K. J. Somaiya College of Engineering, Mumbai

(A constituent College of Somaiya Vidyavihar University)

Operating System

Module 2. Process Concept and scheduling

Dr. Prasanna Shete

Dept. of Computer Engineering

prasannashete@somaiya.edu

Mobile/WhatsApp 9960452937

Module 2: Process Concept and scheduling

- Process: Concept of a Process, Process States, Process Description, Process Control Block, Operations on Processes.
- Threads: Definition and Types, Concept of Multithreading
- Multicore processors and threads
- Scheduling: Uniprocessor Scheduling - Types of Scheduling:
- Preemptive and, Non-preemptive, Scheduling Algorithms:
- FCFS, SJF, SRTN, Priority based, Round Robin, Multilevel Queue scheduling.
- Multi Processor Scheduling
- Introduction to Thread Scheduling
- Linux Scheduling

OS Fundamental Task: Process Management

- The Operating System must-
  - **Interleave** the execution of multiple processes
  - **Allocate resources** to processes, and **protect the resources** of each process from other processes
  - Enable processes to **share and exchange information**
  - Enable **synchronization** among processes

What is a "process"?

- 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**

- 2 essential elements of a process:

  Program code and set of data associated with it

Process Elements

- A process is comprised of:
- –Program code (possibly shared)
- –A set of data
- –A number of attributes describing the (state of the) process

Process Elements

- Process can be uniquely characterized by following
  - Identifier: unique identifier to distinguish it from other processes
  - State: Running, not running etc.
  - Priority: priority level relative to other processes
  - Program counter: Address of the next instruction in the program to be executed
  - Memory pointers: pointers to program code and associated data, plus any memory blocks shared with other processes
  - Context data: data that are present in processor registers while the process is executing
  - I/O status information: includes outstanding I/O requests, I/O devices assigned to the process, list of files in use by the process
  - Accounting information: includes amount of processor time and clock time used, time limits, account numbers etc.
  - → stored in a data structure called PCB (process control block), created & managed by OS

Process Control Block

- Created and managed by the OS
- PCB contains sufficient information so that it is **possible to interrupt a running process and resume execution** as if the interrupt had not occurred
- Key tool that enables the OS to support multiple processes; multiprocessing
- → A process consists of Program Code and associated data plus PCB

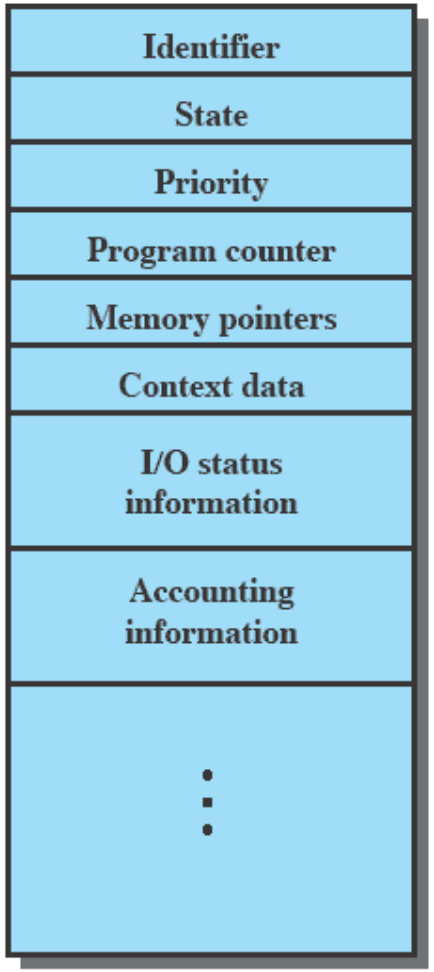


Figure 3.1 Simplified Process Control Block

Trace of the Process

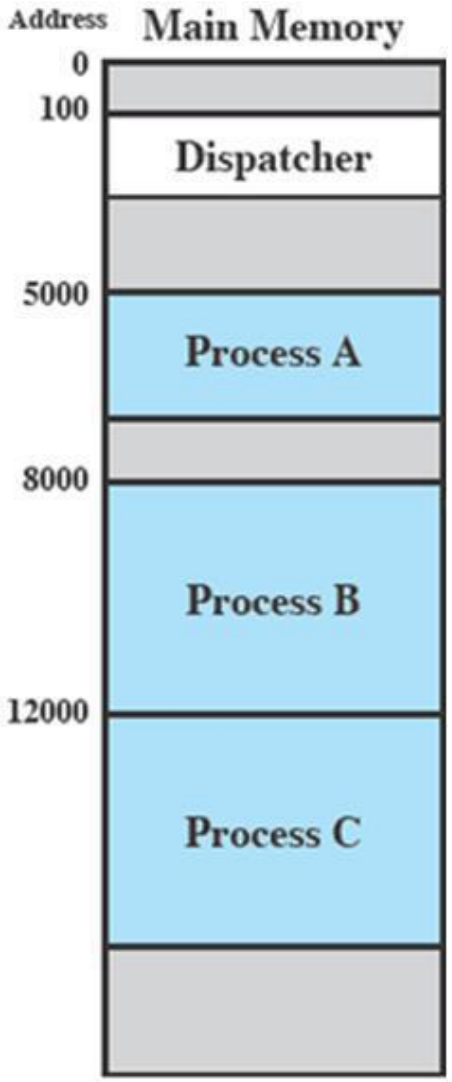
- The behavior of an individual process can be characterized by listing the **sequence of instructions that are executed** for that process.

- This list is called a ***Trace*** of the process

Process Execution

- Consider three processes being executed
- All are in memory
- plus the dispatcher
  - -a small program which switches the processor from one process to another

