Processes

- Process Concept
- Process Scheduling
- Operation on Processes

Process

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

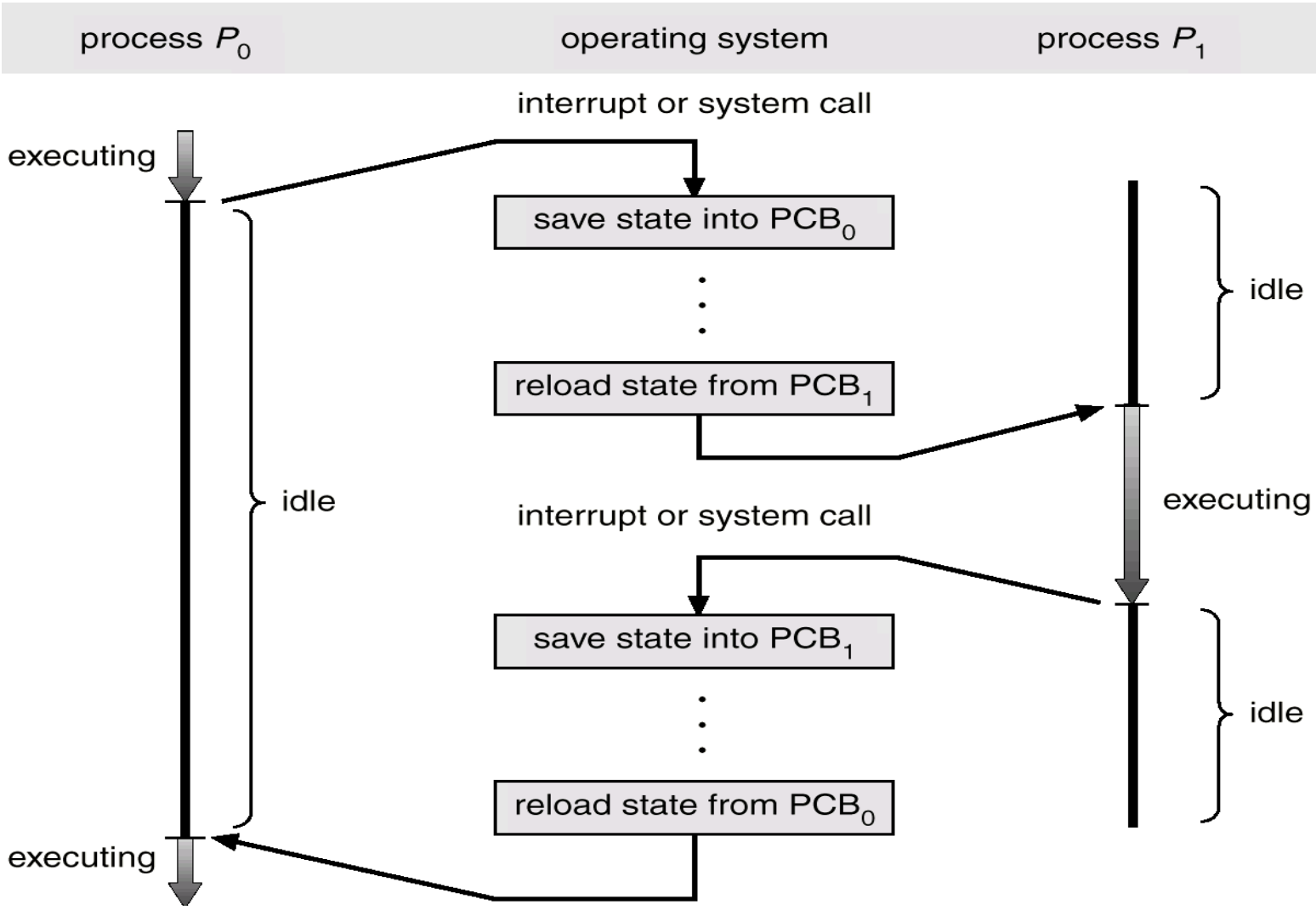
Process Concept

- An operating system executes a variety of programs:
  - Batch system – jobs
  - Time-shared systems – user programs or tasks
- Textbook uses the terms *job* and *process* almost interchangeably.
- Process – a program in execution; process execution must progress in sequential fashion.
- Programs are passive while Processes are active.
- A process includes:
  - program counter
  - stack
  - data section
  - code or text section

Process State

- **Only one process can be running on a CPU at a given instant.**
- **As a process executes, it changes *state***
  - **new: The process is being created.**
  - **running: Instructions are being executed.**
  - **waiting: The process is waiting for some event to occur.**
  - **ready: The process is waiting to be assigned to a processor.**
  - **terminated: The process has finished execution.**

CPU Switch From Process to Process



Process Creation

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

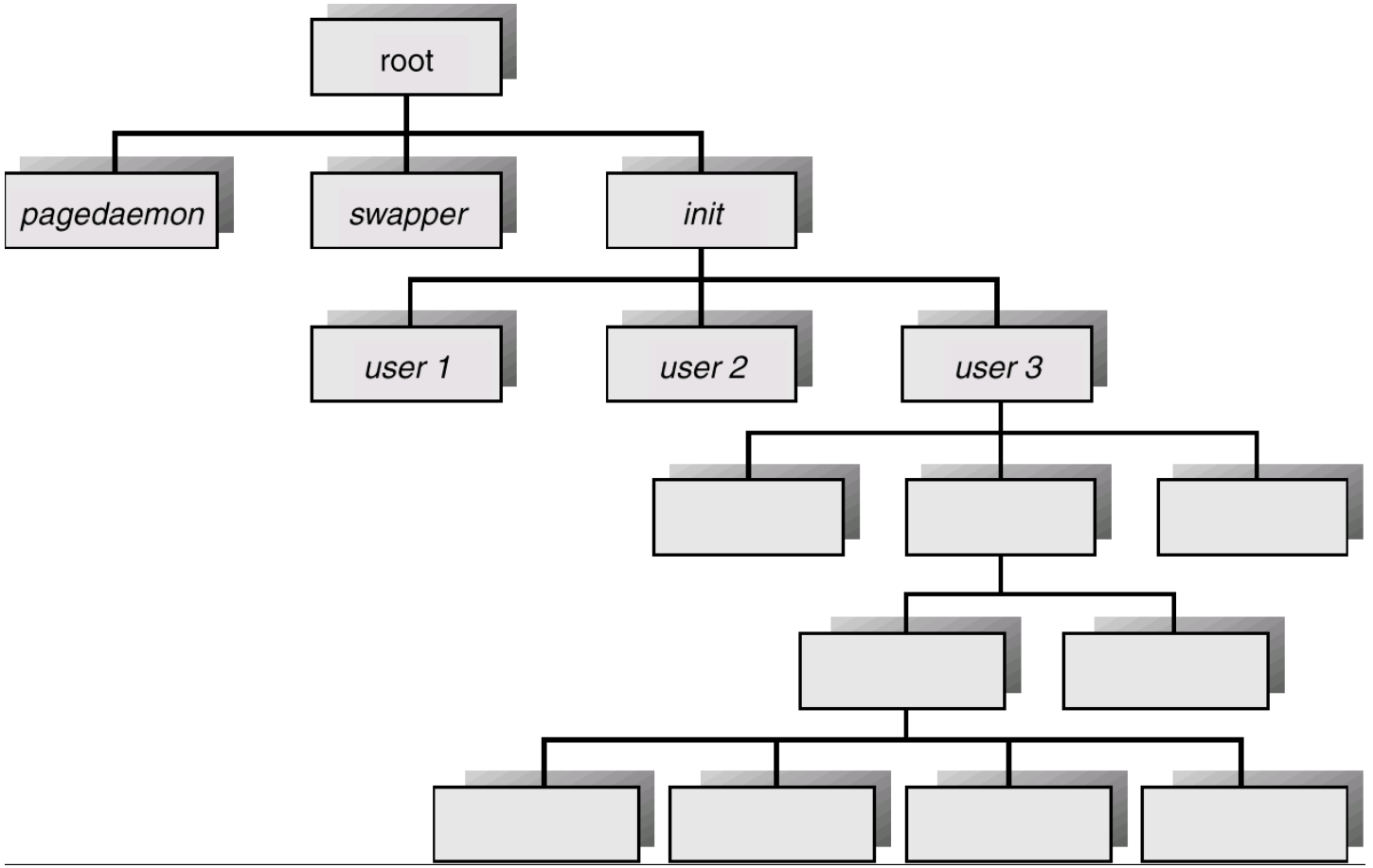
Process Creation

- **Parent process creates child processes, which, in turn create other processes, forming a tree of processes.**
- **Resource sharing (memory, files etc.)**
  - **Parent and children share all resources.**
  - **Children share subset of parent's resources. (prevents system overload)**
  - **Parent and child share no resources.**
- **Execution**
  - **Parent and children execute concurrently.**
  - **Parent waits until children terminate.(some or all)**

Process Creation (Cont.)

