1. Introduction to O.S.

1.1 O.S. Definitions:

Operating systems is a program that acts as an intermediary between a user of a computer and the computer hardware (H/W). The purpose of an operating system is to provide the environment in which the user can execute programs. The O.S. is a resource allocator (managers all resources, decides between conflicting requests for efficient and fair resource use) and is a control program (controls execution of programs) to prevent errors and improper use of the computer.

1.2 Computer System Components:

An O.S. is an important part of almost every computer system. A Computer System can be divided roughly into four components:

1. The (H/W) (cpu, memory, I/O devices).
2. Operating system (O.S.).
3. Application programs (Assembler, data base, compiler, text, and editor).
4. The users (people, machines, other computers). See figure (1.1).

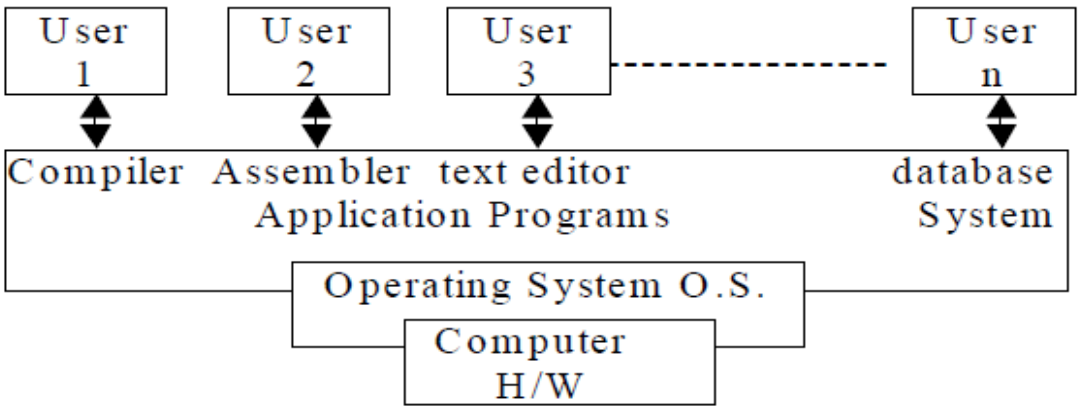

Figure 1.1 Abstract view of the components of computer system

1.3 O.S. goals:

1. The primary goal of an O.S. is to make the C/S. convenient to use. Operating systems exist because they are supposed to make it easier to compute with them than without them. This view is particularly clear when you look at operating systems for small personal computers.

2. A secondary goal is to use the computer H/W in an efficient manner. This goal is particularly important for large, shared multiuser systems. These systems are typically expensive, so it is desirable to make them as efficient as possible. These two goals-convenience and efficiency-are sometimes contradictory. In the past, efficiency considerations were often more important than convenience. Thus, much of operating-system theory concentrates on optimal use of computing resources.

1.4 The O.S. Functions:

O.S. performs many functions such as: -

1. Implementing the user interface.

2. Sharing H/W among users.

3. Allowing users to share data among them selves.

4. Preventing users from interfering with one another.

5. Scheduling resources among users.

6. Facilitating I/O.

7. Recovering from errors.

8. Accounting for resource usage.

9. Facilitating parallel operations.

10. Organizing data for secure and rapid access.

11. Handling network communications.

1.5 The O.S. Development history:

Operating system has developed over the last 50 years through a number of distinct phases or generations to the decades: -

- Early history (The 1940's and 1950's): In this time the earliest Electronic digital computers had no O.S.
  - Machines of that period were so primitive.
  - The programs were often entered one Bit at a time on rows of mechanical witches.
  - Eventually machine language programs were entered on punched cards, and assembly languages were developed to speed the programming process.
  - The 1st O.S. implemented in the early 1950's, this system run one job at a time and smoothed The transition between jobs to get maximum utilization of the computer system.
  - This type of O.S. called single-stream Batch processing system, because program and data were submitted in groups or batches.

