1.3 PROCESS MANAGEMENT

Various process management functions include

(a) Creating and removing processes from the main memory of a computer.

(b) Controlling the progress of processes, i.e. ensuring that each process makes progress towards its completion at a positive rate.

(c) Acting on exceptional or error conditions, arising during the execution of a process, including interrupts and arithmetic errors.

(d) Allocating hardware resources among processes.

(e) Providing a means of communication among processes through messaging or signalling.

1.3.1 Process

From the Operating System's point of view, a process is the smallest individually schedulable entity, consisting of *code*, *data*, *attributes* and *state*.

Code is composed of machine instructions and system service calls (OS calls).

Attributes include such things as priority and access rights and are associated with a process by the system programmer or OS itself.

As the process executes, it changes *state* according to its circumstances. The four general categories of process states are:

(a) Dormant State

(b) Ready State

(c) Running State

(d) Suspended State or Waiting State

1.3.2 Dormant State

In this state, processes are not known to the OS and are therefore not tracked by the OS. All programs not yet submitted to the OS, as well as processes awaiting activations are considered to be dormant.

Ready State

A process is in a *ready* state, if it possesses all resources needed for execution, except for the processor. Processes usually assume the ready state immediately upon creation. All ready processes are waiting to have the processor time allocated to them by the OS so that they can run. The OS module, *scheduler*, selects one ready process out of the available ready processes for execution.

Running State

A process is in a *running* state, if it has been given all resources needed for its execution, including the processor. On a single-processor system, only one process can be in the running state at any point of time. The process in *running* state executes its sequence of machine instructions. It may also call OS to perform services such as an I/O operation via system calls. Depending on the particular scheduling policy in effect. The OS may return control to the running process after performing the I/O services or the scheduler may select another process if it is ready to be run.

Suspended State

A process is in a *suspended* or *waiting* state, if it lacks some resources other than the processor. A running process may get into *suspended* state if it requires execution of an I/O routine by the OS. Or if it requires some synchronization signals from another process. The OS then records the reason for suspension so that it can resume the process when the suspending condition is fulfilled.

Figure 1.3 illustrates the process state-transitions. If a process in the state of running is not able to get the resource needed to execute, then it gets into ready state. Or, if another process with a higher priority than this process arrives, then also it will get into the ready state. The act of taking control of the CPU from one process and giving it to another process is called *preempting*. This is clearly shown in Figure 1.3.