- **Address space**
  - **Child is duplicate of parent.**
  - **Child has a different program loaded into it.**
- **UNIX example (each process identified by a process identifier (PID) [integer]**
  - **fork system call creates a new process. PID of child is returned to the parent.**
  - **execve system call used after a fork to replace the process' memory space with a new program.**

A Tree of Processes On A Typical UNIX System



Process Creation

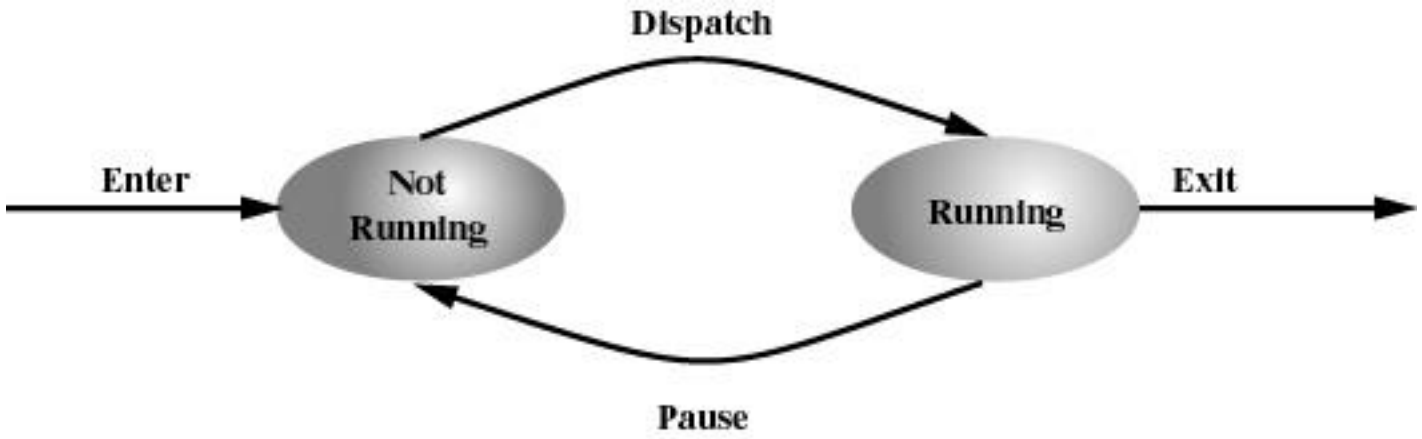
Tasks to be done:

- 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

Process States

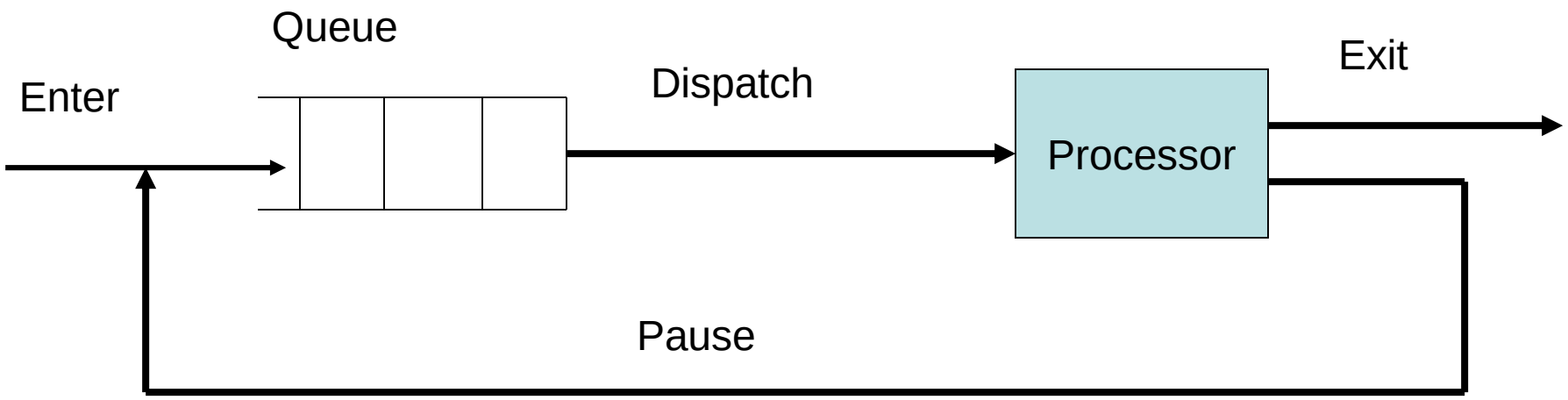
Two state process model

- Process may be in any of the 2 states
  - Running
  - Not running



(a) State transition diagram

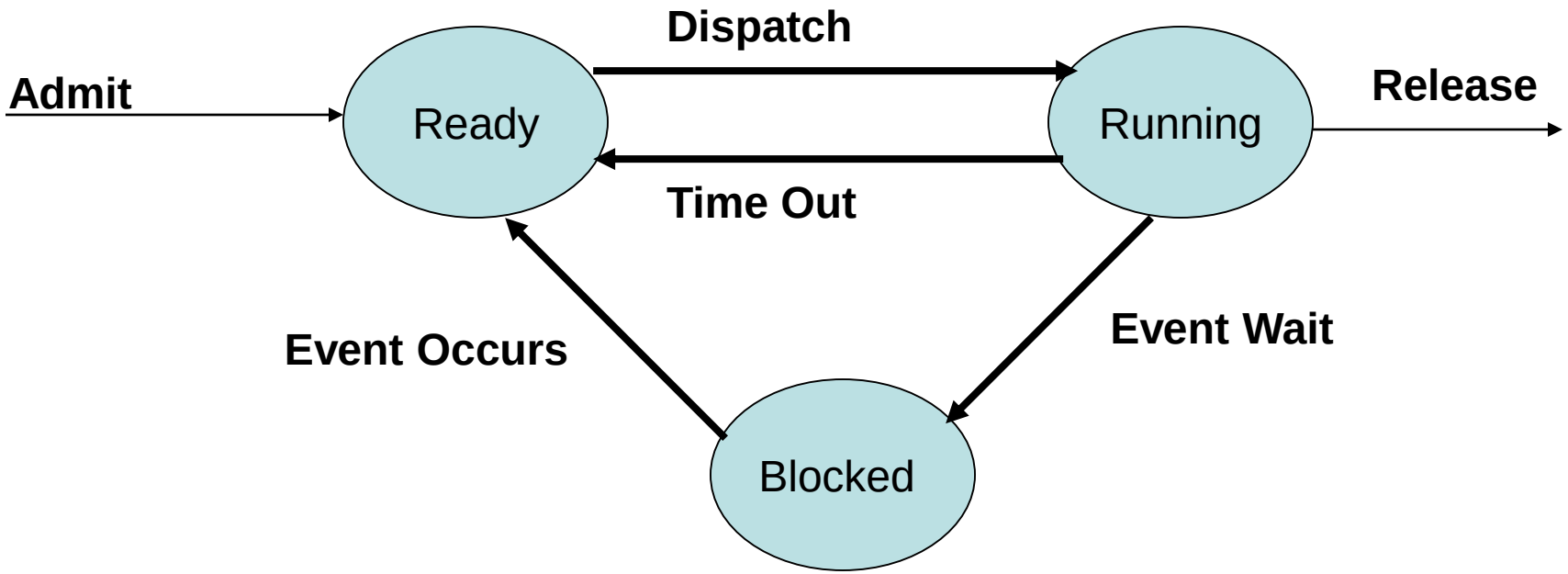
Queuing Diagram



Queue May contain ready and blocked processes

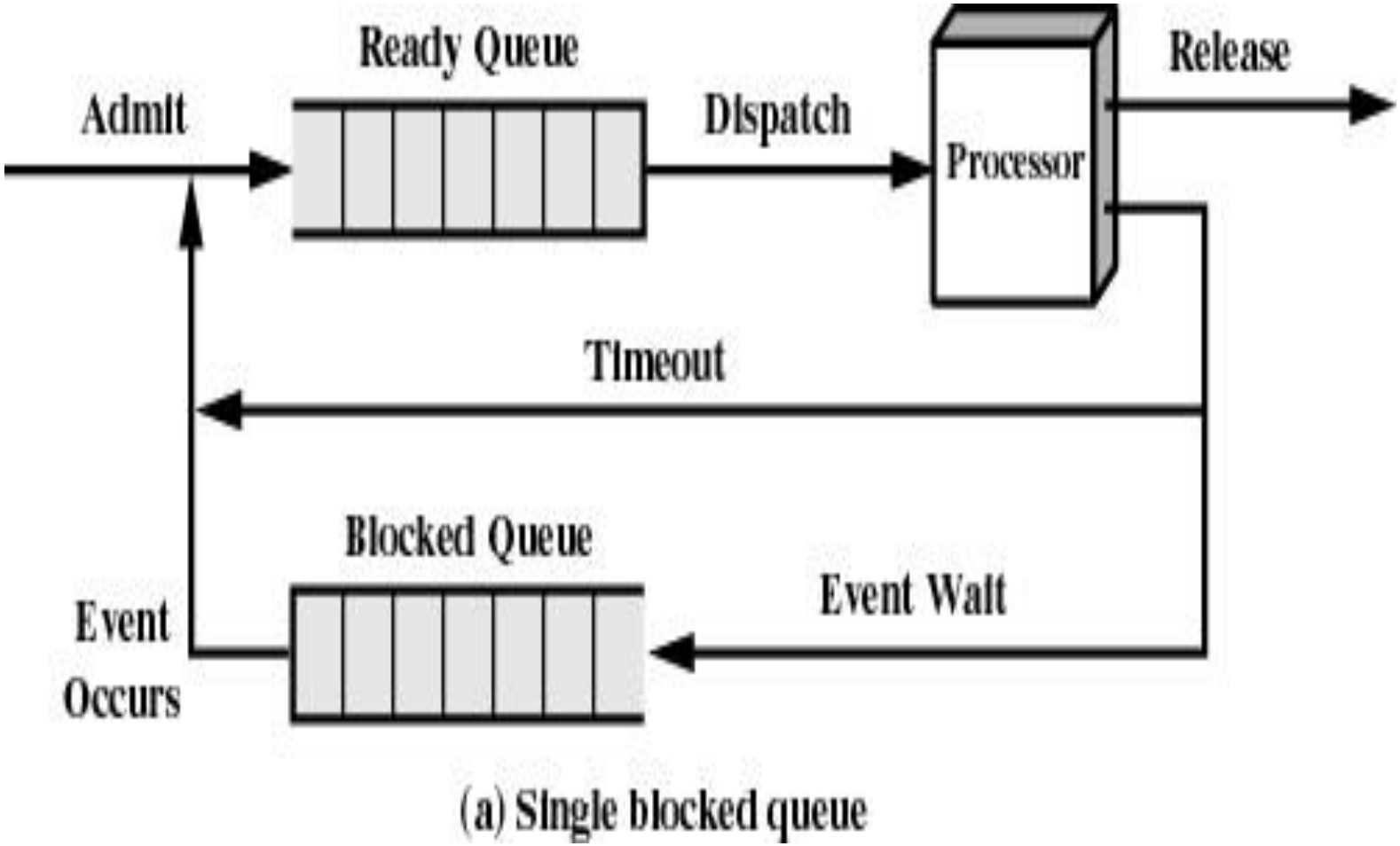
- Dispatcher schedules a Ready process to CPU for execution.
- In Two State model Not running state contains processes which are :
  - Ready to run
  - Blocked
- Dispatcher is required to linearly search the queue to find Ready to run process which Increases dispatcher overhead.
- Solution : Split not running state into Ready state & Blocked state

Three state model

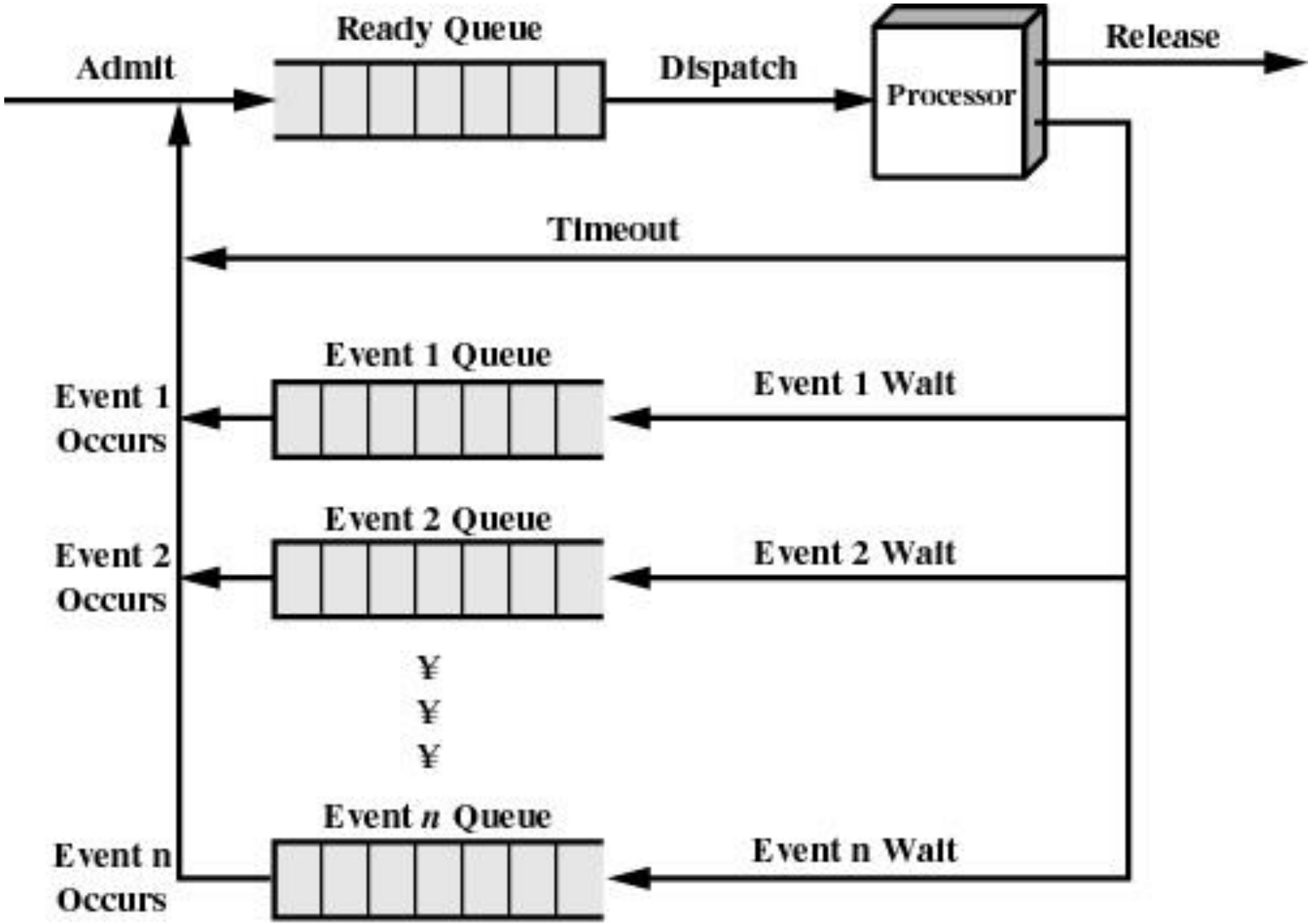


- Running: The process that is currently being executed
- Ready: A process that is ready to execute when given the opportunity
- Blocked : A process that cannot execute until some events occur
- Occurrence of event is usually indicated by interrupt signal

Using Two Queues



(a) Single blocked queue



(b) Multiple blocked queues

A Five-state Model

- Running
- Ready
- Blocked
- New
- Exit

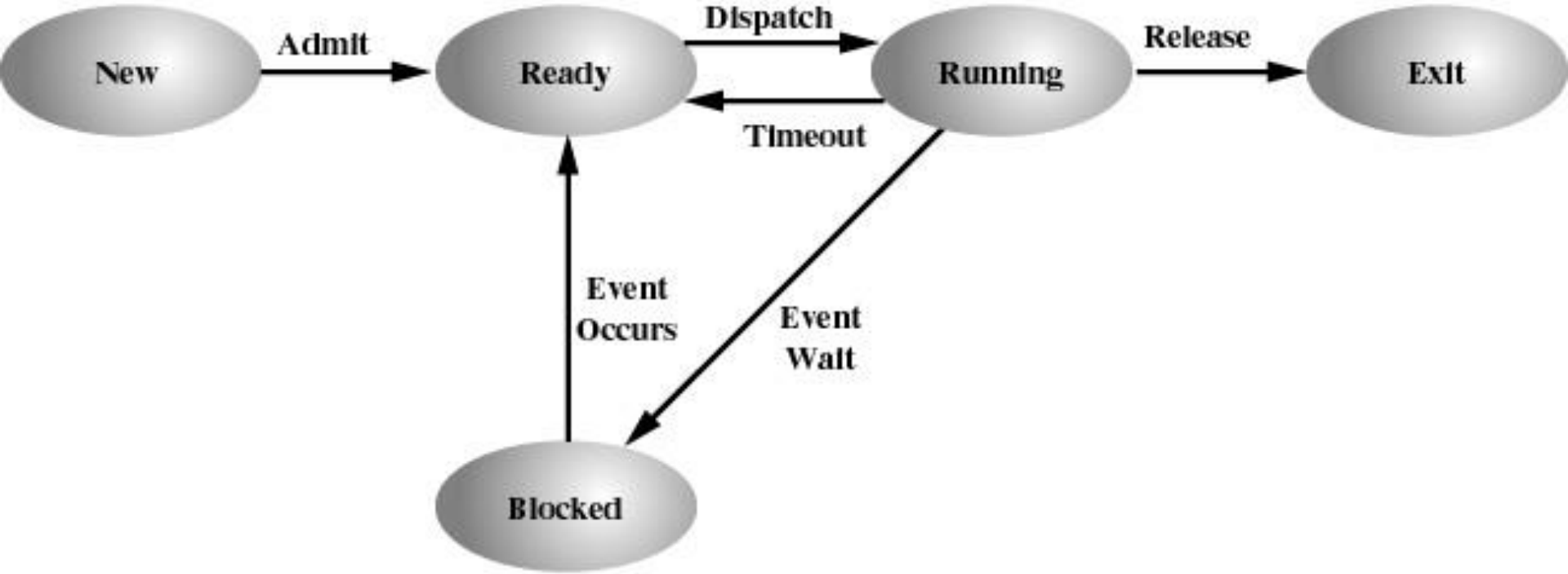


Figure 3.5 Five-State Process Model

Why do we need “New” state?