•Lets ignore virtual memory



Trace of the Process- from the processes point of view

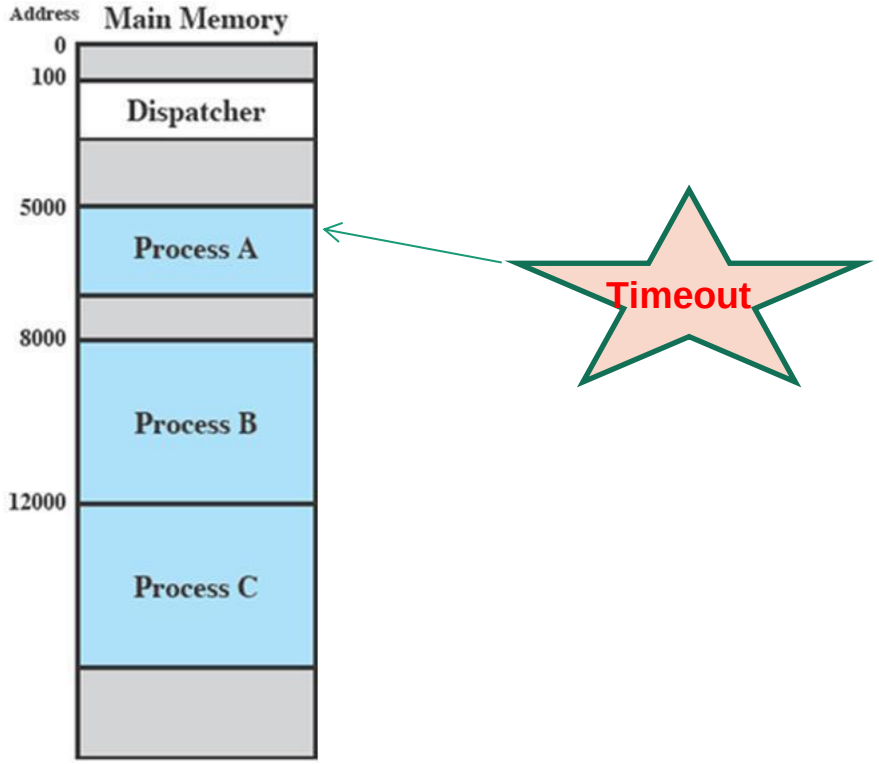
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

(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

Trace of the Process- from the Processors' point of view



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

Somaiya TRUST

Process Creation

- When a new process is to be added to the set of processes currently being managed-
- OS builds the data structures that are used to manage the process (i.e PCB) and allocates address space in main memory to the process→ creation of new process
- Events Leading to Process Creation:

1. **Submission of a new job** in batch processing
2. **New user attempts to logon** in interactive environment
3. **Created by OS on behalf of an application** to provide a service
   - If user application requests printing of a file OS creates a process that will manage printing
4. **Spawned by existing process- Created by OS at the explicit request of another process**
   - Requesting process referred a Parent process and spawned process as Child process

Process Termination

- Events Leading to Process Termination:

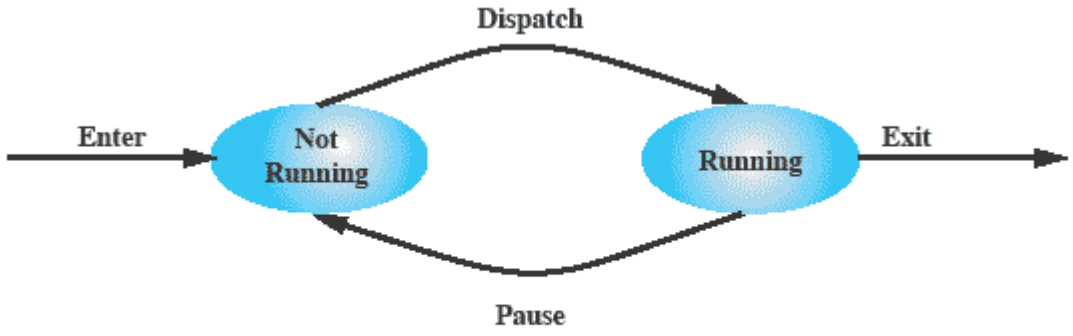
1. User specified
   logout or termination of application
2. Normal Completion
3. Time Limit exceeded
   Process has run longer than specified time limit
4. Memory unavailable
   Process requires more memory than available
5. Bounds violation
   Attempts to access memory location that is not allowed to access
6. Protection Error
   Attempts to use a resource (e.g. file) that is not allowed to use or improper access (write a read only file)
7. Arithmetic error
   attempt of prohibited computation- division by zero

Process Termination

8. Time overrun
   process has waited longer than specified maximum for certain event to occur
9. I/O failure
   an error during i/p or o/p; inability to find a file; failure to read/write after number of tries
10. Invalid instruction
    attempt to execute non existent instruction
    (branching into data area and attempting to execute data)
11. Privileged instruction
    Process attempts to execute an instruction reserved for OS
12. Data misuse
    data of wrong type or not initialized
13. Operator or OS intervention
    operator or OS terminates the process (e.g. in case of deadlock)
14. Parent termination
15. Parent request
    - Parent process has authority to terminate any of its offsprings (child processes)

Two-State Process Model

- Process may be in one of two states
  - –Running
  - –Not-running
- When a new process is created the OS enters the process into the system in Not-running state
- PCB is created and OS knows that the process exists; waiting for opportunity to execute



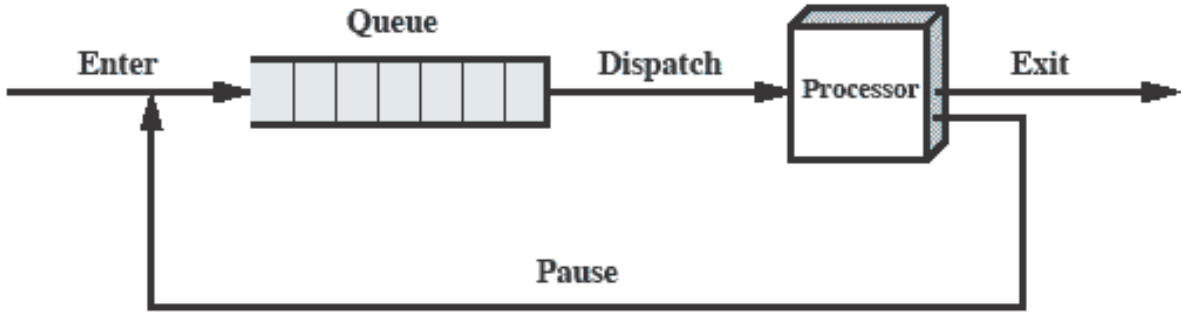
(a) State transition diagram

Two-State Process Model

- From time to time the currently running process will be interrupted and the dispatcher will select some other process to run
- → Process in Running state will move to Not-running state and one of the process from the Not-running state moves to running state

- Not-running processes are kept in a **single queue, in which each entry is a pointer to the PCB**

Two-State Process Model: Queuing Diagram



(b) Queuing diagram

- Processes moved by the dispatcher to the CPU then back to the queue until the task is competed

Five-State Process Model

- 2 state process model- inadequate
  - Would have been effective only if all processes were always ready to execute
- Some processes in not-running state are ready to execute, while others are blocked; waiting for completion of I/O operation
- **Dispatcher should scan the list looking for the process that is not blocked** and has been in the queue the longest

