Process Description and Control

Process Control Block (continued…)

- Process Control Information
  - *Process state*
  - *Priority*
  - *Scheduling-related information*
  - *Event*
  - Inter-process Communication
  - Memory Management
  - Resource Ownership and Utilization

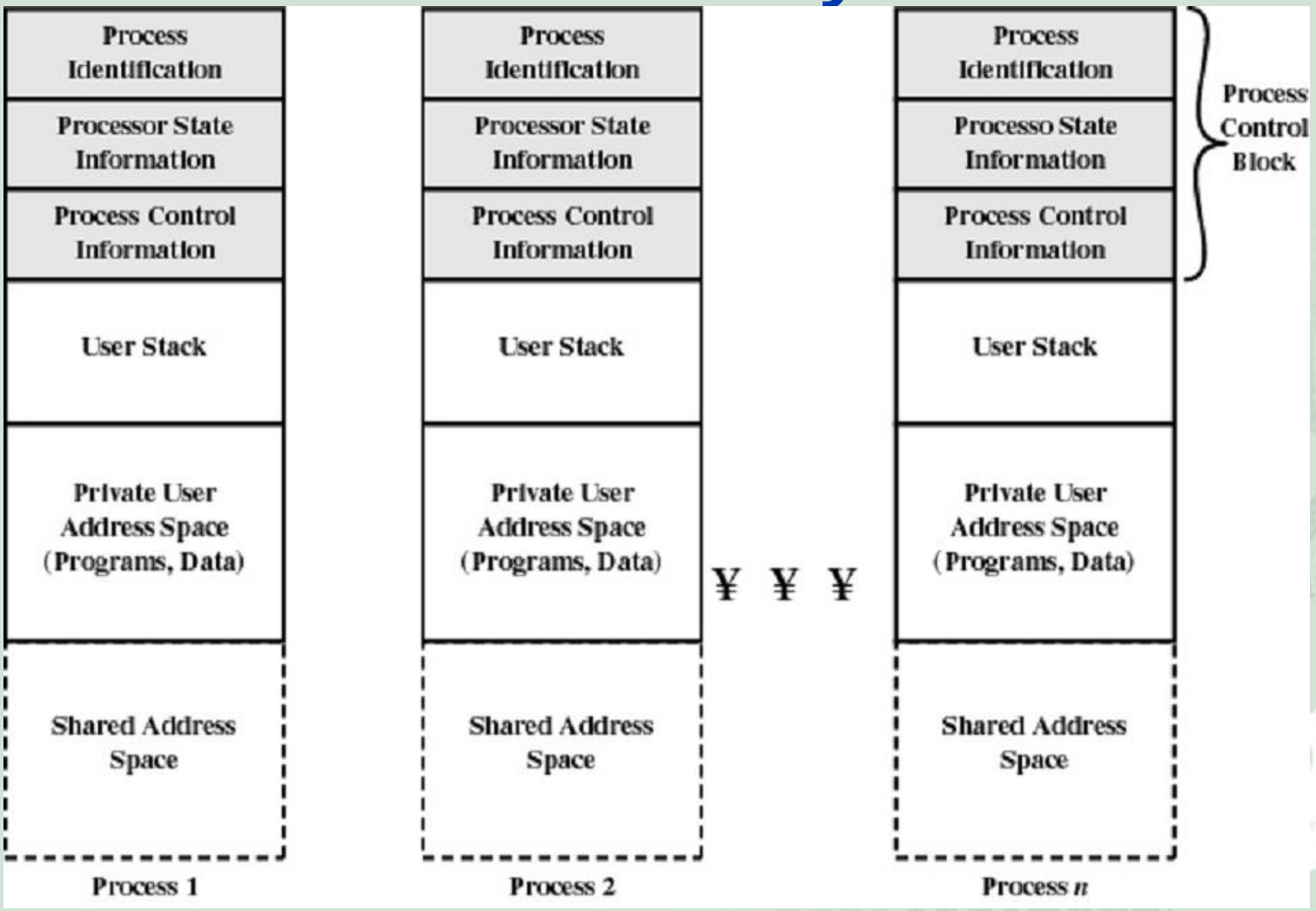
Process Control Block (continued…)

- 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.

Process Control

- Process Control Block (PCB)
- Process Execution
- Context Switch
- Process Scheduling