- Whenever a job is submitted , OS creates data structure for keeping track of the process context and then it tries to load the process.
- While loading the process
  - System May not have enough memory to hold the process
  - If we attempt to load Jobs as they are submitted, there may not be enough resource to execute processes in efficient manner
  - To efficiently execute processes, system may put a maximum limit on processes in ready queue

Valid state transitions

- New to ready
- Ready to running
- Running to exit
- Running to ready
- Running to blocked
- Blocked to ready
- Ready to exit
- Blocked to exit

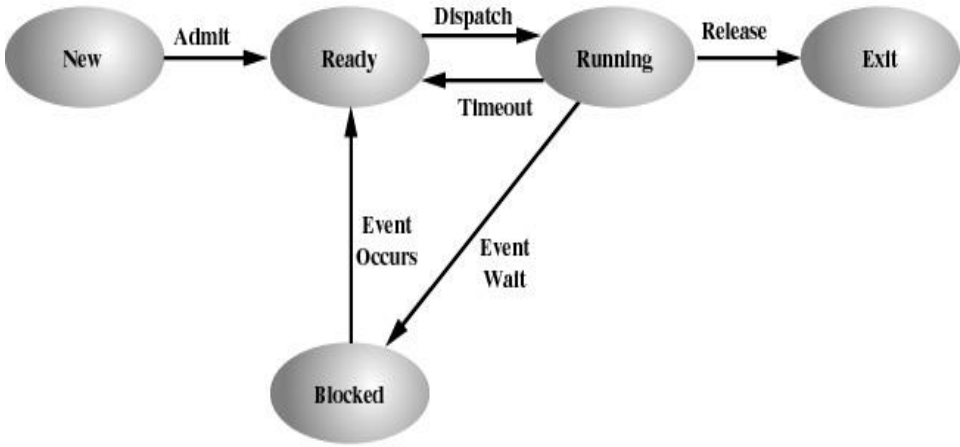


Figure 3.5 Five-State Process Model

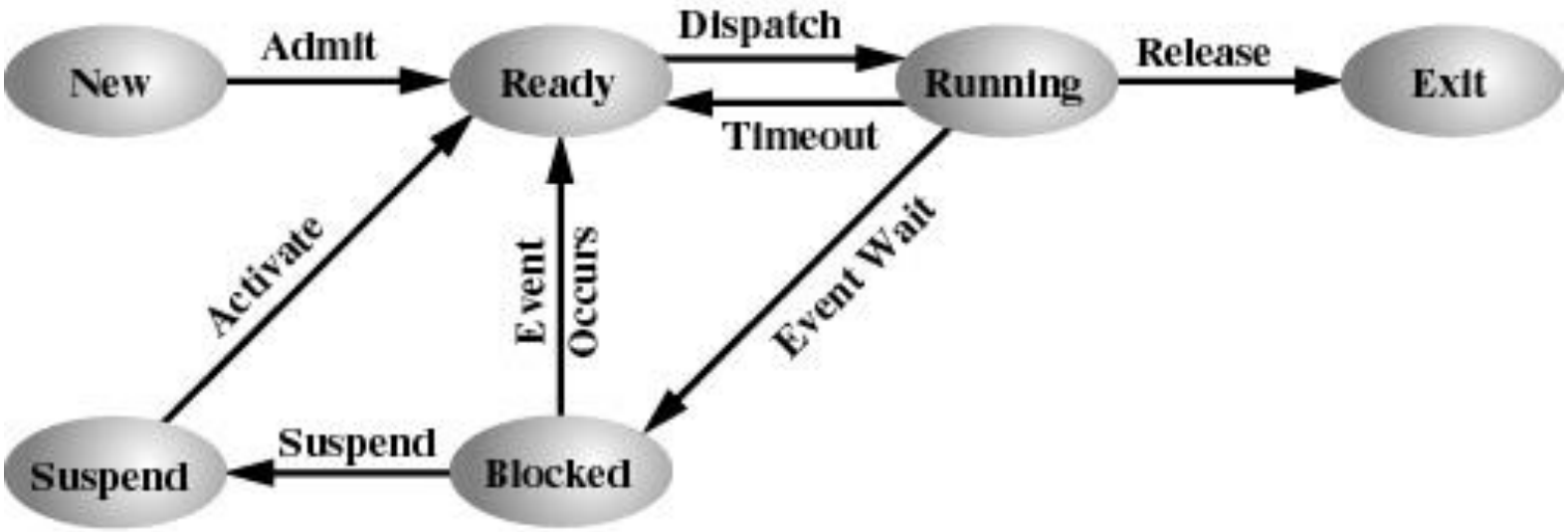
- What do we do If all processes get blocked on I/O and when system tries to bring in a new process from new to ready state no memory is available ?

- Solution :swapping

Suspended Processes

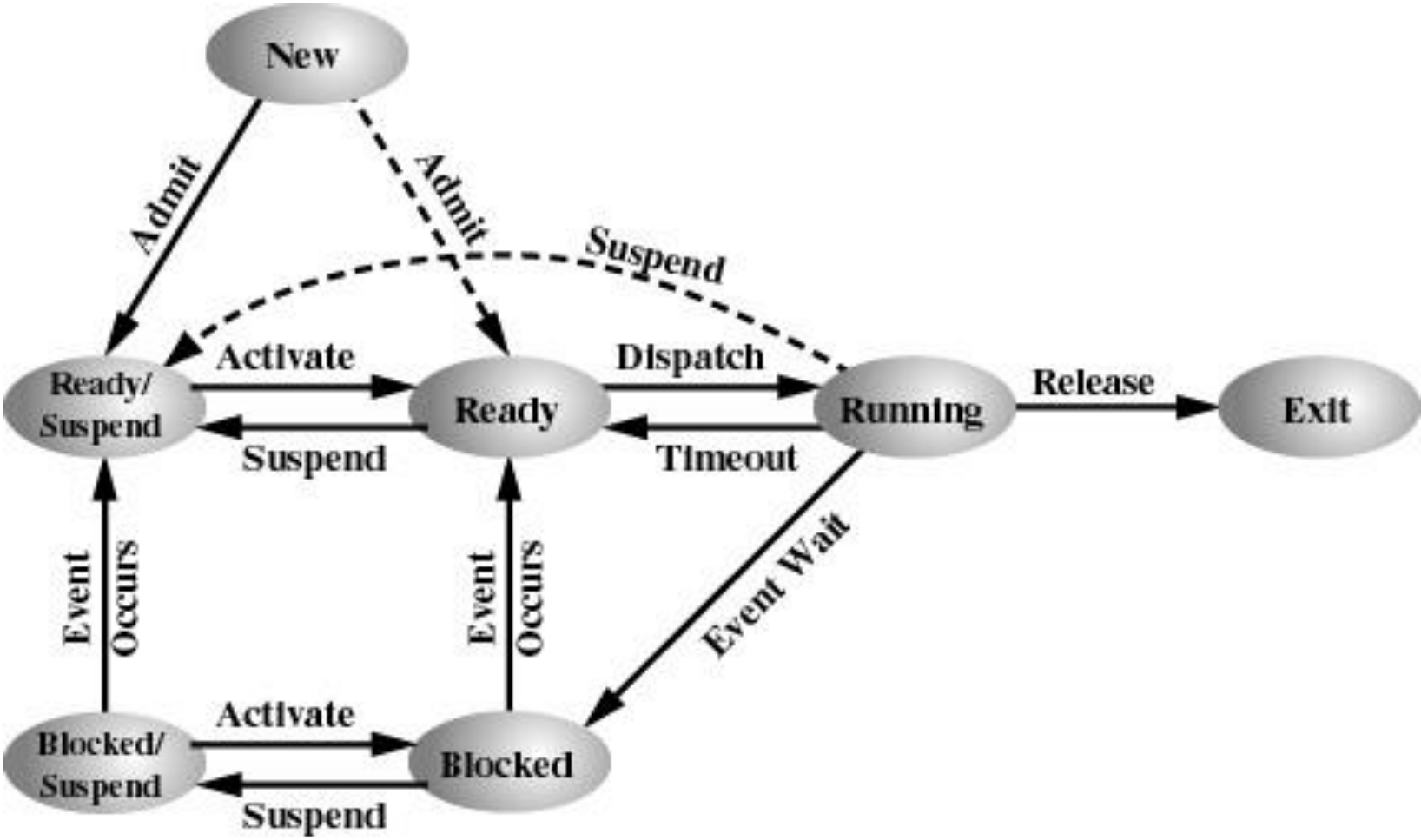
- Processor is faster than I/O so all processes could be waiting for I/O
- Swap these processes to disk to free up more memory
- Blocked state becomes suspend state when swapped to disk
- Two new states
  - Blocked, suspend
  - Ready, suspend

One Suspend State



(a) With One Suspend State

Two Suspend States



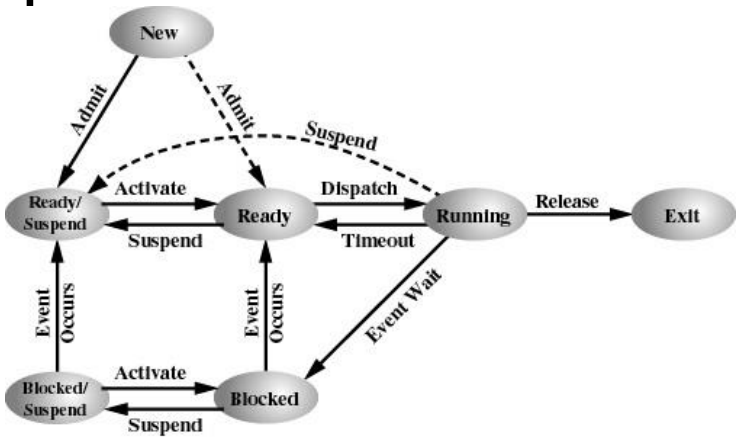
(b) With Two Suspend States

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.