- Split the Not-Running state into 2 states: **Ready** and **Blocked**

Five-State Process Model

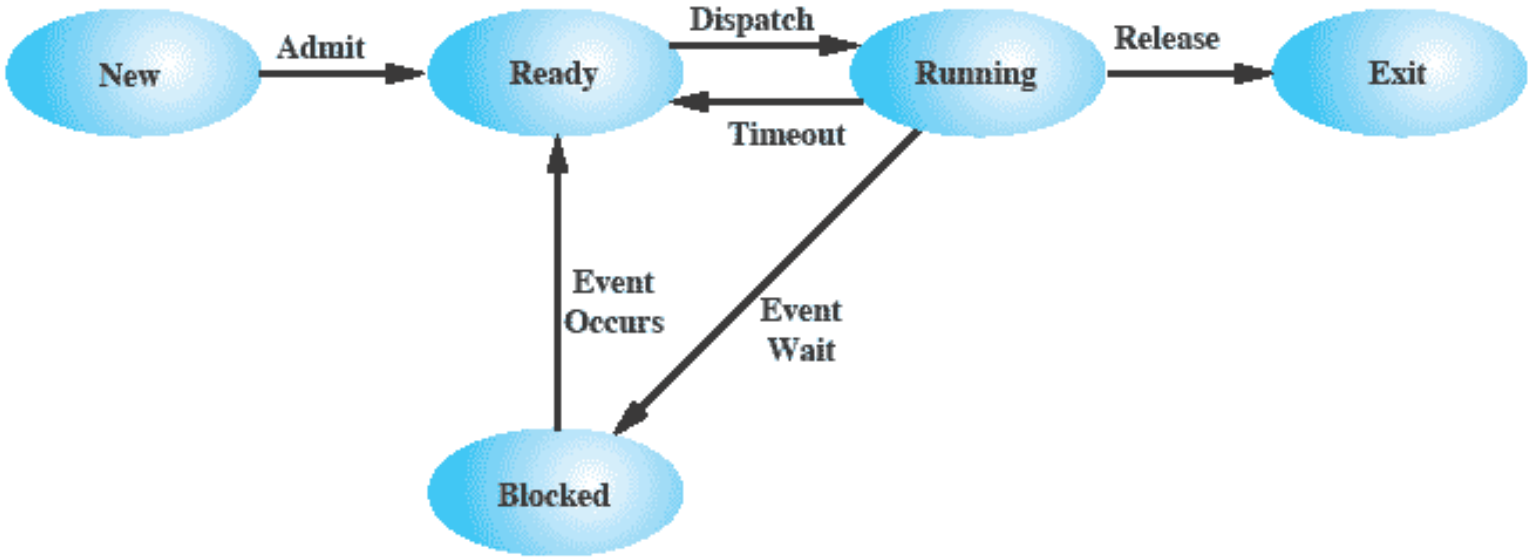
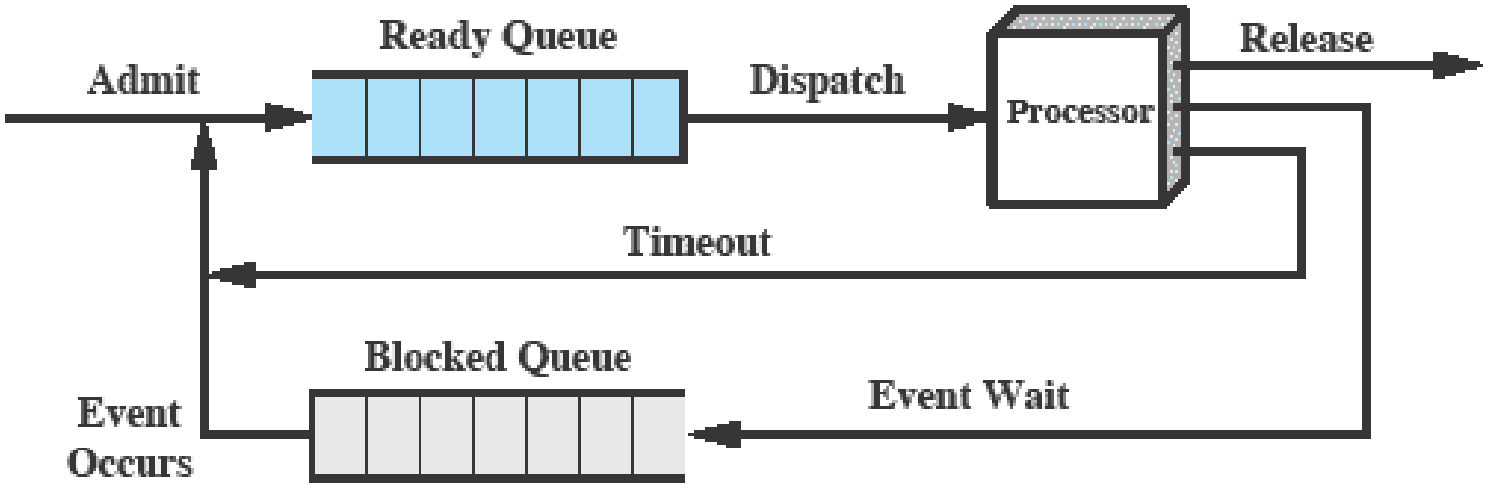


