Module 1: Introduction to Operating System

Operating System (or shortly OS) primarily provides services for running applications on a computer system.

Need for an OS:

The primary need for the OS arises from the fact that user needs to be provided with services and OS ought to facilitate the provisioning of these services. The central part of a computer system is a processing engine called CPU. A system should make it possible for a user's application to use the processing unit. A user application would need to store information. The OS makes memory available to an application when required. Similarly, user applications need use of input facility to communicate with the application. This is often in the form of a key board, or a mouse or even a joy stick (if the application is a game for instance).

Keyboard

Mouse

Joystick

Monitor

Printer

The output usually provided by a video monitor or a printer as some times the user may wish to generate an output in the form of a printed document. Output may be available in some other forms. For example it may be a video or an audio file.

Let us consider few applications.

- Document Design
- Accounting
- E-mail
- Image processing

- Games

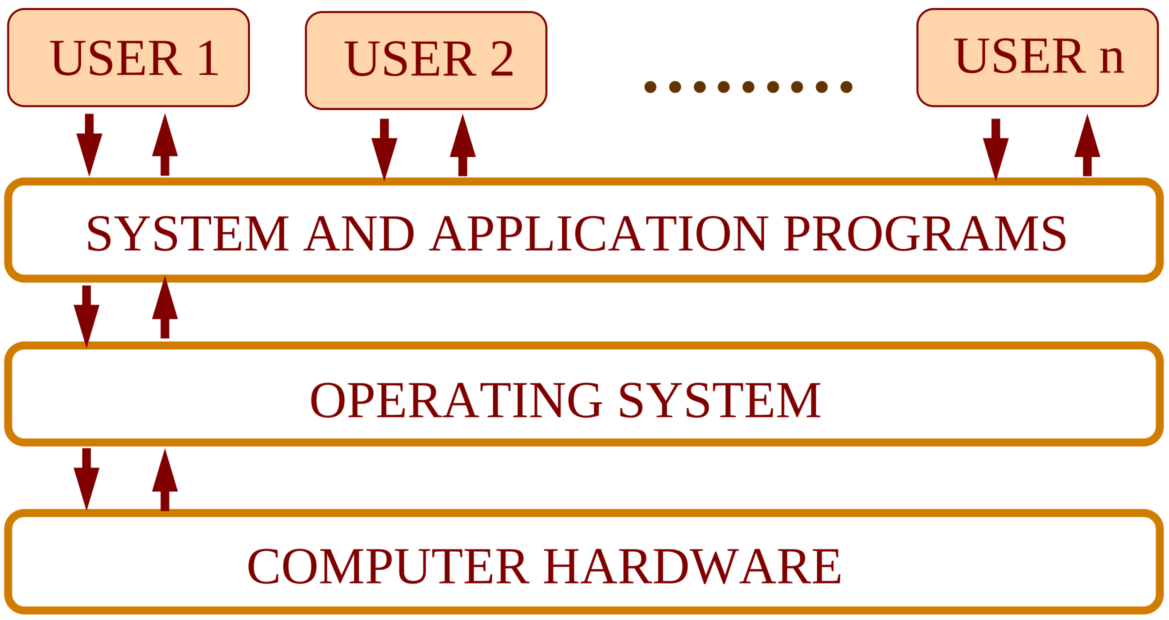
We notice that each of the above application requires resources for

- Processing information
- Storage of Information
- Mechanism to inputting information
- Provision for outputting information
- These service facilities are provided by an operating system regardless of the nature of application.

The OS offers generic services to support all the above operations. These operations in turn facilitate the applications mentioned earlier. To that extent an OS operation is ***application neutral*** and ***service specific.***

User and System View:

From the user point of view the primary consideration is always the convenience. It should be easy to use an application. In launching an application, it helps to have an icon which gives a clue which application it is. We have seen some helpful clues for launching a browser, e-mail or even a document preparation application. In other words, the human computer interface which helps to identify an application and its launch is very useful. This hides a lot of details of the more elementary instructions that help in selecting the application. Similarly, if we examine the programs that help us in using input devices like a key board – all the complex details of character reading program are hidden from the user. The same is true when we write a program. For instance, when we use a programming language like C, a printf command helps to generate the desired form of output. The following figure essentially depicts the basic schema of the use of OS from a user stand point. However, when it comes to the view point of a system, the OS needs to ensure that all the system users and applications get to use the facilities that they need.



