3 OPERATING SYSTEM STRUCTURES

An operating system provides the environment within which programs are executed. The design of a new operating system is a major task. The goals of the system must be well defined before the design begins. The type of system desired is the basis for choices among various algorithms and strategies.

3.1 System Components

We can create a system as large and complex as an operating system only by partitioning it into smaller pieces. Each piece should be a well-delineated portion of the system, with carefully defined inputs, outputs, and functions. Obviously, not all systems have the same structure.

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. A time-shared user program such as a compiler is a process. A word-processing program being run by an individual user on a PC is a process. A system task, such as sending output to a printer, is also a process.

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, various initialization data (or input) may be passed along.

the 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 be sequential. The CPU executes one instruction of the process after another, until the process completes. Further, at any 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:

_ Creating and deleting both user and system processes

_ Suspending and resuming processes

_ Providing mechanisms for process synchronization

_ Providing mechanisms for process communication

_ Providing mechanisms for deadlock handling

3.1.2 Main-Memory Management

The central processor reads instructions from main memory during the instruction-fetch cycle, and it 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 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 the CPU and the speed of the computer's response to its users, we must keep several programs in memory. Many different memory-management schemes are available, 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:

_ Keeping track of which parts of memory are currently being used and by whom.

_ Deciding which processes are to be loaded into memory when memory space becomes available.

_ Allocating and deallocating memory space as needed.

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.

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, that also has unique characteristics. These properties include access speed, capacity, data-transfer rate, and access method (sequential or random).

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.

The operating system implements the abstract concept of a file by managing mass storage media, such as disks and tapes, and the devices that control them. Also, files are normally organized into directories to ease their use. Finally, when multiple users have access to files, we may want to control by whom and in what ways (for example, read, write, append) files may be accessed. The operating system is responsible for the following activities in connection with file management:

- Creating and deleting files
- Creating and deleting directories
- Supporting primitives for manipulating files and directories
- Mapping files onto secondary storage
- Backing up files on stable (nonvolatile) storage media

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

- A memory-management component that includes buffering, caching, and spooling

- A general device-driver interface
- Drivers for specific hardware devices

only the device driver knows the peculiarities of the specific device to which it is assigned.

3.1.5 Secondary-Storage Management

The main purpose of a computer system is to execute programs. These programs, with the data they access, must be in main memory, or **primary storage,** during execution. Because main memory is too small to accommodate all data and programs, and because the data that it holds are lost when power is lost, the computer system must provide **secondary storage** to back up main memory.

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

- Free-space management
- Storage allocation
- Disk scheduling

Because secondary storage is used frequently, it must be used efficiently. The entire speed of operation of a computer may hinge on the speeds of the disk subsystem and of the algorithms that manipulate that subsystem.

3.1.6 Networking

A distributed system is a collection of processors that do not share memory, peripheral devices, or a clock. Instead, each processor has its own local memory and clock, and the processors communicate with one another through various communication lines, such as high-speed buses or networks. The processors in a distributed system vary in size and function. They may include

small microprocessors, workstations, minicomputers, and large, general-purpose computer systems.

The processors in the system are connected through a **communication network,** which can be configured in a number of different ways. The network may be fully or partially connected. The communication-network design must consider message routing and connection strategies, and the problems of contention and security.

A distributed system collects physically separate, possibly heterogeneous, systems into a single coherent system, providing the user with access to the various resources that the system maintains. Access to a shared resource allows computation speedup, increased functionality, increased data availability, and enhanced reliability. Operating systems usually generalize network access as a form of file access, with the details of networking being contained in the network interface's device driver. The protocols that create a distributed system can have a great effect on that system's utility and popularity. The innovation of the World Wide Web was to create a new access method for information sharing. It improved on the existing file-transfer protocol (FTP) and network file-system (NFS) protocol by removing the need for a user to log in before she is allowed to use a remote resource. It defined a new protocol, hypertext transfer protocol (http), for use in communication between a web server and a web browser. **A** web browser then just needs to send a request for information to a remote machine's web server, and the information (text, graphics, links to other information) is returned.

3.1.7 Protection System

If a computer system has multiple users and allows the concurrent execution of multiple processes, then the various processes must be protected from one another's activities. For that purpose, mechanisms ensure that the files, memory

segments, CPU, and other resources can be operated on by only those processes that have gained proper authorization from the operating system.

For example, memory-addressing hardware ensures that a process can execute only within its own address space. The timer ensures that no process can gain control of the CPU without eventually relinquishing control. Device control registers are not accessible to users, so that the integrity of the various peripheral devices is protected.

Protection is any mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system. This mechanism must provide means for specification of the controls to be imposed and means for enforcement.

Protection can improve reliability by detecting latent errors at the interfaces between component subsystems. Early detection of interface errors can often prevent contamination of a healthy subsystem by another subsystem that is malfunctioning. An unprotected resource cannot defend against use (or misuse) by an unauthorized or incompetent user. A protection-oriented system provides a means to distinguish between authorized and unauthorized usage.

3.1.8 Command-Interpreter System

One of the most important systems programs for an operating system is the command interpreter, which is the interface between the user and the operating system. Some operating systems include the command interpreter in the kernel. Other operating systems, such as MS-DOS and UNIX, treat the command interpreter as a special program that is running when a job is initiated.

Many commands are given to the operating system by control statements. When a new job is started in a batch system, or when a user logs on to a time-shared

system, a program that reads and interprets control statements is executed automatically. This program is sometimes called the control-card interpreter or the command-line interpreter, and is often known as the shell.

Its function is simple: To get the next command statement and execute it. Operating systems are frequently differentiated in the area of the shell, with a user-friendly command interpreter making the system more agreeable to some users. One style of user-friendly interface is the mouse-based window and menu system used in the Macintosh and in Microsoft Windows. The mouse is moved to position the mouse pointer on images, or icons, on the screen that represent programs, files, and system functions. Depending on the mouse pointer's location, clicking a button on the mouse can invoke a program, select a file or directory-known as a folder-or pull down a menu that contains commands. More powerful, complex, and difficult-to-learn shells are appreciated by other users. In some of these shells, commands are typed on a keyboard and displayed on a screen or printing terminal, with the enter (or return) key signaling that a command is complete and is ready to be executed. The MS-DOS and UNIX shells operate in this way. The command statements themselves deal with process creation and management, I/O handling, secondary-storage management, main-memory management, file-system access, protection, and networking.