3.1.1 Process Management

A program does nothing unless its instructions are executed by a CPU. A *process* can be thought of as a program in execution, but its definition will broaden as we explore it further. Typically, a batch job is a process. A time-shared user program is a process. A system task, such as spooling output to a printer, also is a process. For now, you can consider a process to be a job or a time-shared program, but the concept is actually more general. As we shall see in Chapter 4, it is possible to provide system calls that allow processes to create subprocesses to execute concurrently.

A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task. These resources are either given to the process when it is created, or allocated to it while it is running. In addition to the various physical and logical resources that a process obtains when it is created, some initialization data (input) may be passed along. For example, consider a process whose function is to display the status of a file on the screen of a terminal. The process will be given as an input the name of the file, and will execute the appropriate instructions and system calls to obtain the desired information and display it on the terminal. When the process terminates, the operating system will reclaim any reusable resources.

We emphasize that a program by itself is not a process; a program is a *passive* entity, such as the contents of a file stored on disk, whereas a process is an *active* entity, with a *program counter* specifying the next instruction to execute. The execution of a process must progress in a sequential fashion. The CPU executes one instruction of the process after another, until the process completes. Further, at any point in time, at most one instruction is executed on behalf of the process. Thus, although two processes may be associated with the same program, they are nevertheless considered two separate execution sequences. It is common to have a program that spawns many processes as it runs.

A process is the unit of work in a system. Such a system consists of a collection of processes, some of which are operating-system processes (those that execute system code) and the rest of which are user processes (those that execute user code). All these processes can potentially execute concurrently, by multiplexing the CPU among them.

The operating system is responsible for the following activities in connection with process management:

- The creation and deletion of both user and system processes
- The suspension and resumption of processes
- The provision of mechanisms for process synchronization
- The provision of mechanisms for process communication
- The provision of mechanisms for deadlock handling

Process-management techniques will be discussed in great detail in Chapter 4 through Chapter 7.

3.1.2 Main-Memory Management

As discussed in Chapter 1, the main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes, ranging in size from hundreds of thousands to hundreds of millions. Each word or byte has its own address. Main memory is a repository of quickly accessible data shared by the CPU and I/O devices. The central processor reads instructions from main memory during the instruction-fetch cycle, and both reads and writes data from main memory during the data-fetch cycle. The I/O operations implemented via DMA also read and write data in main memory. The main memory is generally the only large storage device that the CPU is able to address and access directly. For example, for the CPU to process data from disk, those data must first be transferred to main memory by CPU-generated I/O calls. Equivalently, instructions must be in memory for the CPU to execute them.

For a program to be executed, it must be mapped to absolute addresses and loaded into memory. As the program executes, it accesses program instructions and data from memory by generating these absolute addresses. Eventually, the program terminates, its memory space is declared available, and the next program can be loaded and executed.

To improve both the utilization of CPU and the speed of the computer's response to its users, we must keep several programs in memory. There are many different memory-management schemes. These schemes reflect various approaches to memory management, and the effectiveness of the different algorithms depends on the particular situation. Selection of a memory-management scheme for a specific system depends on many factors — especially on the *hardware* design of the system. Each algorithm requires its own hardware support.

The operating system is responsible for the following activities in connection with memory management:

- Keep track of which parts of memory are currently being used and by whom
- Decide which processes are to be loaded into memory when memory space becomes available
- Allocate and deallocate memory space as needed

Memory-management techniques will be discussed in great detail in Chapters 8 and 9.

3.1.3 File Management

File management is one of the most visible components of an operating system. Computers can store information on several different types of physical media. Magnetic tape, magnetic disk, and optical disk are the most common media. Each of these media has its own characteristics and physical organization. Each medium is controlled by a device, such as a disk drive or tape drive, with its own unique characteristics. These properties include speed, capacity, data transfer rate, and access method (sequential or random access method).

For convenient use of the computer system, the operating system provides a uniform logical view of information storage. The operating system abstracts from the physical properties of its storage devices to define a logical storage unit, the *file*. The operating system maps files onto physical media, and accesses these files via the storage devices.

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data. Data files may be numeric, alphabetic, or alphanumeric. Files may be free-form, such as text files, or may be formatted rigidly. A file consists of a sequence of bits, bytes, lines, or records whose meanings are defined by their creators. The concept of a file is an extremely general one.

The operating system implements the abstract concept of a file by managing mass storage media, such as tapes and disks, and the devices which control them. Also, files are normally organized into directories to ease their use. Finally, when multiple users have access to files, it may be desirable to control by whom and in what ways files may be accessed.

The operating system is responsible for the following activities in connection with file management:

- The creation and deletion of files
- The creation and deletion of directories
- The support of primitives for manipulating files and directories
- The mapping of files onto secondary storage
- The backup of files on stable (nonvolatile) storage media

File-management techniques will be discussed in Chapters 10 and 11.

3.1.4 I/O System Management

One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user. For example, in UNIX, the peculiarities of I/O devices are hidden from the bulk of the operating system itself by the *I/O subsystem*. The I/O subsystem consists of