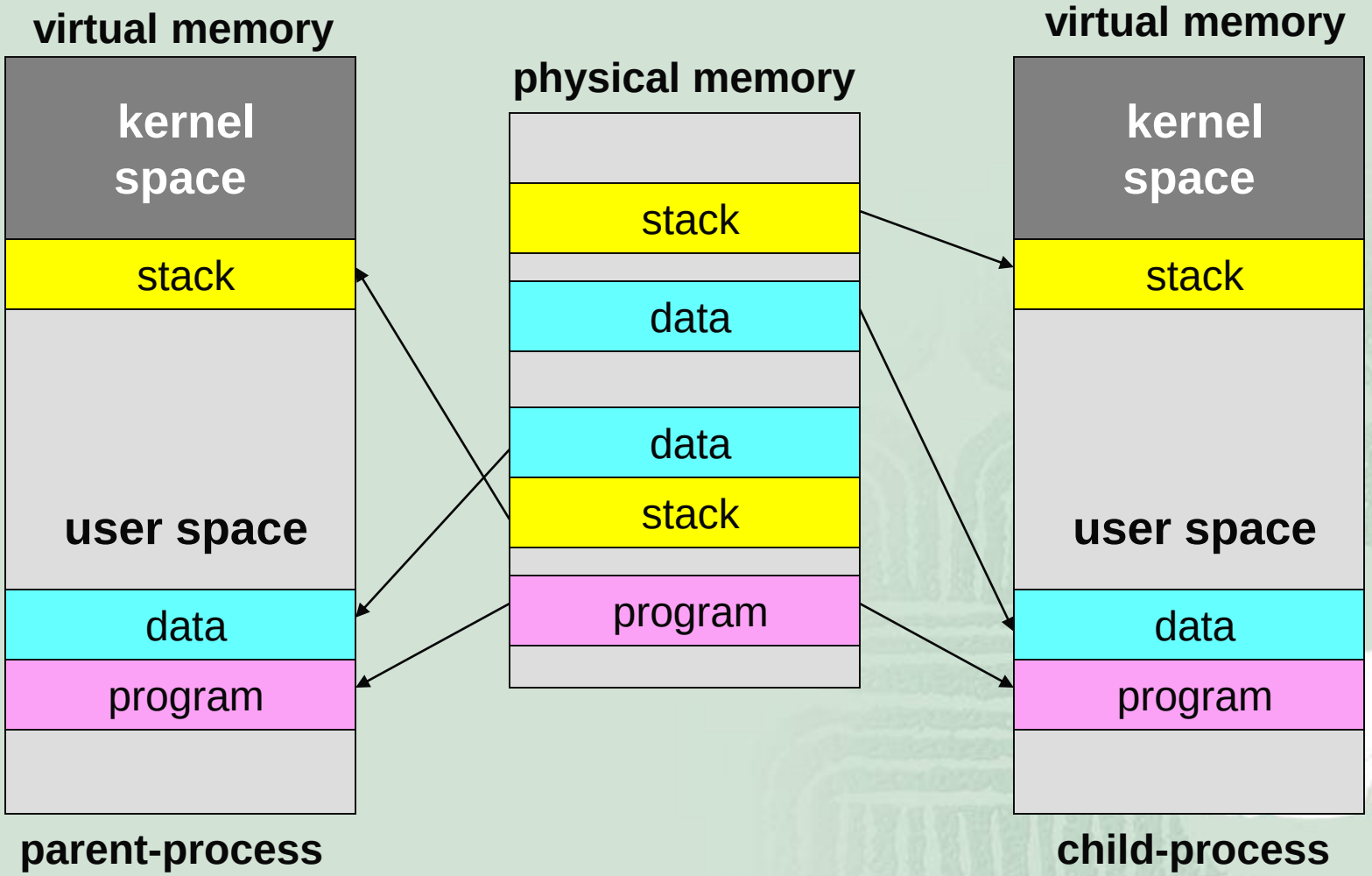
User Processes in Virtual Memory



When a program ‘forks()’

- We already know how new processes get created in the UNIX/Linux environment (i.e., via the ‘fork()’ function or system-call)
- A child-process gets constructed using the same code and data as its parent-process
- Yet each task’s memory-regions are kept ‘private’ (i.e., accessible to it alone)
- So how exactly is this feat achieved?

Similar memory-mappings



Process Creation

- Assign a unique process identifier
- Allocate space for the process
- Initialize process control block (PCB)
- 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