Also, OS needs to ensure that system resources are utilized efficiently. For instance, there may be many service requests on a Web server. Each user request need to be serviced. Similarly, there may be many programs residing in the main memory. The system need to determine which programs are active and which need to await some form of input or output. Those that need to wait can be suspended temporarily from engaging the processor. This strategy alone enhances the processor throughput. In other words, it is important for an operating system to have a control policy and algorithm to allocate the system resources.

The Evolution:

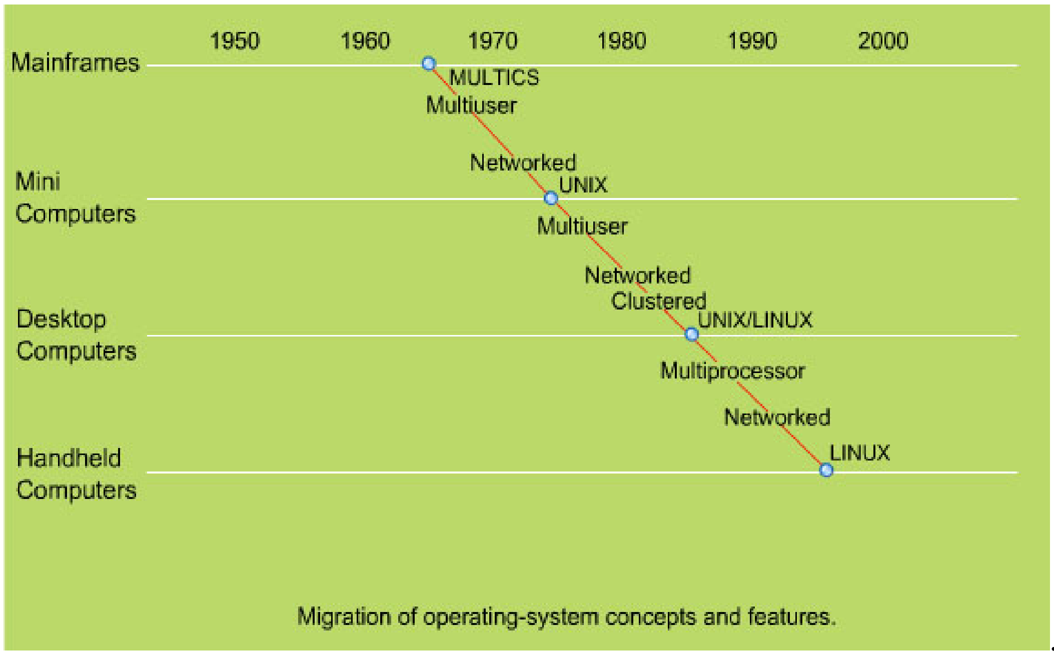
It would be worthwhile to trace some developments that have happened in the last four to five decades. In the 1960s, the common form of computing facility was a mainframe computer system. The mainframe computer system would be normally housed in a computer center with a controlled environment which was usually an air conditioned area with a clean room like facility. The users used to bring in a deck of punched cards which encoded the list of program instructions.

The mode of operation was as follows:

- User would prepare a program as a deck of punched cards.
- The header cards in the deck were the “job control” cards which would indicate which compiler was to be used (like Fortran / Cobol compilers).
- The deck of cards would be handed in to an operator who would collect such jobs from various users.

- The operators would invariably group the submitted jobs as Fortran jobs, Cobol jobs etc. In addition, these were classified as “long jobs” that required considerable processing time or short jobs which required a short and limited computational time.

