

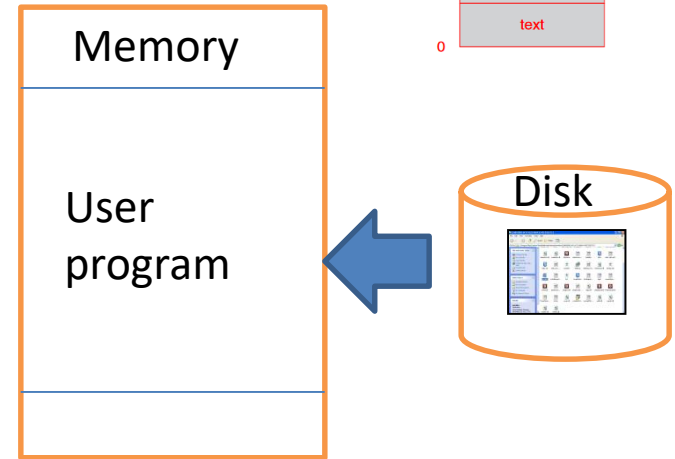
Process management

What are we going to learn?

- ***Processes*** : Concept of processes, process scheduling, co-operating processes, inter-process communication.
- ***CPU scheduling*** : scheduling criteria, preemptive & non-preemptive scheduling, scheduling algorithms (FCFS, SJF, RR, priority), algorithm evaluation, multi-processor scheduling.
- ***Process Synchronization*** : background, critical section problem, critical region, synchronization hardware, classical problems of synchronization, semaphores.
- ***Threads*** : overview, benefits of threads, user and kernel threads.
- ***Deadlocks*** : system model, deadlock characterization, methods for handling deadlocks, deadlock prevention, deadlock avoidance, deadlock detection, recovery from deadlock.

Process concept

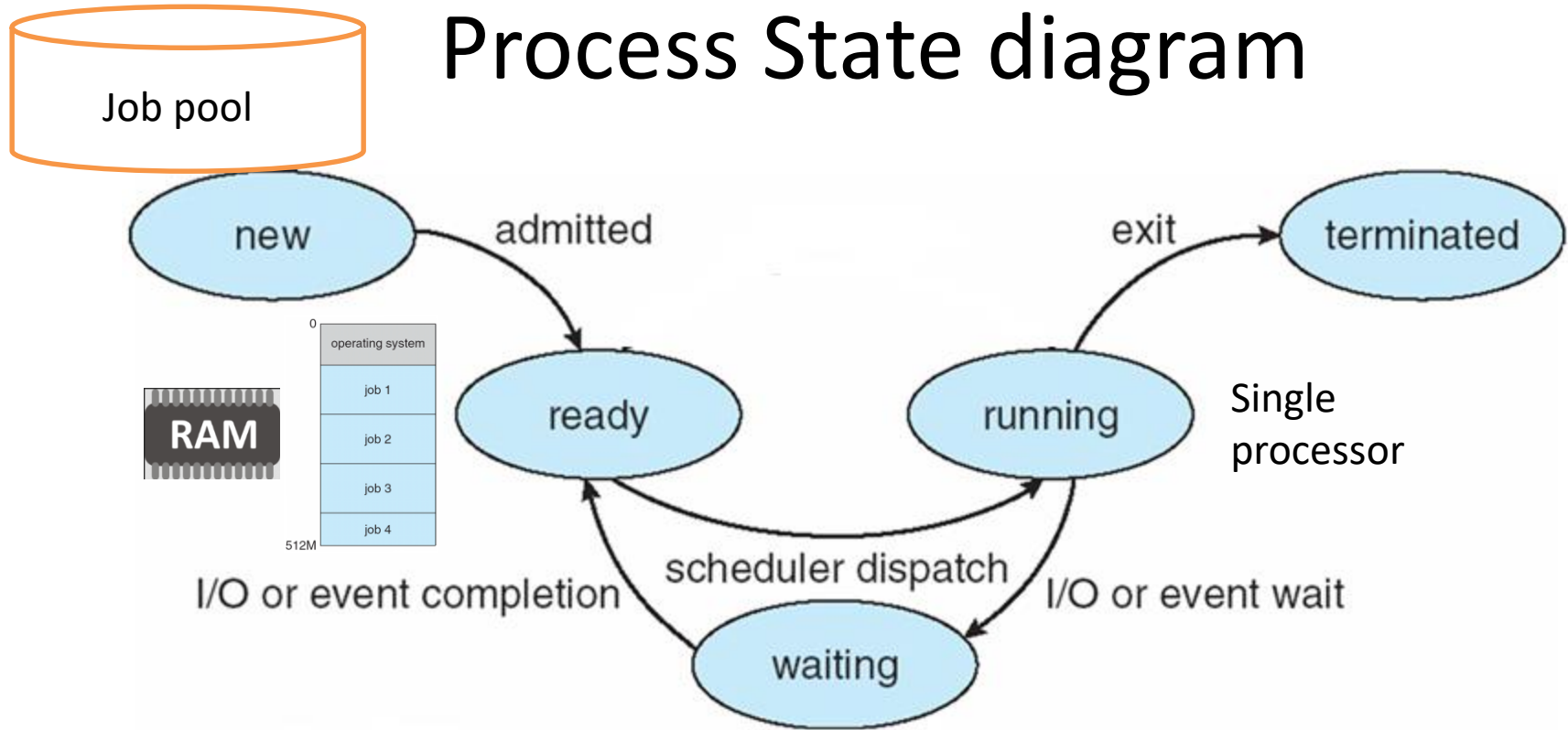
- Process is a dynamic entity
 - Program in execution
- Program code
 - Contains the text section
- Program becomes a process when
 - executable file is loaded in the memory
 - Allocation of various resources
 - Processor, register, memory, file, devices
- One program code may create several processes
 - One user opened several MS Word
 - Equivalent code/text section
 - Other resources may vary



Process State

- As a process executes, it changes *state*
 - **new**: The process is being created
 - **ready**: The process is waiting to be assigned to a processor
 - **running**: Instructions are being executed
 - **waiting**: The process is waiting for some event to occur
 - **terminated**: The process has finished execution