Figure 3.6 Five-State Process Model

Five-State Process Model: 5 states

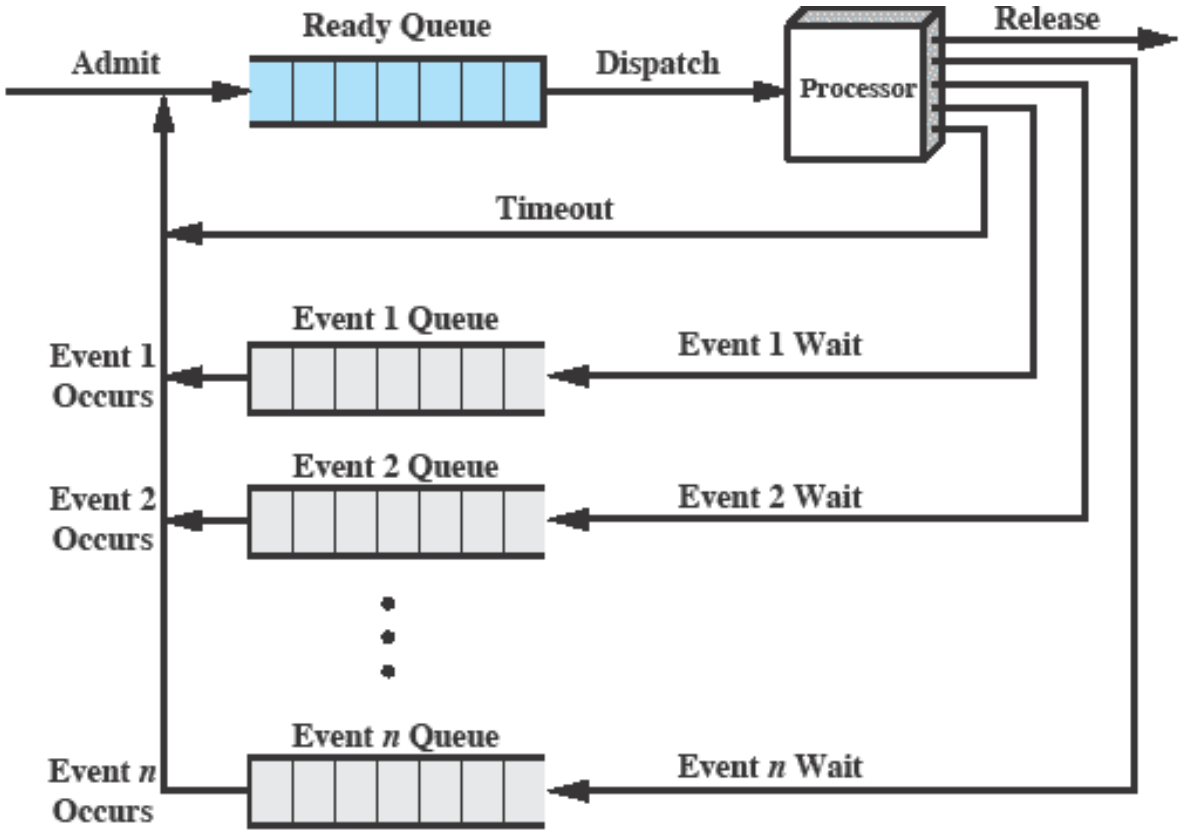
- Running- process that is currently being executed
- Ready- process that is prepared to execute when given the opportunity
- Blocked- process that cannot execute until some event occurs
- New- process that has just been created but not yet admitted / loaded in main memory; its PCB is created
- Exit- process that has been released from the pool of executable processes by the OS (halted or aborted)

Five-State Process Model: 2 Queues



(a) Single blocked queue

Five-State Process Model: Multiple blocked Queues

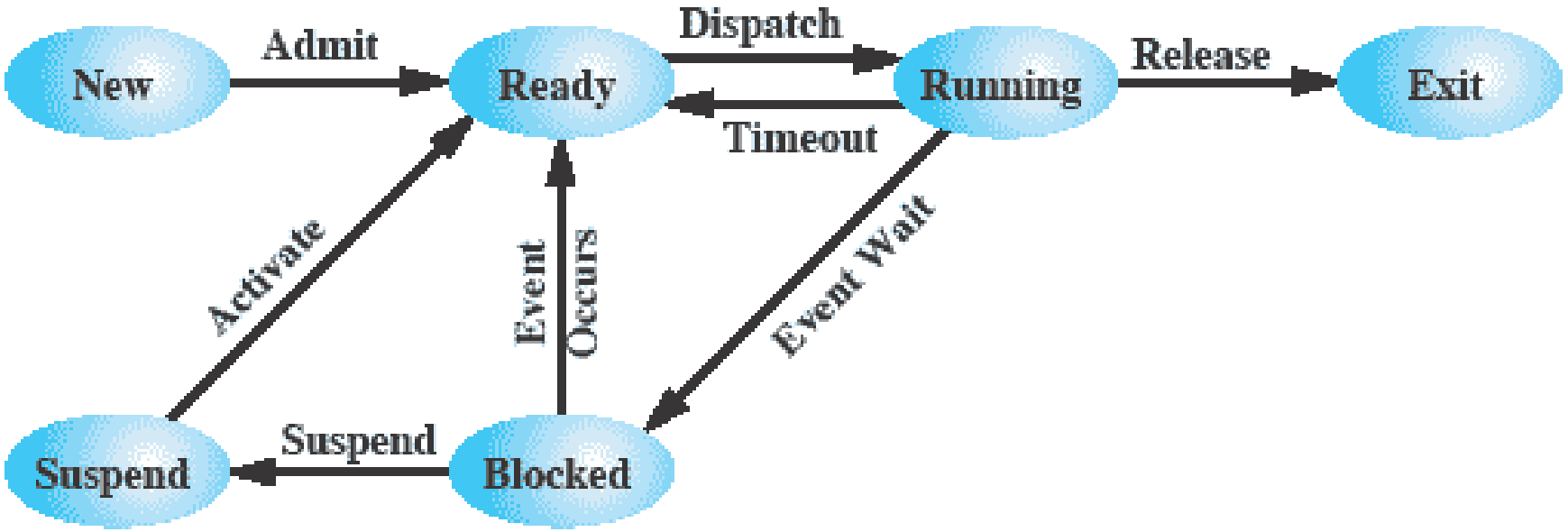


(b) Multiple blocked queues

Suspended Processes

- Process to be executed must be residing in main memory
- Processor is faster than I/O so all processes could be waiting for I/O
  - Processor remaining idle most of the time (ex)
- Solution:
  - A. Expand Main memory to accommodate more processes
  - B. Swap the processes to disk to free up more memory
- Swapping: involves moving part or all of a process from main memory to disk
- When no process in the main memory is in the Ready state, OS swaps the blocked process out onto the disk into a suspend queue
  - (queue of processes that have been temporarily moved out the memory or suspended)
- Space freed is used to bring in another process (either newly admitted process or previously suspended process)
- Swapping is an I/O operation