Each set of jobs was considered as a batch and the processing would be done for a batch. Like for instance there may be a batch of short Fortran jobs. The output for each job would be separated and turned over to users in a collection area. This scenario clearly shows that there was no interactivity. Users had no direct control. Also, at any one time only one program would engage the processor. This meant that if there was any input or output in between processing then the processor would wait idling till such time that the I/O is completed. This meant that processor would idling most of the time as processor speeds were orders of magnitude higher than the input or output or even memory units. Clearly, this led to poor utilization of the processor. The systems that utilized the CPU and memory better and with multiple users connected to the systems evolved over a period of time as shown in Table1.1.

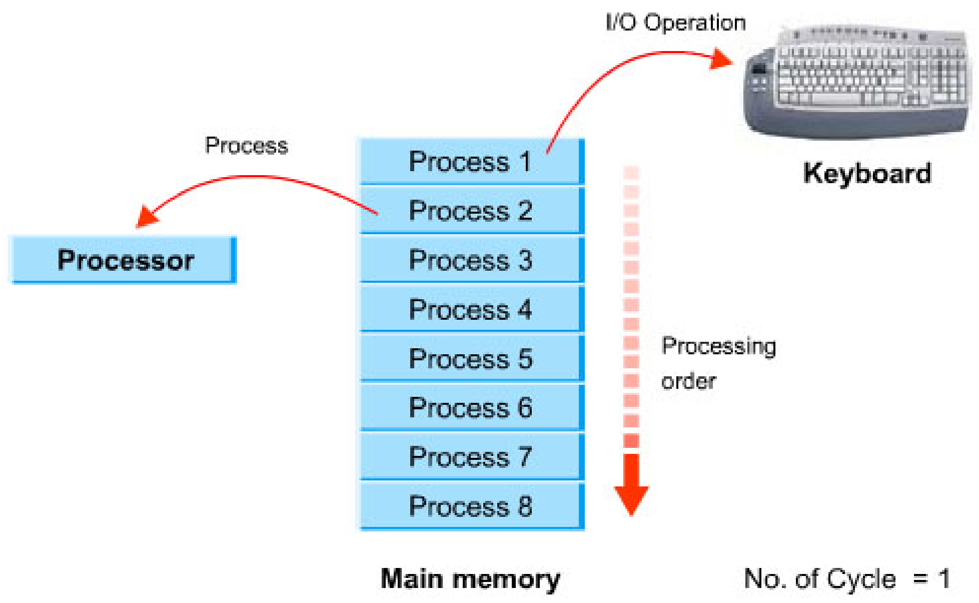


.

At this time we would like to invoke Von - Neumann principle of stored program operation. For a program to be executed it ought to be stored in the memory. In the scheme of things discussed in the previous paragraph, we notice that at any time only one program was kept in the memory and executed. In the decade of 70s this basic mode of operation was altered and system designers contemplated having more than one program resident in the memory. This clearly meant that when one program is awaiting completion of an input or output, another program could, in fact, engage the CPU..

Late 60's and early 70's

Storing multiple executables (at the same time) in the main memory is called multiprogramming. With multiple excutables residing in the main memory, the immediate consideration is: we now need a policy to allocate memory and processor time to the resident programs. It is obvious that by utilizing the processor for another process when a process is engaged in input or output the processor utilization and, therefore, its output are higher. Overall, the multiprogramming leads to higher throughput for this reason.



Multiprogramming

While multiprogramming did lead to enhanced throughput of a system, the systems still essentially operated in batch processing mode.

1980's

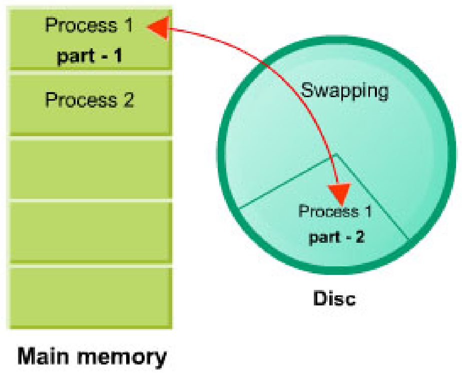
In late 70s and early part of the decade of 80s the system designers offered some interactivity with each user having a capability to access system. This is the period when the timeshared systems came on the scene.

Basically, the idea is to give every user an illusion that all the system resources were available to him as his program executed. To strengthen this illusion a clever way was devised by which each user was allocated a slice of time to engage the processor. During the allocated time slice a users' program would be executed. Now imagine if the next turn for the same program comes quickly enough, the user would have an illusion that the system was continuously available to his task. This is what precisely time sharing systems attempted – giving each user a small time slice and returning back quickly enough so that he never feels lack of continuity. In fact, he carries an impression that the system is entirely available to him alone.

Timeshared systems clearly require several design considerations. These include the following: How many programs may reside in the main memory to allow, and also sustain timesharing? What should be the time slice allocated to process each program? How would one protect a users' program and data from being overwritten by another users' program? Basically, the design trends that were clearly evident during the decade of 1970-80 were: Achieve as much overlapping as may be feasible between I/O and processing. Bulk storage on disks clearly witnessed a phenomenal growth. This also helped to implement the concept to offer an illusion of extended storage. The concept of "virtual storage" came into the vogue. The virtual storage essentially utilizes these disks to offer enhanced addressable space. The fact that only that part of a program that is currently active need be in the main memory also meant that multi-programming could support many more programs. In fact this could be further enhanced as follows:

1. Only required active parts of the programs could be swapped in from disks.
2. Suspended programs could be swapped out.

This means that a large number of users can access the system. This was to satisfy the notion that "computing" facility be brought to a user as opposed to the notion that the "user go to compute". The fact that a facility is brought to a user gives the notion of a utility or a service in its true sense. In fact, the PC truly reflects the notion of "computing utility" - it is regarded now as a personal productivity tool.



Swapping of program parts main memory - disc, vice-versa

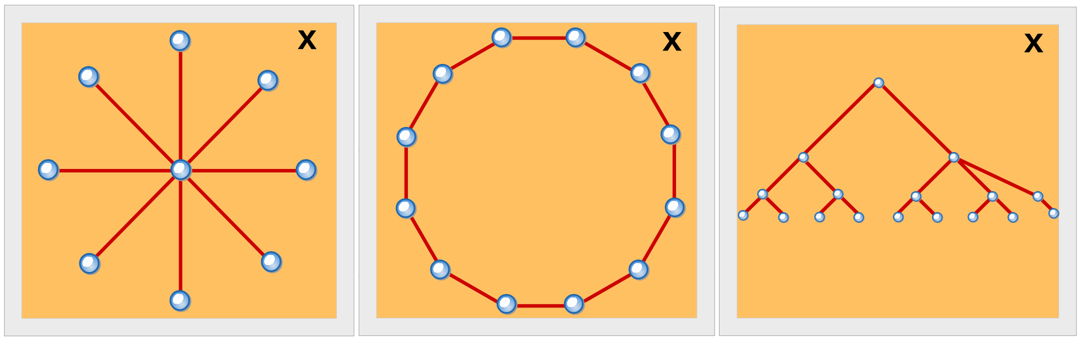
It was in early 1970s Bell Laboratory scientists came up with the now well known OS: Unix. Also, as the microcomputers came on scene in 1980s a forerunner to current DOS was a system called CP/M. The decade of 1980s saw many advances with the promise of networked systems. One notable project amongst these was the project Athena at MIT in USA. The project forms the basis to several modern developments. The client-server paradigm was indeed a major fall out. The users could have a common server to the so called X-terminals.

The X windows also provided many widgets to support convenient human computer interfaces. Using X windows it is possible to create multiple windows. In fact each of the windows offers a virtual terminal. In other words it is possible to think about each of these windows as a front-end terminal connection. So it is possible to launch different applications from each of the windows. This is what you experience on modern day PC which also supports such an operating environment.

In our discussions, we shall discuss many of the above issues in greater detail as we move to later chapters. On the micro-computer front the development was aimed at relieving the processor from handling input output responsibilities. The I/O processing was primarily handled by two mechanisms: one was BIOS and the other was the graphics cards to drive the display. The processor now was relieved from regulating the I/O. This made it possible to utilize the processor more effectively for other processing tasks. With the advent of 1990s the computer communication was pretty much the order of the day.

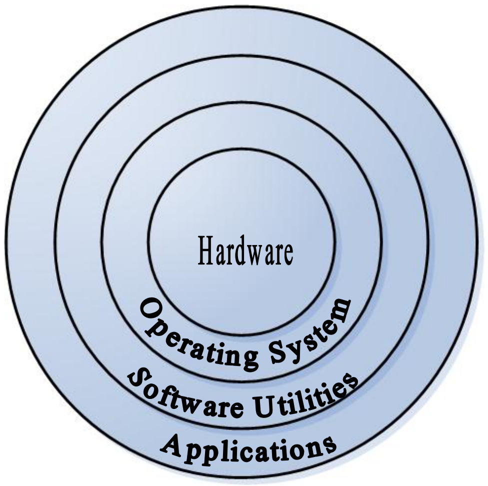
CP/M based computer

Networking topologies like star, ring and general graphs, as shown in the figure, were being experimented with protocols for communication amongst computers evolved. In particular, the TCP/IP suite of network protocols were implemented. The growth in the networking area also resulted in giving users a capability to establish communication between computers. It was now possible to connect to a remote computer using a telnet protocol. It was also possible to get a file stored in a remote location using a file transfer (FTP) protocol. All such services are broadly called network services.



(a) Star (b) Ring (c) Tree

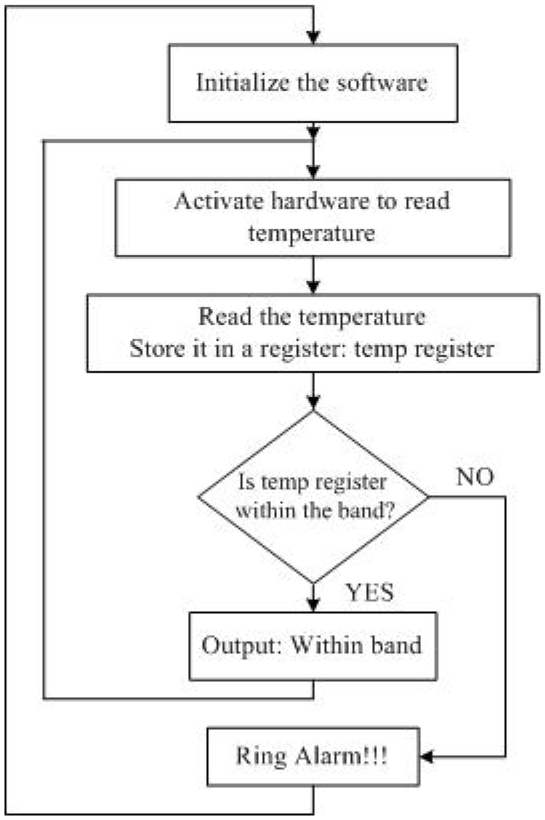
Let's now briefly explore where the OS appears in the context of the software and application.



Let's consider a scenario where we need to embed the computer system in an industrial application. This may be regulating the temperature of a vessel in a process control. In a typical process control scenario

- Monitoring – initializes and activates the hardware.
- Input – Reads the values from sensors and stores it in register.
- Decision – checks whether the readings are within the range.
- Output – responds to the situation.
- Scenario: A temperature monitoring chemical process.

- What we need: A supervisory program to raise an alarm when temperature goes beyond a certain band.
- The desired sequence of operational events: Measure input temperature, process the most recent measurement, perform an output task.


Initialize the software
Activate hardware to read temperature
Read the temperature Store it in a register: temp register
Is temp register within the band?
NO
YES
Output: Within band
Ring Alarm!!!

The computer system may be employed in a variety of operational scenarios like a bank, airlines reservation system, university admissions and several others. In each of these we need to provide the resources for

- Processing
- User access to the system
- Storage and management of information
- Protection of information against accidental and intentional misuse
- Support for data processing activities
- Communication with I/O devices
- Management of all activities in a transparent manner.

Let’s now review What Does an OS Do?

