Chapter 3: Process Management

3,1 Process Concept

A question that arises in discussing operating systems involves what to call all the CPU activities, A batch system executes jobs, whereas time-shared system has user programs or tasks, Even on a single-user system such as Microsoft Windows, 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 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.* The terms Job and process are used almost interchangeably in this text. Although prefer the term *process,*

3.1.1 The Process

A process is a program in execution. A process is more than the program code. It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers. A process generally also includes the process stack, which contains temporary data (such as function parameters, return addresses, and local variables), and a data section, which contains global variables. A process may also include a heap, which is memory That is dynamically allocated during process run time. The structure of a process in memory is shown in Figure 3.1.

We emphasize that a program by itself is not a process; a program is a *passive* entity, such as a file containing a list of instructions stored on disk (often called an executable file), whereas a process is an *active* entity, with a program counter specifying the next instruction to execute and a set of associated resources. A program becomes a process when an executable file is loaded into memory. Two common techniques for loading executable files are double-clicking an icon representing the executable file and entering the name of the executable file on the command line (as in prog.exe or a.out.)



figure 3.1 Process in memory

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 Web browser program. Each of these is a separate process; and although the text sections are equivalent, the data, heap, and stack sections vary. It is also common to have a process that spawns many processes as it runs. .

3.1.2 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/0 completion or reception of a signal).
* Ready. The process is waiting to be assigned to a processor.
* 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 3.2.



Figure 3.2 Diagram of process state.

3.1.3 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 3.3.



Figure 3.3 Process control block (PCB).

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

1. Process state. The state may be new, ready running, waiting, halted, and so on.
2. Program counter. The counter indicates the address of the next instruction to be executed for this process.
3. 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 3.4).
4. CPU-scheduling information. This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
5. 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.
6. Accounting information. This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, I/O status information. This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

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



Figure 3.4 Diagram showing CPU switch from process to process.