Chapter Three

Processes

3.1. Process Management

A process can be thought of as a program in execution. A process will need certain resources such as CPU time, memory, files, and I/O devices to accomplish its task. These resources are allocated to the process either when it is created or while it is executing.

Early C/S allowed only one program to be executed at a time. This program had complete control of the system and had access to all of the system resources. Today C/S allows multiple programs to be loaded into memory and to be executed concurrently. The more complex the O.S the more it is expected to do on behalf of its users. A system therefore consists of a collection of processes:

O.S processes executing system code and user processes executing user code. By switching the CPU between processes the O.S can make the C/S more productive.

3.2 Process Concept

A question that arises in discussing operating systems involves what to call all the CPU activities. A batch system executes jobs, whereas a time-shared system has user programs, or tasks. Even on a single-user system, a user may be able to run several programs at one time: a word processor, a Web browser, and an e-mail package. And even if a user can execute only one program at a time, such as on an embedded device that does not support multitasking, the operating system may need to support its own internal programmed activities, such as memory management. In many respects, all these activities are similar, so

we call all of them processes. The terms job and process are used almost interchangeably in this text.

Although we personally prefer the term process, much of operating-system theory and terminology was developed during a time when the major activity of operating systems was job processing. It would be misleading to avoid the use of commonly accepted terms that include the word job (such as job scheduling) simply because process has superseded job. The execution of a process must progress in a sequential fashion. A process is more than the program code: sometimes known as the Text section, the value of program counter and the contents of the processor's registers.

A Process is a program in execution; process execution must progress in sequential fashion.

3.3 Process State

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states:

1. **New:** The process is being created.
2. **Running:** Instructions are being executed.
3. **Waiting:** The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
4. **Ready:** The process is waiting to be assigned to a processor.
5. **Terminated:** The process has finished execution.

These names are arbitrary, and they vary across operating systems. The states that they represent are found on all systems, however. Certain operating systems also more finely delineate process states. It is important to realize that only one process can be running on any processor at any instant. Many processes may be ready and waiting, however. The state diagram corresponding to these states is presented in Figure 3.1.