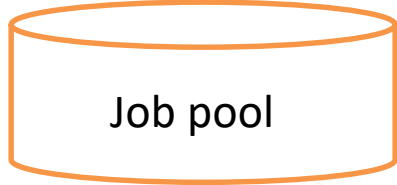
Process State diagram



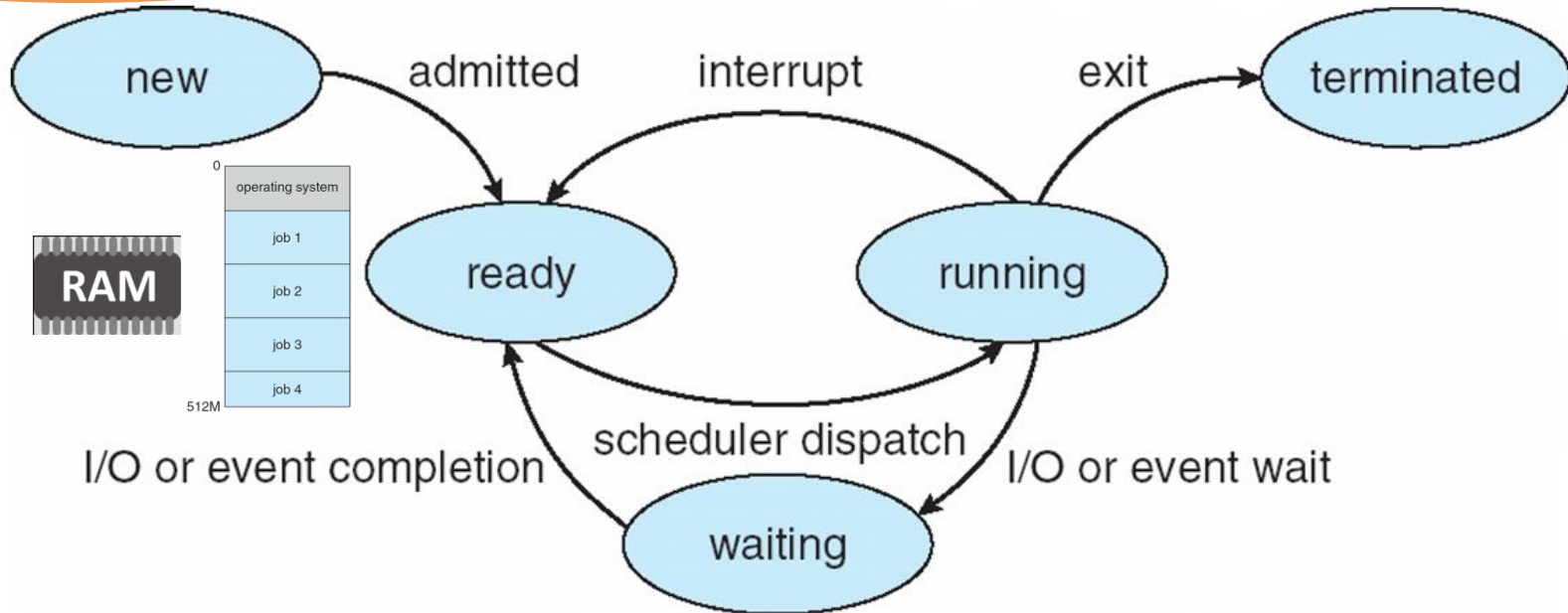
Multiprogramming

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



Process State diagram




Multitasking/Time sharing

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

How to represent a process?

- Process is a dynamic entity
 - Program in execution
- Program code
 - Contains the text section
- Program counter (PC)
- Values of different registers
 - Stack pointer (SP) (maintains process stack)
 - Return address, Function parameters
 - Program status word (PSW) 
 - General purpose registers
- Main Memory allocation
 - Data section
 - Variables
 - Heap
 - Dynamic allocation of memory during process execution

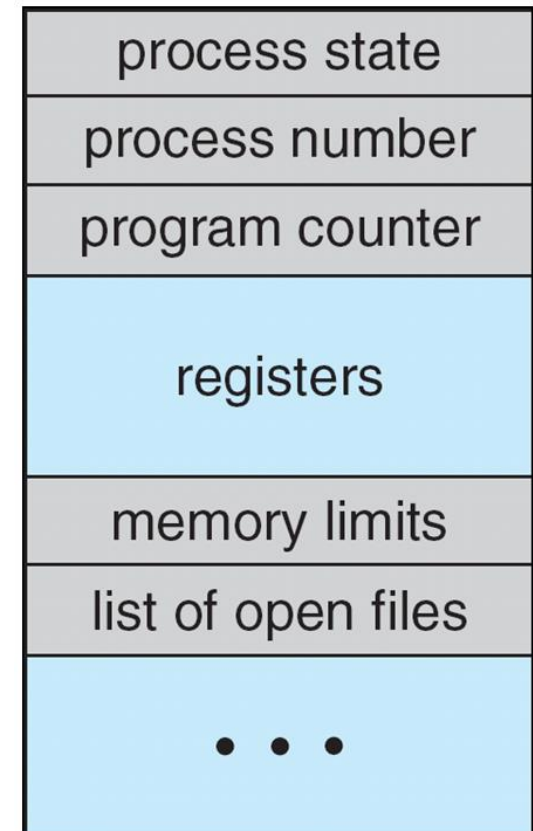


Process Control Block (PCB)

- Process is represented in the operating system by a Process Control Block

Information associated with each process

- Process state
- Program counter
- CPU registers
 - Accumulator, Index reg., stack pointer, general Purpose reg., Program Status Word (PSW)
- CPU scheduling information
 - Priority info, pointer to scheduling queue
- Memory-management information
 - Memory information of a process
 - Base register, Limit register, page table, segment table
- Accounting information
 - CPU usage time, Process ID, Time slice
- I/O status information
 - List of open files=> file descriptors
 - Allocated devices