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

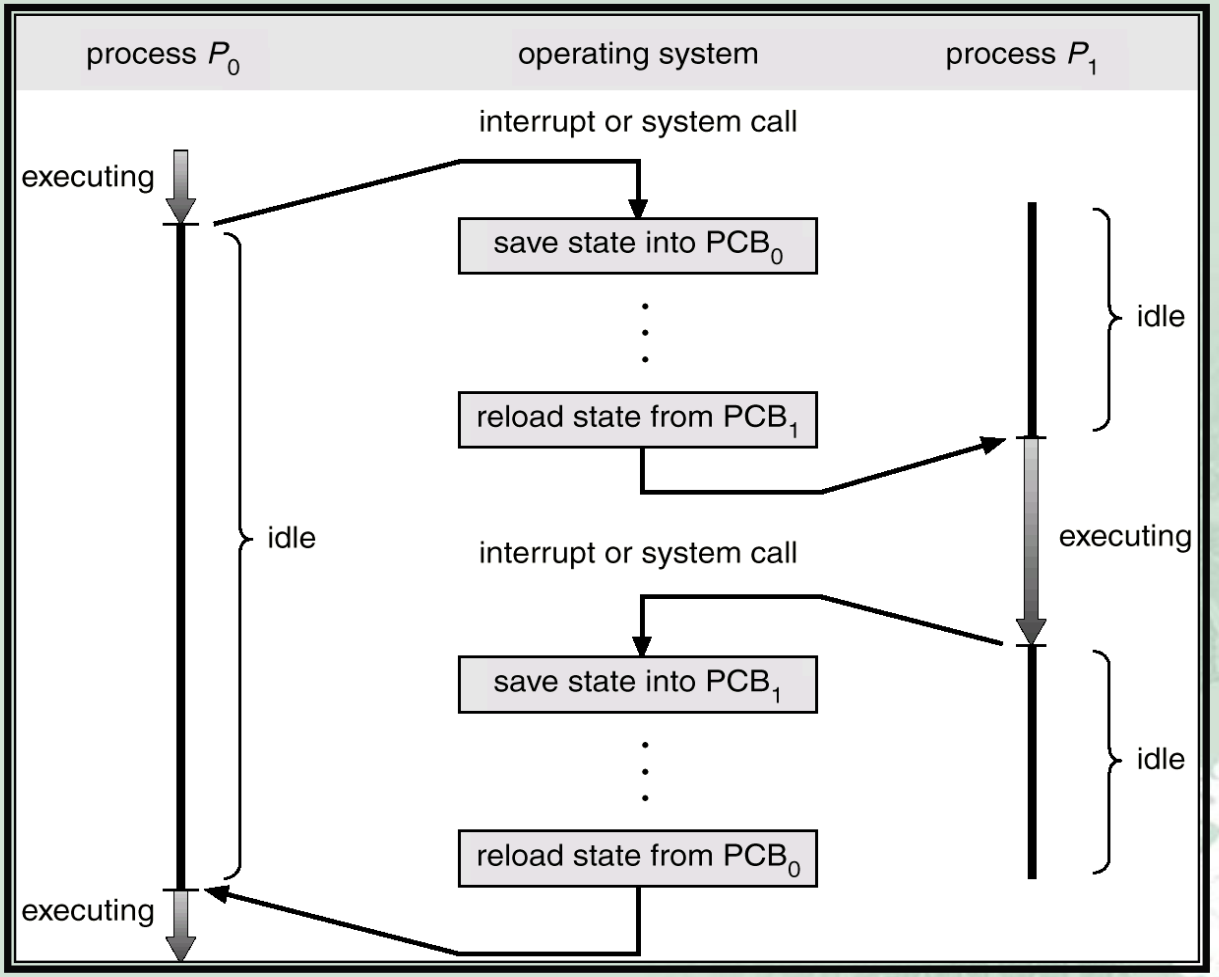
Process Control

- Process Control Block (PCB)
- Process Execution
- Context Switch
- Process Scheduling

Context Switch

- Saving the state of one process and loading another process onto the CPU
- Pure Overhead
  - length of time varies from machine to machine
  - a modern CPU can perform hundreds of context switches per second, but???
  - hardware support makes it faster (Intel 80386), but most modern OS's perform software context switching

CPU Switch From Process to Process



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

Change of Environment

- Caches
  - Virtual addresses: cache must either have process tag or must flush cache on context switch
- Translation Lookaside Buffer (TLB)
  - Each entry must have process tag or flush TLB on context switch
- Page table
  - Typically have page table pointer (register) that must be reloaded on context switch

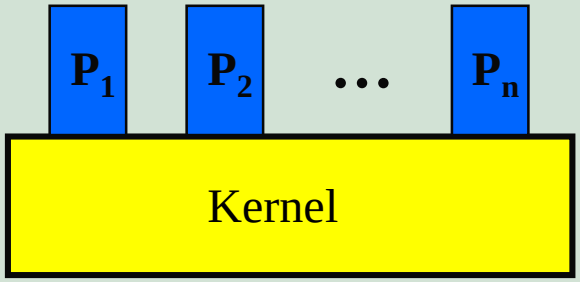
The Cost of Context Switching

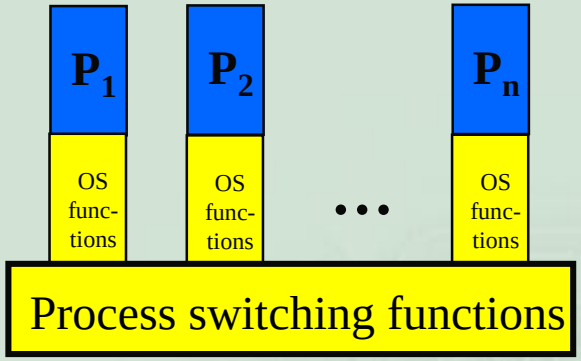
- context switching represents a substantial *cost* to the system in terms of CPU time and can, in fact, be the most costly operation on an operating system.
- a major focus in the design of operating systems has been to avoid unnecessary context switching to the extent possible.

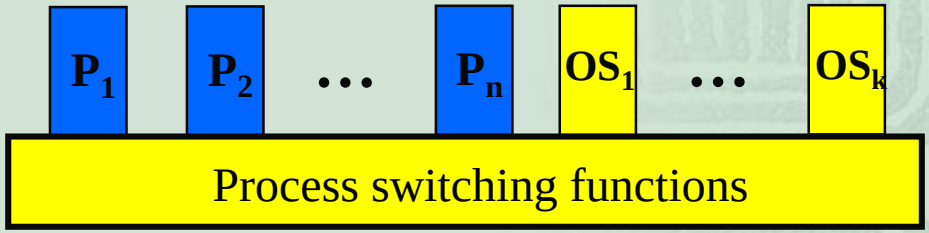
Execution of the Operating System

- Non-process Kernel
  - execute kernel outside of any process
- Execution Within User Processes
  - operating system software within context of a user process
- Process-Based Operating System
  - major OS functions are separate processes

Where does the OS execute?