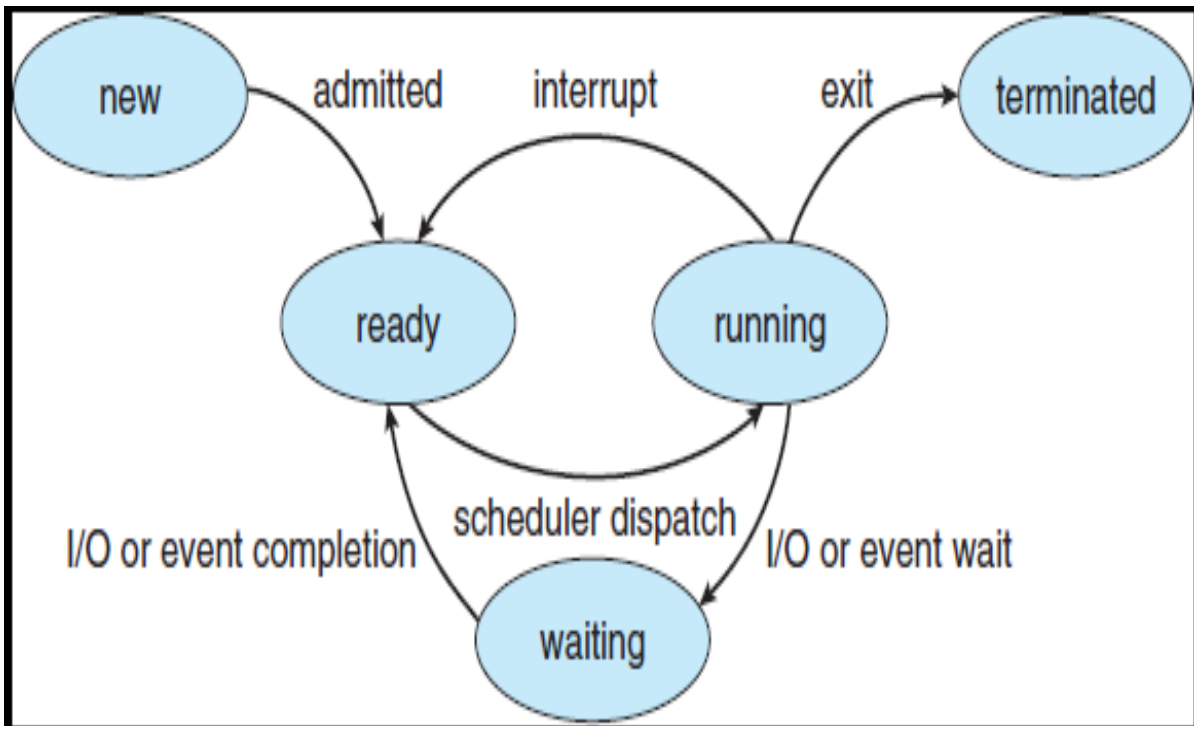

Figure 3.1 Diagram of process state.

3.4 Process Control Block

Each process is represented in the operating system by a **Process Control Block** (**PCB**), also called a **Task Control Block**. A **PCB** is shown in Figure 3.2. It contains many pieces of information associated with a specific process, including these:

- **Process state:** It maybe new, ready, running, waiting, halted, and so on.
- **Program counter:** the address of the next instruction to execute for the process.
- **CPU registers:** The registers include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information.
- **CPU-scheduling information:** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- **Memory-management information:** This information may include the value of the base and limit registers, the page tables, … etc.
- **Accounting information:** It includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- **I/O status information:** it includes the list of I/O devices allocated to the process, a list of open files, and so on.

The PCB simply serves as the repository for any information that may vary from process to process.

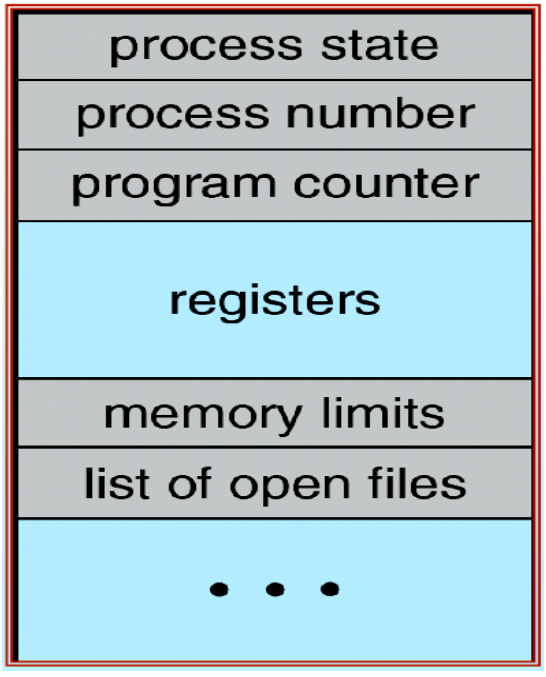

Figure 3.2 Process control block (PCB).

3.5 Process Scheduling

- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
- The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running. To meet these objectives, **the process scheduler** selects an available process (possibly from a set of several available processes) for program execution on the CPU.
- For a single-processor system, there will never be more than one running process. If there are more processes, the rest will have to wait until the CPU is free and can be rescheduled.