Process Model: One Suspend State



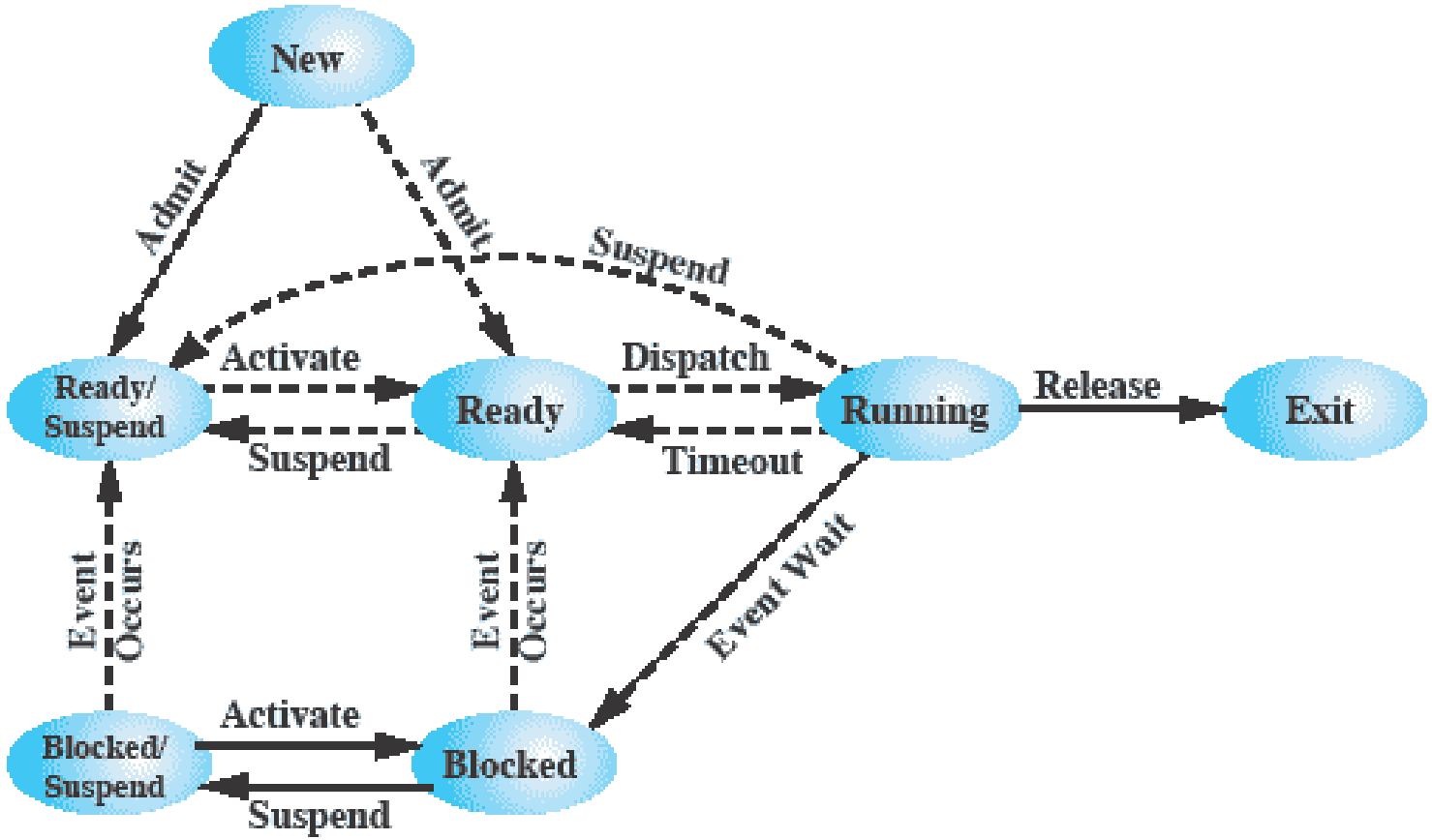
(a) With One Suspend State

Suspended Processes

- Two possibilities about the Process
  - Process is waiting on an event (blocked or not)
  - Process has been swapped out of main memory (suspended or not)

- Two new states
  - **Blocked/Suspend**: process is in secondary memory and awaiting an event
  - **Ready/Suspend**: process is in secondary memory and available for execution as soon as loaded into main memory

Process Model: Two Suspend State



(b) With Two Suspend States

Suspended Processes: State Transitions

- Blocked → Blocked/Suspend
- Blocked/Suspend → Ready/Suspend
- Ready/Suspend → Ready
- Ready → Ready/Suspend
  - If large block of memory is required to be freed up
  - Low priority ready process and high priority blocked process
- New → Ready/Suspend; New → Ready
  - Preferred to create a process as late as possible and reduce overhead
- Blocked/Suspend → Blocked
  - Low priority process in Ready/suspend state & high priority process in Blocked/suspend state; if process knows that the event is going to occur soon
- Running → Ready/Suspend
  - Process transition fom Running to ready state on timeout or process premption;
  - direct transition of prempted process to ready-suspend state to free space
- Various → Exit (Running →Exit; All →Exit)

Reasons for Processes Suspension

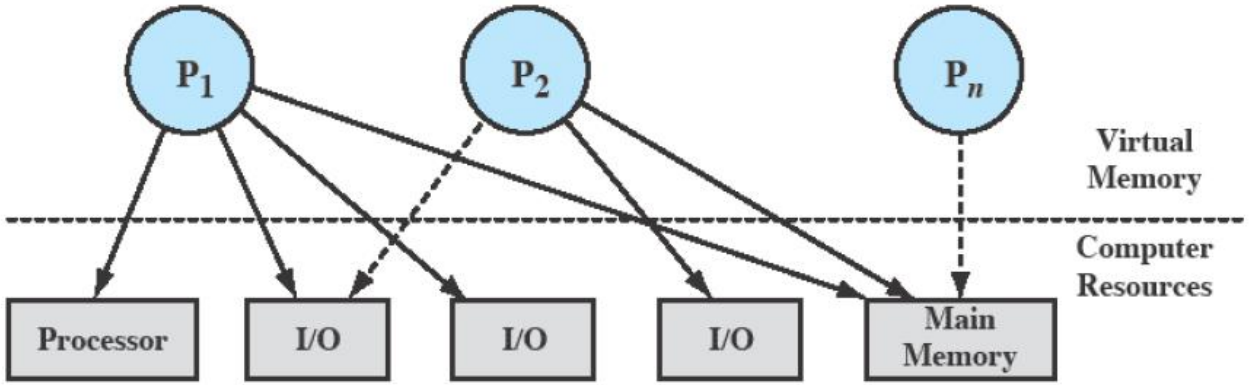
Reason	Comment
Swapping	The OS needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS Reason	OS may suspend: background utility process, or process suspected of causing a problem.
Interactive User Request	e.g. debugging or in connection with the use of a resource.
Timing	A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next interval.
Parent Process Request	A parent process may wish to suspend execution of a descendent to examine or modify or to coordinate the activity of various descendants.

