Operating System

Chapter 1: Types and Structure of Operating Systems

Prepared By:
Amit K. Shrivastava
Asst. Professor
Nepal College Of Information Technology

1.1 Introduction and History of Operating Systems

- An operating system acts as an intermediary between the user of a computer and the computer hardware. The purpose of an Operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.
- An operating system is software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

History of Operating System:

- Operating systems have been evolving through the years. Since operating systems have historically been closely tied to the architecture of the computers on which they run, we will look at successive generations of computers to see what their operating systems were like.

The First Generation (1945- 55) Vacuum Tubes and Plug boards

- First of all calculating engines is built and mechanical relays were used but were very slow, with cycle times measured in seconds. Relays were later replaced by vacuum tubes. All programming was done in absolute machine language, often by wiring up plugboards to control the machine's basic functions. By the early 1950s, the routine had improved somewhat with the introduction of punched cards. It was now possible to write programs on cards and read them in instead of using plugboards;

History of Operating System(contd..):

The Second Generation (1955-65) Transistors and Batch Systems

- The introduction of the transistor in the mid-1950s changed the picture radically. These machines, now called mainframes, and to run a job a programmer would first write the program on paper (in FORTRAN or possibly even in assembly language), then punch it on cards. But it was time consuming and costly. The solution generally adopted was the batch system. The idea behind it was to collect a tray full of jobs in the input room and then read them onto a magnetic tape using a small (relatively) inexpensive computer, such as the IBM 1401.

they were largely programmed in FORTRAN and assembly. Typical OS were FMS(Fortran Monitor System) and IBSYS (IBM's operating system for 7094)

The Third Generation (1965-1980) ICs and Multiprogramming

- The 360 developed by IBM was the first major computer line to use (small-scale) Integrated Circuits (ICs), thus providing a major price/ performance advantage over the second-generation machines, which were built up from individual transistors. And the most important advantage was multiprogramming. Here, the memory was partitioned into several pieces, with a different job in each partition, while one job was waiting for the I/O to complete, another job could be using the CPU.

IBM came with the idea of family of compatible computers. This made them design an operating system that would be compatible to all of the computers in one family. The IBM 360 had Operating system named OS/360.

ONe feature in third generation computers was that whenever the running job finished, the operating system could bring a new job from the disk to the empty partition and then run it.

Foundation of UNIX was developed in third generation computers

MINIX was conceived

History of Operating System(contd..):

The Fourth Generation (1980Present) Personal Computers

- With the development of LSI (Large Scale Integration) circuits chips containing thousands of transistors on a square centimeter of silicon, the age of personal computer dawned. First kildall wrote a disk base operating system called CP/M(Control Program for Microcomputers) for Intel in 1974, then in early 1980's DOS(Disk Operating System) was invented and after that Microsoft revised it and renamed MS-DOS(Microsoft Disk Operating System). All these operating systems were all based users typing in commands from the keyboard after that GUI(Graphical User Interface) was invented, complete with windows, icons, menus, and mouse. After that different version of windows and Unix came in to light.

FreeBSD is a popular UNIX derivative

1.2 Operating System Concepts and Functionalities

- The interface between the operating system and the user programs is defined by the set of "extended instructions" that the operating system provides. These extended instructions have been traditionally known as system calls, which have generally three parameters: one to specify the file, one to tell where the data are to be put, and one to tell how many bytes to read"). This three parameter 'read' system call belongs to UNIX

1.2.1. Processes

- A key concept in all operating systems, is the process. A process is basically a program in execution. Associated with each process is its address space, a list of memory locations from some minimum (usually 0) to some maximum, which the process can read and write. The address space contains the executable program, the program's data, and its stack. Also associated with each process is some set of registers, including the program counter, stack pointer, and other hardware registers, and all the other information needed to run the program. Process is like multiprogramming systems. Periodically, the operating system decides to stop running one process and start running another. But this suspended process must later be started in exactly the same state it had when it was stopped. For that purpose, all the information about each process, other than the contents of its own address space, is stored in an operating system table called the process table, which is an array (or linked list) of structures, one for each process currently in existence.
- If a process can create one or more other processes (usually referred to as child processes) and these processes in turn can create child processes, a tree structure can be built. And the communication between these process is called interprocess communication.