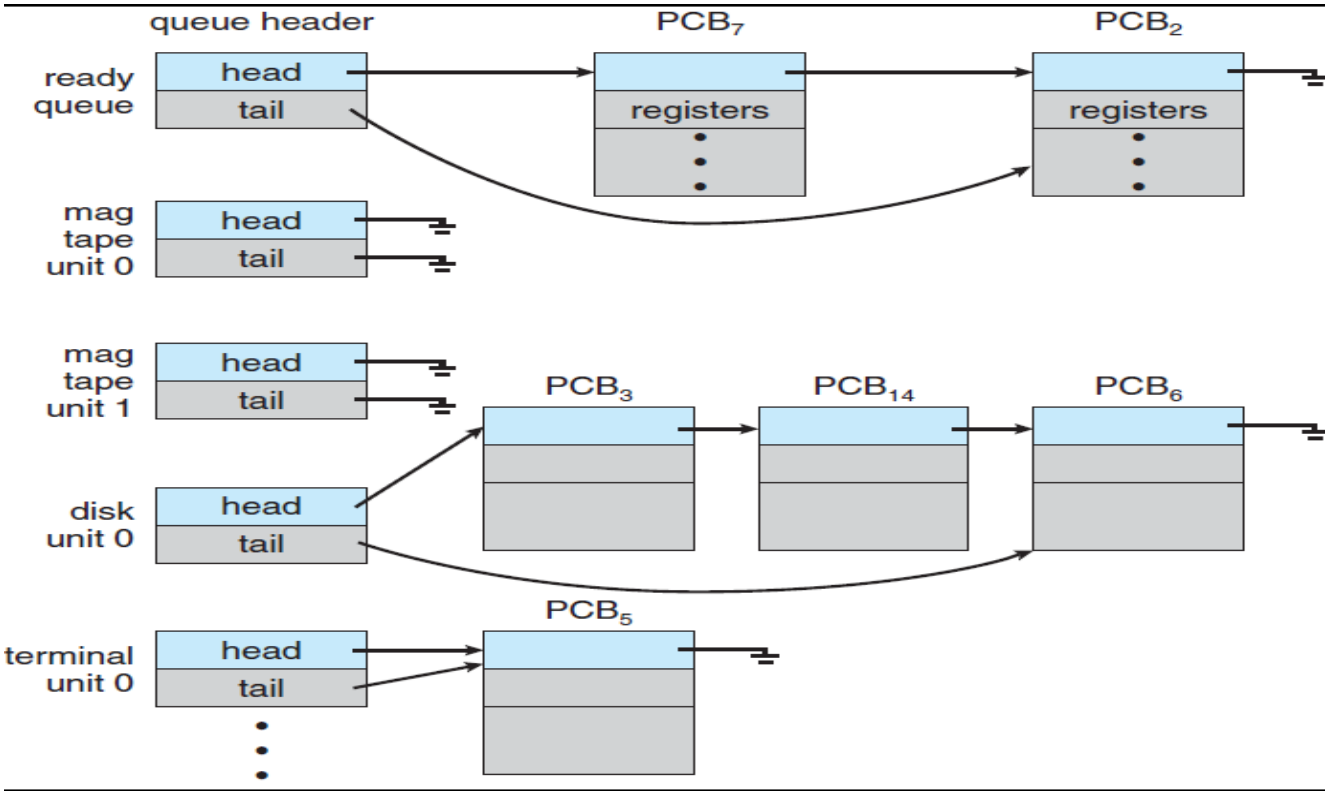

Figure 3.3 The ready queue and various I/O device queues

3.6 Scheduling Queues

As processes enter the system, They are put into a **Job queue**, which consists of all processes in the system.

- The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the **Ready queue**.

- Each PCB has a pointer field that points to the next process in the queue. Each device has its own device queue (Figure 3.4).

