EVOLUTION OF OPERATING SYSTEMS

Operating system and computer architecture have had a great deal of influence on each other. Operating systems were developed mainly to facilitate the use of the hardware and to bring it to the best advantage. Here we will briefly make a sketch of the evolutionary path of OS development.

Serial Processing

Before 1950's the programmers directly interact with computer hardware, there was no OS at that time. If the programmer want to execute the program on those days, he has to follow some serial steps:

- Type the program on punched card.
- Convert the punched card to card reader.
- Submit to the computing machine, if any error in the program, the error condition was indicated by lights.
- The programmer examine the registers and main memory to identify the cause of error.
- Take the output on the printers.
- Then the programmer is ready for the next program.

This type of processing is difficult for users, it takes much time and next program should wait for the completion of previous one. The programs are submitted to the machine one after the other. So, this method is called as "Serial processing".

Batch Processing

In olden days(before 1960's), it is difficult to execute a program using computer. Because the computer is located in different rooms, one room for card reader and one for executing the program and another room for printing the output. The user or machine operator, running between these three rooms to complete a job. This problem was solved by batch processing system.

In batch processing technique similar type of jobs batch together and execute at a time. The operator carries the group of jobs at a time from one room to another. Therefore the programmer need not run between these three rooms several times.

The batch processing had an advantage .In that for one batch, the compiler, assembler, the loader etc had to be loaded only once, thus reducing the setup time to some extent. For example, FORTRAN programs were grouped together as one batch say batch 1, the PASCAL programs into another batch say Batch 2, the COBOL programs into another batch say Batch 3, and so on. Now the operator can arrange for the execution of these source programs which has been batched together one by one. After the execution of batch1 was over, the operator would load the compiler, assembler and loader, etc for the batch 2 and so on.

Setup time for batch 1	Runtime for batch 1	Setup time for batch 2	Runtime for batch 2	...	...	...

Fig. *Batch Processing*

The main advantage of batch processing is setup time will be reduced to a large extent, but the disadvantage is that the CPU is idle for the time in between two batches.

If the programs were not batched up together, the set up time would be much more higher.

Setup time for program 1	Runtime for program 1	Setup time for program 2	Runtime for program 2	...	...	...

Multiprogramming

Multiprogramming is a rudimentary form of parallel processing in which several programs are run at the same time on a uniprocessor. Since there is only one processor, there can be no true simultaneous execution of different programs. Instead the processor executes part of one program, then part of another, and so on. But to the user it appears that all programs are executing at the same time.

In multiprogramming, number of processes are reside in main memory at a time. The OS picks and begins to execute one of the jobs in the main memory. For example, consider the main memory consisting of 5 jobs at a time, the CPU executes one by one.

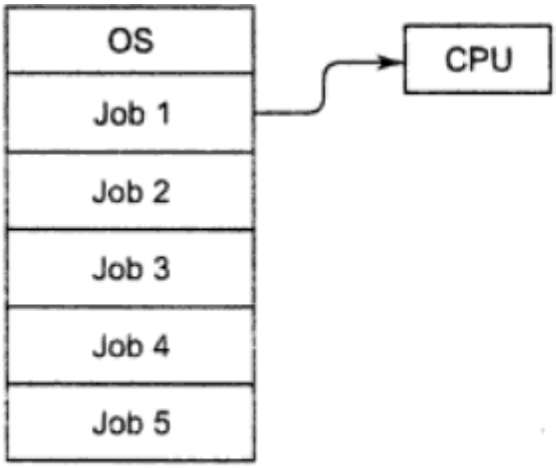

Fig. *Multiprogramming*

In non-multiprogramming system, the CPU can execute only one program at a time, if the running program waiting for any I/O device, the CPU becomes idle, so it will effect on the performance of the CPU.

But in multiprogramming environment, any I/O wait happened in a process, then the CPU switches from that job to another job in the job pool. If enough jobs could be held in main memory at once, the CPU is not idle at any time.

For Example: The idea is common in other life situations. The doctor does not have only one patient at a time, number of patients reside in the hospital under treatment. If the doctor has enough patients a doctor never needs to be idle.

Distributed Systems

A recent trend in computer system is to distribute computation among several processors. The processors in distribute system may vary in size and function, and referred by a number of different names such as sites, nodes, computers and so on depending on the context.

A distributed system is basically a collection of autonomous (independent by function) computer systems which co-operate with one another through their hardware and software interconnections.

In distributed systems, the processors cannot share memory or time, each processor has its own local memory. The processors communicate with one another through various communication lines such as high speed buses .These systems are also called as *"Loosely Coupled systems"*.

Distributed system = Network + Transparency(Invisible)

Advantages

1. **Resource sharing:** If a number of sites connected by a high speed communication lines, it is possible to share the resources from one site to another site.

   For example, S_1 and S_2 are two sites, these are connected by some communication lines, the site S_1 having the printer, but S_2 does not having the printer. Then the system can use the printer at S_1 without moving from S_2 to S_1.Therefore resource sharing is possible in distributed systems.
2. **Computation speedup:** A big computation is partitioned into number of partitions, these sub-partitions run concurrently in distributed systems.

   For example, site S_1 need to execute a big computation, this computation is divided into sub computations and these are executed by some other machines in different sites.
3. **Reliability:** If a resource or a system failed in one site due to technical problems. We can use other systems or other resources in some other sites.
4. **Communication:** Distributed systems provides communication which is not at all possible, that much in a centralized system. For Example, E-mail

Time Sharing Systems

Multiprogramming features were superimposed on batch processing to ensure good utilization of CPU but from the point of view of a user the service was poor as the response time, *i.e*, the time elapsed between submitting a job and getting the results was unacceptably high. Development of interactive terminals changed the scenario. Computation became an on-line activity. A user could provide inputs to a computation from a terminal and could also examine the output of the computation on the same terminal. Hence the response time needed to be drastically reduced. This was achieved by storing programs of several users in memory and providing each user a slice of time on CPU to process his/her program.