Characteristics of Suspended Processes

- Process is not immediately available for execution
- Process may or may not be waiting for an event
- Occurrence of blocking event does not enable a process in blocked-suspend state for execution
- Process was placed in suspended state by an agent
  - Either itself or parent process or the OS
- Process is not removed from the suspend state until the suspending agent explicitly orders removal

OS Control Structure

- During the execution, every process needs access to certain resources e.g. processor, memory, I/O devices etc.
- OS manages use of system resources by processes



OS Control Structure

- What information does OS need to control processes and manage resources for them?
  - Current status of each process and resources
  - OS constructs and maintains tables of information about each entity that it manages
  - Tables maintained by OS:
    - Memory table
    - I/O Table
    - File Table
    - Process Table

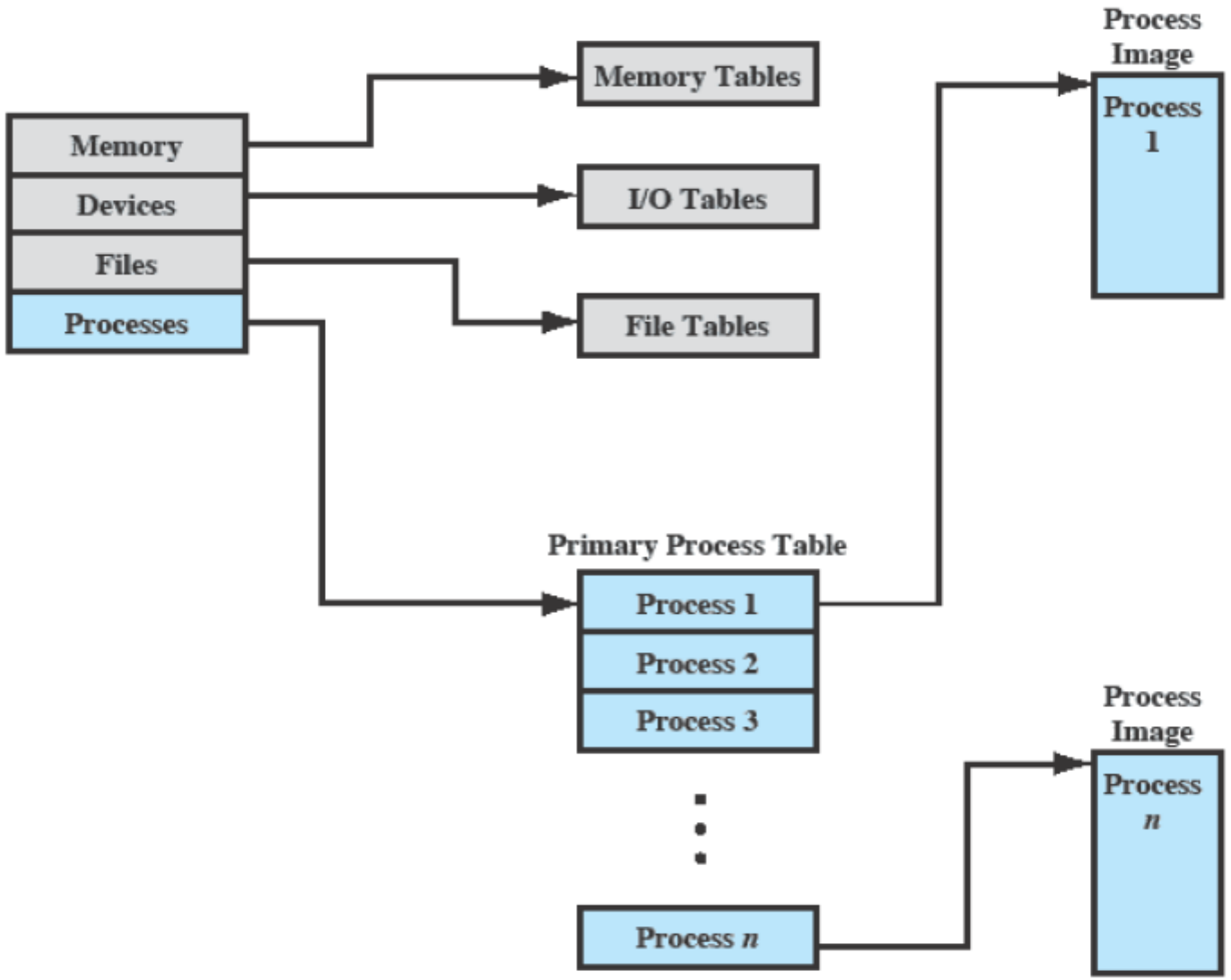
OS Control Structure: Memory Tables

- Used to keep track of main (real) and secondary (virtual) memory
  - Some main memory is reserved for OS; remainder is available for processes
  - Processes are maintained on secndary memory using virtual memory or swapping
  - Memory Tables must include following information:
    - Allocation of main memory to processes
    - Allocation of secondary memory to processes
    - Protection attributes of blocks of main or virtual memory (which process may access certain shared memory reasons)
    - Any information needed to manage virtual memory

OS Control Structure: I/O and File Tables

- I/O tables: Used by OS to manage I/O devices and channels
  - Whether the I/O device is available or assigned?
  - OS needs to know status of I/O operation
  - Location in main memory being used as source/destination of I/O transfer
- File Tables
  - Provide information about
    - existence of files,
    - their location in secondary memory,
    - current status and other attributes
  - Maintained and used by File management system in some systems

OS Control Tables



Process Control Structure

- In order to manage and control a process what are the things that OS must know?
- –Location of Process
- –Processor state information
- –Attributes of process that are necessary for its management (Process ID, state..)
- *Collection of attributes is called PCB*
- *Collection of program, data, stack and attributes is called "process image"*

