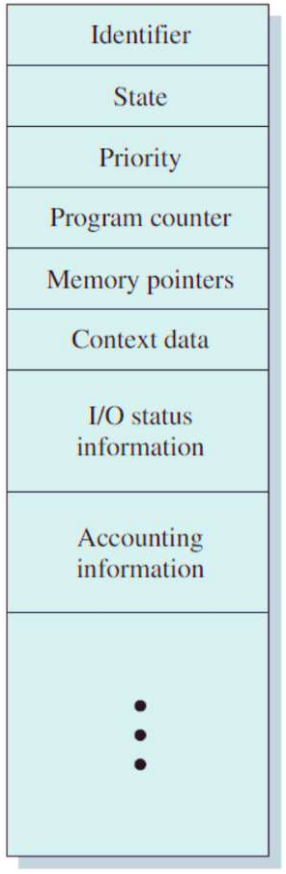
CSE 306 Operating Systems
Processes

YoungMin Kwon

What is a Process

- A process consists of
  - Program code
  - Set of data associated with the code
  - Process Control Block (PCB)
    - Process Id
    - State (running, ready, blocked…)
    - Program counter
    - Memory pointers
    - Context data (registers)
    - I/O status (I/O devices assigned to, files in use…)
    - Accounting info. (processor time, time limits…)

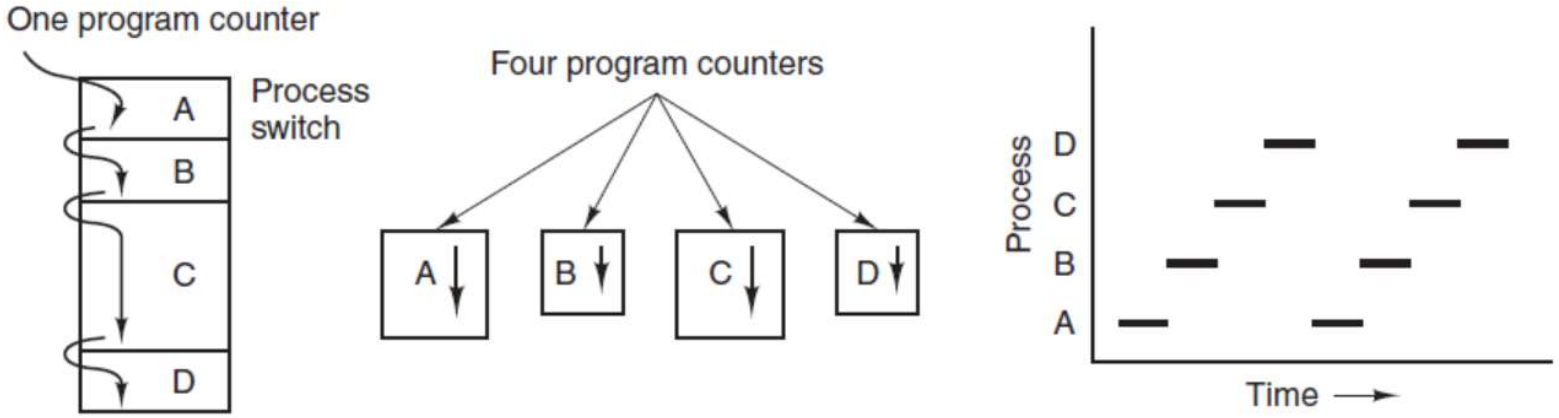


Some Entries of PCB

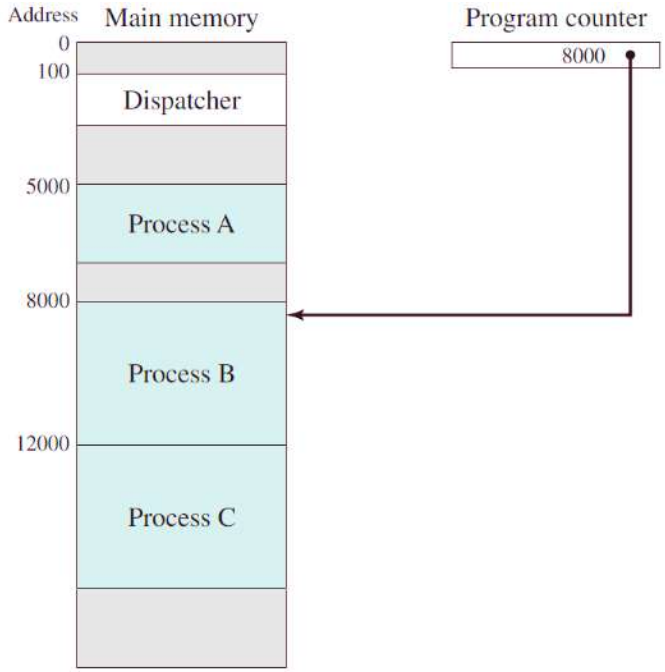
Process management	**Memory management**	**File management**
Registers	Pointer to text segment info	Root directory
Program counter	Pointer to data segment info	Working directory
Program status word	Pointer to stack segment info	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Execution of Programs