- O.S. 1960's:
  - They were also batch-processing systems, but they were able to take advantage of computers resources by running several jobs at once.
  - The O.S. designer developed the concept of multiprogramming, and software engineering field was appeared.

- O.S. 1970's:
  - They were multimode time-sharing system that supported batch processing, timesharing, and real-time applications.
  - The P.C was in its 1st development stage.
  - Communication between C/S became widely used.

- Communication in local area networks (LAN) was made practical and economical by Ethernet standard.

O.S. 1980's:

- The 1980's was decade of P.C and the work station.
- Microprocessor technology it became possible to build desktop computers as power full as the mainframe of 1970's.
- Individuals could have their own computers for performing their work and they could use communication facilities for transmitting data between systems.
- Computing was distributed to the sites, which it was needed, rather than bringing the data to be processed to some central-scale computer installation.
- Application software packages are available such as:

1. Spread sheet programs.

2. Word processors.

3. Database packages.

4. Graphics packages.

5. Transfer information between computers in computer network (E-mail, Remote DB access application . . . etc) were widely used.

6. The client/server model became wide spread: clients are network users that need various services performed; servers are H/W, S/W and network components that perform these services.

The O.S. 1990's and beyond:

- In the 1990's we enter the area of true distributed computing in which computations will be divided into sub-computations that can be executed on other processors in multiprocessor computer network.
- Networks will be dynamically configured, they will continue operating even as new devices and S/W are added or removed by using registration procedure.

1.6 O.S. Categories:

The main categories of modern O.S. may be classified into three groups, which are distinguished by the nature of interaction that takes place between the computer and the user:

1. Batch -System

In this type of O.S. Users submit jobs on a regular schedule (e.g, daily, weekly, monthly) to a central place where the user of such system did not interact directly with C/S. To speed up processing, jobs with similar needs were batched together and were run through the computer as a group. Thus, the programmers would leave their programs with the operator. The major task of this type was to transfer control automatically from one job to the next. The O.S always resident in memory as in the figure (1.2). The output from each job would be send back to the appropriate programmer.

- **Advantages** : the batch system are very simple.
- **Disadvantages:**
  - There is no direct interaction between the user and the job while that job is executing.
  - The delay between job submission and job completion (called turnaround time) may result from amount of computing time needed.

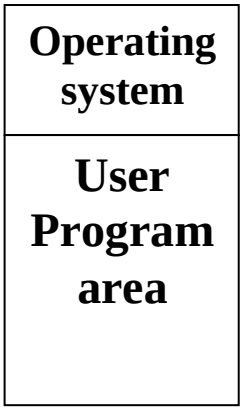

Figure 1.2 Memory layout for a simple batch system

2. Time-sharing system:

This type of O.S. provides online communication between the user and the system, where the user will give instruction to the O.S. or to the program directly (usually from terminal) and receives an immediate response, therefore some time called an interactive system.

The Time-Sharing system allows many users simultaneously share the computer system where little cpu time is needed for each user. As the system switches rapidly from one user to the next user is given the impression that they each have their own computer, while actually one C/S shared among the many users. The figure 1.3 show the time-sharing layout.

Advantage: Reduce the cup idle time.

Disadvantage: more Complex.