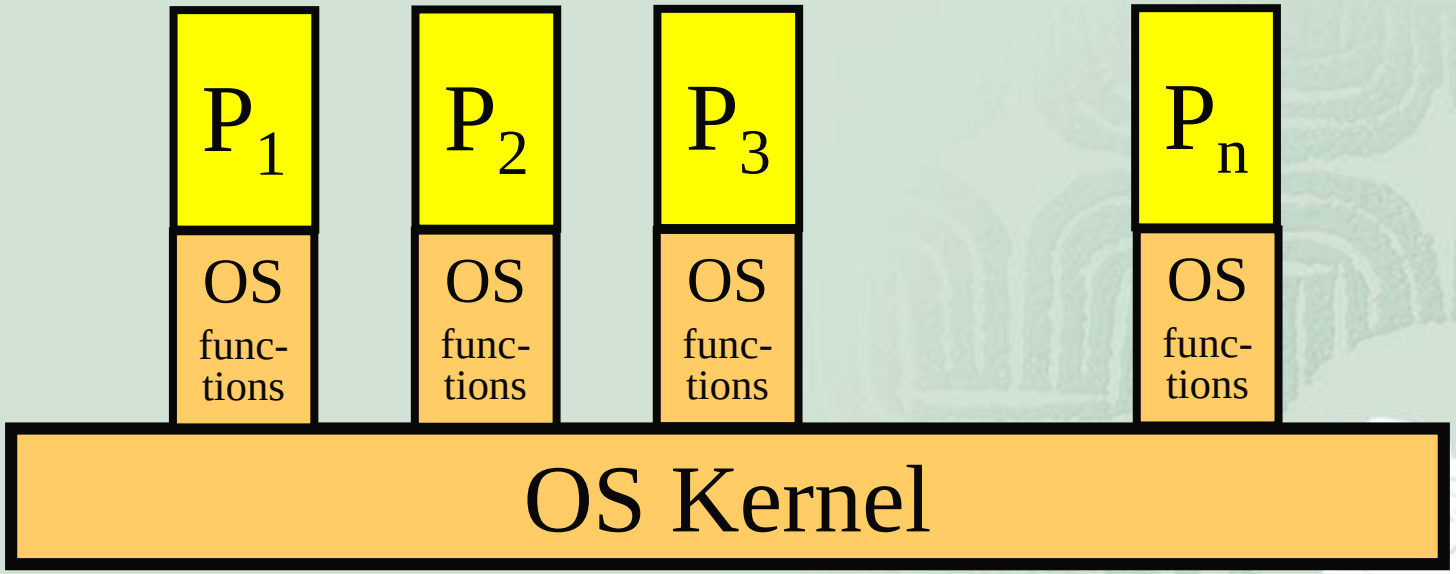






UNIX SVR4 Process Management

- operating system executes within the environment of a user process in order to reduce the cost of context switching