State Transitions

- Blocked ⟶ Blocked/suspended:
  - If Ready queue is empty and insufficient memory is available then one of the blocked process can be swapped out
  - If currently running process requires more memory
  - If OS determines that ready process will require more memory

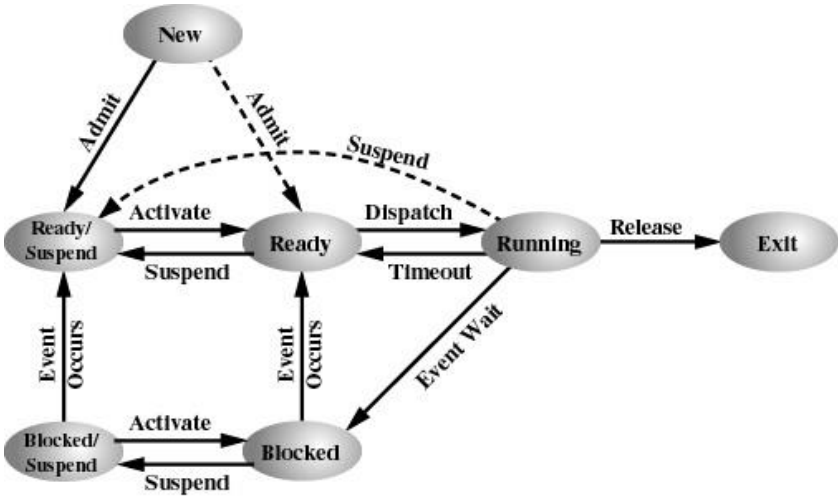


(b) With Two Suspend States

- Blocked/ Suspended ⟶ Ready/suspended
  - When the event for which process has been waiting occurs

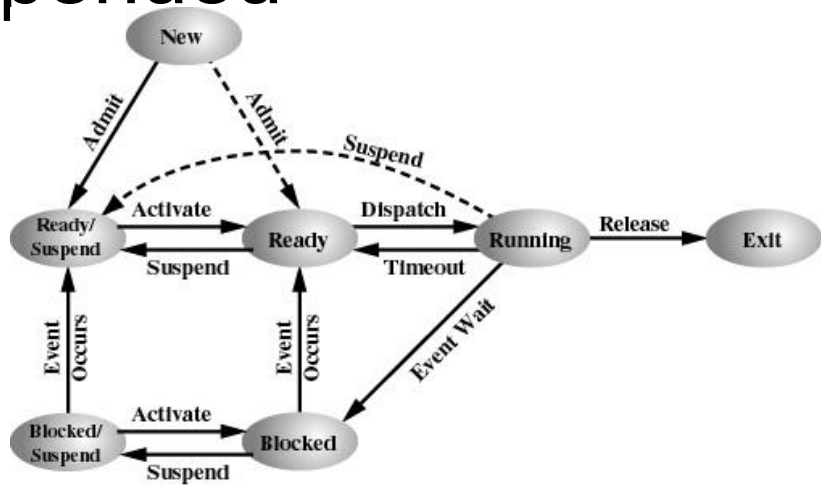
Note: State information concerning the suspended process must be accessible to the OS

- Ready-suspended ⟶ Ready
  - If Ready queue is empty
  - If Process in Ready-suspended state has higher priority than process in Ready state



(b) With Two Suspend States

- Ready ⟶ Ready/Suspend
  - Normally a blocked process is suspended
  - Suspend Ready process if it is the only way to free memory
  - Suspend a lower priority ready process than higher priority blocked process
- Blocked Suspended ⟶ Blocked
- Running ⟶ Ready suspended
- Various ⟶ Exit



(b) With Two Suspend States

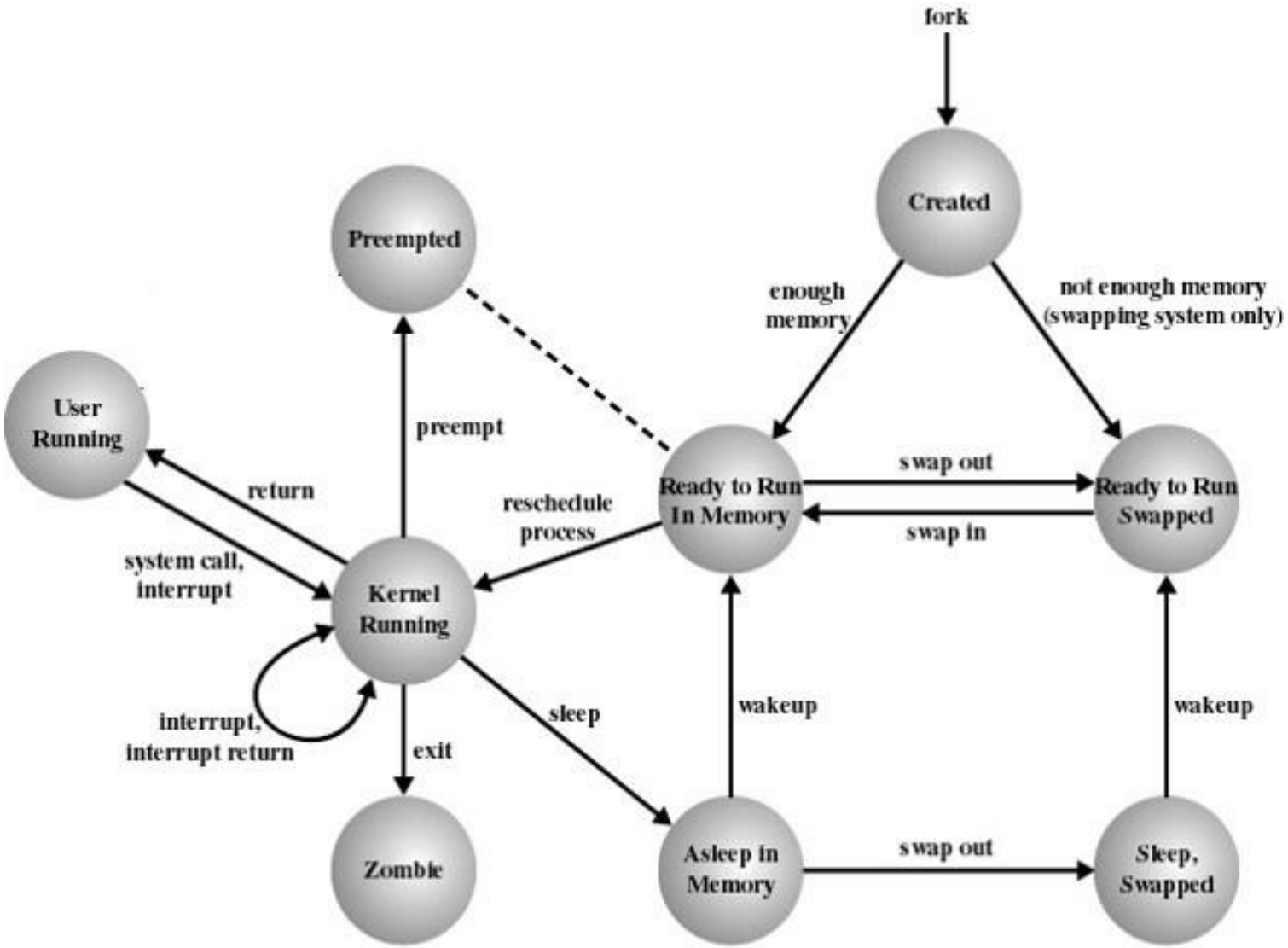


Figure 3.16 UNIX Process State Transition Diagram

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.

Process Termination

- Process executes last statement and asks the operating system to delete it (exit).
  - Output data from child to parent (via wait).
  - Process' resources are deallocated by operating system.
- Parent may terminate execution of children processes (abort).
  - Child has exceeded allocated resources.
  - Task assigned to child is no longer required.
  - Parent is exiting.
    - Operating system does not allow child to continue if its parent terminates.
    - Results in cascading termination.