- Power On Self Test (POST)
- Resource management

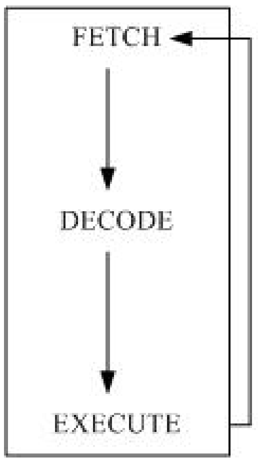
- Support for multi-user
- Error Handling
- Communication support over Network
- (Optional) Deadline support so that safety critical application run and fail gracefully

Operational View:

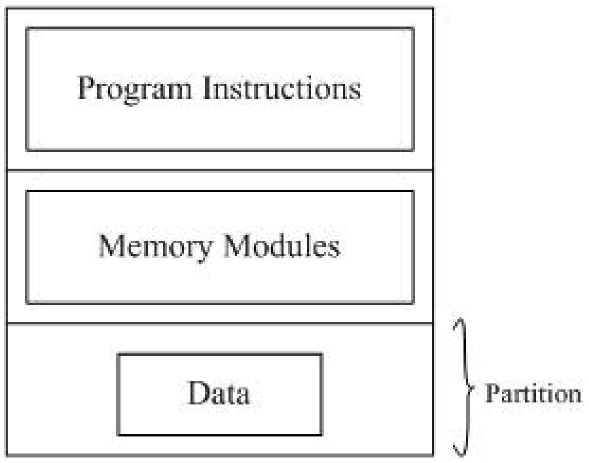
Let's briefly look at the underlying principle of operation of a computer system. Current systems are based on The Von-Neumann principle. The principle states that a program is initially stored in memory and executed by fetching an instruction at a time.

The basic cycle of operation is

- ➢ Fetch an instruction (Fetch)
- ➢ Interpret the instruction (Decode)
- ➢ Execute the instruction (Execute)



Operating Cycle



Memory Map

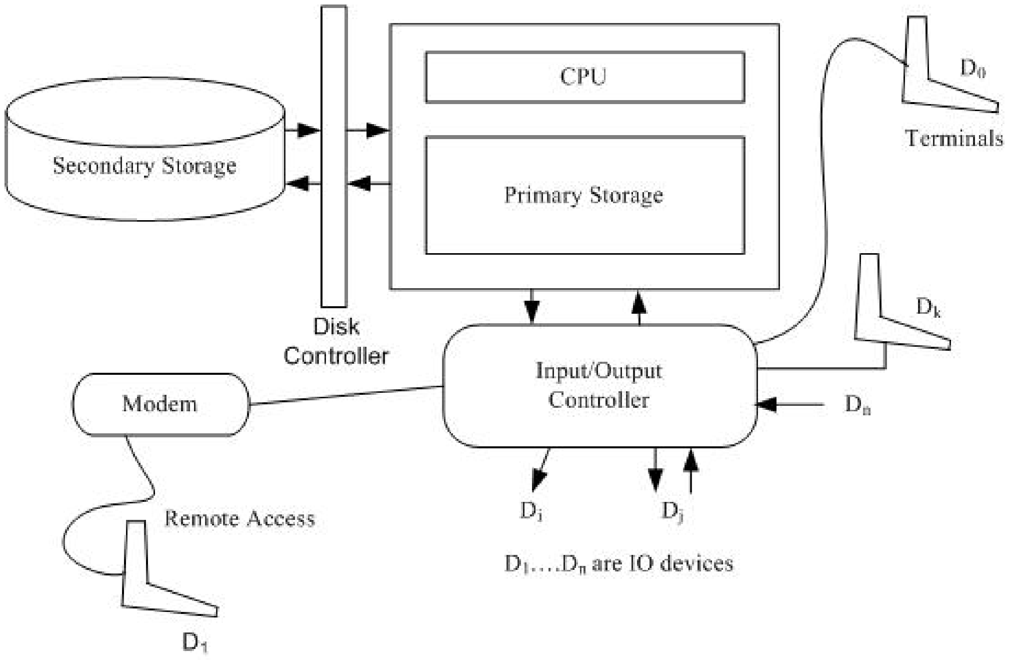
Modern systems allow multiple users to use a computer system. Even on a stand alone PC there may be multiple application which are running simultaneously. For instance, we have a mail program receiving mails, a clock to display time while we may be engaged in browsing a word process.

In other words OS need to schedule the processor amongst all the application simultaneously without giving an impression that the processor time is being divided and scheduled per an application.

An Operational Overview:

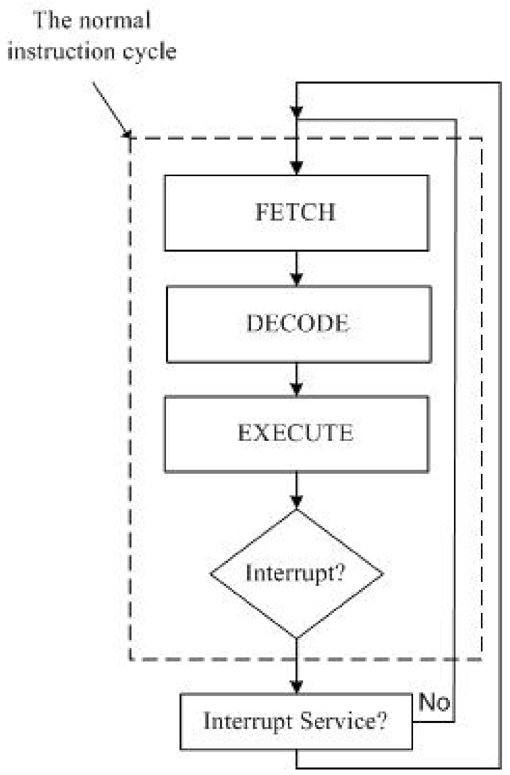
- Processor – schedule and allocate processor time.

- Memory – executes program and access data
- Input output devices
- Communication with devices
- Mutual exclusion – schedule the usage of shared device and fair access
- Shell of an OS
- Human computer interface (HCI/CHI)



A Modern Computer System

The peripheral devices communicate in a mode known as interrupt mode .Typically human input is considered important and often uses this mode. This is so because human desire to guide the operation. For instance, we use a mouse click or a key board input. These require immediate attention by needing an appropriate interruption service.



Processes and Tools:

Processes:

Most OSs support a notion that a program in execution may be regarded as a process. Clearly, with multiple applications active at the same time there are many processes that are active. The OS needs to manage all these processes. Often applications may spawn processes that need to communicate with each other. Such inter process communication forms the primary basis of distributed computing.

With the coming together of communications network and computers it is possible to create processes on one machine and seek services from another machine. The machine seeking the services is called ***client machine*** and the machine offering services is called ***server machine***. When you use a web browser, your computer is the client and the machine which provides the content for browsing is the web server. All modern OSs support client-server operations for service provisioning.

Tools:

One important issue is: how does one use the services efficiently? In addition to the device handling utilities, most OSs provide many other general purpose utilities as a set of packaged tools. The tools help us to think in terms of higher level of operations. In the absence of tools and tool kits we would be required to write a fresh program each time.

As an example, consider a sort utility in UNIX. It supports identification of fields in a file with multi field structured data. Sort allows us to sort on a chosen field on a file. Imagine if we had to write a sort program every time we had a set of records making a file. Also look at the support we derive from the compression software tools like compression of files for long-term storage or transmission (Zip tools in windows environment). It would be stupendous task to write a compression program for every file transfer.

Trends in Computing:

The emergence of micro-computer, in particular Apple mackintosh, offered improvement in the manner in which human computer interfaces could be utilized for I/O and also for launching the applications. Apple was the first computer system to use mouse clicks to launch application. It also was the first environment to offer a "drag and drop" facility. In fact this set trends to offer icon driven convenient ways of launching applications.