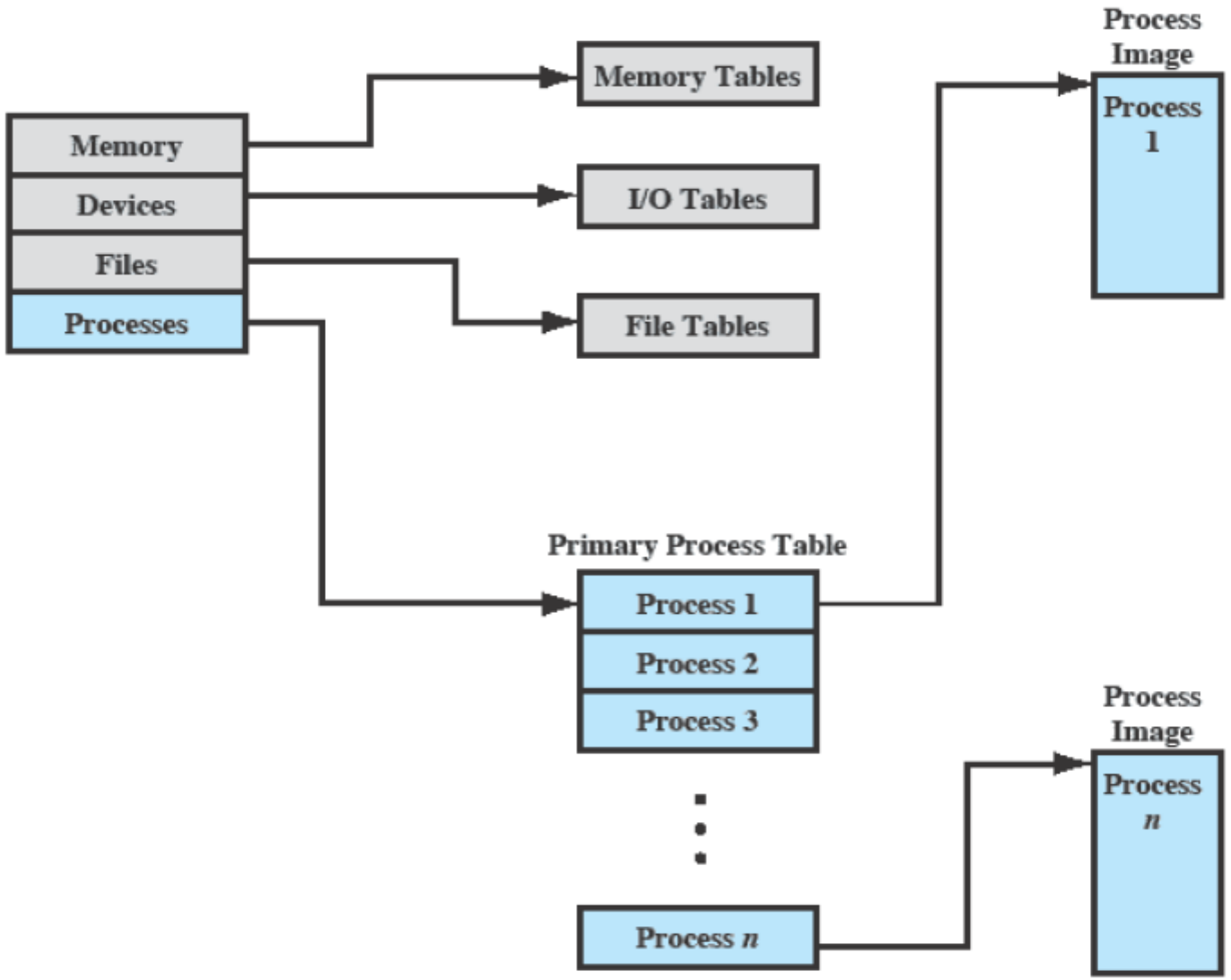
Process Control Structure

- Elements of process image
- –User data
- –User program
- –System stack
- –PCB
- Process image is maintained as contiguous block in the secondary memory
  - –Small portion in main memory
- To execute the process, entire process image must be loaded into main memory or virtual memory

Process Control Structure

- Modern OS use paging hardware that allows non-contiguous physical memory to support partially resident processes
  - Portion of process image may be in main memory, and remainder in secondary
- Process tables must show the location of each page of each process image
  - Primary process table has one entry for each process → contains pointer to the process image
- Elements of Process Image: User Data, User Stack area, and programs that may be modified
  - User Program- program to be executed
  - System stack- LIFO system stack used to store parameters and calling addresses for procedure and system calls
  - PCB

OS Control Tables



Process Attributes

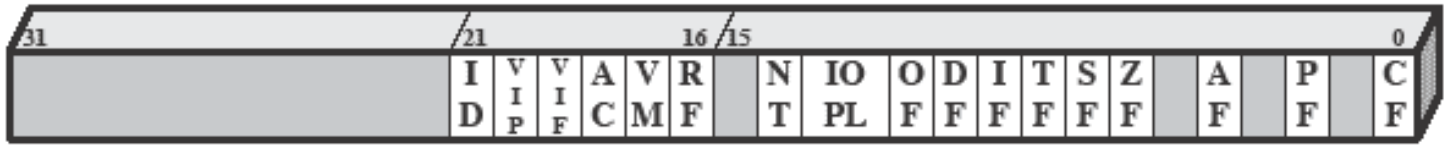
- Information required by OS for managing each process can be grouped into three general categories:

1. Process Identification
2. Processor State information
3. Process Control information

Process Identification

- Each process is assigned a unique numeric identifier
  - Identifier of Process (PID)
  - Identifier of the Process that created this process (PPID)
  - User identifier
  - -can be used by other tables to cross reference process tables
- Processor State information: consists contents of processor registers
- User visible registers: that might be referenced by means of machine language (8-32 registers; RISC >100)
- Control and status registers: processor registers to control its operation
  - Program Counter
  - Condition cods
  - Status information
    - PSW, EFLAGS register
- Stack Pointers: stack is used to store parameters and calling addresses for procedure and system calls; SP points to Top of the Stack

Processor State information- 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 3.12 Pentium II EFLAGS Register

Process Control Information

- **Scheduling and State information**
  - Process State (running/ready..)
  - Priority (default, current, highest allowable)
  - Scheduling related information (scheduling algorithm, waiting time, execution time..
  - Event- identity of event the process is awaiting
- **Data structuring**
  - A process may be linked to other process in a queue, ring or some other structure
    - All waiting processes of a particular priority may be linked in a queue
  - PCB contains pointers to other processes to support these structures

Process Control Information

- **Inter-process Communication**
  - Flags, signals, messages for communication between two independent processes; this info may be maintained in PCB
- **Process privileges**
  - Memory locations that may be accessed; types of instructions that may be executed; privileges applicable to system utilities/services
- **Memory Management**
  - Pointers to segment &/or page tables that describe virtual memory assigned to process
- **Resource Ownership and Utilization**
  - resources controlled by the process- Opened files, etc
  - history of utilization of the processor and other resources- reqd by scheduler