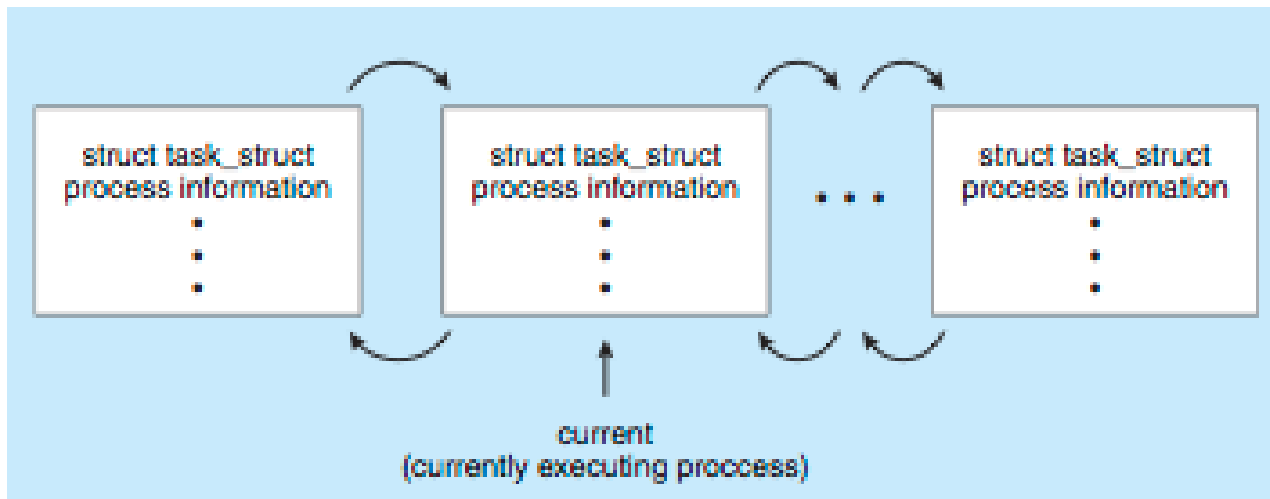


Process Representation in Linux

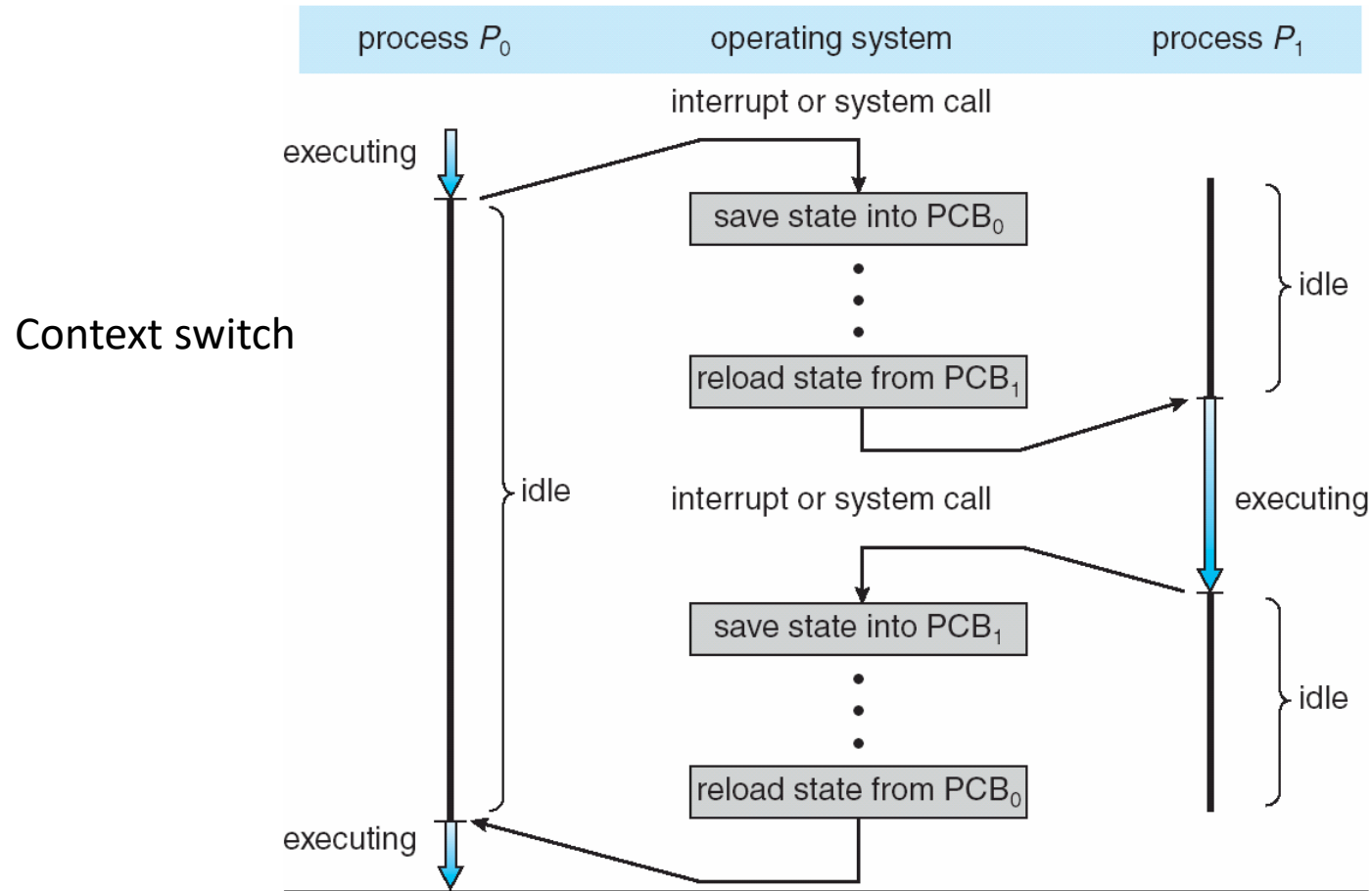
Represented by the C structure `task_struct`

```
pid_t pid; /* process identifier */
long state; /* state of the process */
unsigned int time_slice /* scheduling information */
struct task_struct *parent; /* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files; /* list of open files */
struct mm_struct *mm; /* address space of this pro */
```