- Processor's view
  - Executions of instructions from multiple programs
- Process's view
  - Executions of a sequence of instructions within that program



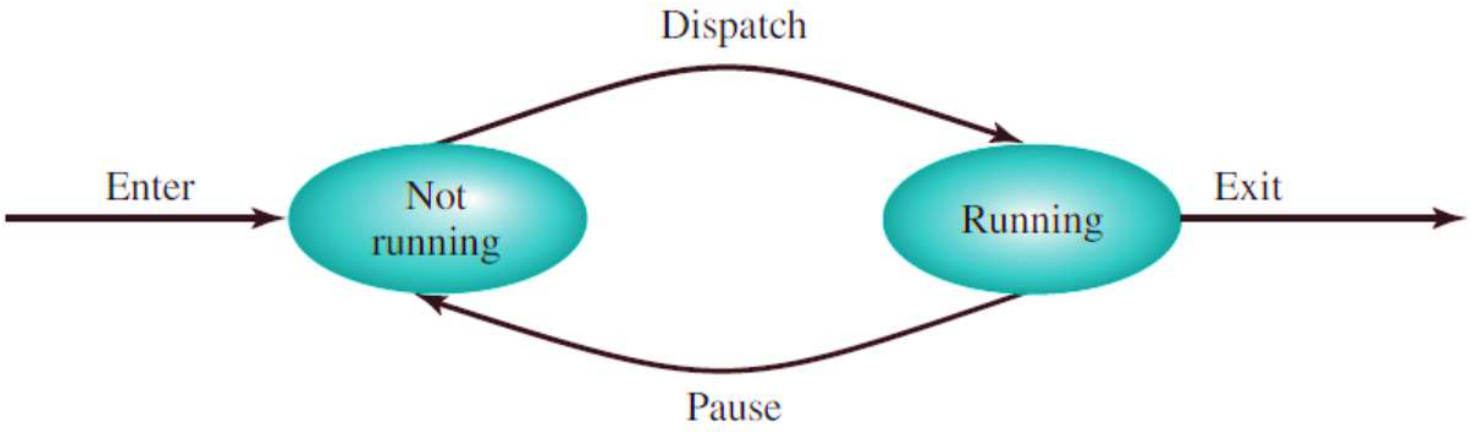
Execution of Programs



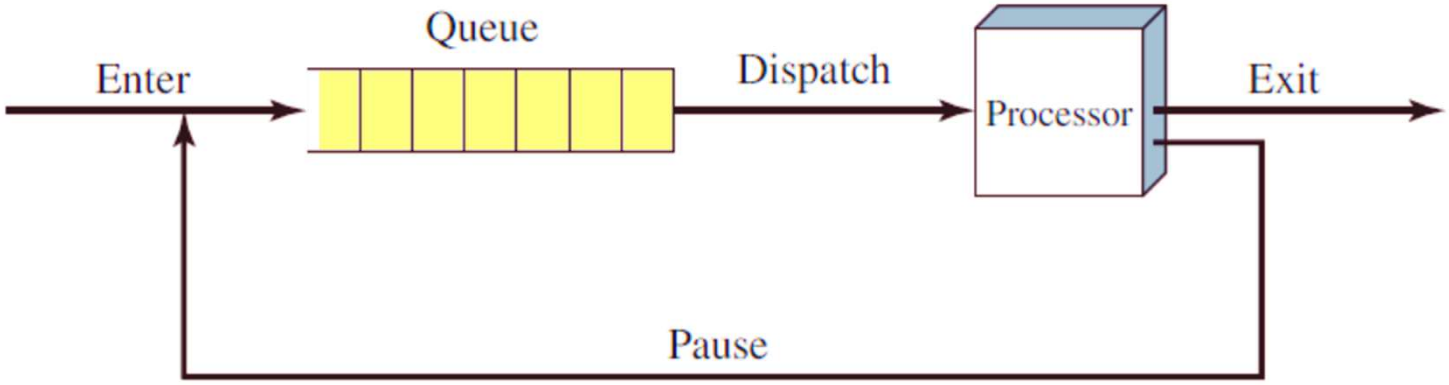
(a) Trace of process A	(b) Trace of process B	(c) Trace of process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

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		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	----------Time-out	
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
		----------Time-out	

A Two-State Process Model



(a) State transition diagram



(b) Queueing diagram

A Two-State Process Model

- Process Creation
  - OS builds the data structures to manage the process
  - Allocates address space for the process