Structure of Process Images in Virtual Memory

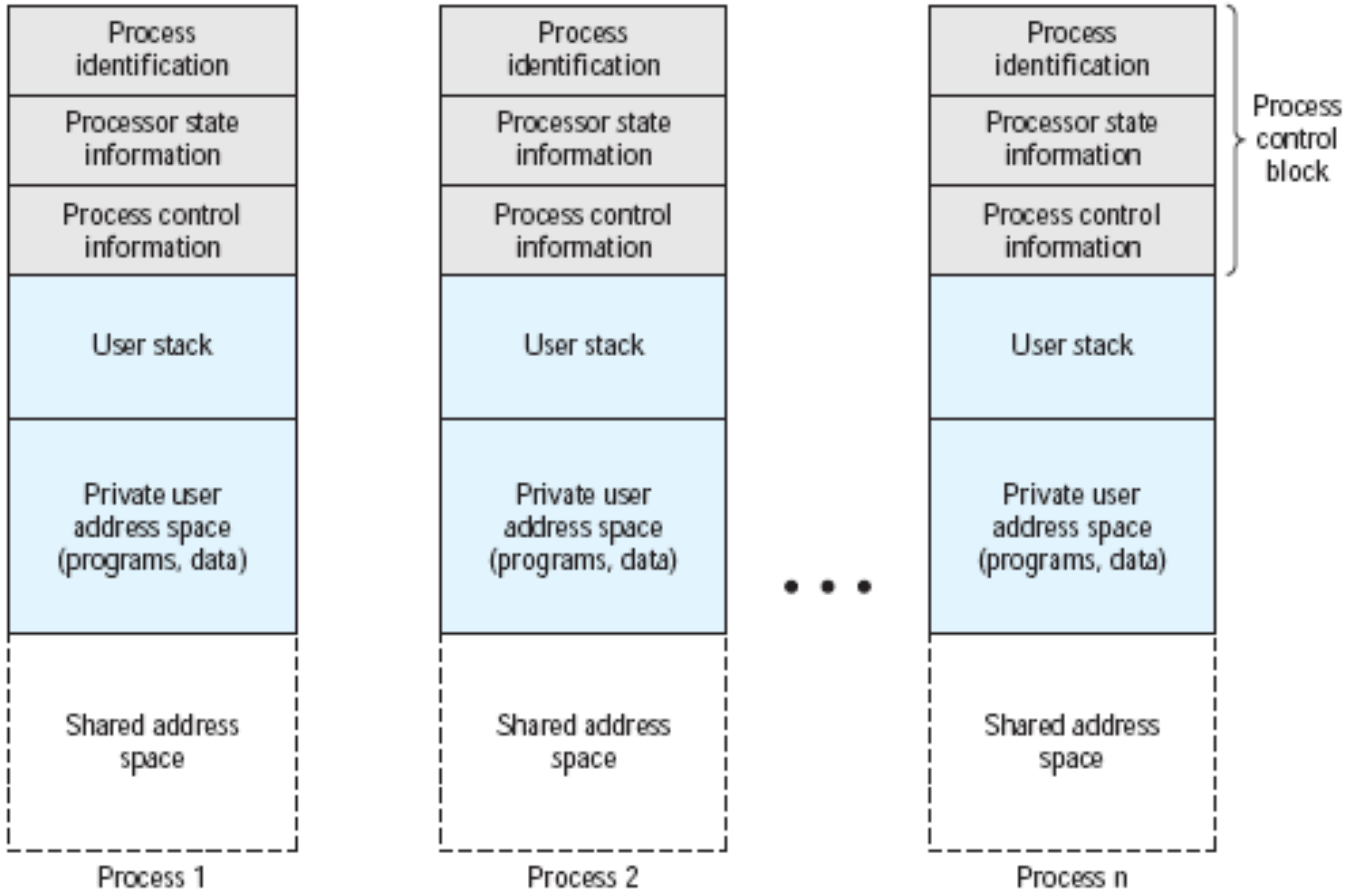


Figure 3.13 User Processes in Virtual Memory

Role of the Process Control Block

- Most important data structure in an OS; contains all information about a process that is needed by OS
- – set of PCBs define the state of the OS

- Process Control Block requires protection
  - faulty routine could cause damage to the block, destroying the OS's ability to manage the process
  - Any design change to the block could affect many modules of the OS

Modes of Execution

- Most processors support at least two modes of execution
  - Purpose: protecting OS and key OS Tables/data structures from interference by user programs
- User mode
- –Less-privileged mode; User programs typically execute in this mode
- System mode (control mode/ kernel mode/ protected mode)
- –More-privileged mode
  - Instructions executed in more privileged manner; reading or altering control register, primitive I/O instructions, MM
- –Kernel of the operating system

Modes of Execution..

- How does the processor know in which mode it is to be executing? And how does it change?
- Typically a flag bit in the PSW indicates mode of execution
- changed in response to certain events
- When a user makes a call to an OS service or when an interrupt triggers execution of an OS routine, the mode is set to the kernel mode- *CHM instruction*
  - upon return from the service to the user process, the mode is set to user mode

Typical functions of OS Kernel

- Process Management
- –Process Creation and Termination
- –Process Scheduling and Dispatching
- –Process Switching
- –Process Synchronization and IPC
- –Management of PCB
- Memory Management
- –Allocation of Address space
- –Swapping
- –Page and Segment management
- I/O Management
- –Buffer management
- –Allocation of Channels and devices to processes
- Support functions
- –Interrupt Handling
- –Accounting
- –Monitoring

Process Creation

Once the OS decides to create a new process it:

- –Assigns a unique process identifier
  - Entry in primary process table
- –Allocates space for the process
  - Space for including process image
- –Initializes process control block
  - State info initialized with most zero; except PC and SP
  - Control info based on standard defaults + attributes
- –Sets up appropriate linkages
  - Scheduling queue as linked list
- –Creates or expands other data structures
  - Accounting file for each process

Process Switching

- At times running process is interrupted and OS assigns another process to running state and turns control over to that process →Process Switching

- Design issues raised
  - What events trigger a process switch?
  - distinction between mode switching and process switching
  - What must the OS do to the various data structures under its control to achieve a process switch?

When to switch processes

- A process switch may occur any time that the OS has gained control from the currently running process. Possible events giving OS control are:

Mechanism	Cause	Use
Interrupt (Ex.Clock interrupt, I/O interrupt, Memory Fault)	External to and independent of 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 (system call)	Call to an operating system function

System Interrupts, Trap and Supervisor call

1. Clock interrupt
2. I/O interrupt
3. Memory fault

- Trap: Used by OS to determine if error or exception condition is fatal
  - If so, currently running process is moved to Exit sate and process switch occurs
  - If not, action depends on nature of error- OS may attempt some recovery procedure or notify user →Process switch or resumption of process
- Supervisor call
  - During execution, an instruction requests an I/O operation e.g file open
  - This call results in transfer to a routine that is part of OS code
  - → system call may place user program in blocked state

Steps involved in full Process Switch

1. Save context of processor, including program counter and other registers
2. Update the PCB of the process that is currently in the Running state
   - (changing current state to other; ready; blocked; ready/suspend or Exit)
3. Move PCB of this process to appropriate queue
   - ready; blocked; ready/suspend
4. Select another process for execution
5. Update the PCB of the selected process
6. Update memory-management data structures
   - Required for how address translation is managed
7. Restore context of the selected process

Mode Switching

- Mode switch may occur without changing the state of the process that is currently in the Running state
  - Occurrence of interrupt not necessarily means a process switch
  - Save the processor state information when interrupt occurs and restore it after interrupt handler has executed, and control is returned to the program
  - Saving and restoring performed in hardware
- Involves little overhead associated with context saving and restoral
- Process switch involves a state change and thus requires more efforts than a mode switch