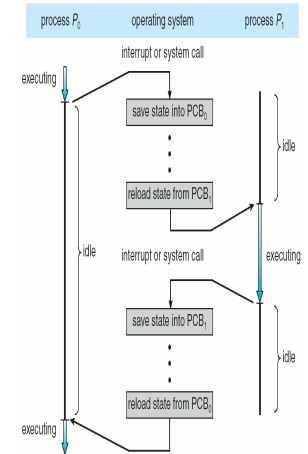
Doubly
linked list



CPU Switch From Process to Process



Context Switch

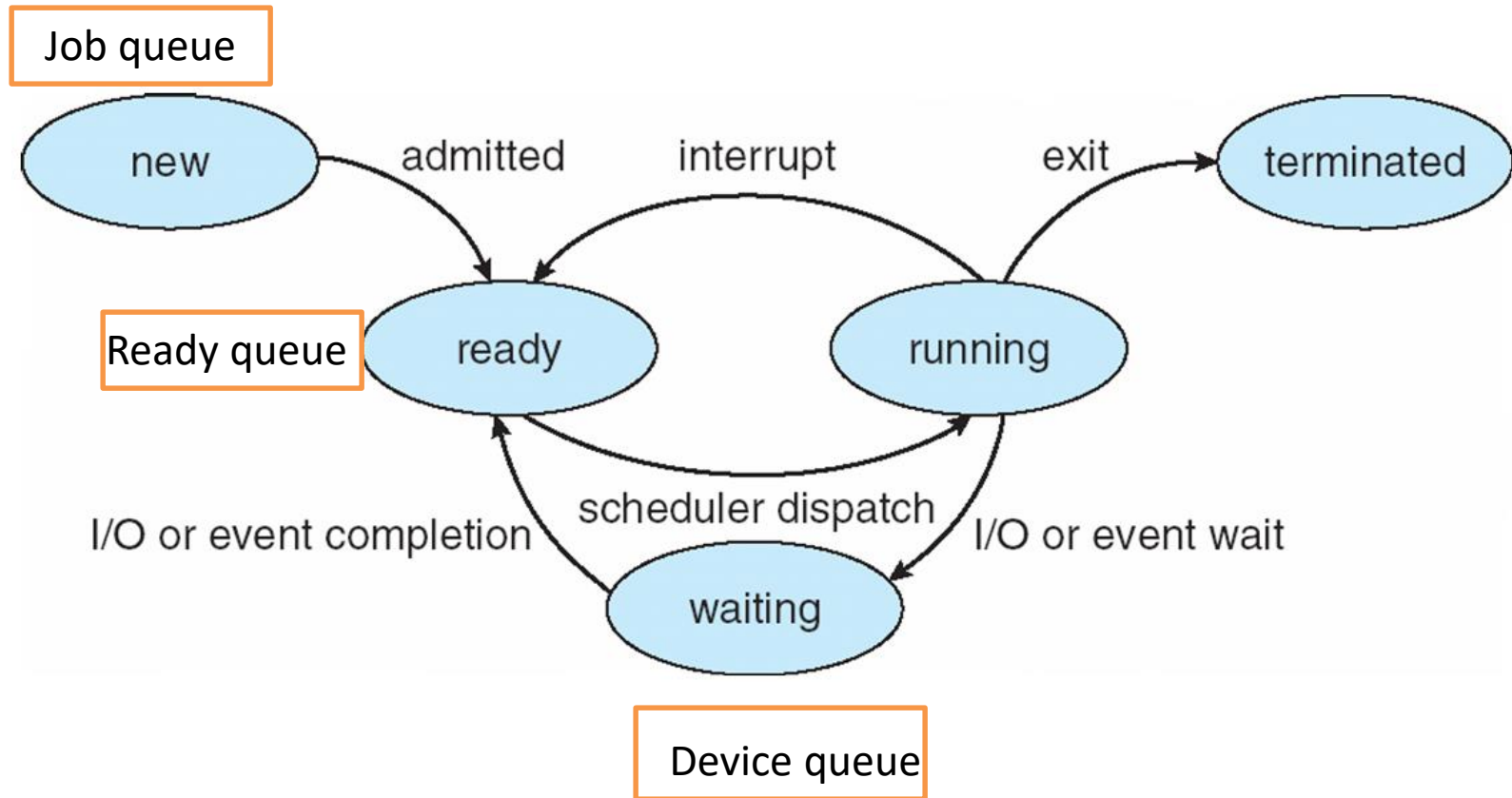


- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a **context switch**.
- **Context** of a process represented in the PCB
- Context-switch time is overhead; the system does not do useful work while switching
 - The more complex the OS and the PCB -> longer the context switch
- Time dependent on hardware support
 - Some hardware provides multiple sets of registers per CPU -> multiple contexts loaded at once

Scheduling queues

- Maintains **scheduling queues** of processes
 - **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
- Processes migrate among the various queues

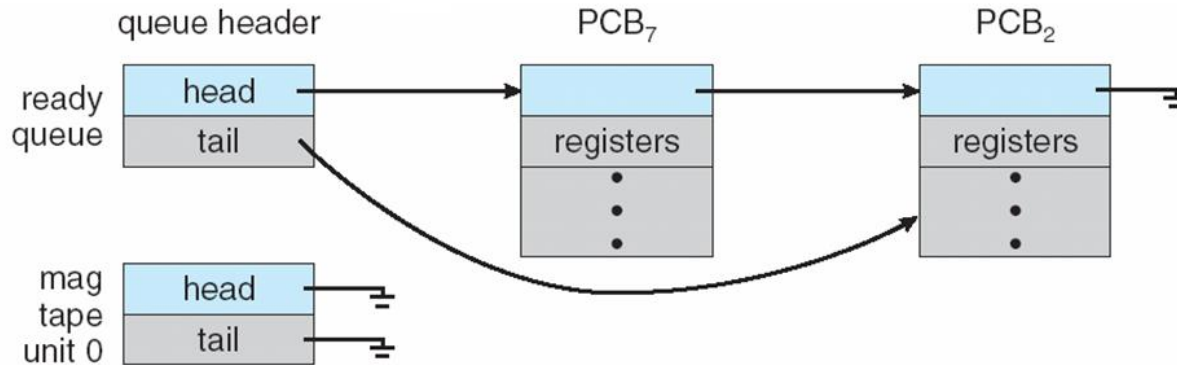
Scheduling queues



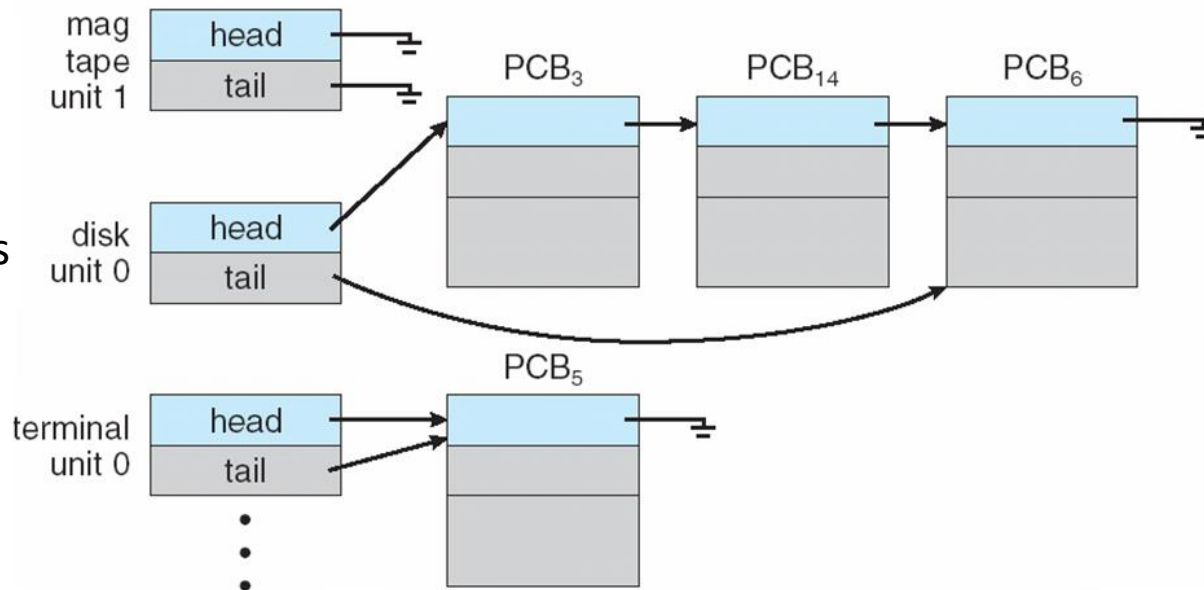
Ready Queue And Various I/O Device Queues