Reasons for Process Termination

- Normal completion
- Time limit exceeded
- Bounds violation
- Protection error
  - example write to read-only file
- Arithmetic error
- Time overrun
  - process waited longer than a specified maximum for an event

Reasons for Process Termination

- I/O failure
- Invalid instruction
  - happens when try to execute data
- Privileged instruction
- Operating system intervention
  - such as when deadlock occurs
- Parent terminates so child processes terminate
- Parent request

Processes- Topics Covered

- ✓ Process concept and definition
- ✓ 5 State Process model
- ✓ Process Creation => Reasons, Unix- fork(), execve()
- ✓ Process Termination => Reasons, Unix- exit(), abort()
- ✓ 7 State Process model => block-suspend/ready-suspend

Further……..

Operating System Control Structures

Contain:

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

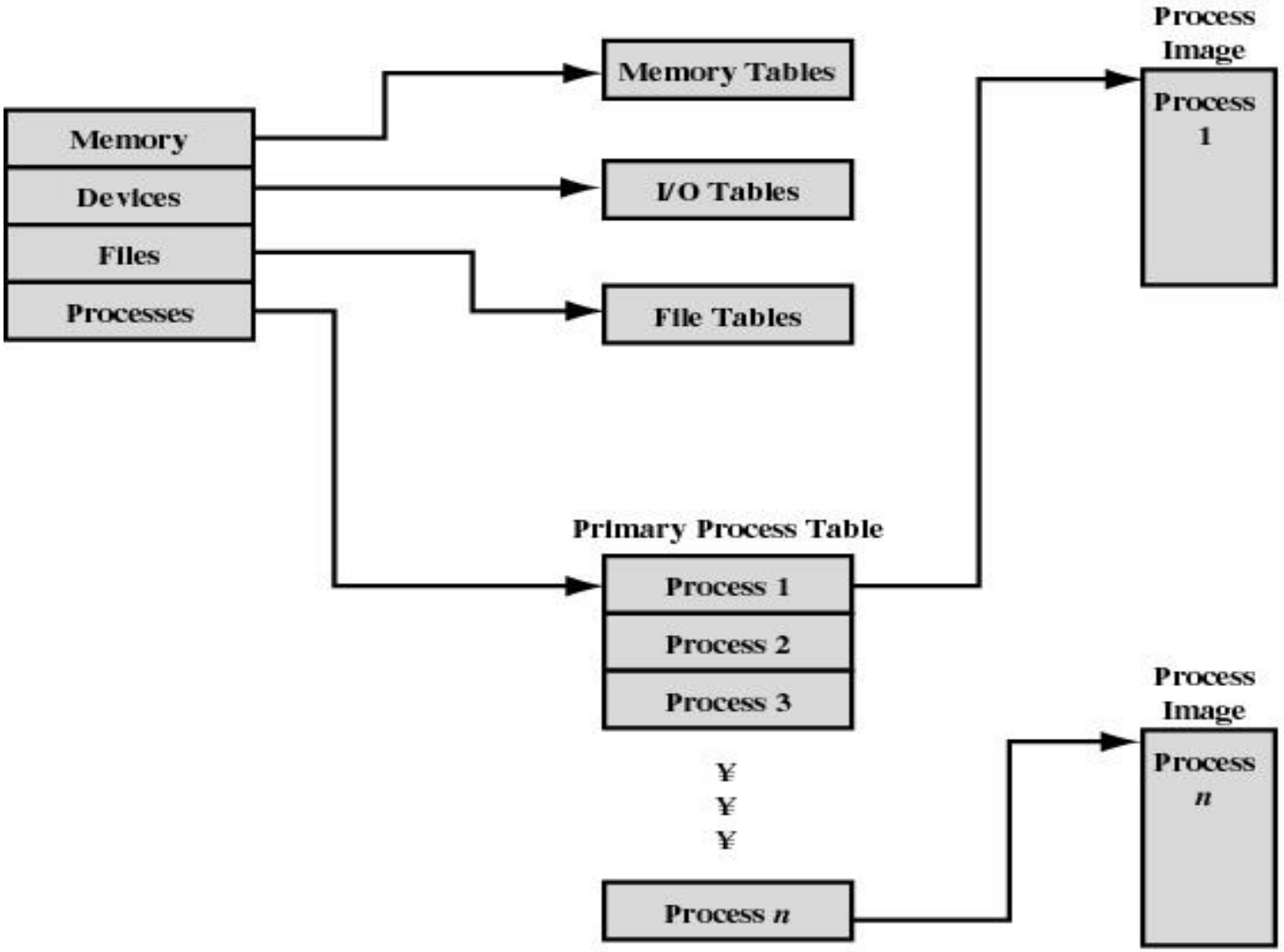


Figure 3.10 General Structure of Operating System Control Tables

Memory Tables

Contain

- 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

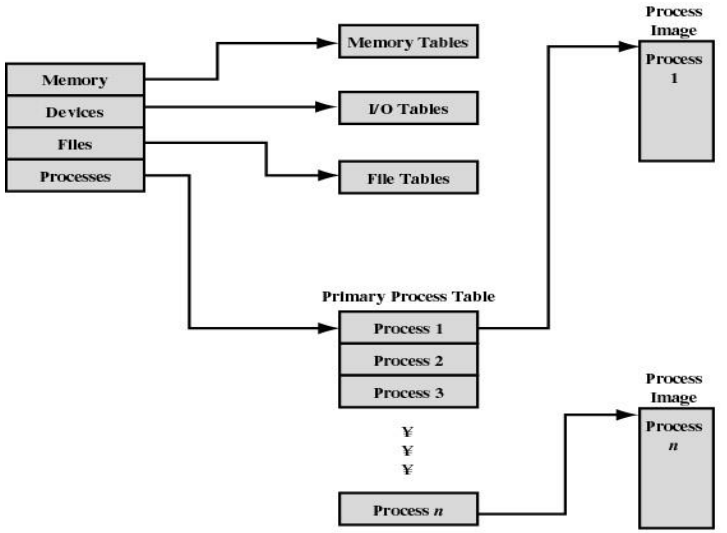


Figure 3.10 General Structure of Operating System Control Tables

I/O Tables

Information about:

- 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

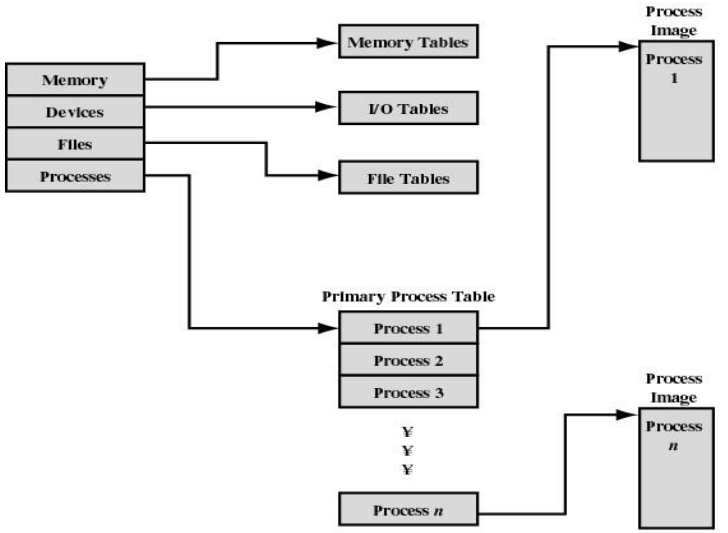


Figure 3.10 General Structure of Operating System Control Tables

File Tables

- Existence of files
- Location on secondary memory
- Current Status
- Attributes