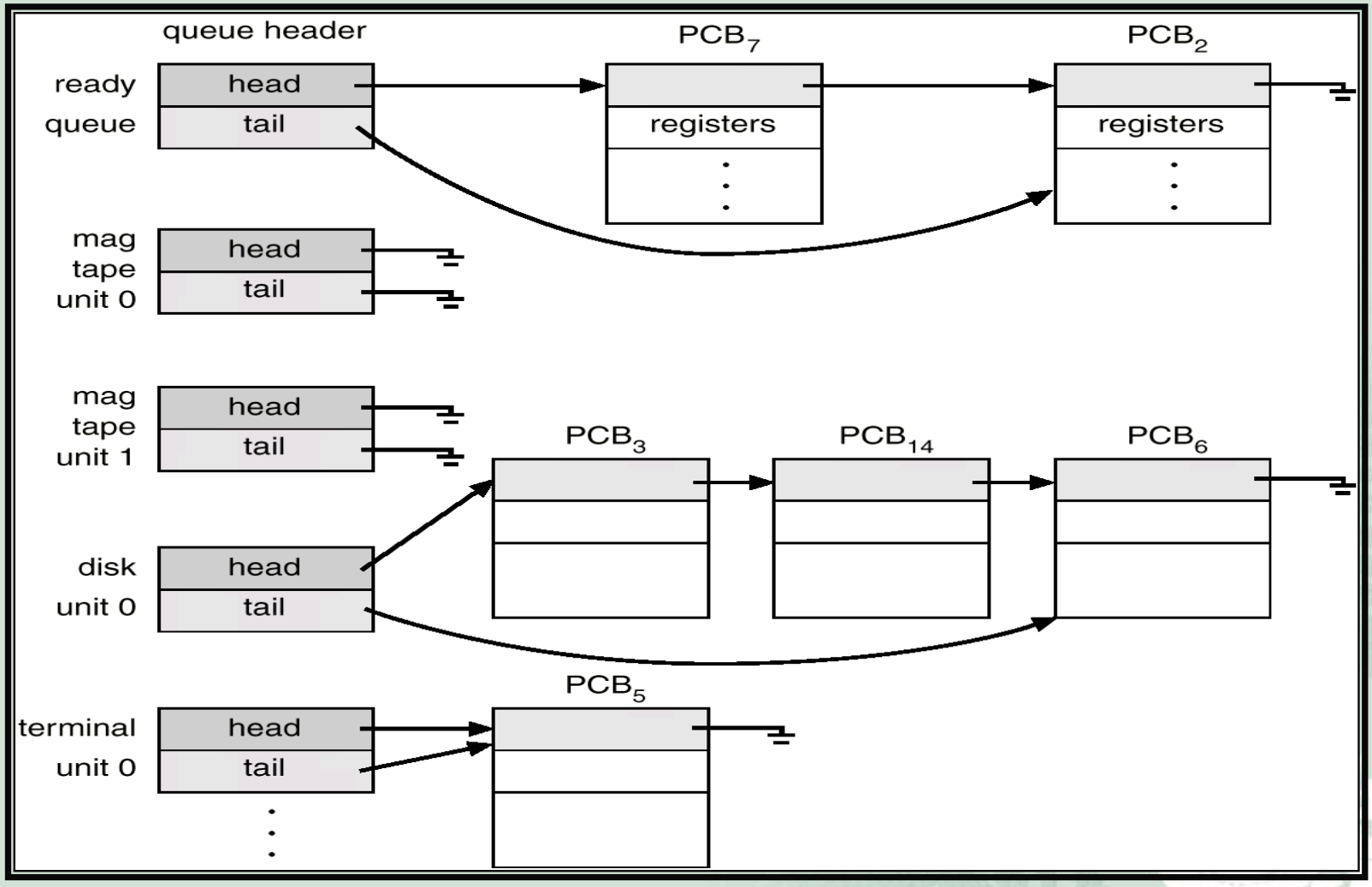
Process Control

- Process Control Block (PCB)
- Process Execution
- Context Switch
- Process Scheduling

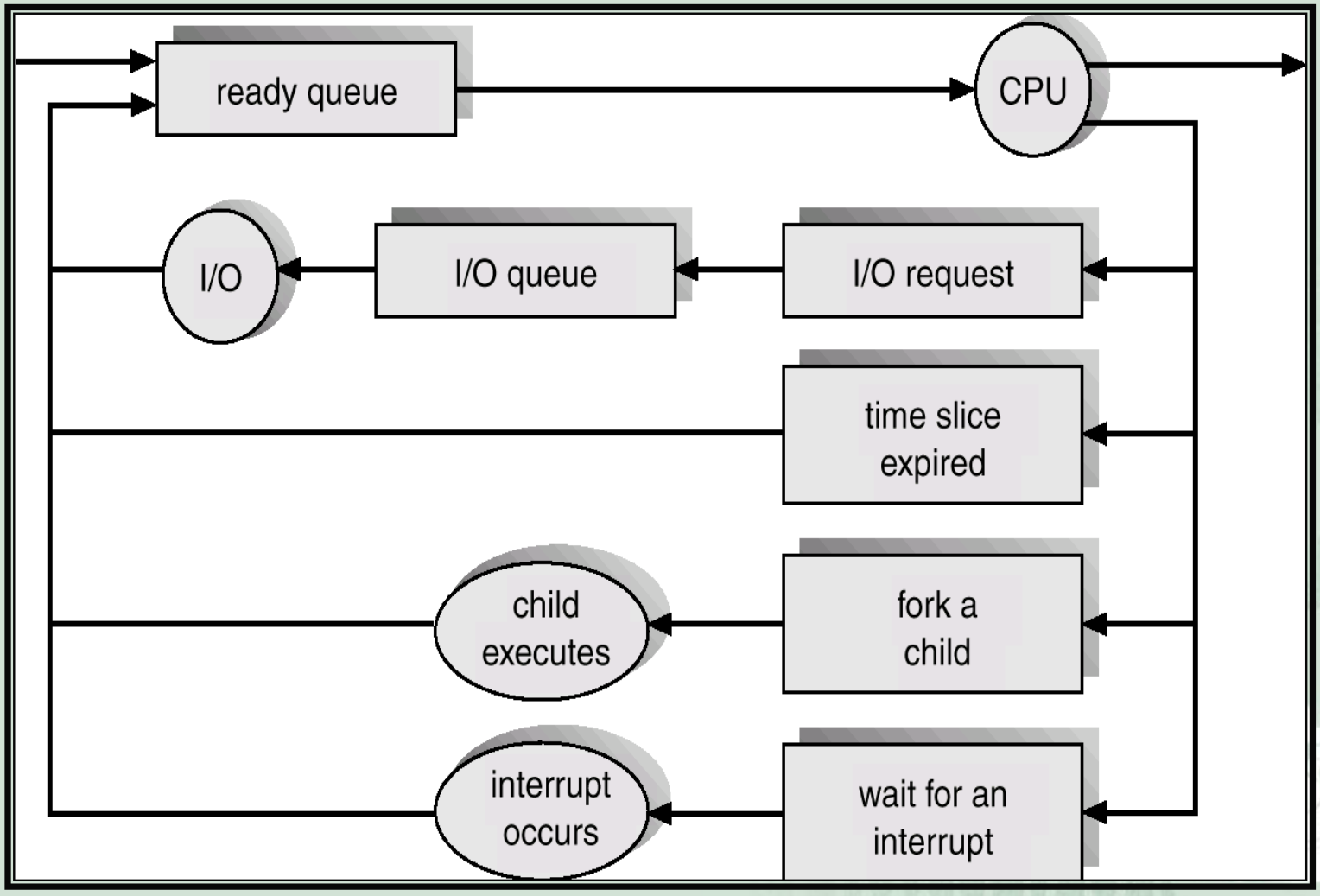
Process Scheduling Queues

- Job queue – set of all processes in the system.
- Ready queue – set of all processes residing in main memory, ready and waiting to execute.
- Device queues – set of processes waiting for an I/O device.
- Process migration between the various queues.