1.2.2. Files

- The other broad category of system calls relates to the file system. As noted before, a major function of the operating system is to hide the peculiarities of the disks and other I/O devices and present the programmer with a nice, clean abstract model of device-independent files. System calls are obviously needed to create files, remove files, read files, and write files. Before a file can be read, it must be opened, and after it has been read it should be closed, so calls are provided to do these things.
- To provide a place to keep files, there is a concept of a directory as a way of grouping files together. Directory entries may be either files or other directories. This model also gives rise to a hierarchythe file systemas shown in Fig. 1.
- Every file within the directory hierarchy can be specified by giving its path name from the top of the directory hierarchy, the root directory. Such absolute path names consist of the list of directories that must be traversed from the root directory to get to the file, with slashes separating the components. In Fig. 1, the path for file CS101 is

  /Faculty/Prof.Brown/Courses/CS101.

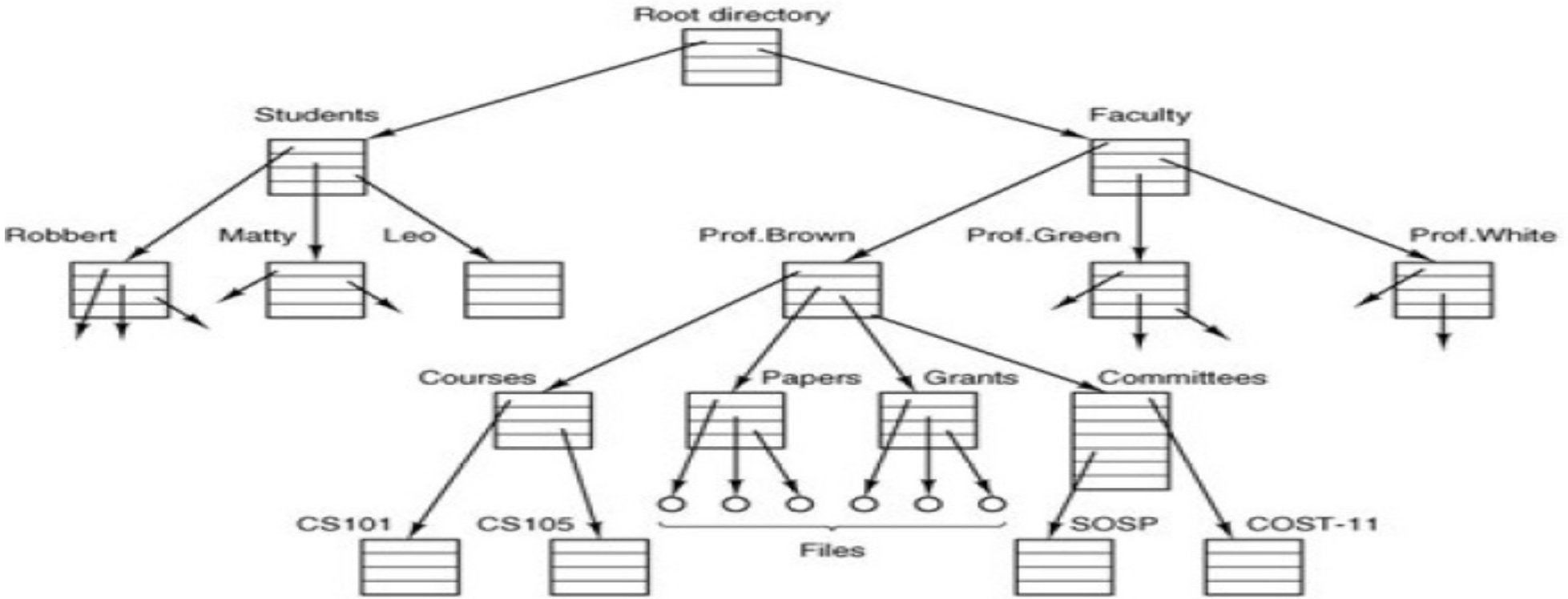


Fig. 1:A file system for a university department

1.2.3 The Shell

▪The operating system is the code that carries out the system calls. Shell is a UNIX command interpreter. Although it is not part of the operating system, it makes heavy use of many operating system features and thus serves as a good example of how the system calls can be used. It is also the primary interface between a user sitting at his terminal and the operating system, unless the user is using a graphical user interface. Many shells exist, including csh, ksh, zsh, and bash. All of them support the functionality described below, which derives from the original shell (sh). E.g. Standard input can be redirected as in ***sort <file1 >file2*** which invokes the sort program with input taken from file1 and output sent to file2.