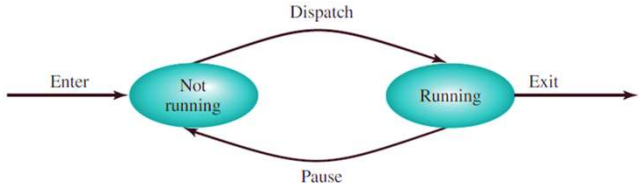
- Reasons for Process Creation
  - New Batch job
  - Interactive log-on
  - Created by OS to provide service
  - Spawned by existing process

A Two-State Process Model

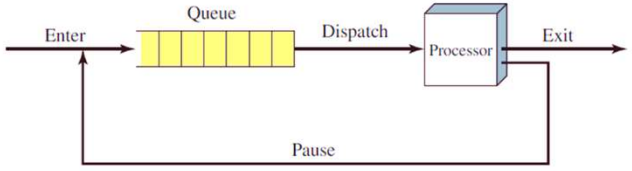
- Process Termination
  - Halt instruction generates an interrupt to alert the OS
  - Action of a user
    - log off, turn off a terminal, quit an application
    - Result in a service request to OS to terminate the process
  - Errors or Faults

A Five-State Model

- A problem with the two-state model
  - Some processes in Not-running sate are blocked, waiting for an I/O to complete
  - The dispatcher has to scan the queue looking for the process that is
    - Not blocked
    - Has been in the queue the longest



