Evolution of Multi-user operating system

Multi-user operating system:

Is a computer operating system (OS) that allows multiple users on different computers or terminals to access a single system with one OS on it. These programs are often quite complicated and must be able to properly manage the necessary tasks required by the different users connected to it. The users will typically be at terminals or computers that give them access to the system through a network, as well as other machines on the system such as printers.

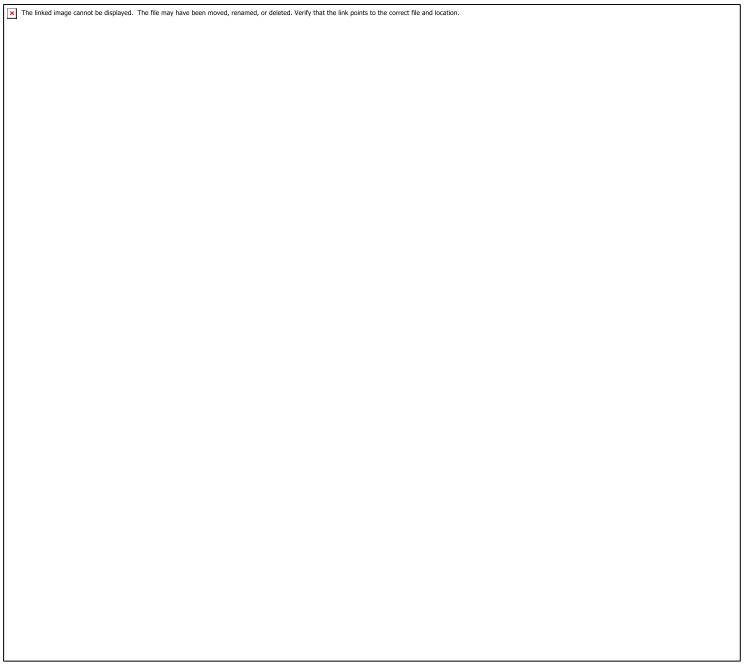
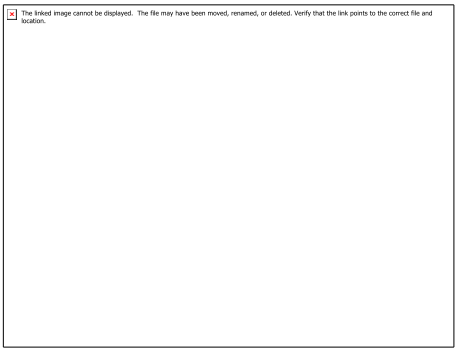


Diagram shows how OS is handling three user through 3 different terminals

- Ubuntu
- macOS
- All linux based OS
- Unix
- IBM AS400
- Windows 10

Single user operating system:

Becomes a mode where the user has a multipurpose computer screen to run the program and the operating system boots into a single super user that controls all the activities. The primary usage for such a system comes whenever the maintenance for several users takes place at the same time on the network servers. Single client mode is a mode in which a multi-user PC is working framework boots into a single super user.



- Windows 95
- MS-Dos
- Windows NT Workstation
- Windows 2000 professional

Key Difference: A single-user operating system is a system in which only one user can access the computer system at a time. On the other hand, a multi-user operating system allows more than one user to access a computer system at one time.

Comparison between Single User and Multi-User Operating System:

	Single User	**Multi-User**
Definition	A single user operating system provides facilities to be used on one computer by only one user.	A multi-user operating system has been designed for more than one user to access the computer at the same or different time.
Types	Single user, single task: A single task is performed by one user at a time. Example- The Palm OS for Palm handheld computers. Single user, multi-task: Several programs are run at the same time by a single user. For example- Microsoft Windows.	Time sharing systems: These systems are multi-user systems in which CPU time is divided among the users. The division is made on the basis of a schedule. Most batch processing systems for the mainframe computers can also be considered as 'multi user.'
Attributes	Simple	Complex
Examples	Windows 95, Windows NT Workstation and Windows 2000 professional.	Unix, Linux and mainframes such as the IBM AS400.

EVOLUTION OF OPERATING SYSTEMS

Serial Processing

Users access the computer in series. From the late 1940's to mid 1950's, the programmer interacted directly with computer hardware i.e., no operating system. These machines were run with a console consisting of display lights, toggle switches, some form of input device and a printer. Programs in machine code are loaded with the input device like card reader (Punch Card). If an error occurs the program was halted and the error condition was indicated by lights. Programmers examine the registers and main memory to determine error. If the program is success, then output will appear on the printer.

Main problem here is the setup time. That is single program needs to load source program into memory, saving the compiled (object) program and then loading and linking together.

Simple Batch Systems

To speed up processing, jobs with similar needs are batched together and run as a group. Thus, the programmers will leave their programs with the operator. The operator will sort programs into batches with similar requirements.

The problems with Batch Systems are:

- Lack of interaction between the user and job.
- CPU is often idle, because the speeds of the mechanical I/O devices are slower than CPU.