Only system calls enter the kernel, not the procedure calls

1.2.4 System Calls

- System calls provide an interface to the services made available by an operating system. These calls are generally available as routines written in C and C++, and are usually listed in the manuals used by assembly-language programmers.
- System calls can be grouped roughly into six major categories: process control, file manipulation, device manipulation, information maintenance, communications, and protection.

- **Process control**
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
- **File management**
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes

System Calls(contd..)

- **Device management**
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- **Information maintenance**
  - get time or date, set time or date
  - get system data, set system data
  - get process, file, or device attributes
  - set process, file, or device attributes
- **Communications**
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices

1.3 Operating System Structure

- Operating System Structure defines how they look inside. The designs are:

1.3.1 Monolithic Systems:

entire OS runs as a single program in kernel mode.

which are linked to a single large executable program

- In it, the operating system is written as a collection of procedures, each of which can call any of the other ones whenever it needs to. When this technique is used, each procedure in the system has a well-defined interface in terms of parameters and results, and each one is free to call any other one, if the latter provides some useful computation that the former needs.
- To construct the actual object program of the operating system when this approach is used, one first compiles all the individual procedures, or files containing the procedures, and then binds them all together into a single object file using the system linker. In terms of information hiding, there is essentially none-every procedure is visible to every other procedure.
- This organization suggests a basic structure for the operating system:

1. A main program that invokes the requested service procedure.
2. A set of service procedures that carry out the system calls.
3. A set of utility procedures that help the service procedures.

- In this model, for each system call there is one service procedure that takes care of it. The utility procedures do things that are needed by several service procedures, such as fetching data from user programs.

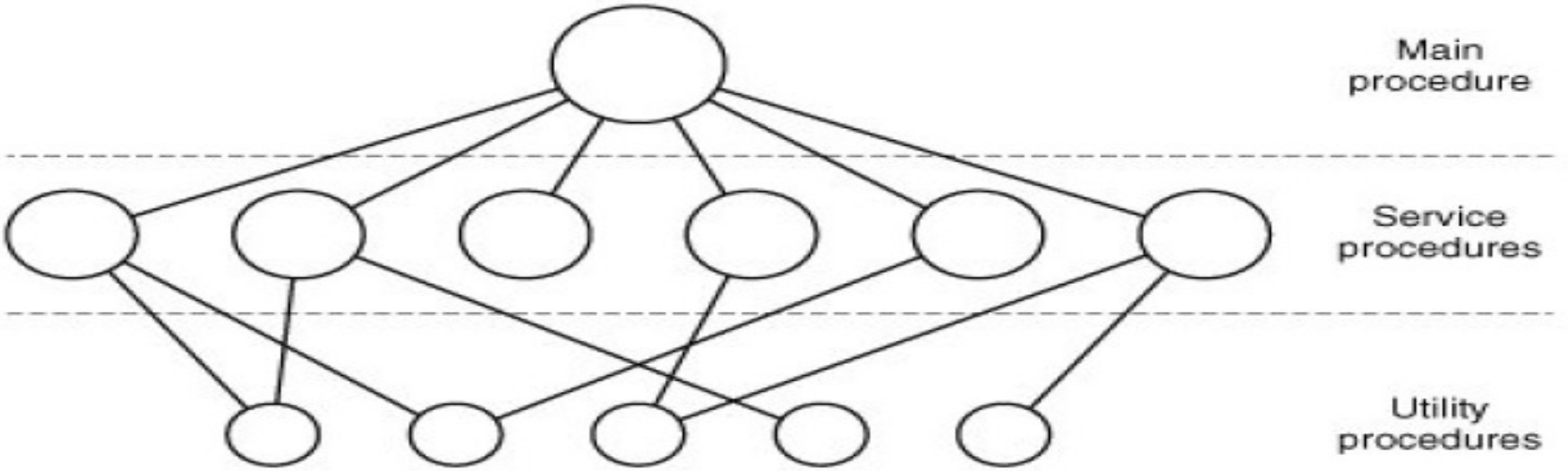


Fig: A simple structuring model for a monolithic system

1.3.2. Layered Systems

- in it, the operating system is organized system as a hierarchy of layers, each one constructed upon the one below it. The first system constructed in this way was the THE system. The system has 6 layers as shown in fig below:

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Layered Systems(contd..)