Time sharing or multitasking is a logical extension of multiprogramming. In time sharing environment, a number of jobs are loaded on to the memory and a number of users are communicating with the computer through different terminals. The OS allocates a fixed time interval (TIME SLICE) to each program in memory. Thus each program in memory is executed for a fixed interval of time.

As soon as the time allotted for a particular program is completed, the CPU starts executing the next program. This process is continued till all the programs in the memory are executed. A program may need number of time slices for its complete execution. Although the computer system is executing one job at a time, due to the speed of the CPU, every user on a terminal has the feeling that his program that is being executed continuously, because, after every time slice, the user gets a response from the computer. The user on the terminal is communicating with his running program, and is able to debug and experiment with his program.

Thus, the OS for a time sharing computer system has all the capabilities of a multiprogramming OS, but along with an additional capacity of allocating a fixed time slice of CPU to each program.

- Main advantage of time sharing system is efficient CPU utilization.
- The user can interact with the job while it is executing, but it is not possible in batch systems.

Personal-Computer Systems(PCs)

A personal computer (PC) is a small, relatively inexpensive computer designed for an individual user. In price, personal computers range anywhere from a few hundred dollars to thousands of dollars. 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.

Parallel Systems

Almost all the systems are uni-processor systems *i.e.*, they have only one CPU. Systems in which there are more than one CPU is called as Multi-processor systems. These systems have been developed to enhance the computing power of a computing system, and the features of this system is that, they share the memory, bus and the peripheral devices. These systems are referred as *"Tightly coupled systems"*. A system consisting of more than one processor and it is a tightly coupled, then the system is called "Parallel system".

In parallel systems number of processors executing their jobs in parallel (simultaneous process). Multi-processor systems are divided into following categories:

- Symmetric
- Asymmetric

In symmetric multi-processing, each processor runs a shared copy of operating system . The processors can communicate with each other and execute these copies concurrently. Thus, in a symmetric system, all the processors share an equal amount of load .Encore's version of UNIX for the Multimax computer is an example of symmetric multiprocessing. In this system various processors execute copies of UNIX operating system, thereby executing M processes if there are M processors.

Asymmetric multi-processing is based on the principle of master-slave relationship. In this system, one of the processors runs the operating system and that processor is called the master processor. Other processors run user processes and are known as slave processors. In other words, the master processor controls, schedules and allocates the task to the slave processors. Asymmetric multi-processing is more common in extremely large systems, where one of the time consuming tasks is processing I/O requests. In the asymmetric systems the processors do not share the equal load.

Advantages:

1. It results in saving money compared to the stand alone systems, since CPU'S can share memory, bus and peripherals.
2. Throughput can be increased
3. They increase the reliability.

Since there are more than one CPU, the failure of one or more of the CPU does not halt the entire system, but only slows down the work. For example, if there are five processors, all the five working together gives full efficiency. If two CPU's fail, then the system still works but only at 60% efficiency. This indicates increased aspect of reliability compared to stand alone systems.

Special purpose systems

a) Real-Time EmbeddedSystems

These devices are found everywhere, from car engines and manufacturing robots to DVDs and microwave ovens. They tend to have very specific tasks.

They have little or no user interface, preferring to spend their time monitoring and

managing hardware devices, such as automobile engines and robotic arms.

b) Multimedia Systems

Most operating systems are designed to handle conventional data such as text files, programs, word-processing documents, and spreadsheets. However, a recent trend in technology is the incorporation of multimedia data into computer systems. Multimedia data consist of audio and video files as well as conventional files. These data differ from conventional data in that multimedia data-such as frames of video-must be delivered (streamed) according to certain time restrictions (for example, 30 frames per second). Multimedia describes a wide range of applications in popular use today. These include audio files such as MP3, DVD movies, video conferencing, and short video clips of movie previews or news stories downloaded over the Internet. Multimedia applications may also include live webcasts (broadcasting over the World Wide Web)

c) Hand held Systems

Handheld Systems include personal digital assistants (PDAs, cellular telephones. Developers of handheld systems and applications face many challenges, most of which are due to the limited size of such devices. For example, a PDA is typically about 5 inches in height and 3 inches in width, and it weighs less than one-half pound. Because of their size, most handheld devices have small amounts of memory, slow processors, and small display screens.

REAL-TIME OS

In a time shared computer system, generally the computer response time is of the order of 0.5 to 2 seconds, which means a user will get computers attention after this much of time. Longer response times may be irritating but not hazardous.

However a real-time OS is needed for the computer systems controlling a process or a real time situation, such as a machine or a satellite. In this case two important points to be noticed are:

- The OS should provide for interactive processing.
- The response time should be very small.

The sensors bring in the data from a device, the OS instructs the computer to analyze the data and send appropriate signals back to the device. Any delay on the part of the computer system or the OS can be catastrophic. Thus, the real-time OS have to work strict time limits and have to be quick. Apart from this, these systems must be highly reliable to avoid failure of the system being controlled.

Here the main job of OS is instant handling of the signals or interrupts sent by the device which is being controlled by the computer system.

Real-time systems are systems that have in-built characteristics as supplying immediate response. A primary objective of the real-time system is to provide quick response time. User convenience and resource utilization are of secondary concern to real-time systems.

Real time System is of two types:

- **Hard real-time**
  - Guarantees that critical tasks complete within time.
  - All the delays in the system are bounded.
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- **Soft real-time**
  - Critical time tasks gets priority over other tasks, and retails that priority until it completes.
  - Limited utility in industrial control of robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.