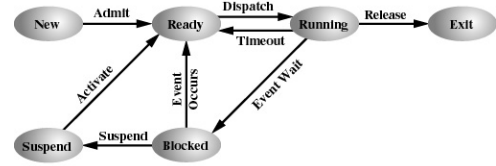


## Suspended Processes

- 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

19

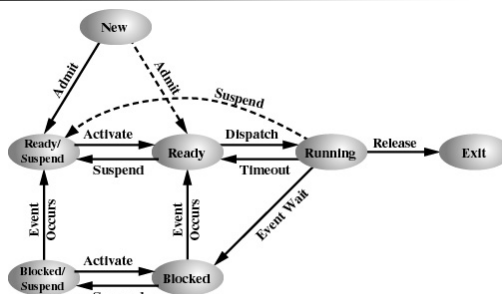
## One Suspend State



(a) With One Suspend State

20

## Two Suspend States



(b) With Two Suspend States

21

## 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.

22

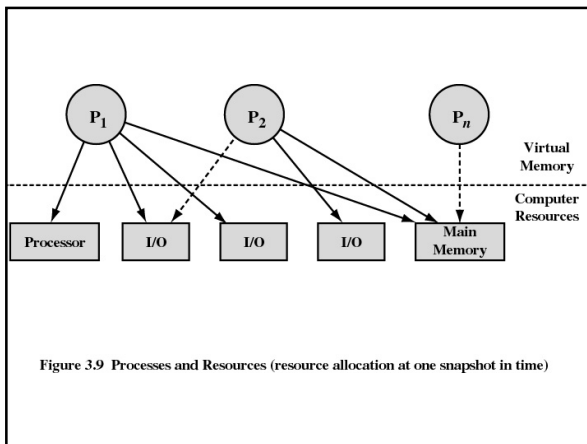


Figure 3.9 Processes and Resources (resource allocation at one snapshot in time)

## Operating System Control Structures

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

24

## 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

25

## 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

26

## File Tables

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

27

## Process Table

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

28

## 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

29

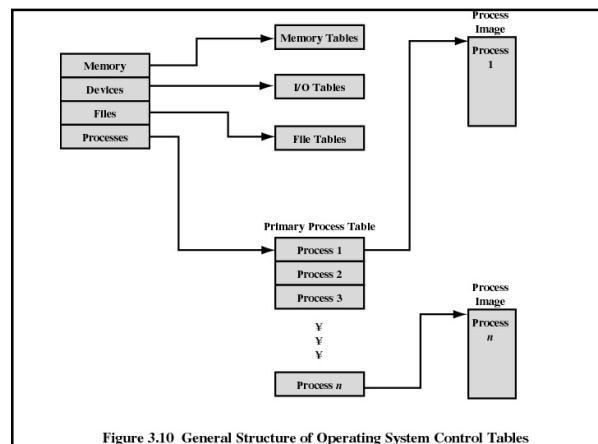


Figure 3.10 General Structure of Operating System Control Tables

## 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

31

## 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.

32

## 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

33

## 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.

34

## 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

35

## 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.

36

## 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.

37

## 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.

38

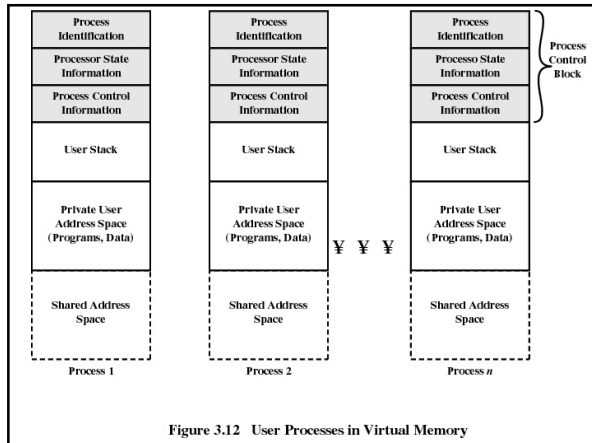


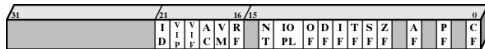
Figure 3.12 User Processes in Virtual Memory

## 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

40

## Pentium II EFLAGS Register



- |                                 |                            |
|---------------------------------|----------------------------|
| 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 = IO privilege level       | CF = Carry flag            |
| OF = Overflow flag              |                            |

Figure 3.11 Pentium II EFLAGS Register

41

## 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

42

## 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 or expand other data structures
  - Ex: maintain an accounting file

43

## 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

44

## When to Switch a Process

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

45

## 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

46

## Change of Process State

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

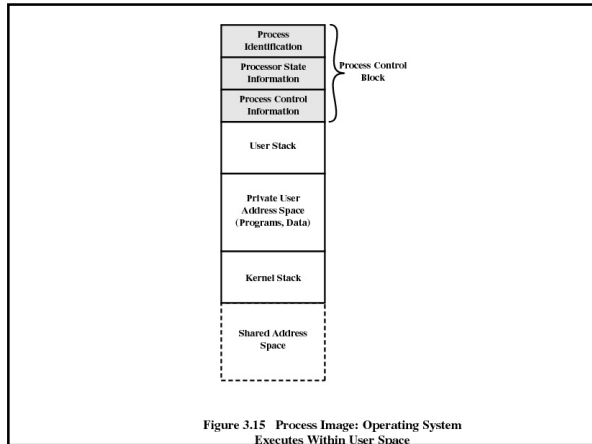
47

## 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

48





## Execution of the Operating System

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

50

## UNIX SVR4 Process Management

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

(b) OS functions execute within user processes

51

## 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.

52