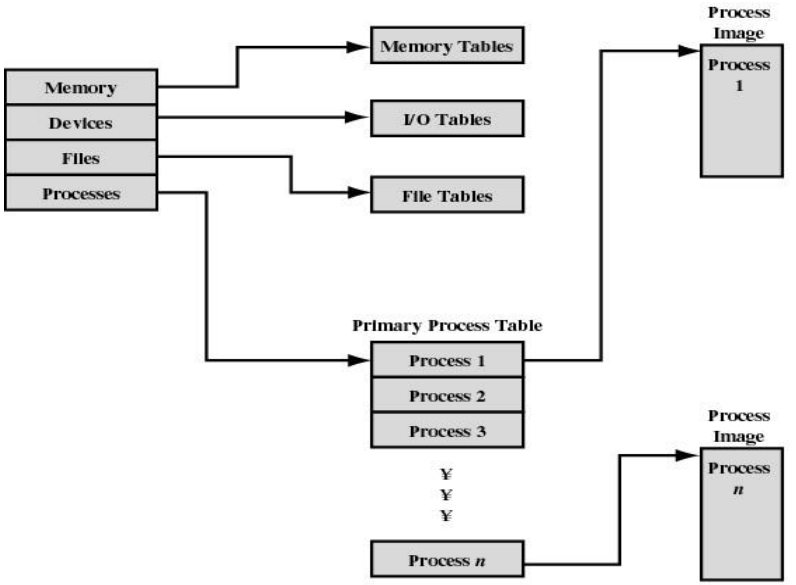


Figure 3.10 General Structure of Operating System Control Tables

Process Table

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

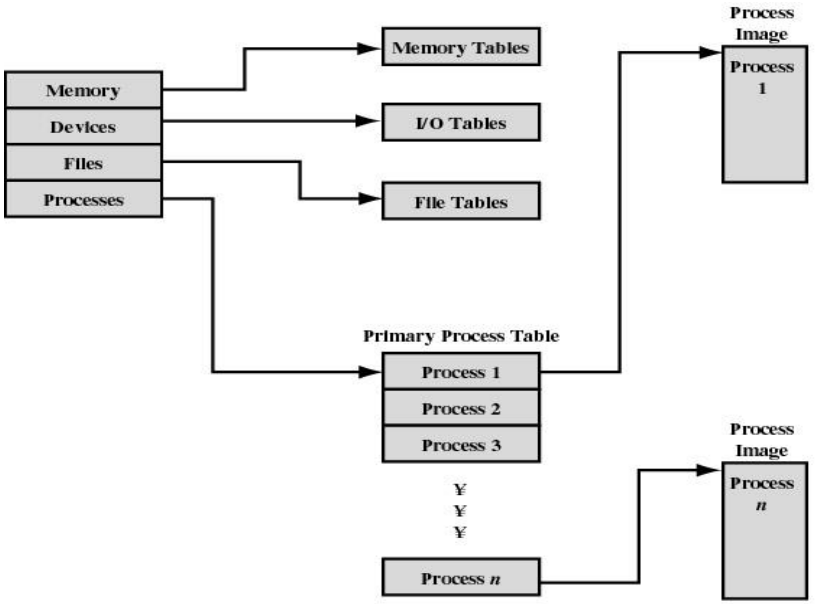


Figure 3.10 General Structure of Operating System Control Tables

Process Image

- 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 (PCB)

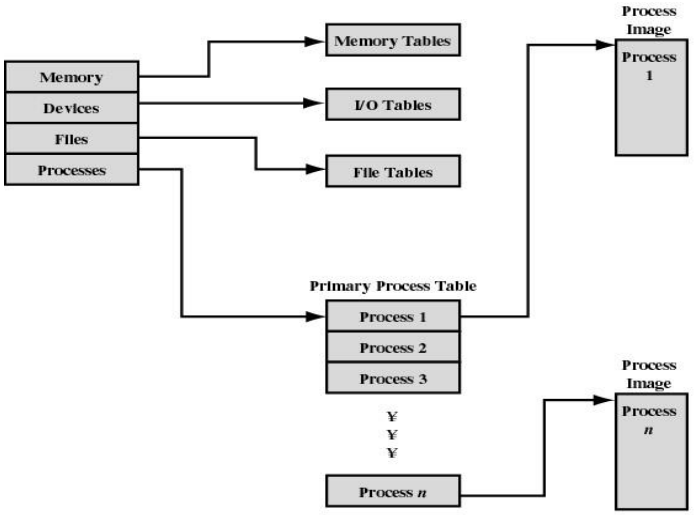


Figure 3.10 General Structure of Operating System Control Tables

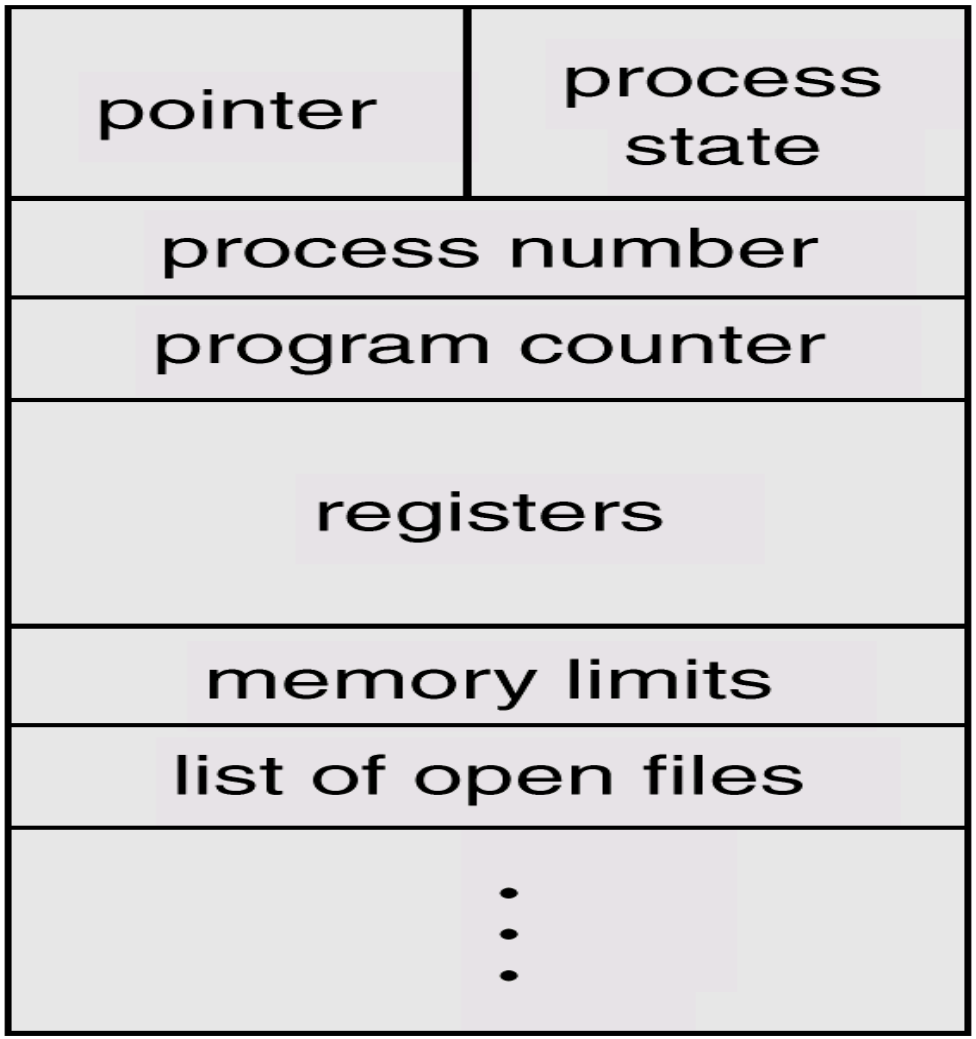
Process Control Block (PCB)

PCB is also referred to as Task control block

Information associated with each process:

- **Process state: new, ready, running, waiting, terminated…**
- **Program counter: for next process instruction.**
- **CPU registers**
- **CPU scheduling information: process priority, pointers to scheduling queues etc.**
- **Memory-management information: base/limit regs, page & segment tables etc.**
- **Accounting information: amount of CPU & real time used, time limits, process number etc.**
- **I/O status information: list of I/O devices allocated and of files open**

Process Control Block (PCB)