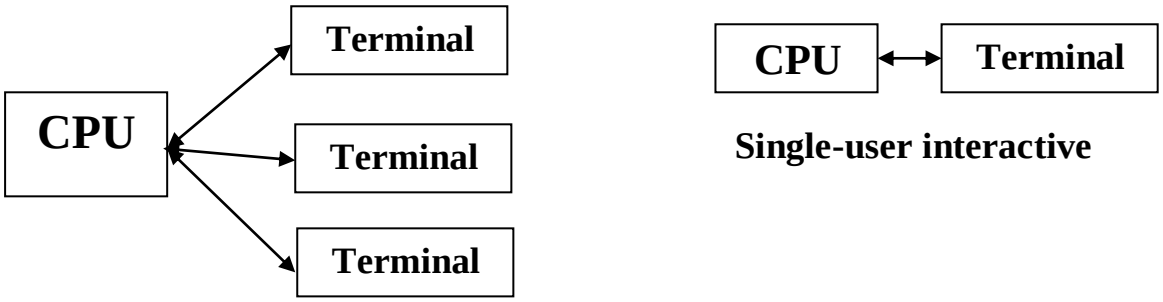

Figure 1.3 Time-sharing system

3. Real-Time system

A Real-Time system is used when there are rigid time requirements on the operation of a processor or the flow of data. A Real-time system guarantees that critical tasks complete on time. The secondary storage of any sort is usually limited; data instead being stored in short-term memory (Rom), the figure 1.4 shows the layout Real-Time system. ((The Airline Reservation system is an example of this type)).

1.7 Performance Development:

O.S. attempted to schedule computational activities to ensure good performance, where many facilities had been added to O.S. some of these are:

1.7.1 On-Line and off-Line operations:

A special subroutine was written for each I/O device called a device-driver, and some peripherals (I/O devices) has been equipped for either On-Line operation, in which they are connected to the processor, or off-line operations in which they are run by control units not connected to the central C/S. card-to-tape or Tape-to-Print operations are performed by off-line units. The figure 1.5 illustrates the on-line and off-line operation

The main advantage of off-line operation was that, the CPU was no longer constrained by the speed of C/R and L/T, but only by the speed of M/T unit.

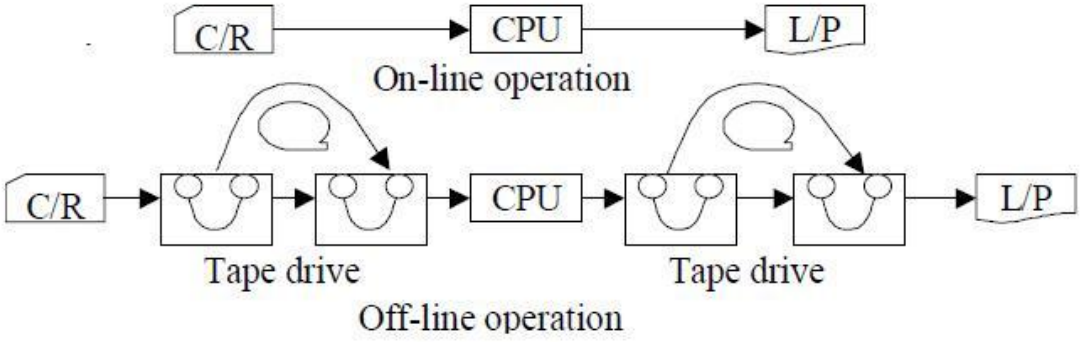

Figure 1.4 On-line and Off-line layout

1.7.2 Buffering

A buffer is an area or primary storage for holding data during I/O transfers, where the I/O transfer speed depends on many factors related to I/O Hardware but normally unrelated to processor operation.

On input the data placed in the buffer by an I/O channel when the transfer is complete the data may be accessed by the processor.

There are two types of buffering:

1.The single-buffered:

The channel deposits data in a buffer the processor will accessed that data the channel deposits the next data, etc. while the channel is depositing data no, processing on that data may occur.

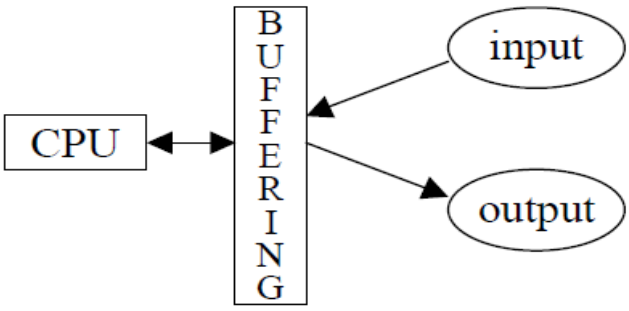

Figure 1.5 Buffering layout

2. The Double-buffering:

This system allows overlap of I/O operation with processing; while the channel is depositing data in one buffer the processor may be processing data in the other buffer. When the processor is finished processing data in one buffer it may process data in the second buffer. In buffering the CPU and I/O are both busy.