Queues are linked list of PCB's

Device queue



Many processes are waiting for disk



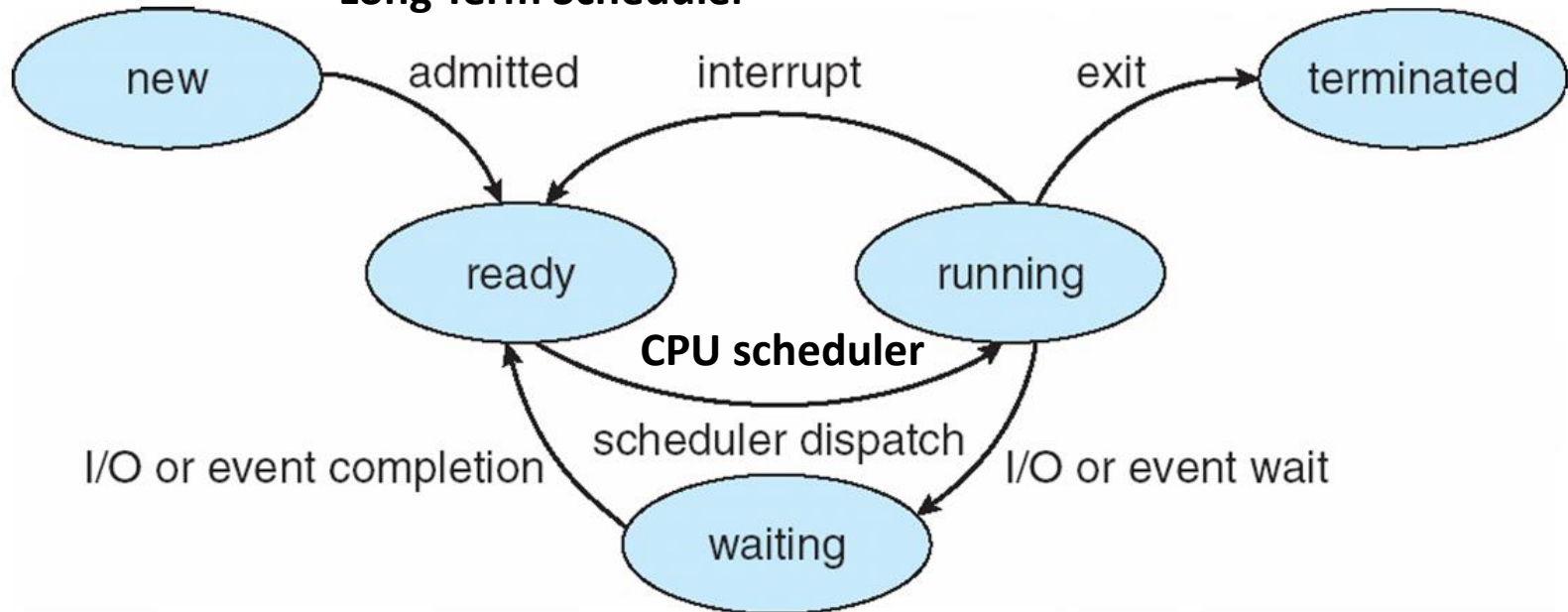
Process Scheduling

- We have various queues
- Single processor system
 - Only one CPU=> only one running process
- Selection of one process from a group of processes
 - **Process scheduling**

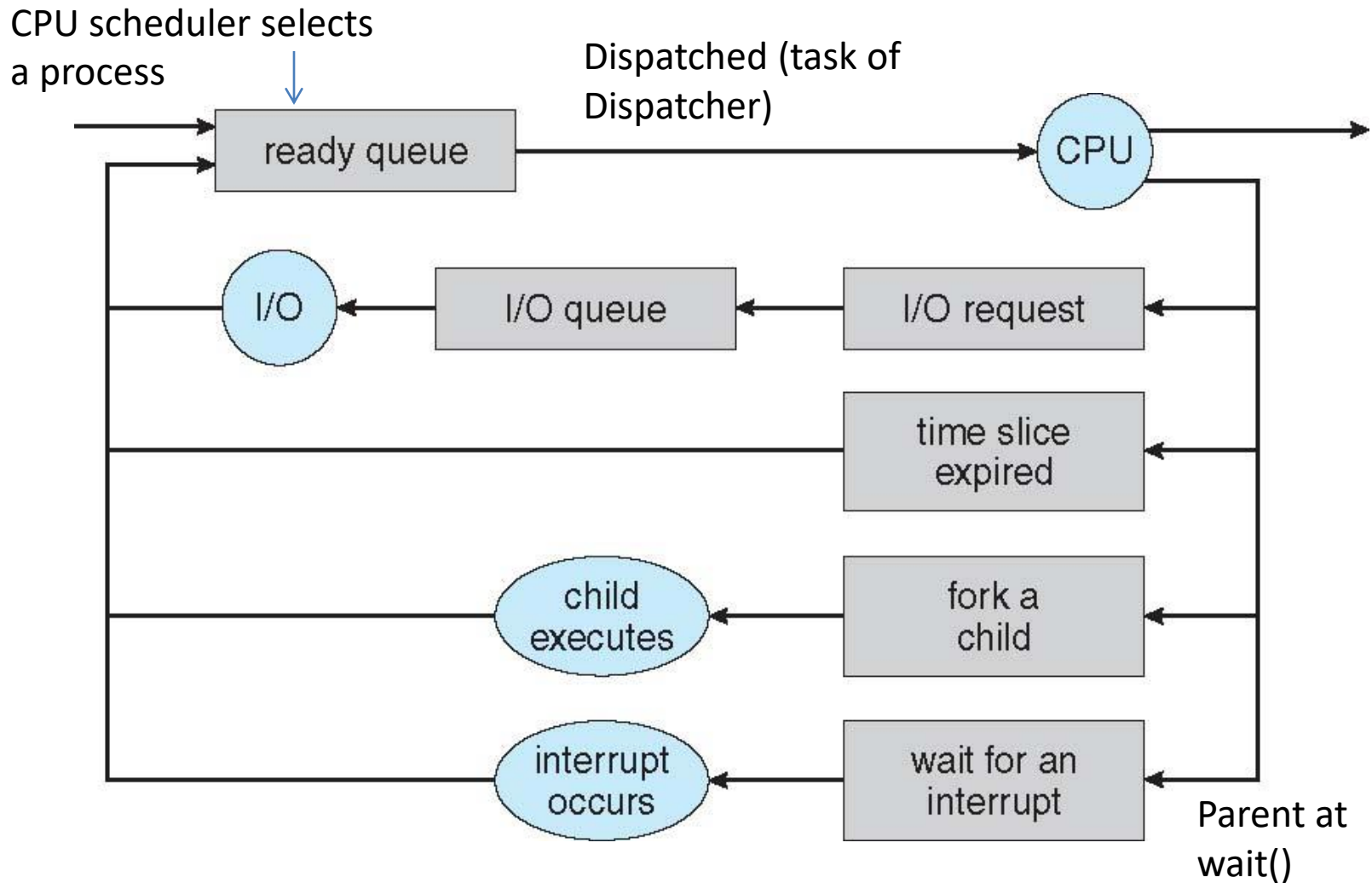
Process Scheduling

- Scheduler
 - Selects a process from a set of processes
- Two kinds of schedulers
 1. Long term schedulers, job scheduler
 - A large number of processes are submitted (more than memory capacity)
 - Stored in disk
 - Long term scheduler selects process from job pool and loads in memory
 2. Short term scheduler, CPU scheduler
 - Selects one process among the processes in the memory (ready queue)
 - Allocates to CPU