(a) State transition diagram

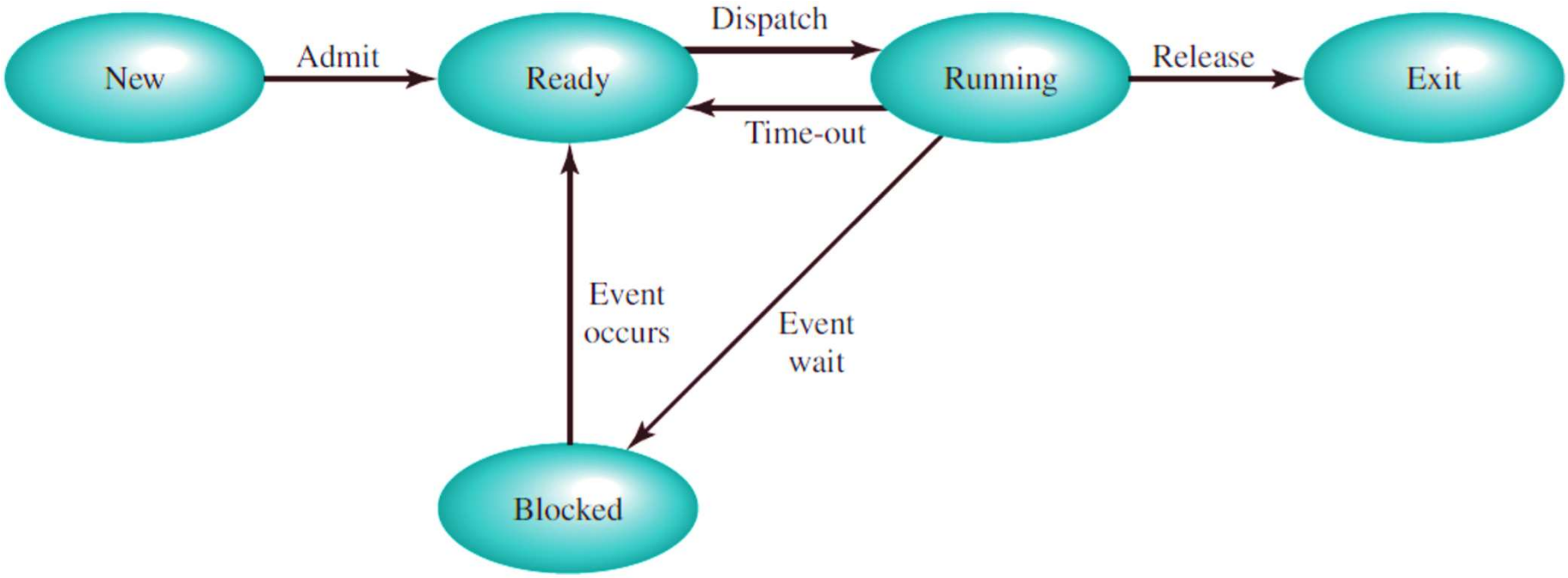


(b) Queueing diagram

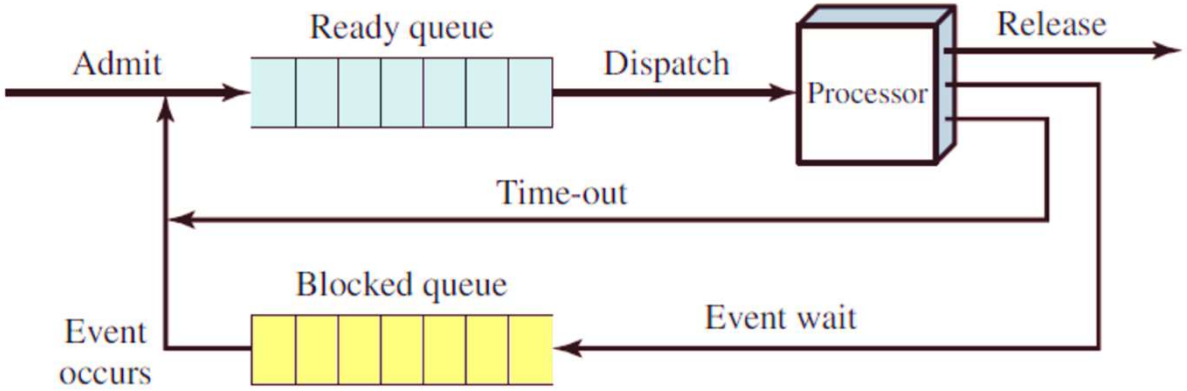
A Five-State Model

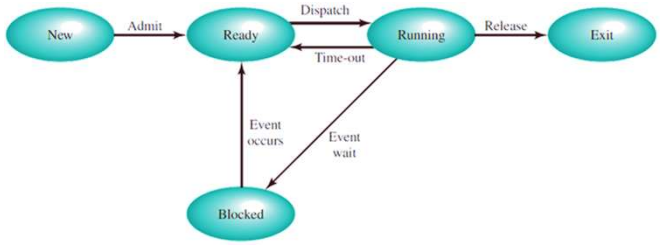
- Solution
  - Split the Not-Running state into Ready and Blocked states
- Five states
  - Running: the process is currently being executed
  - Ready: the process can execute, given the opportunity
  - Blocked/Waiting: the process cannot execute until some event occurs (I/O completion)
  - New: the PCB is created, but the process is not yet loaded into memory
  - Exit: the process has been released from the pool of executable processes

A Five-State Model



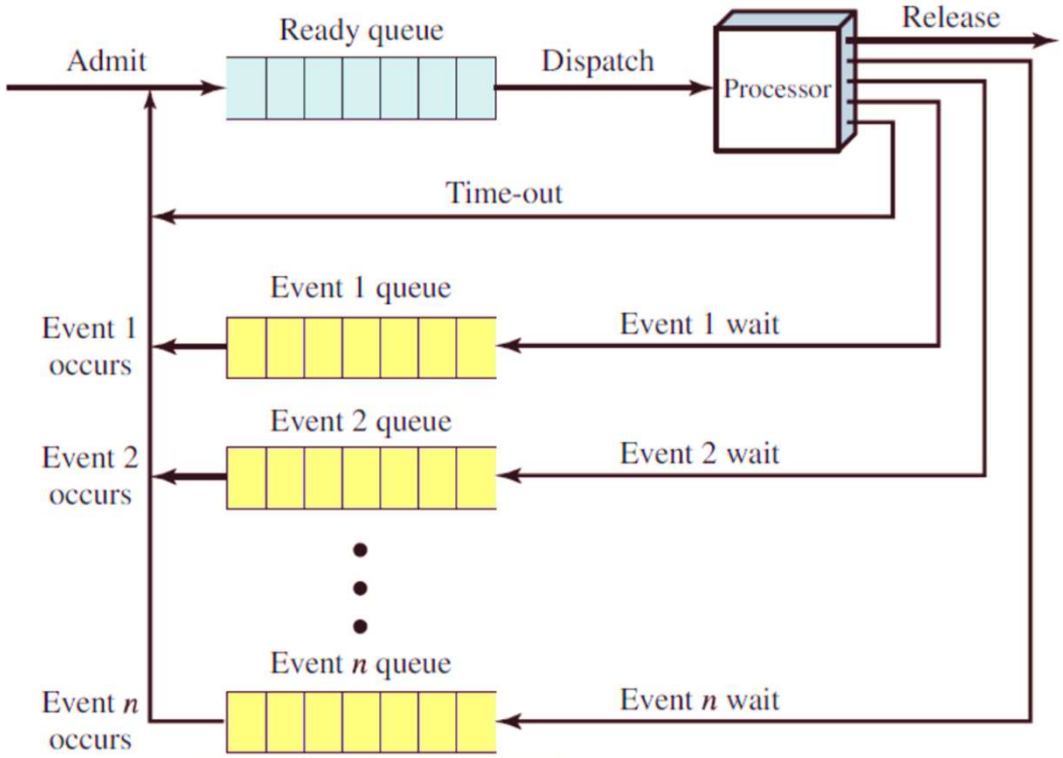
A Five-State Model

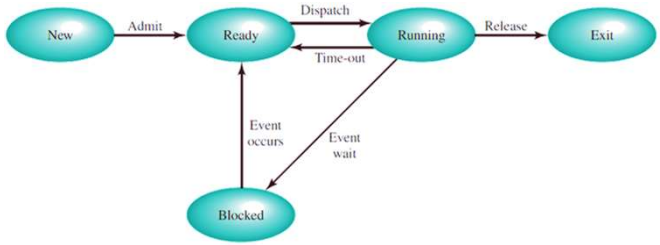




- New processes are placed in the ready queue
- The next process to run are chosen from the ready queue
- A running process can exit or be moved to either the ready queue or the blocked queue
- An event can move processes in the blocked queue waiting for the event to the ready queue

