

## The Process

Informally, as mentioned earlier, 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**. 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). In contrast, **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. Each Program has its own ID.

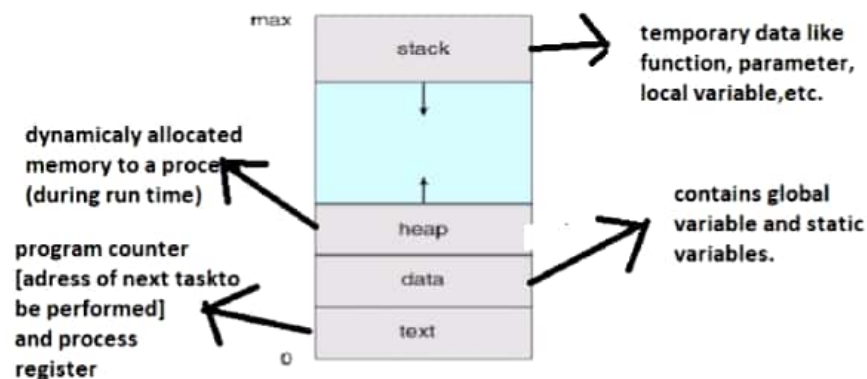
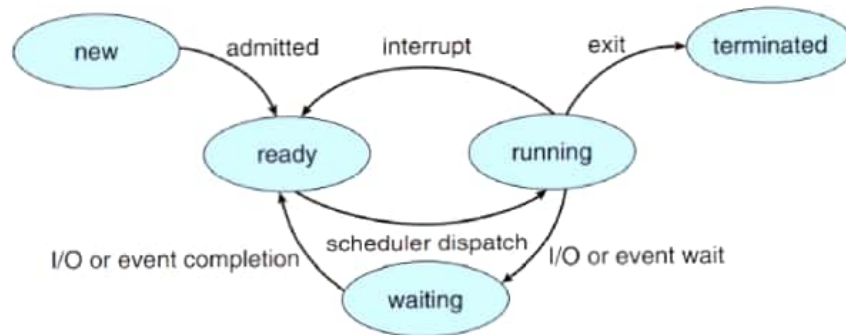


Figure 3.1 Process in memory.

## 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. A 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.

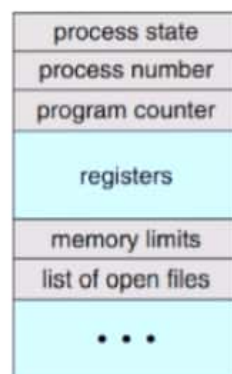


**Figure 3.2** Diagram of process state.

## Process Control Block

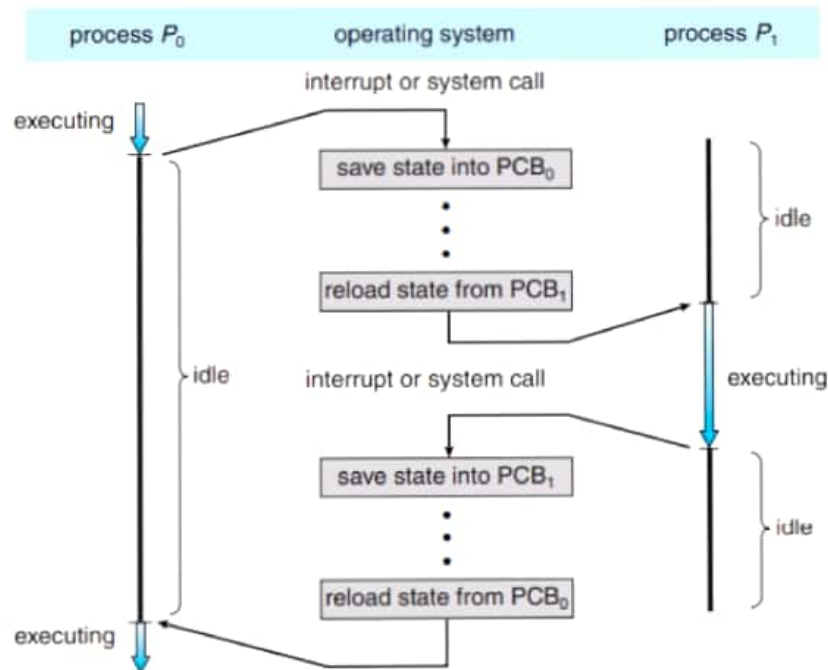
Each process is represented in the operating system by a process control block (PCB)—also called a task control block. A PCB is shown in Figure 3.3. It contains many pieces of information associated with a specific process, including these:

- 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 3.4).
- CPU-scheduling information. This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters. (Chapter 6 describes process scheduling.)
- Memory-management information. This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system (Chapter 8)
- Accounting information. This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- I/O status information. This information includes the list of I/O process, a list of open



This information devices allocated to the files, and so on.

**Figure 3.3** Process control block (PCB).



**Figure 3.4** Diagram showing CPU switch from process to process.

## Threads

Thread is a light weight process. Each process has many thread that can perform their own task [individual task]. A process can have many threads. It is basic unit of CPU execution

The process model discussed so far has implied that a process is a program that performs a single thread of execution. For example, when 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 a time. The user cannot simultaneously type in characters and run the spell checker within the same process, for example. Most modern operating systems have extended the process concept to allow a process to have multiple threads of execution and thus to perform more than one task at a time. This feature is especially beneficial on multicore systems, where multiple threads can run in parallel. On a system that supports threads, the PCB is expanded to include information for each thread. Other changes throughout the system are also needed to support threads. Chapter 4 explores threads in detail.