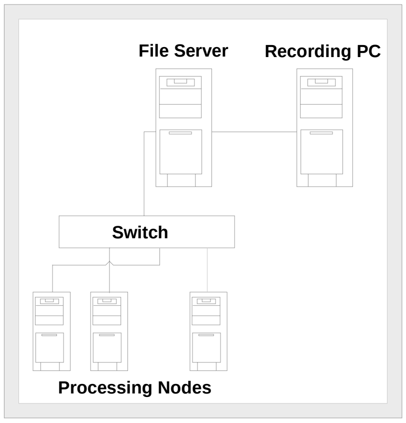
Apple Mac Computer

In this section we shall examine other trends in computing. The application scenario can considerably influence the types of services that need to be managed by an operating system. One such example is the emergence of parallel computing.

Parallel Computing:

There are many problem solving situations like weather forecasting, image processing, statistical data analysis and simulation, pharmaceutical drug synthesis and others where using a single processor becomes a limiting factor

For this class of problems one has to resort to parallel processing. In a parallel processing environment we may have a problem split in such a way that different parts of the same problem use a different processor with minimal communication. Parallel processing together with the network based services supports a mode of computing which is referred some times as distributed computing. In distributed computing environment it is also possible to support distribution of data, files and execution of processes.



Parallel Computing

In some other situation a computer system may be a part of a control system. For instance, the system may be utilized for regulating a utility – like electrical power supply. There is usually an elaborate instrumentation to keep track of the health of plant or a process. In these situations the instrumentation monitors physical quantities like pressure, temperature or speed and the process needs to respond in a real-time mode to be able to regulate the operation of the system.

Real - Time Systems:

Another category of real-time systems are those that are used to offer services like ATM (Automatic teller machine) or airlines reservation systems. The systems of the latter kind are often called transaction oriented systems. This is because the basic information processing involves a transaction.

As an example, consider processing a request like withdrawing money. It is a financial transaction, which requires updating accounts within an acceptable response time. Both the process control systems and transaction oriented systems are real time systems in which the response time to a request is critical. The main objective is to assure timeliness in response. The transaction oriented systems may even use the computer network extensively and often. The process control systems may be bus based or may have local area network as well. Both these systems invariably recognize an event and offer a response. In a transaction oriented system like ATM, it may be a card input. In a process control system an event may be detected when the temperature in a process vessel exceeds a certain limit. The operating system used in such real-time systems is often referred to as RTOS.

ATM

In addition, to the above kind of systems, we are now witnessing emergence of a class of systems that are embedded within an application.

For instance, it is not unusual to find up to four microprocessors in a car. These may be used for regulating fuel injection, cruise control or even operation of brakes. Similarly, there are microcontrollers embedded in washing machines. The presence of these systems is totally transparent to the user. He does not experience the presence of the system while using the gadgets. This is because the operating environment only requires minimal control buttons and response panels. The embedded systems are designed completely with a different philosophy. These are not general purpose computing environments. Instead, these have dedicated mode of operation. In these cases the operating system is not aimed at raising through put with general purpose utility back-ups. Instead these systems are designed with minimal and dedicated services. The minimal service provisioning is done using a minimal OS kernel often called micro-kernel. A well known embedded system is a common mobile phone and a personal digital assistant (PDA). One important Indian contribution in this class of systems is a Simputer. A Simputer can be utilized as a PDA. Simputer can also be used in a dedicated application. Simputers have also been upgraded to support wireless connectivity.

Simputer

Wireless Systems:

Wireless systems in general allow access from any where any time. These are also called ubiquitous systems. The ubiquity comes from the fact that unlike a wired system, the medium of communication is air which can be utilized from anywhere anytime.

Finally, the other major trend which we are witnessing now is driven by Web – the world wide web. All modern systems are internet compliant and allow a user to connect to the rest of the world. Web offers commercial organizations to offer their merchandise on the web. Also, it gives the consumers an opportunity to seek services using the web. In our country now Web is quite extensively utilized for railway reservation and also for the air ticket booking. Web can also offer other services. For example, down loading music is common. Web can, and in due course of time will, offer services which are currently offered by operating systems. In fact, then we will have the paradigm where “network is the computer” as was proclaimed by the SUN CEO, Scott McNealy a few years ago.