A Five-State Model

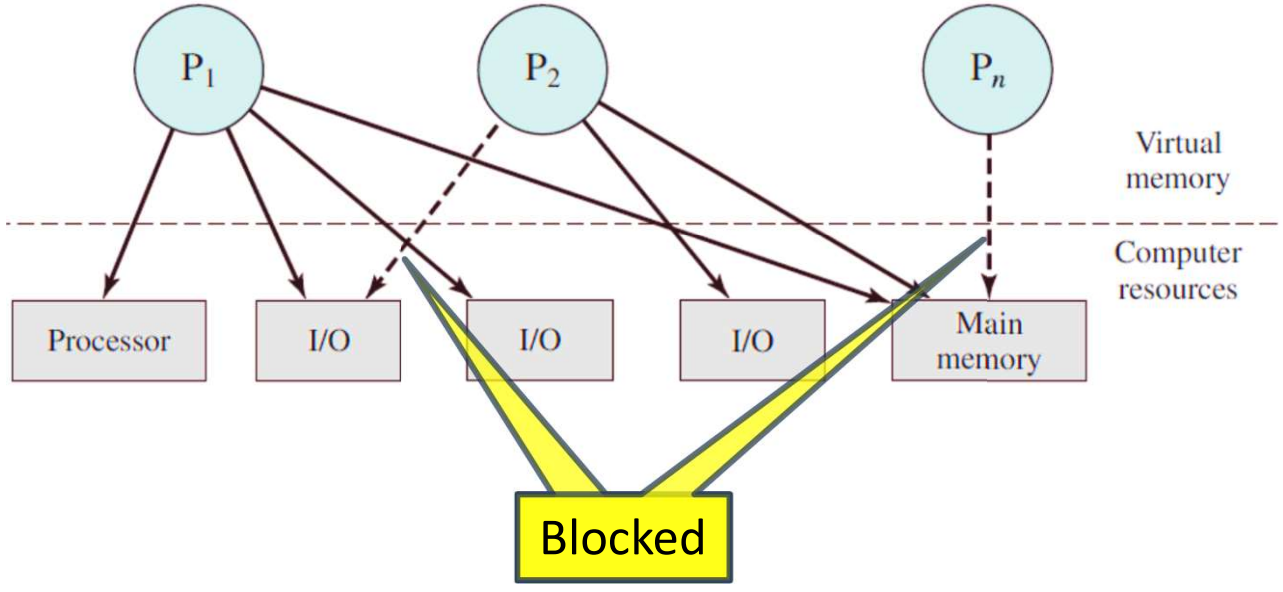




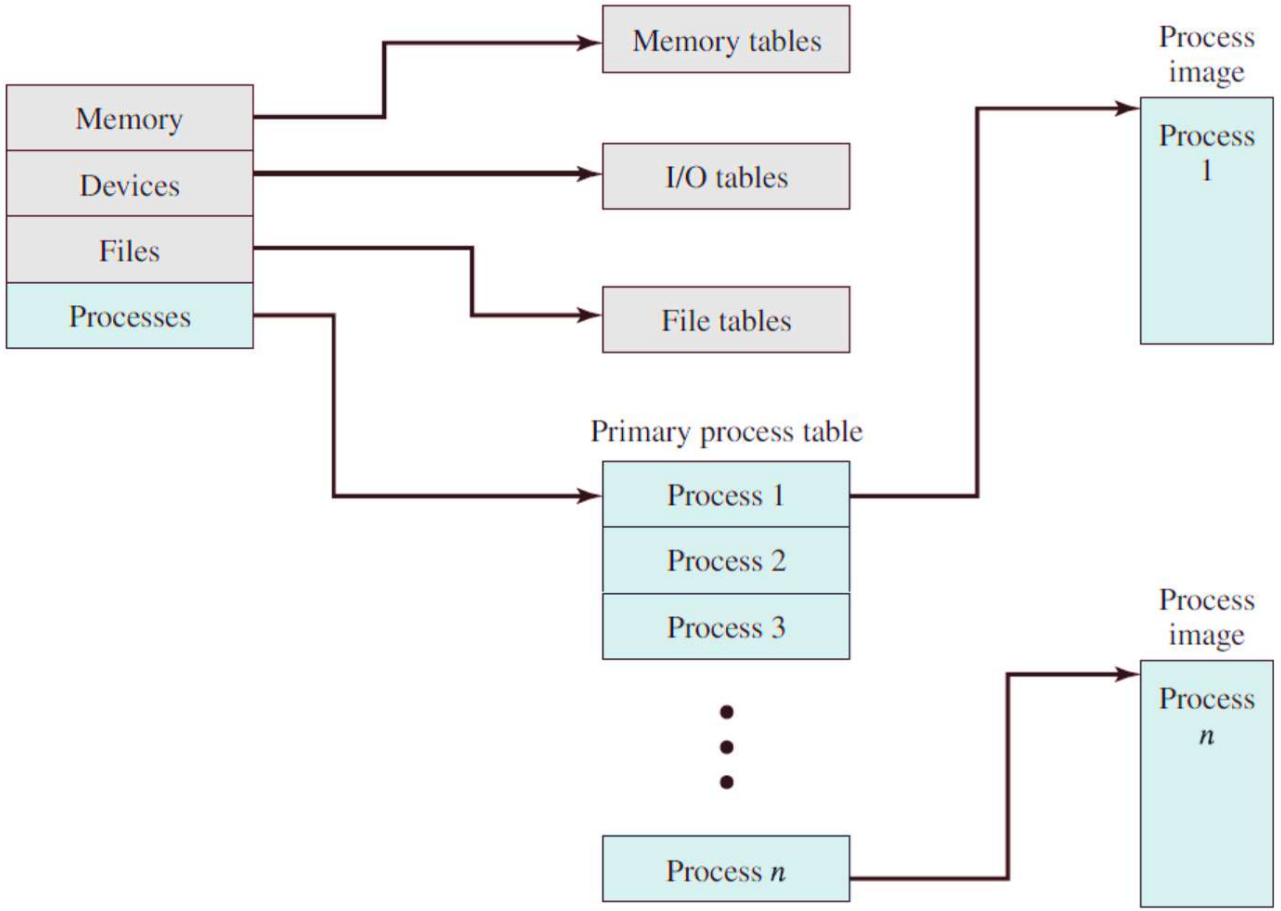
- Multiple blocked queues
  - Single blocked queue: OS has to scan the blocked queue for every event
  - The processes in a certain event queue are moved to the ready queue

Process Description

- OS manages the use of system resources by processes
  - What information does the OS need to control processes and manage resources?
  - Tables for memory, I/O, file, and process



Operating System Control Structures



Typical Functions of an OS Kernel

- Process management
  - Process creation and termination
  - Process scheduling and dispatching
  - Process switching
  - Process synchronization and inter-process communication
  - PCB management
- Memory management
  - Allocating address space to processes
  - Swapping
  - Page and segment management

Typical Functions of an OS Kernel

- I/O management
  - Buffer management
  - Allocation of I/O channels and devices to processes
- Support functions
  - Interrupt handling
  - Accounting
  - Monitoring

Process Control Structures

- Typical elements of a process image
  - User Data
    - The modifiable part of the user space: program data, user stack, and programs that may be modified
  - User Program
    - Instructions to be executed
  - Stack
    - Store parameters, return addresses for function calls
  - Process Control Block
    - Data needed by the OS to control process

Process Control Structures

Typical Elements of a PCB

- Process identification