1.7.3 Spooling: (Simultaneous Peripheral Operation On-line)

1. Spooling uses the disk as a very large buffer for reading as far a head as possible on input devices and for storing output files until the output devices are able to accept them.

2. Spooling is now a standard feature of most O.S.

3. Spooling allows the computation of one job can overlap with the I/O of another jobs, therefore spooling can keep both CPU and the I/O devices working as much higher rates.

4. The figure behind show the spooling layout.

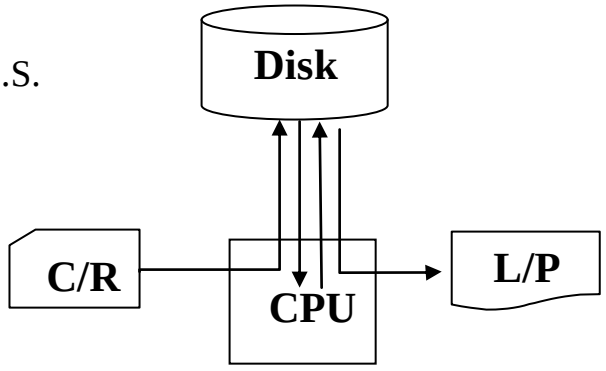

Figure 1.6 the spooling layout

1.7.4 Multiprogramming

1. Spooling provides an important data structure called a job pool kept on disk. The O.S. picks one job from the pool and begin to execute it.

2. In multiprogramming system, when the job may have to wait for any reason such as an I/O request, the O.S. simply switches to and executes another job. When the second job needs to wait the CPU is switches to another job and So on. Then the CPU will never be idle.

3. The figure shows the multiprogramming layout where the O.S. Keeps Several jobs in memory at a time. This set of jobs is a subset of the jobs kept in the job pool.

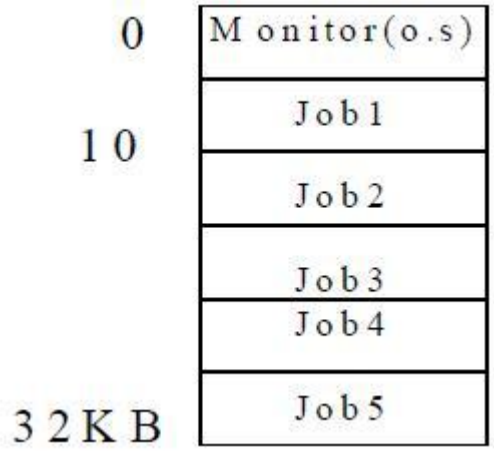

Figure 1.7 the multiprogramming layout

1.7.5 Parallel Systems

1. Most systems today are a single-processor system that is they have one main CPU.

2. There is a trend to have multiprocessor system, where such communication sharing the computer Bus, the clock, and some times memory and peripheral devices, as in the figure behind.

3. The advantage of this type of systems to increase the throughput (the number of jobs completed in unit of time).

4. Multiprocessors can also save money compared to multiple single systems because the processors can share peripherals, cabinets, and power supplies.

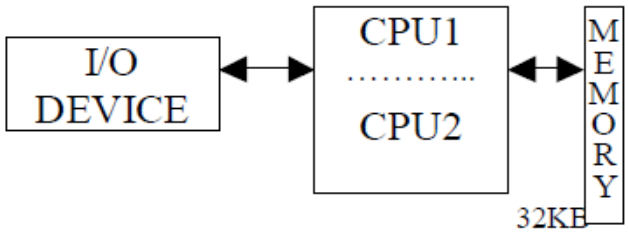

Figure 1.8 Parallel system layouts

5. Another reason for multiprocessor systems is that they increase reliability. The most common multiple-processor systems now use the ***symmetric multiprocessing*** model, in which each processor runs an identical copy of the operating system, and these copies communicate with one another as needed. Some systems use ***asymmetric multiprocessing,*** in which each processor is assigned a specific task. A master processor controls the system; the other processors either look to the master for instruction or have predefined tasks. This scheme defines a master-slave relationship. The master processor schedules and allocates work to the slave processors.