Long Term Scheduler



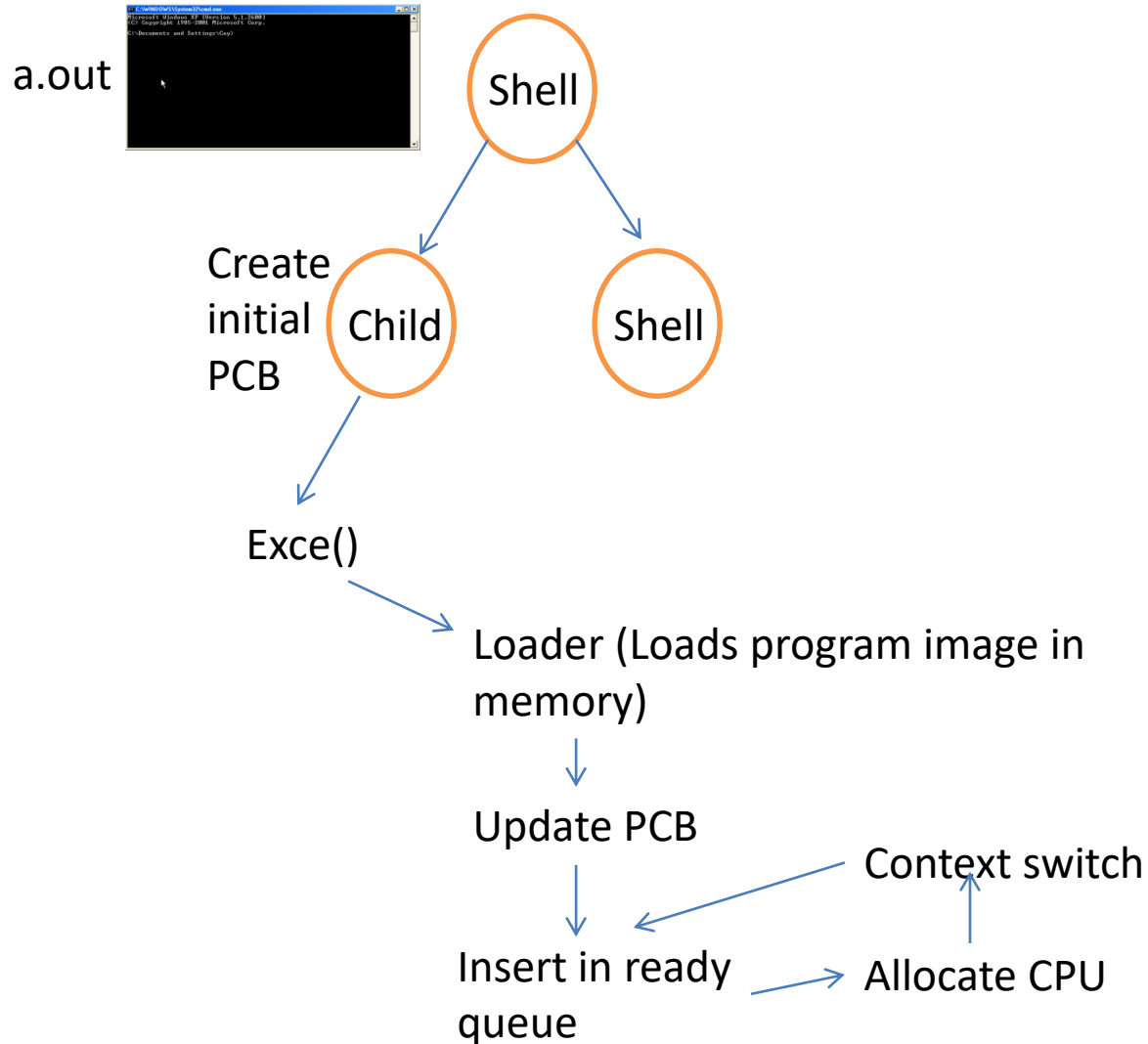
Representation of Process Scheduling



Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
 - switching context
 - switching to user mode
 - jumping to the proper location in the user program to restart that program
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running

Creation of PCB



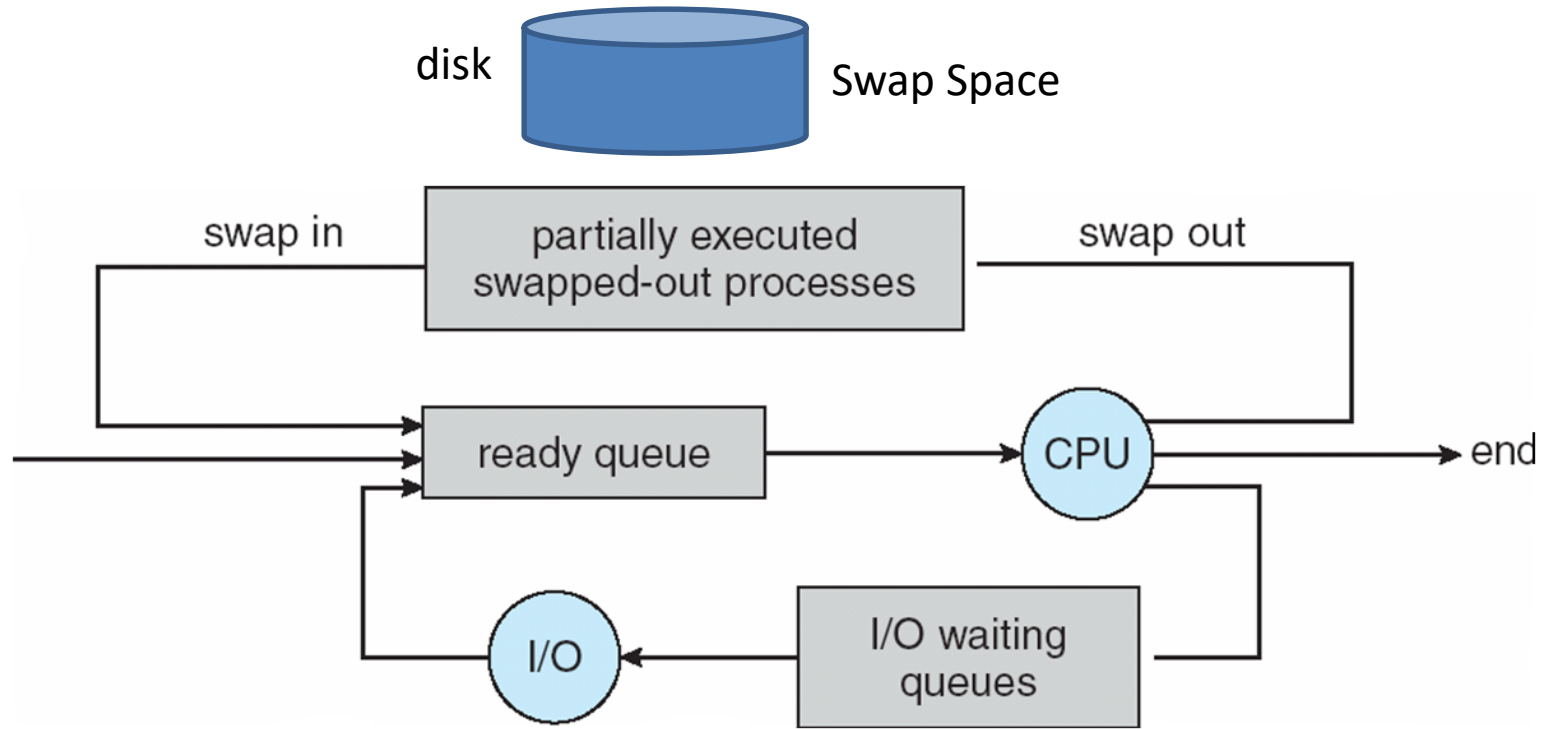
Schedulers

- **Scheduler**
 - Selects a process from a set
- **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
 - Sometimes the only scheduler in a system

Schedulers: frequency of execution

- Short-term scheduler is invoked **very frequently** (milliseconds)
⇒ (must be fast)
 - After a I/O request/ Interrupt
- Long-term scheduler is invoked very infrequently (seconds, minutes) ⇒ (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
 - Ready queue empty
 - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
 - Devices unused
- Long term scheduler ensures good process mix of I/O and CPU bound processes.