  - IDs: PID of this and the parent process, User ID
- Processor status information
  - General purpose registers
  - Program Counter,
  - Program Status Word (PSW)
    - Condition flags: CF, ZF, OF…
    - Status information: interrupt enabled, current privilege level (CPL)…

Process Control Structures

Typical Elements of a PCB

- Process control information
  - Scheduling and State information
    - Process state (running, ready, blocked…)
    - Priority
    - Scheduler dependent information (processor time…)
    - Event (ID of the event the process is waiting)
  - Data structuring (e.g. link to other process in a queue)
  - Processor privilege
    - Memory access, types of instructions that can execute
  - Memory management (e.g. pointers to page tables)
  - Resource ownership and utilization
  - Interprocess communication

Process Control

- Modes of execution
  - User mode and Kernel mode (aka system mode, control mode)
  - Bits in the PSW indicates the mode of execution
    - CPL (Current Privilege Level): 0 for kernel mode, others for user mode
    - On interrupt, CPL is set to 0
    - On IRT (interrupt return), CPL is restored

Process Control: Process Creation

- Assign a unique PID to the new process
- Allocate space for the process
  - Process Image (text, data, stack, …) and PCB
- Initialize the PCB
  - PID, registers, PC, stack pointers, status (Ready), priority, inherited resources, …
- Set the appropriate linkage
  - Ready queue
- Create or expand other data structures
  - Accounting for billing, performance assessment, …