1.7.6 Distributed systems

1. A resent tread in C/S is to distribute computation among several processors.

2. In Contrast to the parallel system, the processors do not share memory and clock.

3. The processors communicate with one another through various communication lines, such as high speed buses or telephone lines. This type of systems called a distributed system.

There is a variety of reasons for building distributed systems, the major ones being these:

1. Resource sharing. If a number of different sites (with different capabilities) are connected to one another, then a user at one site may be able to use the resources available at another.

2. Computation speedup. If a particular computation can be partitioned into a number of sub computations that can run concurrently, then a distributed system may allow us to distribute the computation among the various sites to run that computation concurrently. In addition, if a particular site is currently overloaded with jobs, some of them may be moved to other, lightly loaded, sites. This movement of jobs is called *load sharing.*

3. Reliability. If one site fails in a distributed system, the remaining sites can potentially continue operating.

4. Communication. There are many instances in which programs need to exchange data with one another on one system. Window systems are one example, since they frequently share data or transfer data between displays.

1.8 Summary

Operating systems have been developed over the past 40 years for two main purposes. First, operating systems attempt to schedule computational activities to ensure good performance of the computing system. Second, they provide a convenient environment
for the development and execution of programs.

Initially, computer systems were used from the front console. Software such as assemblers, loaders, linkers, and compilers improved the convenience of programming the system, but also required substantial set-up time. To reduce the set-up time, facilities hired operators and batched similar jobs.

Batch systems allowed automatic job sequencing by a resident operating system and greatly improved the overall utilization of the computer. The computer no longer had to wait for human operation. CPU utilization was still low, however, because of the slow speed of the I/O devices relative to that of the CPU. Off-line operation of slow devices provides a means to use multiple reader-to-tape and tape-to-printer systems for one CPU. Spooling allows the

CPU to overlap the input of one job with the computation and output of other jobs.

To improve the overall performance of the system, developers introduced the concept of multiprogramming. With multi-programming, several jobs are kept in memory at one time; the CPU is switched back and forth among them to increase CPU utilization and to decrease the total time needed to execute the jobs.

Multiprogramming, which was developed to improve performance, also allows time sharing. Time-shared operating systems allow many users (from one to several hundred) to use a computer system interactively at the same time.

Personal computer systems are microcomputers that are considerably smaller and less expensive than are mainframe systems. Operating systems for these computers have benefited from the development of operating systems for mainframes in several ways. However, since individuals have sole use of the computer, utilization is no longer a prime concern. Hence, some of the design decisions that are made in operating systems for mainframes may not be appropriate for smaller systems.

Parallel systems have more than one in close communication; the share the computer bus, and sometimes share memory and peripheral devices. Such systems can provide increased throughput and enhanced reliability.

A distributed system is a collection of processors that do not share memory or a clock. Instead, each processor has its own local memory, and the processors communicate with one another through various communication lines, such as high-speed buses or telephone lines. A distributed system provides the user with access to the various resources located at remote sites.

A hard real-time system is often used as a control device in a dedicated application. A hard real-time operating system has well-defined, fixed time constraints. Processing must be done within the defined constraints, or the

system will fail. Soft real-time systems have less stringent timing constraints, and do not support deadline scheduling. We have shown the logical progression of operating-system development, driven by inclusion of features in the hardware that are needed for advanced operating-system functionality. This trend can be seen today in the evolution of personal computers, with inexpensive hardware being improved enough to allow, in turn, improved characteristics.