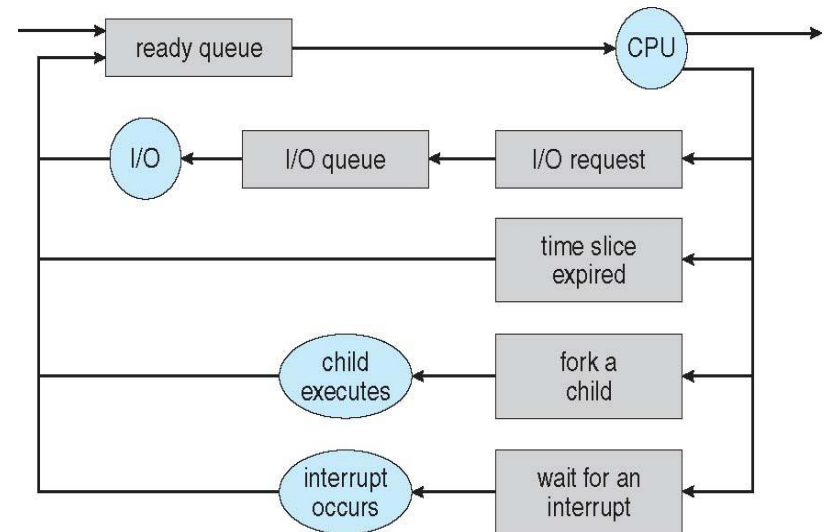
Addition of Medium Term Scheduling



Swapper

ISR for context switch

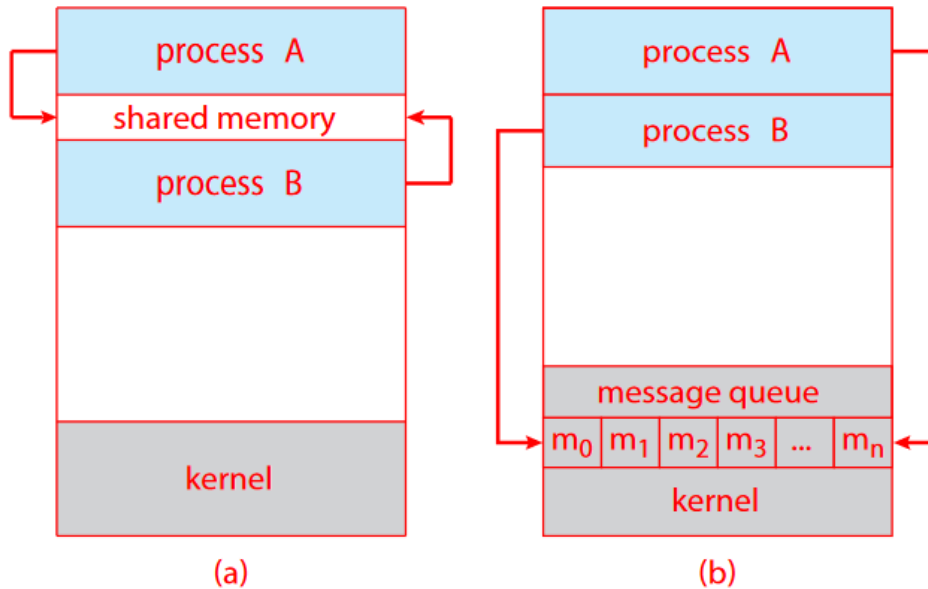
```
Current <- PCB of current process
Context_switch()
{
    Disable interrupt;
    switch to kernel mode
    Save_PCB(current);
    Insert(ready_queue, current);
    next=CPU_Scheduler(ready_queue);
    remove(ready_queue, next);
    Dispatcher(next);
    switch to user mode;
    Enable Interrupt;
}
Dispatcher(next)
{
    Load_PCB(next); [update PC]
}
```



Interprocess Communication

- Processes within a system may be **independent** or **cooperating**
- Cooperating process can affect or be affected by other processes, including sharing data
- Cooperating processes require an interprocess communication (IPC) mechanism that will allow them to exchange data— that is, send data to and receive data from each other.
- Cooperating processes need **interprocess communication (IPC)**
- Two models of IPC
 - Shared memory
 - Message passing

Interprocess Communication



In the **shared-memory model**, a **region of memory** that is shared by the cooperating processes is established.

Processes can then exchange information by reading and writing data to the shared region.

In the **message-passing model**, communication takes place by means of messages exchanged between the cooperating processes
(Kernel involvement, slow)

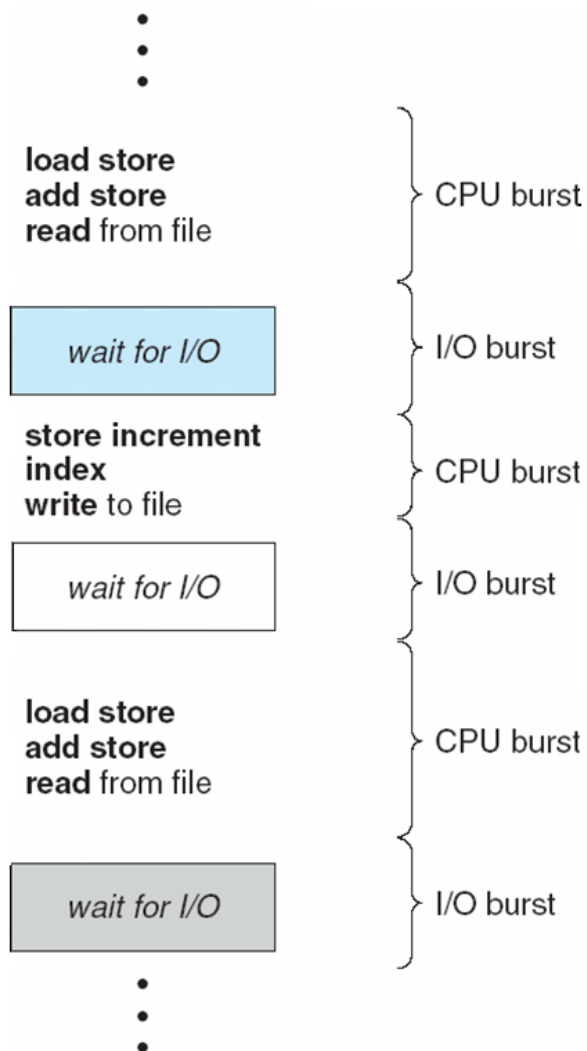
CPU Scheduling

- Describe various CPU-scheduling algorithms
- Evaluation criteria for selecting a CPU-scheduling algorithm for a particular system