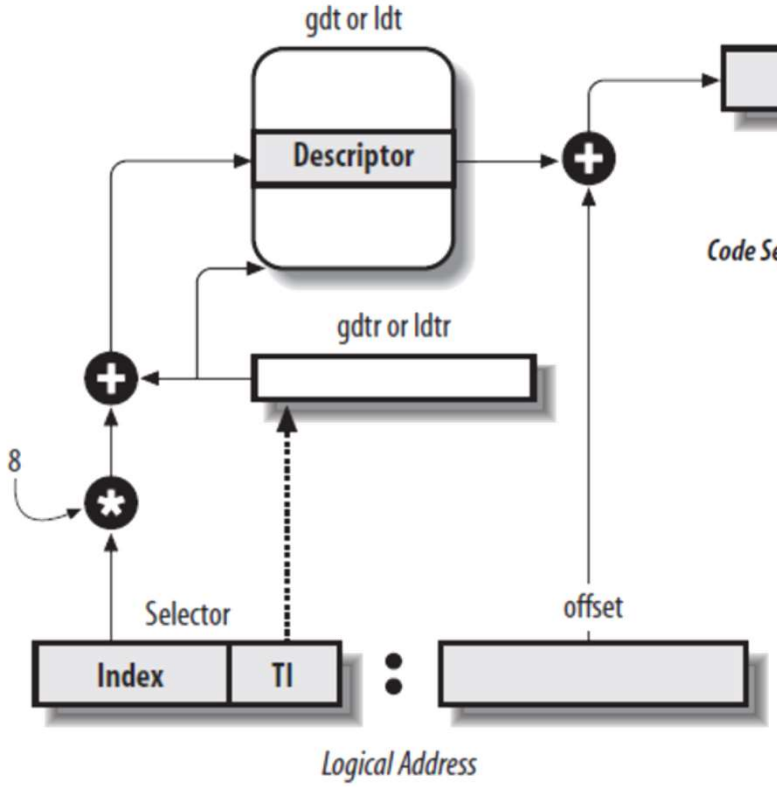
Process Control: Process Switching

- When to switch process
  - A process switch may occur any time when the OS has gained the control
- Mechanisms for interrupting the execution of a process
  - Interrupt: external to the current instruction
  - Trap: associated with the execution of the current instruction
  - Supervisor call: explicit request

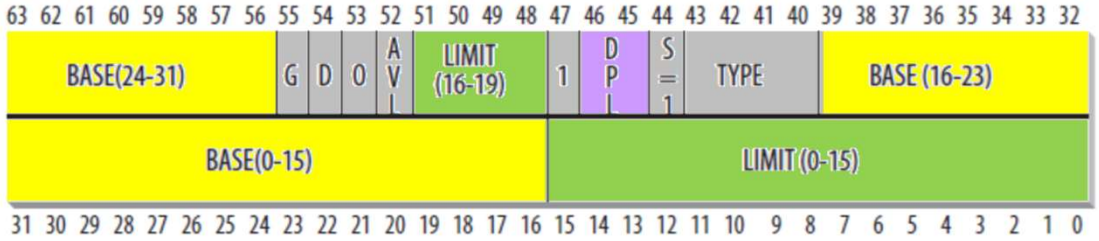
Process Control: Process Switching

- Mode switching on an interrupt
  - Sets the PC to the interrupt handler
  - Switches from user mode to kernel mode if necessary
    - Compare the CPL (Current Privilege Level) with DPL (Descriptor Privilege Level) of interrupt code
    - If CPL != DPL, load ss and esp from TSS and save the previous ss and esp to the new stack
  - Saves the PC, flags, and other registers
  - Mode switch does not necessarily mean process switch

Side Note: Address Translation

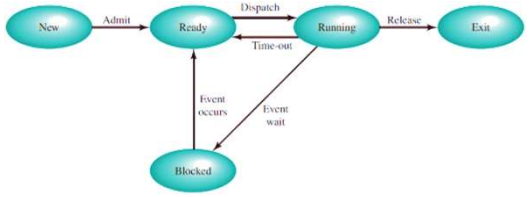


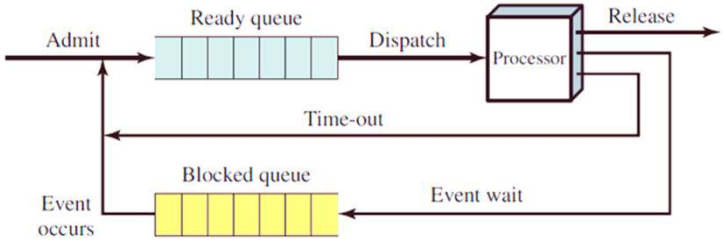
Code Segment Descriptor



Process Control: Change of Process State

- If the currently running process is to be moved to another state (Ready, Blocked)