Ready Queue And Various I/O Device Queues



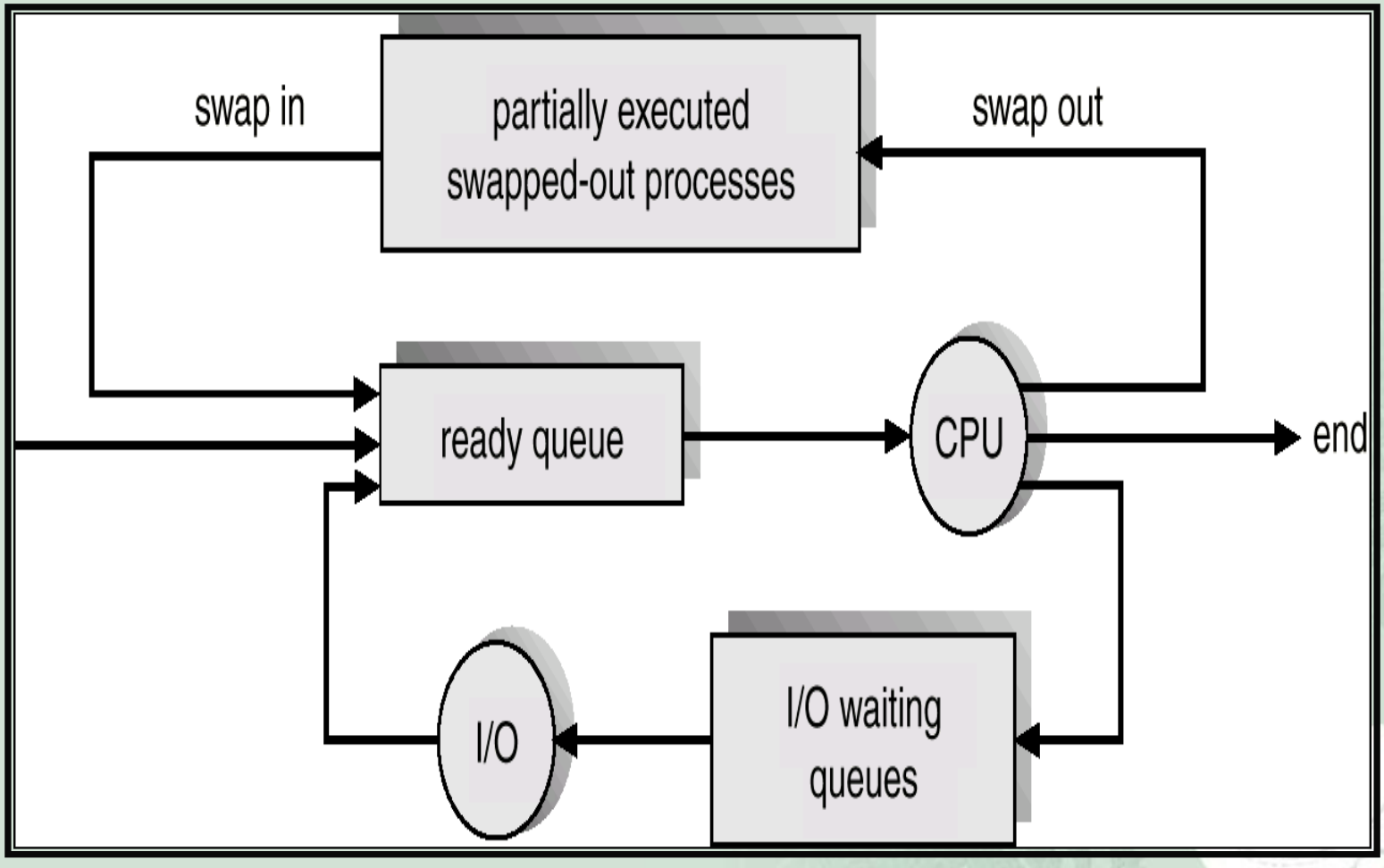
Representation of Process Scheduling



Schedulers

- Long-term scheduler (or job scheduler) – selects which processes should be brought into the ready queue.
- Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU.

Addition of Medium Term Scheduling



Schedulers (Cont.)

- Short-term scheduler is invoked very frequently (milliseconds) $\Rightarrow$ (must be fast).
- Long-term scheduler is invoked very infrequently (seconds, minutes) $\Rightarrow$ (may be slow).
- The long-term scheduler controls the *degree of multiprogramming.*
- Processes can be described as either:
  - I/O-*bound process* – spends more time doing I/O than computations, many short CPU bursts.
  - *CPU-bound process* – spends more time doing computations; few very long CPU bursts.

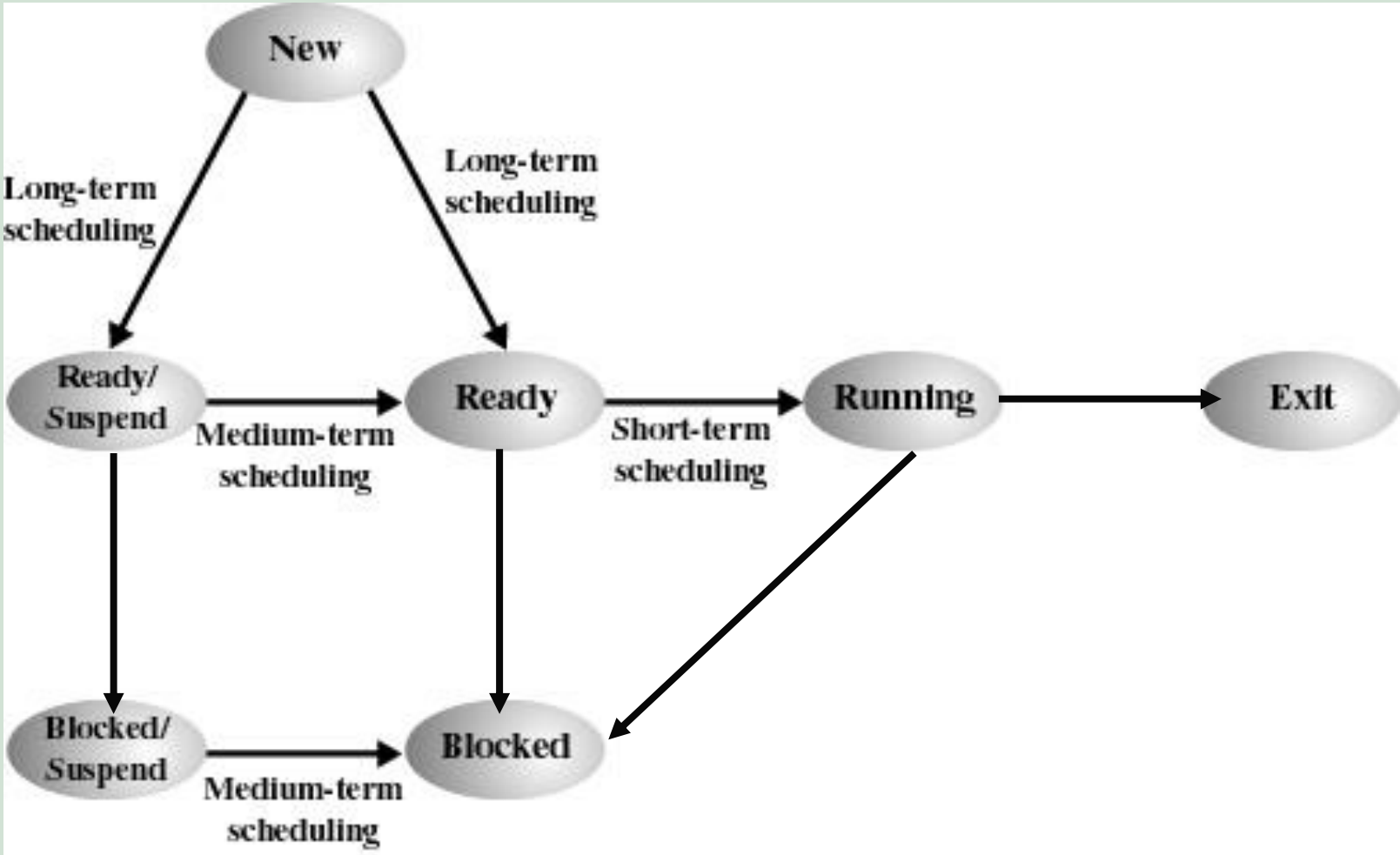


Figure 9.1 Scheduling and Process State Transitions

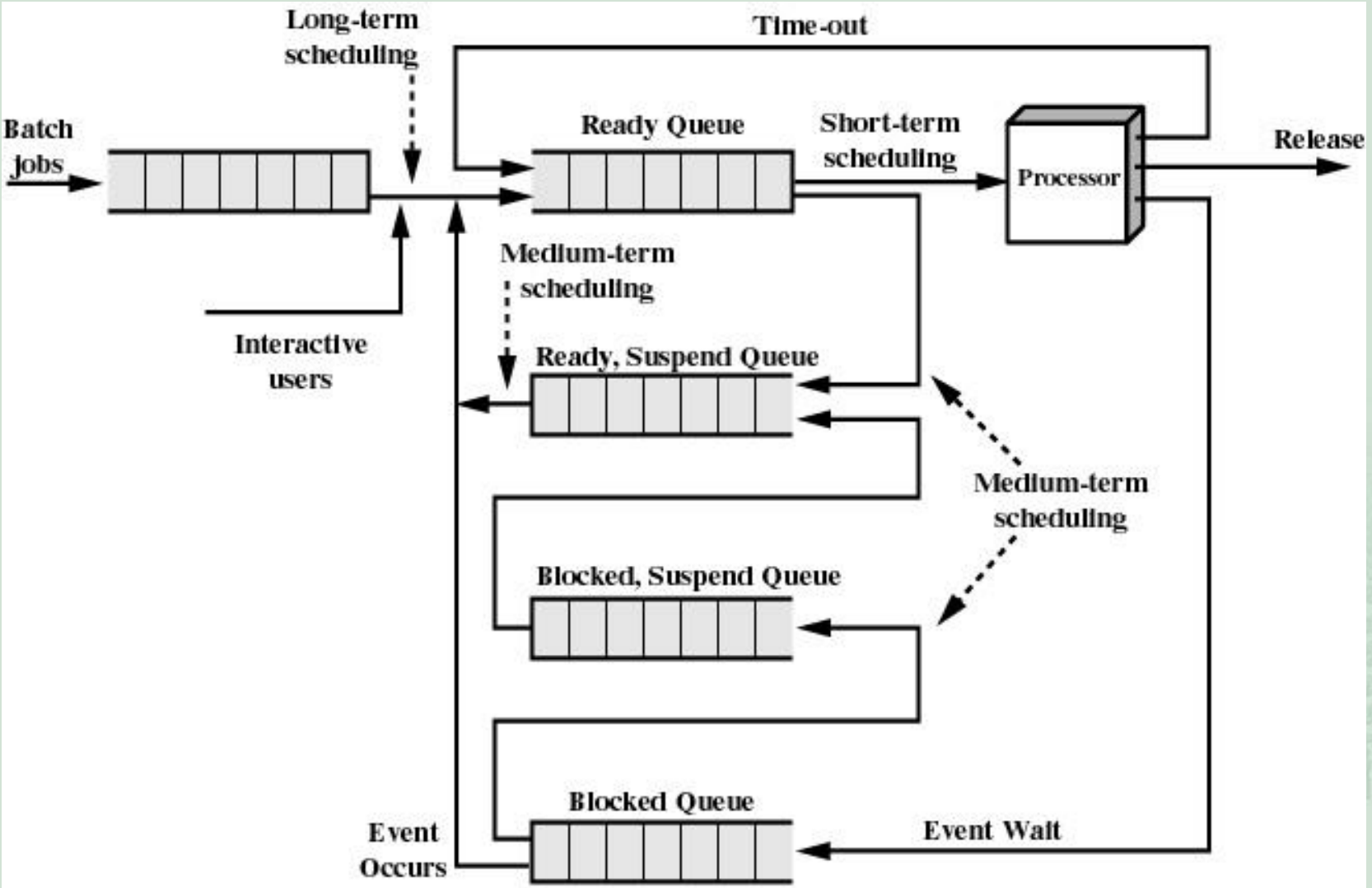


Figure 9.3 Queuing Diagram for Scheduling