Basic Concepts

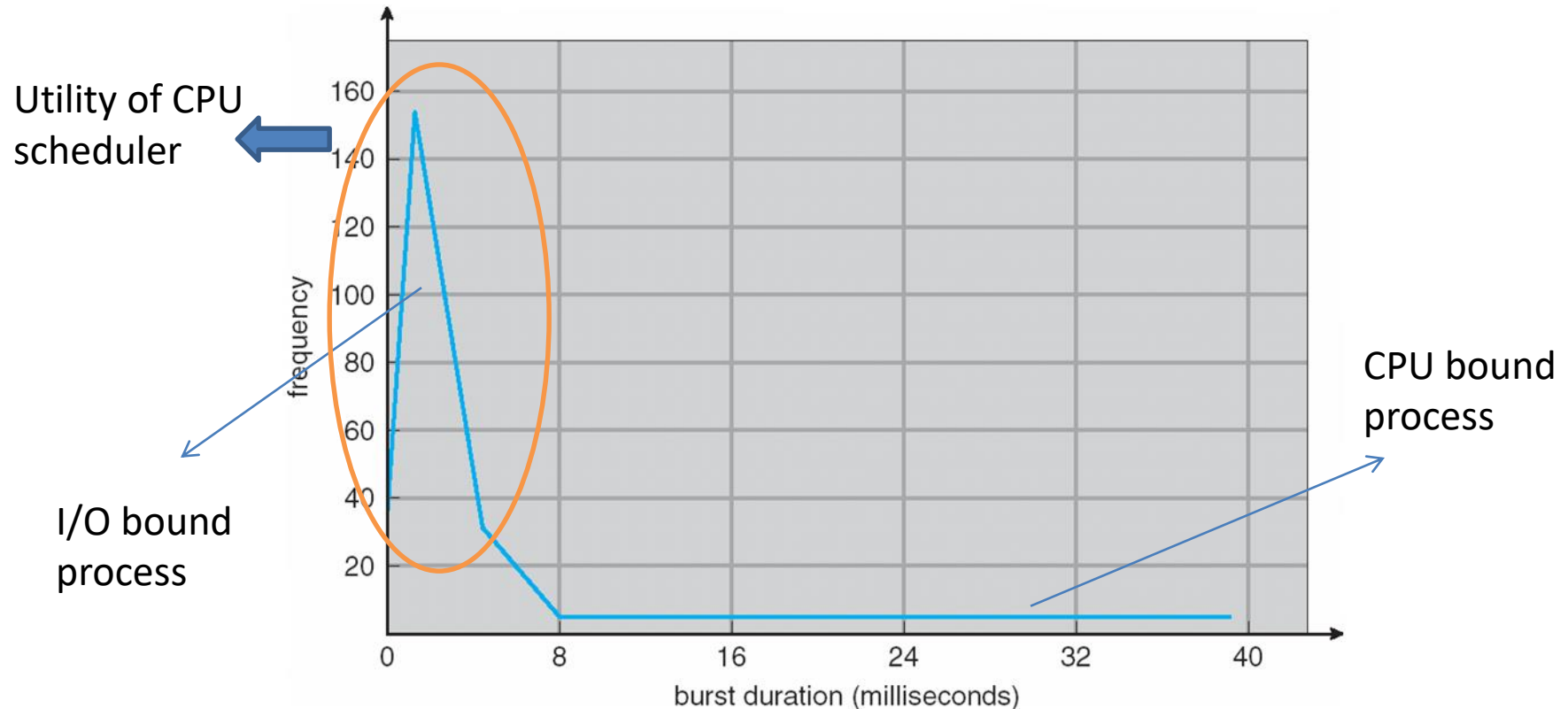
- Maximum CPU utilization obtained with multiprogramming
 - Several processes in memory (ready queue)
 - When one process requests I/O, some other process gets the CPU
 - Select (schedule) a process and allocate CPU

Observed properties of Processes



- CPU–I/O Burst Cycle
- Process execution consists of a *cycle* of CPU execution and I/O wait
- Study the duration of CPU bursts

Histogram of CPU-burst Times

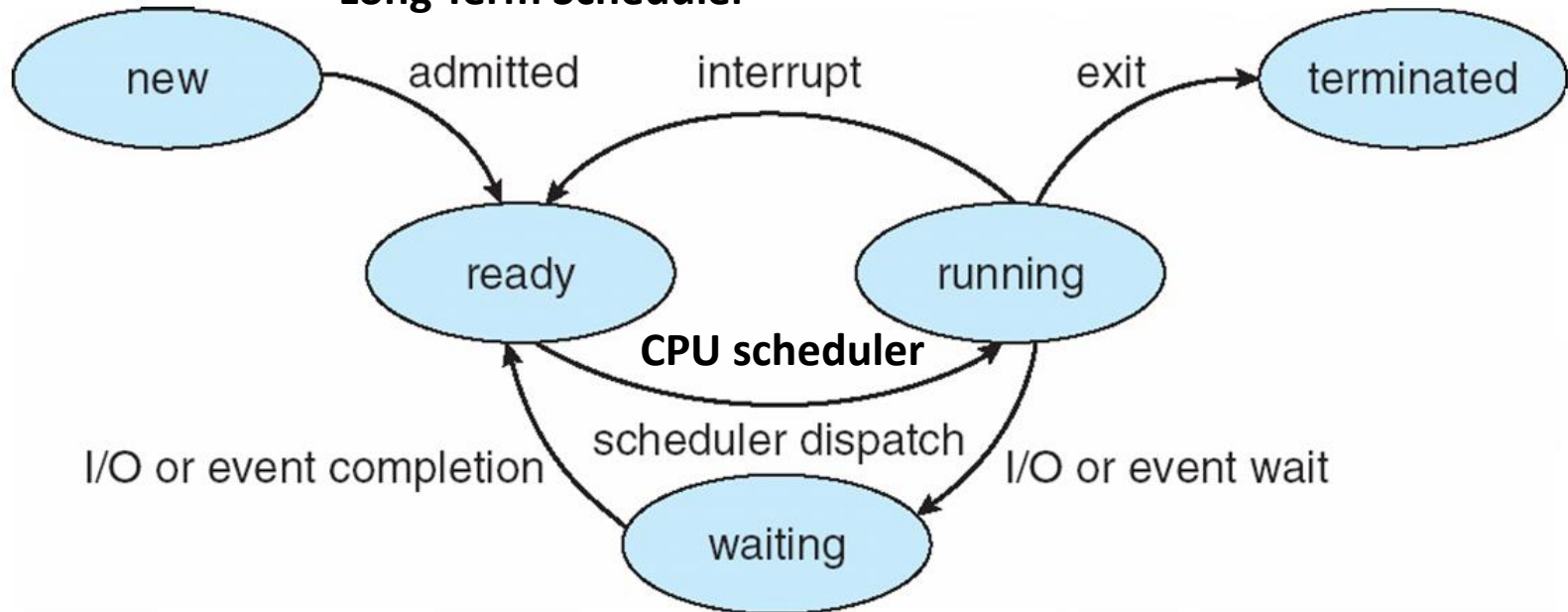


Large number of short CPU bursts and small number of long CPU bursts

Preemptive and non preemptive

- Selects from among the processes in ready queue, and allocates the CPU to one of them
 - Queue may be ordered in various ways (not necessarily FIFO)
- CPU scheduling decisions may take place when a process:
 1. Switches from running to waiting state
 2. Switches from running to ready state
 3. Switches from waiting to ready
 4. Terminates
- Scheduling under 1 and 4 is **nonpreemptive**
- All other scheduling is **preemptive**

Long Term Scheduler



Preemptive scheduling

Preemptive scheduling

Results in cooperative processes

Issues:

- Consider access to shared data
 - Process synchronization
- Consider preemption while in kernel mode
 - Updating the ready or device queue
 - Preempted and running a “ps -el”

Race condition

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output

Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

- Mostly optimize the average
- Sometimes optimize the minimum or maximum value
 - Minimize max response time

- For interactive system, variance is important
 - E.g. response time
- System must behave in predictable way