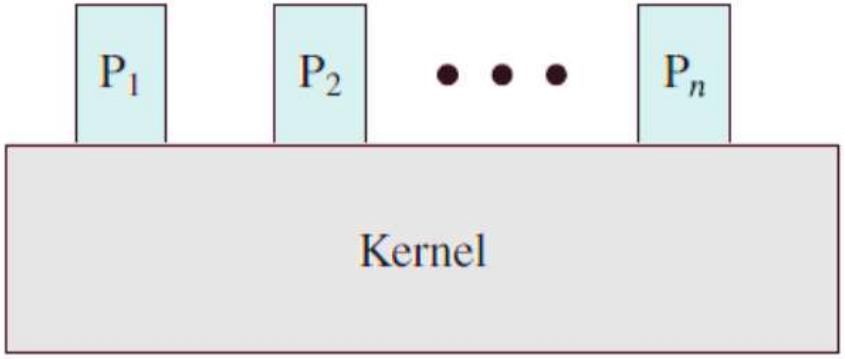




  - Save the context of the processor (PC, other registers)
  - Update the PCB (state, accounting info…)
  - Move the PCB of this process to the appropriate queue
  - Select another process to execute
  - Update the PCB of the process (state to running)
  - Update the memory management data structure
  - Restore the context of the processor

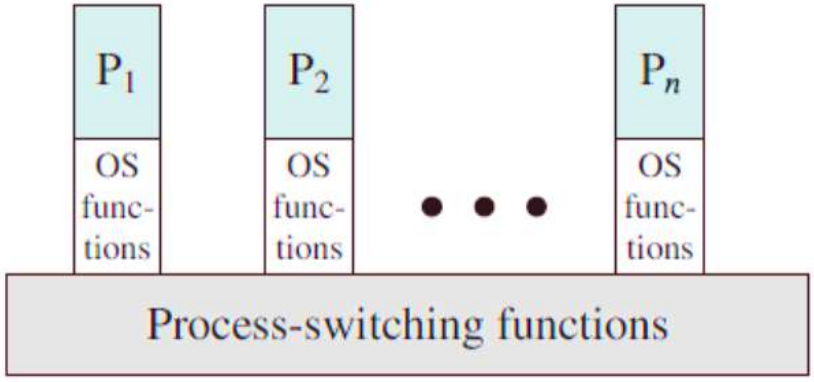
Execution of the Operating System

- Nonprocess Kernel
  - Execute the kernel outside of any process
  - OS has its own memory to use and its own stack for procedure calls

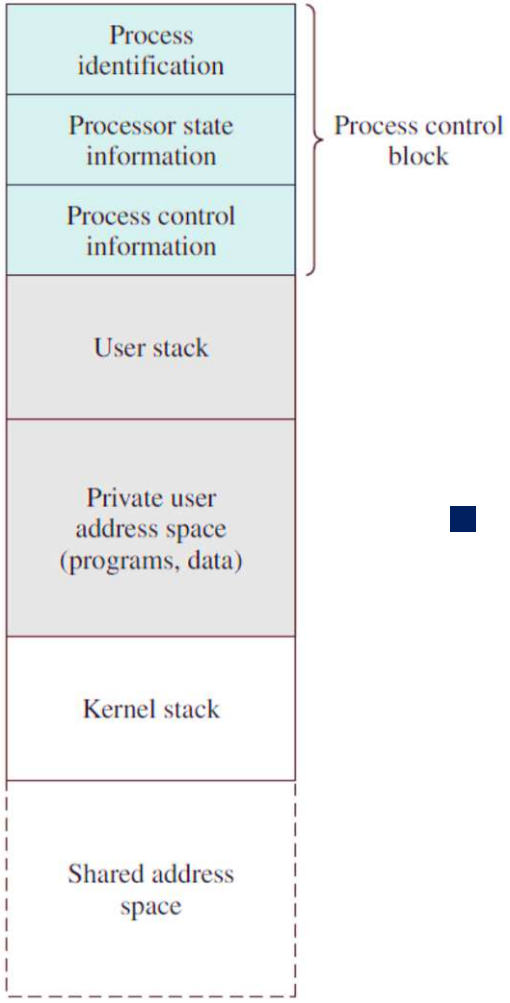


Execution of the Operating System

- Execution within User Processes
  - Execute virtually all OS software in the context of a user process
  - Program data and stack for kernel are included in each process image



Execution of the Operating System



- Execution within User Processes
  - A separate kernel stack is used in the kernel mode
  - OS code and data are in the shared address space
- To pass control to OS
  - Mode switch occurs
  - Process switch is not performed: execution continues within the current user process

Execution of the Operating System

- Process-based Operating System
  - Implement the OS as a collection of system processes
  - Modular OS
  - Noncritical OS functions are implemented as separate processes