- A further generalization of the layering concept was present in the MULTICS system. Instead of layers, MULTICS was organized as a series of concentric rings, with the inner ones being more privileged than the outer ones. When a procedure in an outer ring wanted to call a procedure in an inner ring, it had to make the equivalent of a system call, that is, a TRAP instruction whose parameters were carefully checked for validity before allowing the call to proceed.

1.3.3. Virtual Machines:

- Virtual machines known as VM/370 was based on a very astute observation: a timesharing system provides (1)multiprogramming and (2) an extended machine with a more convenient interface than the bare hardware. The essence of VM/370 is to completely separate these two functions. The heart of the system, known as the virtual machine monitor, runs on the bare hardware and does the multiprogramming, providing not one, but several virtual machines to the next layer up, as shown in Fig. 1
- Different virtual machines can, and frequently do, run different operating systems. Some run one of the descendants of OS/360 for batch or transaction processing, while others run a single-user, interactive system called CMS (Conversational Monitor System) for timesharing users.
- When a CMS program executes a system call, the call is trapped to the operating-system in its own virtual machine, not to VM/370, CMS then issues the normal hardware I/O instructions for reading its virtual disk or whatever is needed to carry out the call.

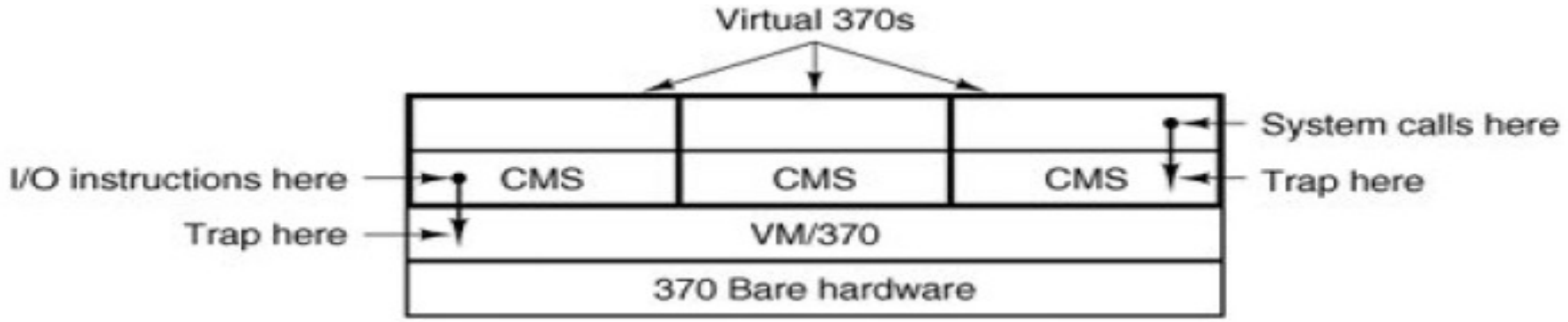
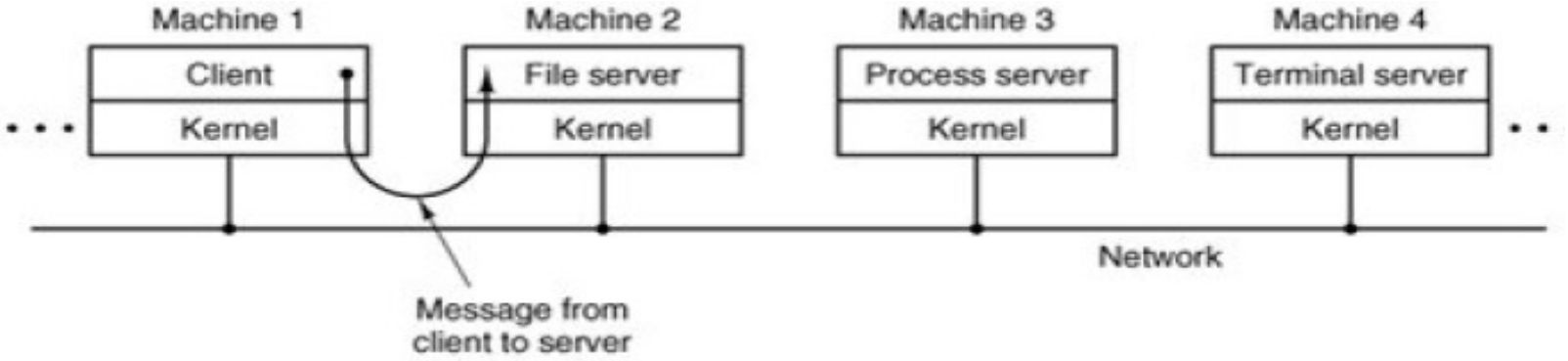