For overcoming this problem use the Spooling Technique. Spool is a buffer that holds output for a device, such as printer, that cannot accept interleaved data streams. That is when the job requests the printer to output a line that line is copied into a system buffer and is written to the disk. When the job is completed, the output is printed. Spooling technique can keep both the CPU and the I/O devices working at much higher rates.

Multi-programmed Batch Systems

Jobs must be run sequentially, on a first-come, first-served basis. However when several jobs are on a direct-access device like disk, job scheduling is possible. The main aspect of job scheduling is multiprogramming. Single user cannot keep the CPU or I/O devices busy at all times. Thus multi-programming increases CPU utilization.

In when one job needs to wait, the CPU is switched to another job, and so on. Eventually, the first job finishes waiting and gets the CPU back. The memory layout for multiprogramming system is shown below:

Time-Sharing Systems

Time-sharing systems are not available in 1960s. Time-sharing or multitasking is a logical extension of multiprogramming. That is processors time is shared among multiple users simultaneously is called time-sharing. The main difference between Multi-programmed Batch Systems and Time-

Sharing Systems is in Multi-programmed batch systems its objective is maximize processor use, whereas in Time-Sharing Systems its objective is minimize response time.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is seconds at most.

Operating system uses CPU scheduling and multi-programming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

For example IBM's OS/360. Time-sharing operating systems are even more complex than multi-programmed operating systems. As in multiprogramming, several jobs must be kept simultaneously in memory.

Personal-Computer Systems (PCs)

A computer system is dedicated to a single user is called personal computer, appeared in the 1970s. Micro computers are considerably smaller and less expensive than mainframe computers. The goals of the operating system have changed with time; instead of maximizing CPU and peripheral utilization, the systems developed for maximizing user convenience and responsiveness.

For e.g. MS-DOS, Microsoft Windows and Apple Macintosh.

Hardware costs for microcomputers are sufficiently low. Decrease the cost of computer hardware (such as processors and other devices) will increase our needs to understand the concepts of operating system. Malicious programs destroy data on systems. These programs may be self-replicating and may spread rapidly via worm or virus mechanisms to disrupt entire companies or even worldwide networks. MULTICS operating system was developed from 1965 to 1970 at the Massachusetts Institute of Technology (MIT) as a computing utility. Many of the ideas in MULTICS were subsequently used at Bell Laboratories in the design of UNIX OS.

Parallel Systems

Most systems to date are single-processor systems; that is they have only one main CPU. Multiprocessor systems have more than one processor.

The advantages of parallel system are as follows:

- throughput (Number of jobs to finish in a time period)
- Save money by sharing peripherals, cabinets and power supplies
- Increase reliability
- Fault-tolerant (Failure of one processor will not halt the system).

Symmetric multiprocessing model: Each processor runs an identical job (copy) of the operating system, and these copies communicate. Encore's version of UNIX operating system is a symmetric model. E.g., If two processors are connected by a bus. One is primary and the other is the backup. At fixed check points in the execution of the system, the state information of each job is copied from the primary machine to the backup. If a failure is detected, the backup copy is activated, and is restarted from the most recent checkpoint. But it is expensive.

Asymmetric multiprocessing model

Each processor is assigned a specific task. A master processor controls the system. Sun's operating system SunOS version 4 is a asymmetric model. Personal computers contain a microprocessor in the keyboard to convert the key strokes into codes to be sent to the CPU.

Distributed Systems

Distributed systems distribute computation among several processors. In contrast to tightly couple systems (i.e., parallel systems), the processors do not share memory or a clock. Instead, each processor has its own local memory.

The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, and computers and so on.

The advantages of distributed systems are as follows:

- Resource Sharing: With resource sharing facility user at one site may be able to use the resources available at another.
- Communication Speedup: Speedup the exchange of data with one another via electronic mail.
- Reliability: If one site fails in a distributed system, the remaining sites can potentially continue operating.

Real-time Systems

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. Real-time operating system has well-defined, fixed time constraints otherwise system will fail. E.g., Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-applicance controllers.

There are two types of real-time systems:

- **Hard real-time systems:** Hard real-time systems guarantees that critical tasks complete on time. In hard real-time systems secondary storage is limited or missing with data stored in ROM. In these systems virtual memory is almost never found.

- **Soft real-time systems**

Soft real time systems are less restrictive. Critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. E.g., Multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers.

System Calls

System calls provide the interface between a process and the OS. These calls are generally available as assembly language instructions. The system call interface layer contains entry point in the kernel code; because all system resources are managed by the kernel any user or application request that involves access to any system resource must be handled by the kernel code, but user process must not be given open access to the kernel code for security reasons. So that user processes can invoke the execution of kernel code, several openings into the kernel code, also called system calls, are provided. System calls allow processes and users to manipulate system resources such as files and processes.

System calls can be categorized into the following groups:

- Process Control
- File Management
- Device Management
- Information maintenance
- Communications