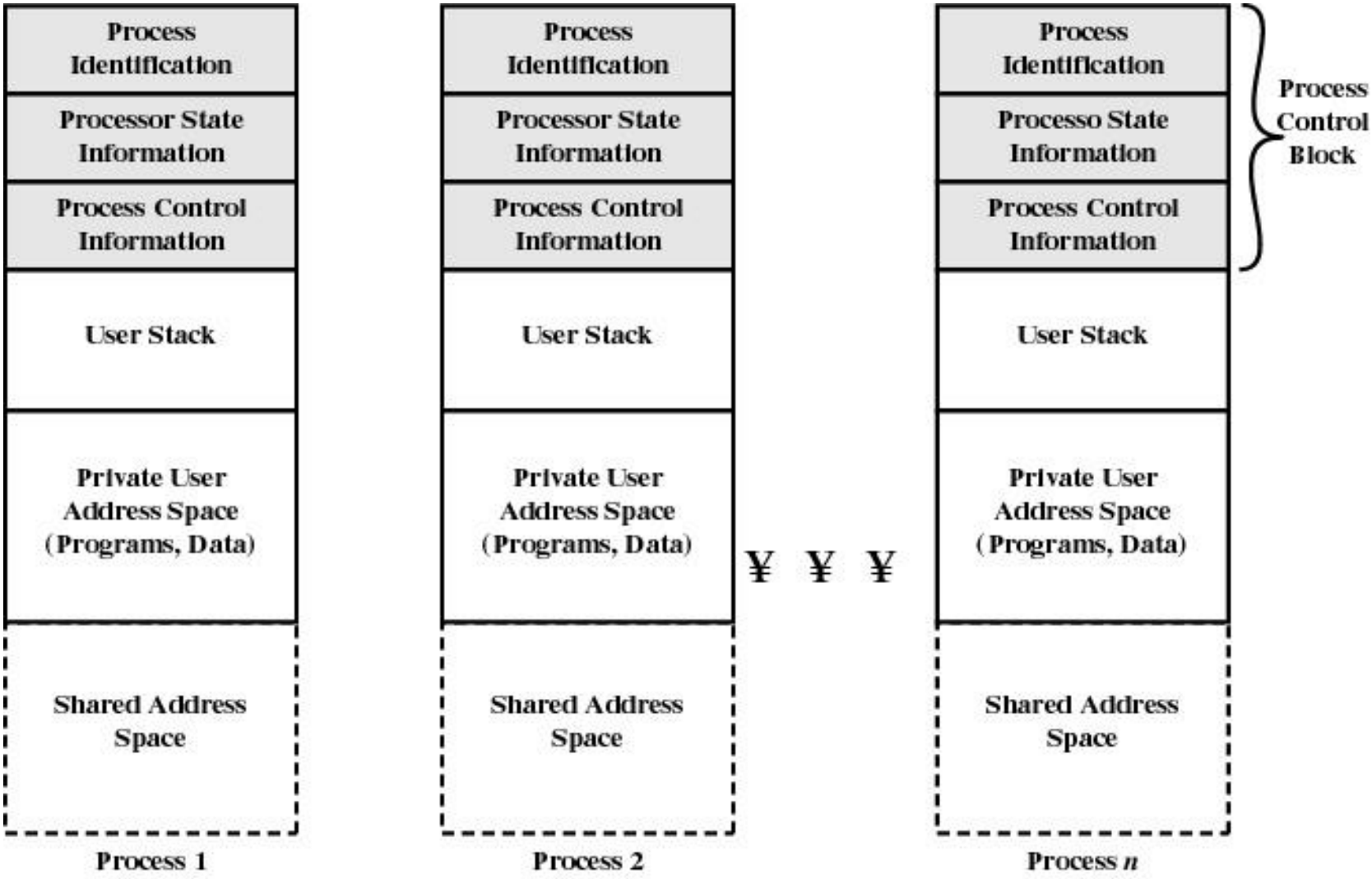


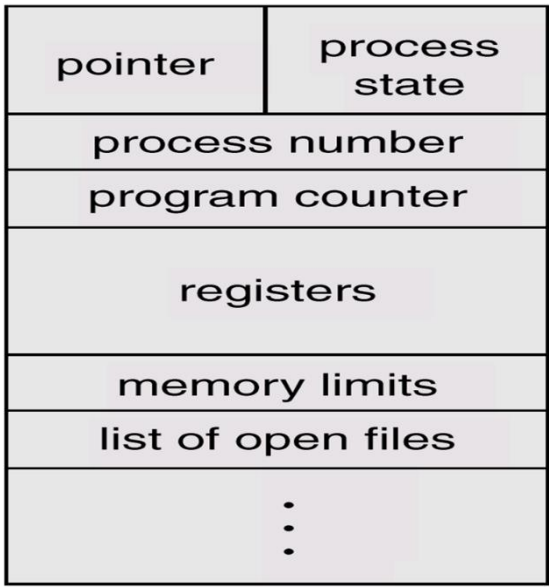
Figure 3.12 User Processes in Virtual Memory

Process Control Block

- Process identification
  - Identifiers

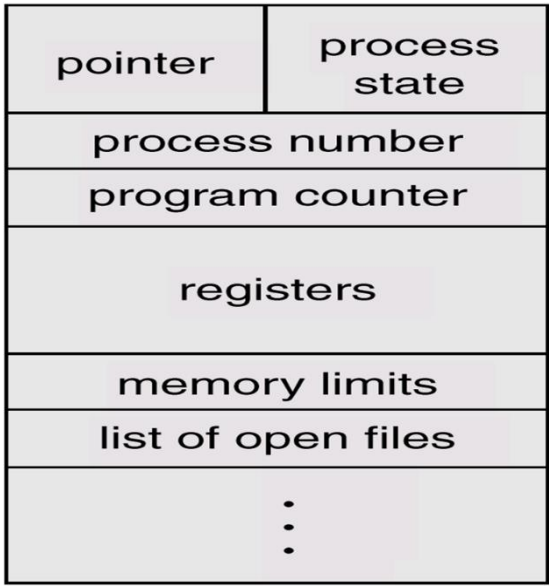
    Numeric identifiers that may be stored with the process control block include

    - Identifier of this process
    - Identifier of the process that created this process (parent process)
    - User identifier



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.



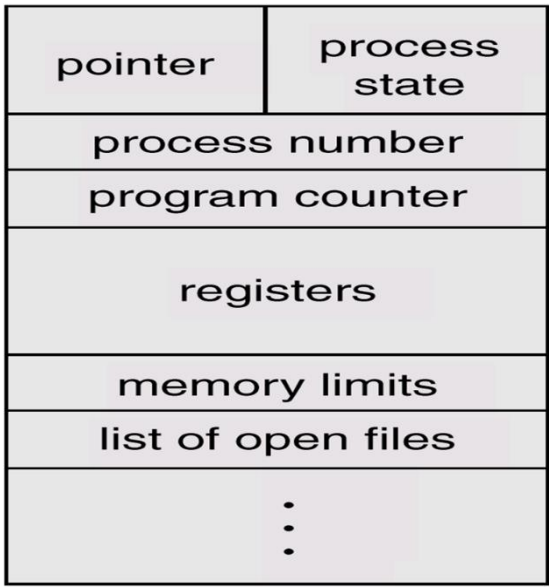
Process Control Block

- 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

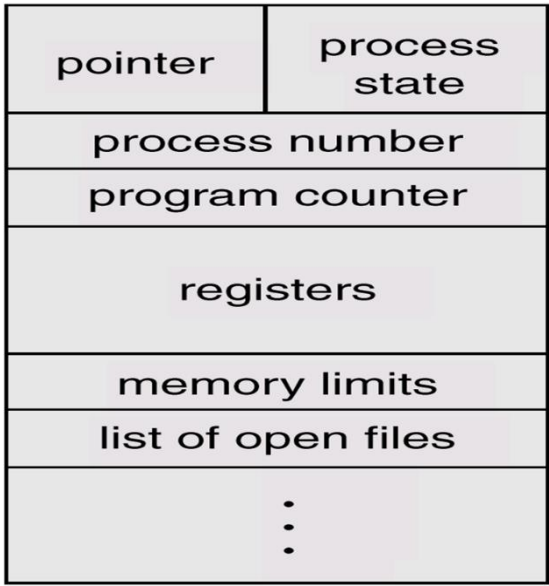
    - •*Program counter:* Contains the address of the next instruction to be fetched
    - •*Condition codes:* Result of the most recent arithmetic or logical operation (e.g., sign, zero, carry, equal, overflow)

    •*Status information:* Includes interrupt enabled/disabled flags, execution mode



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.



Process Control Block

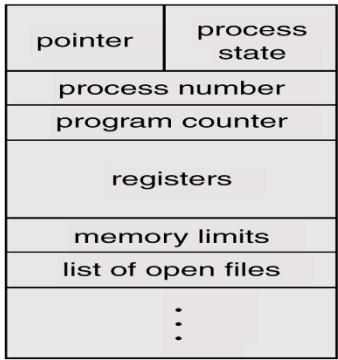
- Process Control Information
  - Scheduling and State Information

    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).

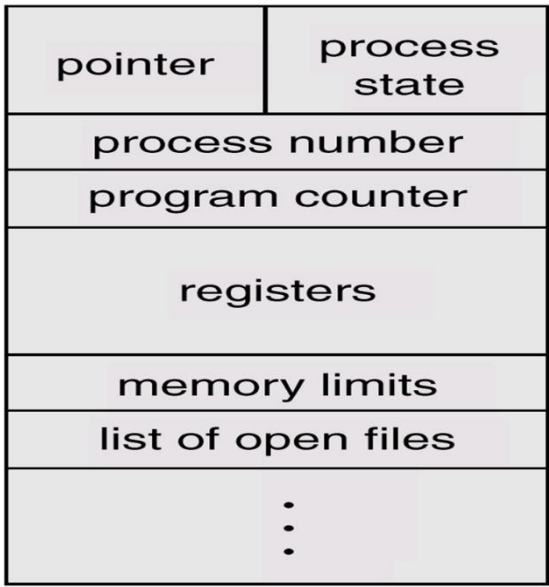
    •*Priority:* One or more fields may be used to describe the scheduling priority of the process. In some systems, several values are required (e.g., default, current, highest-allowable)

    •*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.



Process Control Block

- Process Control Information
  - *Event:* Identity of event the process is awaiting before it can be resumed
  - Pointers
    - 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.



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.

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.

CONTEXT SWITCH

When to Switch a Process

- Clock interrupt
  - process has executed for the maximum allowable time slice
- I/O interrupt
- Page fault
  - memory address is in virtual memory so it must be brought into main memory
- Trap
  - error occurred
  - may cause process to be moved to Exit state
- Supervisor call
  - such as file open

Change of Process State

Tasks to be done for context switch:

- 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

- Update the process control block of the process selected
- Restore context of the selected process

Context Switch

- **Typically the time taken is between 1 and 1000μs**
- **Context-switch time is overhead; the system does no useful work while switching.**
- **Time taken is dependent on h/w support. Like: memory speed, number of regs to be stored, existence of a single instruction each for loading and for saving all regs etc.**
- **Context switch is a big performance improvement bottleneck. Avoidance --- Use Threads**

Summary

- OS Tables= memory, I/o, file, process
- Process image=Collection of program, data, stack, PCB
- PCB= (identification+state info+control info) blocks

  -State info= User-visible registers, Control and status registers, Stack pointers

  -Control info= Scheduling and State Information, data structuring, blocking event, Interprocess Communication, Process Privileges, Memory Management, Resource Ownership and Utilization

- **Context switch is a big performance improvement bottleneck.**

Avoidance --- Use Threads