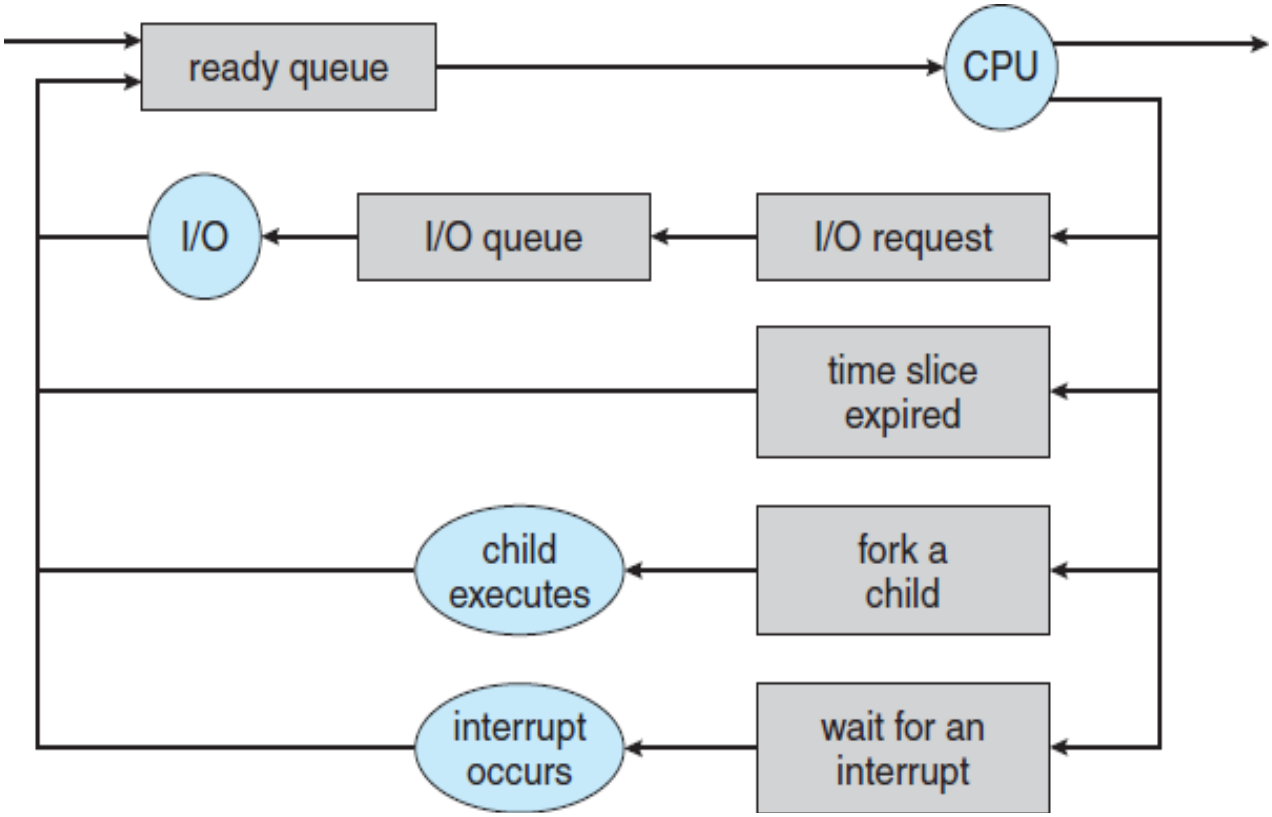

Figure 3.4 Queuing-diagram representation of process scheduling

- The system also other queues in the system. Such as list of processes waiting for a particular I/O device is called a **device queue**.
- A new process is initially put in the ready queue waits there until it is selected for execution, or (**dispatched)**.
- Once the process is allocated, the CPU start execute it, one of several events could occur:
  a. The process could issue an I/O request and then be placed in an I/O queue.
  b. The process could create a new sub-process and wait for its termination.
  c. The process could be removed forcibly from the CPU as a result of an interrupt, and be put back in the ready queue.

In the first two cases, the process eventually switches from the waiting state to the ready state and is then put back in the ready queue. A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.

3.7 Schedulers levels

- A process migrates (يهاجر) among the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate **scheduler (جدولة)**.

There are three levels (terms) of scheduling

1. Long-term scheduler

The long-term scheduler or (Job scheduler) selects processes from the job pool on the disk and loads them into memory for execution. The long-term scheduler execute much less frequently there may be minutes between the creation of new processes in the system. The L.T.S control the degree of multi programming (The number of processes in memory). If

the degree of multi programming is stable than the average rate of processes creation must be equal to the average departure rate of processes leaving the system.

It is important that the L.T.S .make a careful selection. In general most processes can be described as either I/O bound or CPU bound.

- An I/O bound process is one that spends more of its time doing I/O than it spends doing computations.
- A CPU-bound process is one that generates I/O requests infrequently, using more of its time doing computation than an I/O-bound process.

The L.T.S select a good process mix of I/O-bound and CPU-bound processes.

2. The Short-term scheduler (or **CPU scheduler**)

It is selects from among the processes that are ready to execute and allocates the CPU to one of them. The S.T.S must select a new process for the CPU quite frequently. Often the S.T.S must be very fast. If it takes 10 milliseconds to decide to executes a process for 100 milliseconds then 10/(100+10) = 9% of the CPU used (wasted) for scheduling the work. If all processes are I/O bound the ready queue will almost be empty and the S.T.S will have little to do. If all processes are CPU-bound the waiting queue will almost be empty. The system with the best performance will have a combination of CPU bound and I/0-bound processes.

3. The medium-term scheduler

Some O.S such as time-sharing systems may introduce an additional intermediate level of scheduling. The key behind the M.T.S is that sometimes it can be advantageous to remove processes from memory and thus to reduce the degree of multiprogramming. The process can be swapped out and swapped in later by the M.T.S swapping may be necessary to improve the process mix. The figure 3.5 shows the M.T.S.