UNIX Process States

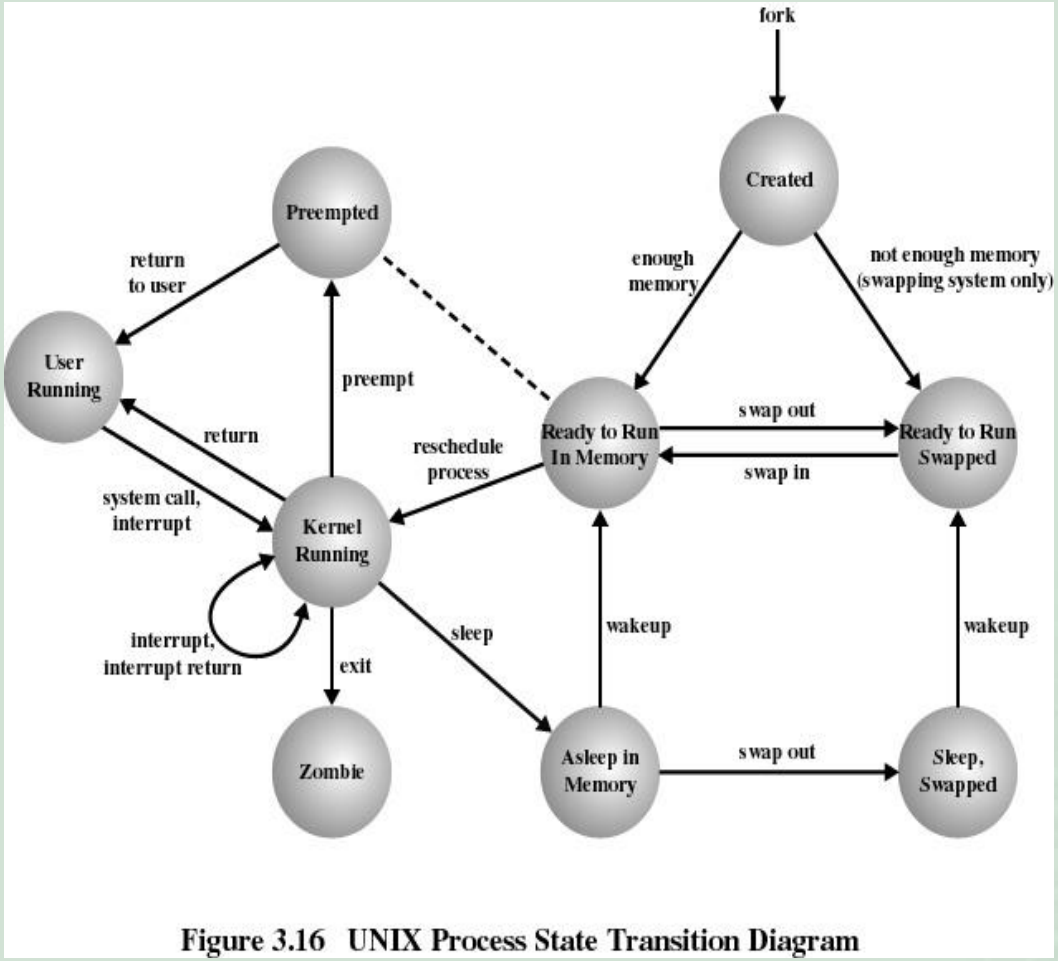


Figure 3.16 UNIX Process State Transition Diagram

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.