Date .....

Evolution of Operating System

Factors of Evolution

1. **Hardware upgrades** :- Advances in hardware technology lead to increased processing power, memory, and storage capacity, driving the need for more sophisticated OS management.
2. **New types of hardware** :- Introduction of new hardware components, such as networks, graphics card, and mobile devices, requires OS support and integration.
3. **New services** :- Emerging user needs and applications lead to the development of new OS services, such as:
   - Networking and communication protocols.
   - Graphical user Interfaces (GUIs).
   - Multitasking and multi-user support
   - security and access control mechanisms.
4. **Fixes** :- continuous improvement and bug fixing to ensure system stability, security, and performance.

=> A brief overview of major OS evolution milestones:

1. Serial Processing (1940s – 1950s)
2. Simple Batch system (1950s – 1960s).
3. Multiprogrammed System (1960s – 1970s)
4. Time-Sharing System (1970s – 1980s)
5. Personal Computer System (1980s – 1990s)
6. Parallel Systems (1990s – present)
7. Distributed Systems (1990s – present)
8. Mobile operating System (2000s – Present)

① Serial Processing

=) Direct Access to Hardware

- users accessed the system hardware directly, without an intermediary operating system.
- Machines were operated from a console with
  (i) display lights (ii) Toggle switches
  (iii) Input devices like card readers (iv) Printers

=) Major Issues

- scheduling: users had to schedule time to work on the machines, leading to inefficiencies.
- setup: The process of loading the compiler, source program, saving the compiled program, and loading and linking was time-consuming and cumbersome.
- CPU wastage: The expensive CPU resources was often kept idle, waiting for user input or output operations to complete.

② Simple Batch System

-> It's first type of operating system.

- Job submission:- users submit jobs (programs) to an operator, who batches similar jobs together.
- Monitor software:- A software called monitor controls the sequence of events and resides in the main memory (Resident Monitor).
- Job Execution:- The current job runs in the user program area, and control is passed to the job. The job returns control to the monitor when finished or if an error occurs.
- Job control language:- A special programming language provides instructions to the monitor,

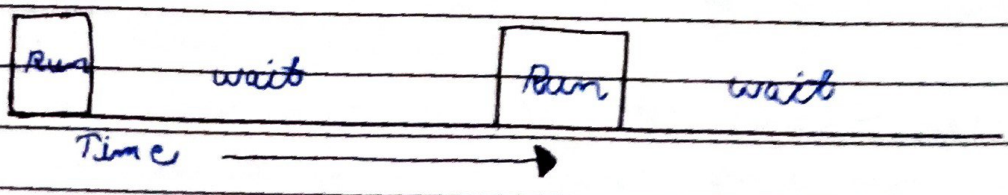
specifying the compiler, data, and hardware requirements

→ Hardware Features

- Memory protection:- Prevents the monitor's memory area from being altered.
- User mode:- user programs execute in user mode, with limited privileges.
- System mode (Kernel mode):- The monitor executes in system mode, allowing privileged instructions and access to protected memory.
- Timer:- Prevents a job from monopolizing the system
- Privileged Instructions:- Certain machine-level instructions can only be executed by the monitor.
- Interrupts:- Allow the OS to relinquish and regain control from user programs.

⇒ Issues with simple Batch systems

- I/O bottleneck:- I/O devices are too slow compared to the processor, causing the process to wait for I/O instructions to complete.
- Processor Idleness:- The processor is often idle, waiting for I/O operations to finish.



(a) Uniprogramming

(3) Multiprogrammed Batch System

- Job switching:- when one job waits for I/O, the processor switches to another job, saving time and improving efficiency.
- DMA (Direct Memory Access):- Allows I/O devices to transfer data directly to/from memory, without processor involvement.
- Memory Management:- Necessary to manage multiple jobs in memory, ensuring efficient use of resources and preventing conflicts.
- I/O Interrupts:- Notify the processor when I/O operations are complete, enabling job switching.
- Memory Residency:- Jobs remain in memory while waiting for I/O or processor time, reducing loading and reloading overhead.

=> Benefits

- Improved processor utilization:- processor switching b/w jobs reduces idle time.
- Increased Throughput: More jobs can be processed in less time.
- Better Resource Utilization: Memory and I/O devices are used more efficiently.

(4) Time Sharing System.

- Multi-user Access: Enabled multiple users to interact with a mainframe computer simultaneously.
- Processor Time sharing: processor time was allocated among users, improving resource utilization.
- System Clock Interrupts: Generated interrupts at regular intervals (~0.2 seconds) to switch b/w users.

- Time Slicing: The technique of allocating processor time to users in fixed intervals, called time slices.
- Context switching: The process of saving the current user program states (programs of data) to disk before loading the next user's programs and data.
- Memory management: user's programs and data were stored on disk and retrieved as needed, allowing efficient use of main memory.

Benefits

- Improved Resource Utilization :- Multiple users shared the processor, increasing overall productivity
- Enhanced User Experience :- users interacted with the system in real-time, without waiting for batch processing.
- Increased System Efficiency :- Time sharing reduced idle time and improved system throughput.

Multiprogramming vs Time sharing

	Batch Multiprogramming	Time sharing
Principal objective	Maximize processor use	minimize response Time
Source of directives to operating system	Job control language commands provided with the job	commands entered at terminal