PROCESSES

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.

A process is the unit of work in most systems. Such a system consists of a collection of processes: Operating-system processes execute system code, and user processes execute user code. All these processes may execute concurrently.
Although traditionally a process contained only a single *thread* of control as it ran, most modern operating systems now support processes that have multiple threads.

4.1. Process Concept

A batch system executes *jobs,* whereas a timeshared system has *user programs,* or *tasks.* Even on a single-user system, such as Microsoft Windows and Macintosh OS, a user may be able to run several programs at one time: a word processor, web browser, and e-mail package. Even if the user can execute only one program at a time, 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.*

4.1.1 The Process

A process is a program in execution. A process is more than the program code, which is sometimes known as the text section. It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers. In addition, a process generally includes the process stack,

which contains temporary data (such as method parameters, return addresses, and local variables), and a data section, which contains global variables.

Although two processes may be associated with the same program, they are nevertheless considered two separate execution sequences. For instance, several users may be running different copies of the mail program, or the same user may invoke many copies of the editor program. Each of these is a separate process, and, although the text sections are equivalent, the data sections vary.

4.1.2 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:

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

These state names are arbitrary, and they vary across operating systems. The states that they represent are found on all systems. Only one process can be *running* on any processor at any instant, although many processes may be *ready* and *waiting.* The state diagram corresponding to these states is presented in Figure 4.1.

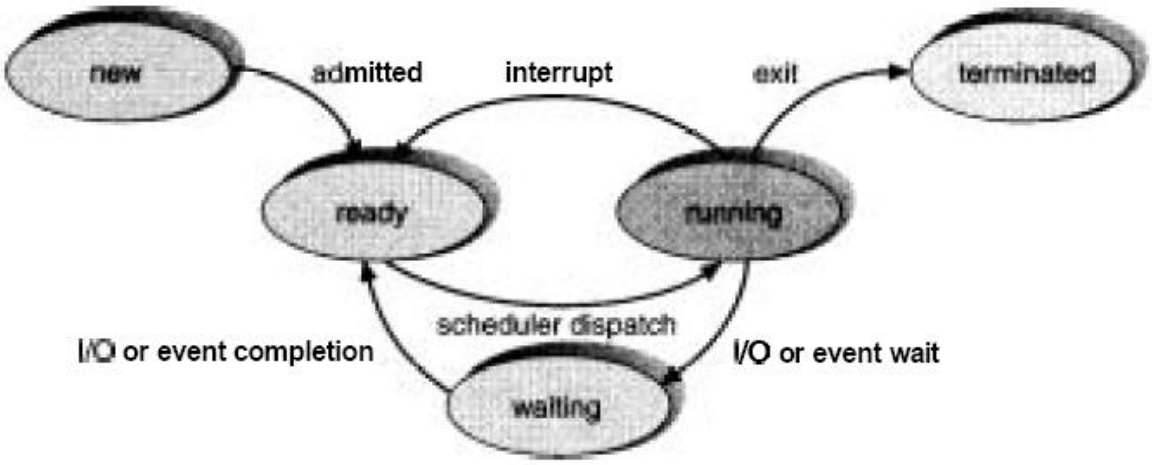

Figure 4.1 Diagram of process state.

4.1.3 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 4.2. It contains many pieces of information associated with a specific process, including these:

- **Process state:** The state may be new, ready, running, waiting, halted, and so on.
- **Program counter:** The counter indicates the address of the next instruction to be executed for this process.
- **CPU registers:** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information.

Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward (Figure 4.3).

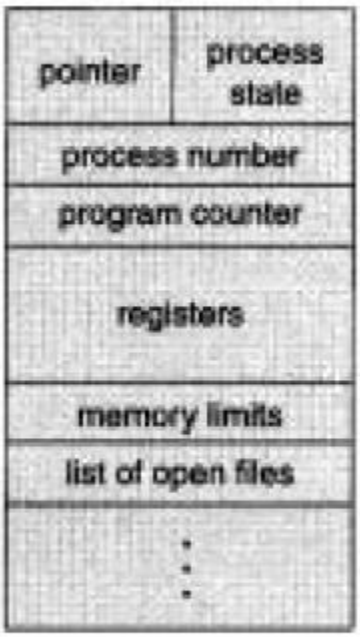

Figure 4.2 Process control block (PCB).

- **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 such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
- **Accounting information:** This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- **status information:** The information includes the list of I/O devices allocated to this 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.

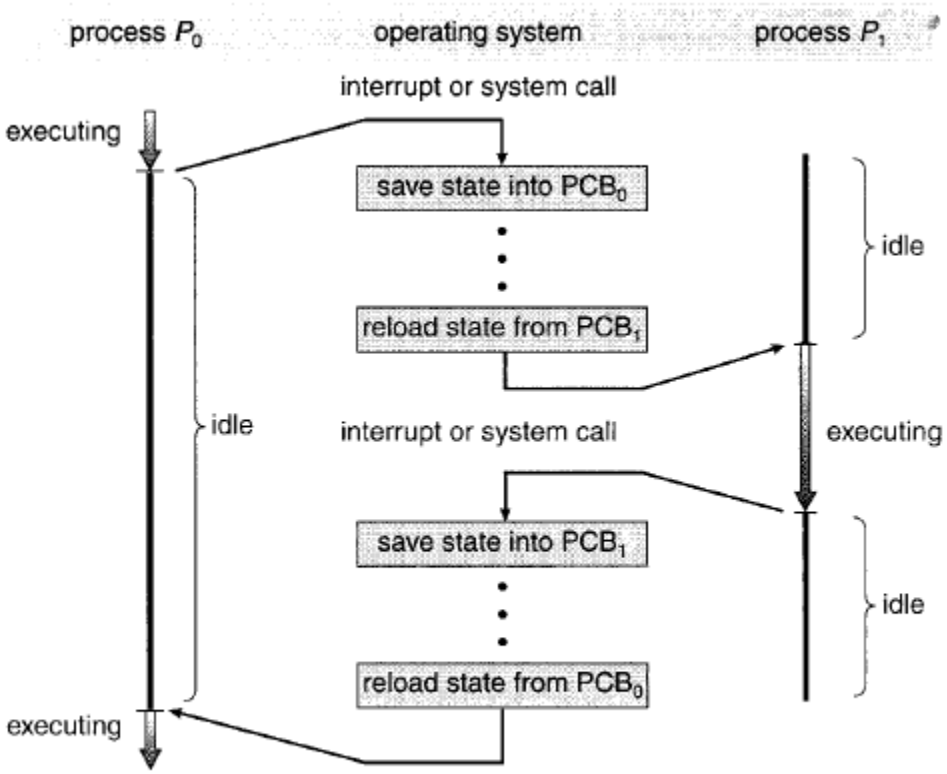

Figure 4.3 Diagram Showing CPU switch From Process to process.

4.1.4 Threads

The process model discussed so far has implied that a process is a program that performs a single thread of execution. For example, if a process is running a word-processor program, a single thread of instructions is being executed. This single thread of control allows the process to perform only one task at one time.

For example, the user could not simultaneously type in characters and run the spell checker within the same process. Many modern operating systems have extended the process concept to allow a process to have multiple threads of execution. They thus allow the process to perform more than one task at a time.