Evolution of Operating Systems

(1) Serial Processing

→ Time period :- Late 1940s to mid-1950s
→ Programmer interacted directly with the computer hardware.
→ There was no OS; Run from a console.
→ Programs were in machine code.
→ Error condition was indicated by the lights in the console.
→ The output was taken via printer.

Two Main Problems.

(1) Scheduling.

→ To reserve computer time, a hardcopy sign up sheet was maintained.
→ A block of time was usually 30 mins.
→ So, if one were to finish early, it would result in wastage of processing time due to the unused time left.
→ Similarly, if one were not to finish in time, he/she is forced to stop before completion of his/her problem.

(2) Set-up time.

→ A single program (JOB), to run, would take the removal and assembly of multiple elements wrt the source & object program.
→ Each step could involve mount/dismount/setting up card decks.
→ If an error occured, the user would have to remove it all, and start from the first.
→ So, just for the running of the program, a considerable amount of time is spent.

∴ This mode of operation is called serial processing, as the user has access to the computer in series.

→ Linkers, loaders, debuggers have been used to make serial processing more efficient.

Simple Batch Systems

→ To battle high expense, set-up time and scheduling time, batch OS was developed in the mid 1950s.

→ First batch OS was developed by General Motors for use on IBM 701.

→ IBSYS for the 7090/7094 computers had the most widespread influence.

→ The central idea of BS OS was around the monitor.

Monitor

→ Computer operator takes the jobs from users via cards, and batches them sequentially, and places them in input device which is used by the monitor.

The programs branch back to the monitor after completion; the monitor then goes to the next program to load.

- **Monitor point of view :-**
  - → Monitor controls the sequence of events.
    majority
    Its, must be in the main memory & ready for execution.
    This portion is called resident monitor.
  - → Rest of the monitor consists of utilities & functionalities that are loaded as sub-routines at the beginning of any job.
  - → From input device, monitor reads in jobs one at a time. As it is read, the respective job is placed in the user program area, and the job gets control. After job completion, monitor regains control, which immediately reads the next job. The results of each job are sent to an output device.

Processor Point of view :-

→ Processor executes instructions in the resident memory which causes the next job to be read.

→ The job is then executed from the start of user-program in accordance to the branch instruction. It is executed until the ending or an error occurs. After which, next instruction is fetched.

→ "control is passed to a job" :- processor fetch/execute i in user program

→ "control is returned to monitor" :- " " " in monitor program

i → instructions.

→ The monitor performs a scheduling function where a batch of jobs are queued up and are executed rapidly as possible. No time waste. It improves job set up as well.

JCL :- Job control language.
Programming language used to provide instructions to the monitor.

ex: User submitting a program in FORTRAN plus some data to be used by the program.

Desirable features in Monitor (hardware).

- Memory protection.
- Timer.
- Privileged instructions.
- Interrupts.

User mode :- certain areas are protected from user's use in which certain instructions aren't executed.

Kernel mode :- in which, privileged instructions may be executed & in which protected areas of memory may be accessed.

∴ In a batch OS, execution of user program & monitor is executed by in alternating processor time.

2 sacrifices.
→ some main memory is given to monitor
→ Some processor time is consumed by monitor.

Multiprogrammed Batch Systems

→ As I/O devices are slower than the processor, in simple Batch OS, the processor is often idle.

∴ Multiprogramming.

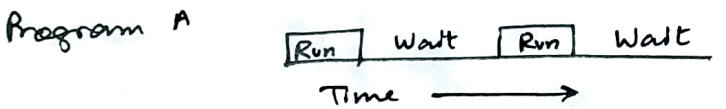
→ There must be enough memory to hold the resident monitor and one user program.

→ If there were to be multiple user programs, instead of staying idle for sequential execution, the processor can alternate between jobs and can avoid time wasted on waiting.

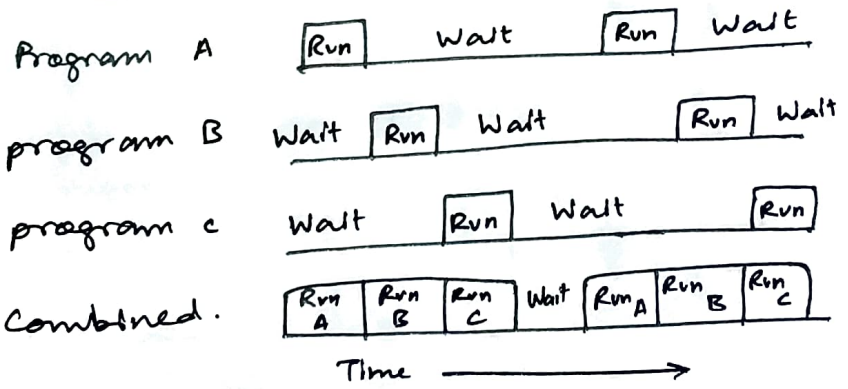
→ This is the central theme of modern operating systems.

Diagrams.

(a) Uniprogramming.



(b) Multiprogramming (with three programs)



→ Hardware that supports I/O interrupts and DMA is essential for multiprogramming.
This will be useful while passing control while jumping between jobs while execution.

∴ Multiprogramming is more sophisticated than uniprogramming.

Time sharing systems

→ A mode where user interacts directly with the computer is essential.

→ In time-sharing system, multiple users simultaneously access the system through terminals, with the OS leaving (interleaving) the execution of each user program in a short burst of computation.

ie, if there are n users actively requesting service at 1 time, each user will only see on the average 1/n of the effective computer capacity.

→ One of the first time-sharing OS was developed by CTSS at MIT known as Project MAC.
It was developed for the IBM 709 in 1961.

CTSS ran on a computer with 32,000 36-bit words of main memory. Resident memory consumed 5000 of that.

Time slicing :- At each clock interrupt, the OS regains control and reassign the processor to another user.

- To preserve old user program status, the old programs / data were written out to disk.
- To minimize disk traffic, user memory was only written out when the incoming program program would overwrite it.

	Batch Multiprogramming	Time sharing.
Objective	Maximize processor use	Minimize response time
Source of directive to OS	JCL ~~provided~~ commands provided with the job	commands entered at the terminal.

∴ Main advantage → quick response by reducing CPU idle time.

Major Achievments

→ Four major theoretical advances in the development of operating systems.

- **The process**
  → Central to the design of OS.

process
→ Program in execution
→ Instance of a program running on a computer
→ Entity that can be assigned/executed on a processor.

Three major lines of computer system development

Multiprogramming.
→ Designed to keep the processor & I/O devices simultaneously busy to achieve maximum efficiency.
→ In response to signals indicating the completion of I/O transactions, the processor is switched among the various programs residing in main memory.

Time sharing
→ Be responsive to the needs of the individual user & be able to support many users simultaneously.
→ They're compatible because of the relatively slow reaction time of the user.

Real time Transaction
→ 'n' no. of users are entering queries/updates against a database.
→ limited to 1 or few applications.
→ system response time is important.
→ These three created problems in timing & synchronization that contributed to the development of the concept of the process.

Process contains three components
→ An executable program
→ The associated data needed by the program
→ The execution context of the program.

errors caused.

→ Improper Synchronization
  → Routine must be suspended awaiting an event elsewhere in the system.

→ Failed mutual Exclusion
  → More than one user or program will attempt to make use of a shared resource at the same time.

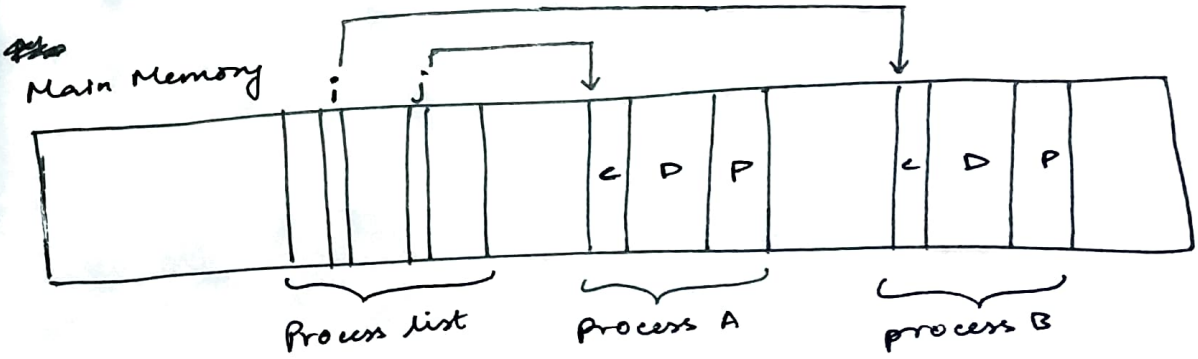
→ Nondeterminate Program Operation
  → Results of a program should depend only on the input of that program & not on the activities of other programs in a shared system.

→ Deadlocks.
  → Possibility for 2 or more programs to be hung up waiting for each other.

Process Management

→ The entire state of the process at any instant is contained in its context.

→ New features can be designed & incorporated into the OS by expanding the context to include any new info needed to support the feature.



C → context
D → Data
P → Program Code.

Typical Process Implementation.

Thread:- A single process, which is assigned certain resources, can be broken up into multiple concurrent threads that execute cooperatively to perform the work of the process.

Memory Management

→ System managers need efficient & orderly

& principal storage management responsibilities:-

- **Process isolation**:-
  → Prevention of independent processes from interfering with each other's memory.

- **Automatic allocation & management**
  → Programs should be allocated across the memory hierarchy as required
  Allocations should be transparent to the programmer.

- **Support of Modular Programming**
  → Programmers should be able to define, create, destroy & alter the size of modules dynamically.

- **Protection & access Control.**
  → OS must allow portions of memory to be accessible in various ways by various users.
  Sharing of memory, at any ~~time~~ level of the memory hierarchy, creates the potential for one program to address the memory space of another.

- **Long-term storage**
  → Many application programs require means for storing info for extended periods of time, after the computer has been powered down.

⇒ OS meets these requirements via,

File system facility :- Implements a long-term store, with information stored in named objects, ie files.

virtual memory :- A facility that allows programs to address memory from a logical point of view without regard to the amount of main memory physically available.

(i) Main Memory → Consists of a no. of fixed-length frames, each equal to the size of a page.
For a program to execute, some or all of its pages must be in main memory.

(ii) Secondary Memory → Can hold many fixed-length pages.

Paging → Pages :- fixed size blocks of comprised processes.
→ Provides for dynamic mapping between the virtual & real address in the program & main memory respectively.

Information Protection and Security

Four categories

→ Availability :-

Protecting the system against interruption

→ Confidentiality :-

Unauthorized users can't access all data

→ Data Integrity :-

Protection of data from unauthorized modification.

→ Authenticity :-

Validation/Verification of users & data sent.

Scheduling and Resource Management

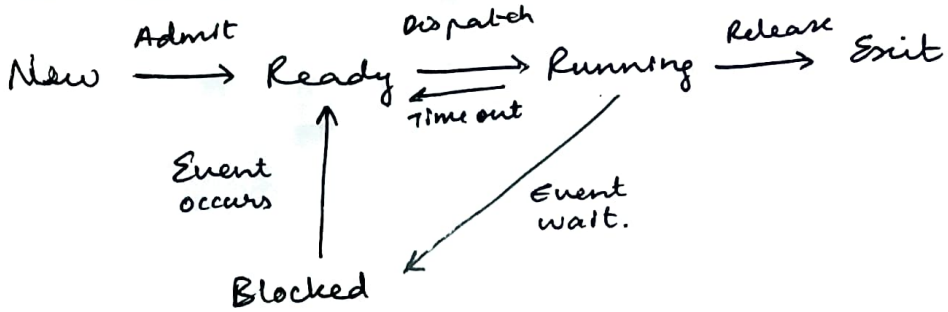
Three factors :-

- Fairness :- Equal and fair access of resources to all processes. Useful with jobs of similar demands.
- Differential Responsiveness :- Different service requirements will have different allocation of resources It should be allocated & scheduled wrt the total set of requirements.
- Efficiency :- Maximum throughput
  minimum response time
  Maximum user accomodation.

5 - State process Model.

⇒ Non Running state → Ready, Blocked.

Five state process Model.



Running :- Process that is currently being executed

Ready :- Process that is prepared to execute when given the opportunity.

Blocked :- A process that can't execute until some event occurs.

New :- A process that has just been created but hasn't been admitted into the pool of executable processes by the OS.

Exit :- A process that has been released from the executable processes by the OS.

— Bothe new & exit are two stage processes. They're both useful constructs for managing the process.

Drawbacks :-

- When process is terminated / exited by the OS, data isn't preserved.
- CPU stays idle if due to some error the processes are in blocked state.

Possible Transitions.

- Running → Ready. :- Reason :- Running process has reached the maximum allowable time for uninterrupted execution. Another reason may be differed levels of priority.
- Running → Blocked
- Running → Exit

Put in block state if it must wait for something it has requested.
Request is made as a system service call.

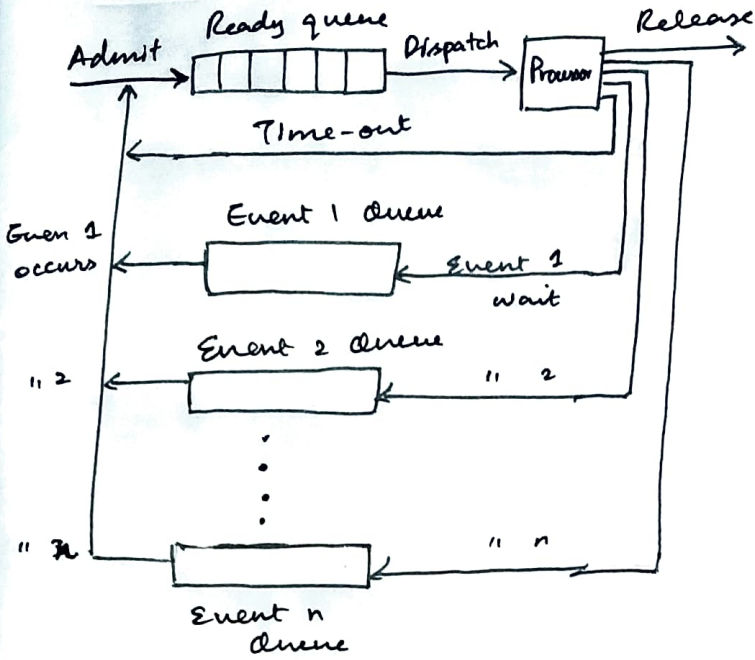
Current running process is terminated by the OS if the process is completed or aborted.

- Ready → Running :- When it is time to select a process to run, OS chooses one of the processes in the ready state.
  Job of dispatcher/scheduler.
- Ready → Exit

Not shown on state diagram.
Parent may terminate child processes.
Termination of parent may terminate child processes.

- Blocked → Ready :- Blocked state to ready state when event being waited for occurs.
- Blocked → Exit.
- New → Ready :- New to ready state when the OS is prepared to take on an additional process.
- Null → New :- A new process is created to execute a program.

Queueing Model.



Queueing Model can be used in absence of priority scheme.

First-In-First-out scheme.

Suspension & it's uses.

- Needed when process in main memory isn't immediately available for execution, whether or not it is awaiting an event.

→ Suspension is when the process isn't immediately available for execution.

→ The process may/may not be waiting for an event. Block condition is independant to suspend condition.

→ Placed in suspension state either by the process or it's parent process.

→ Process may not be removed from the state until agent asks explicitly order the removal.