Fig: The structure of VM/370 with CMS

1.3.4. Client-Server Model:

- This concepts is based on of two classes of processes, the servers, each of which provides some service, and the clients, which use these services. This model is known as client-server model. Communication between clients and servers is often by message passing. To obtain a service, a client process constructs a message saying what it wants and sends it to the appropriate service. The service then does the work and sends back the answer. An obvious generalization of this idea is to have the clients and servers run on different computers, connected by a local or wide-area network.



1.4 Types and Evolution of Operating Systems:

Batch Systems

- Jobs with similar needs are batched together and run through the computer as a group by an operator or automatic job sequencer. Performance is increased by attempting to keep CPU and I/O devices busy at all times through buffering, off-line operation, spooling, and multiprogramming. Batch is good for executing large jobs that need little interaction; it can be submitted and picked up later. The problems with Batch Systems are following.
  - Lack of interaction between the user and job.
  - Difficult to provide the desired priority.

Time-Sharing Systems

- This systems uses CPU scheduling and multiprogramming to provide economical interactive use of a system. The CPU switches rapidly from one user to another. Instead of having a job defined by spooled card images, each program reads its next control card from the terminal, and output is normally printed immediately to the screen. Advantages of Timesharing operating systems are - Provide advantage of quick response, Avoids duplication of software, Reduces CPU idle time.

Personal-Computer Systems

- A Personal Computer(PC) is a small, relatively inexpensive computer designed for an individual user. All are based on the microprocessor technology that enables manufacturers to put an entire CPU on one chip. At home, the most popular use for personal computers is for playing games. Businesses use personal computers for word-processing, accounting, desktop publishing, and for running spreadsheet and database management applications. The goals of these operating systems is not only maximizing CPU and peripheral utilization, but also maximizing user convenience and responsiveness.

Parallel Systems

- Parallel operating systems are used to interface multiple networked computers to complete tasks in parallel. The architecture of the software is often a UNIX-based platform, which allows it to coordinate distributed loads between multiple computers in a network. Parallel operating systems are able to use software to manage all of the different resources of the computers running in parallel, such as memory, caches, storage space, and processing power. Parallel operating systems also allow a user to directly interface with all of the computers in the network. Its one advantage is increased throughput.

Real-Time Systems

- Often used in a dedicated application, this system reads information from sensors and must respond within a fixed amount of time to ensure correct performance. In this Response Time is already fixed. Means time to Display the Results after Possessing has fixed by the Processor or CPU. Real Time System is used at those Places in which we Requires higher and Timely Response.

Distributed Systems

- This system distributes computation among several physical processors. The processors do not share memory or a clock. Instead, each processor has its own local memory. They communicate with each other through various communication lines, such as a high-speed bus or telephone line. The advantages of distributed systems are- With resource sharing facility user at one site may be able to use the resources available at another, Speedup the exchange of data with one another via electronic mail, If one site fails in a distributed system, the remaining sites can potentially continue operating, Better service to the customers, Reduction of the load on the host computer.

1.4 Types and Evolution of Operating Systems: