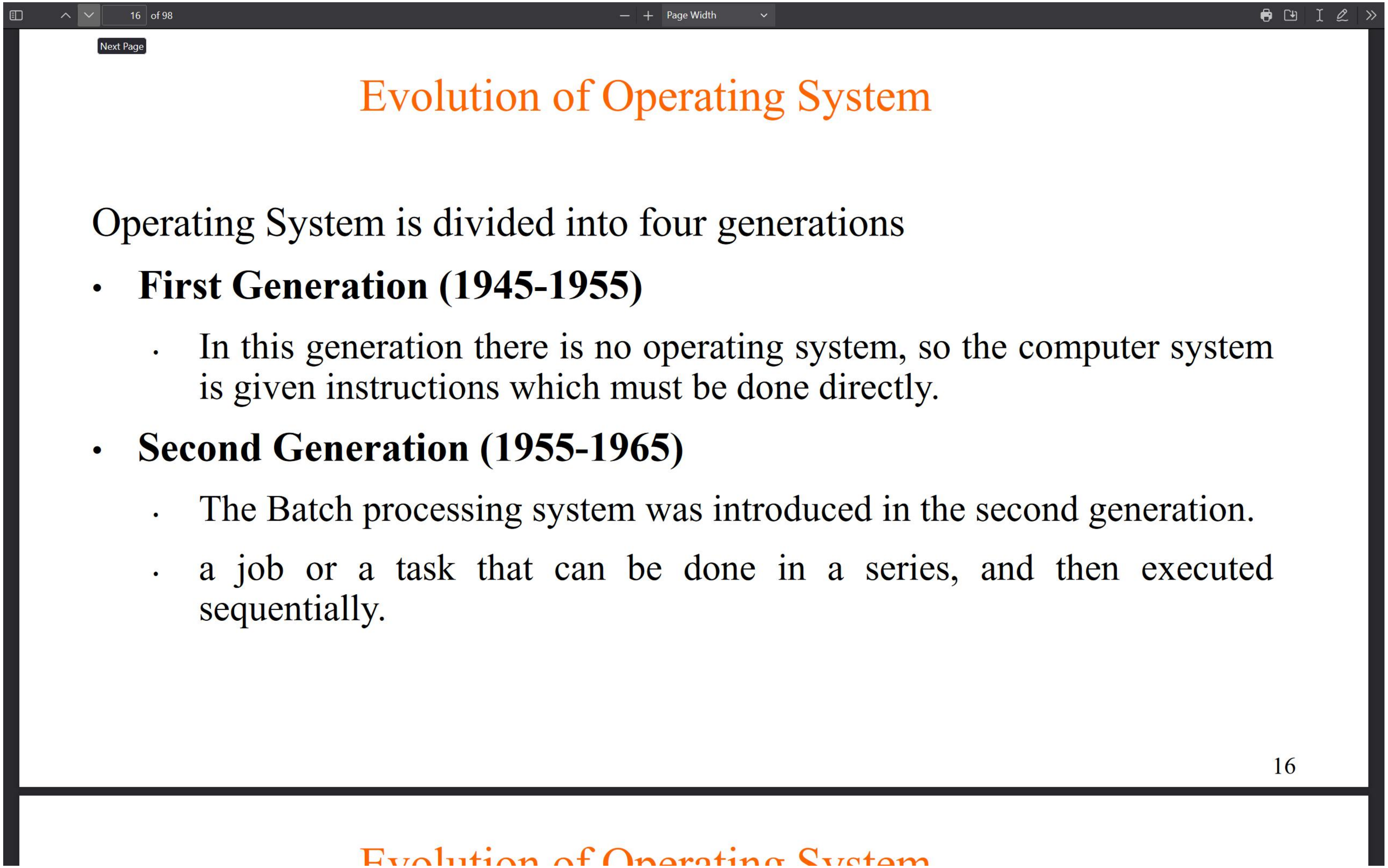


Evolution of Operating System

Operating System is divided into four generations

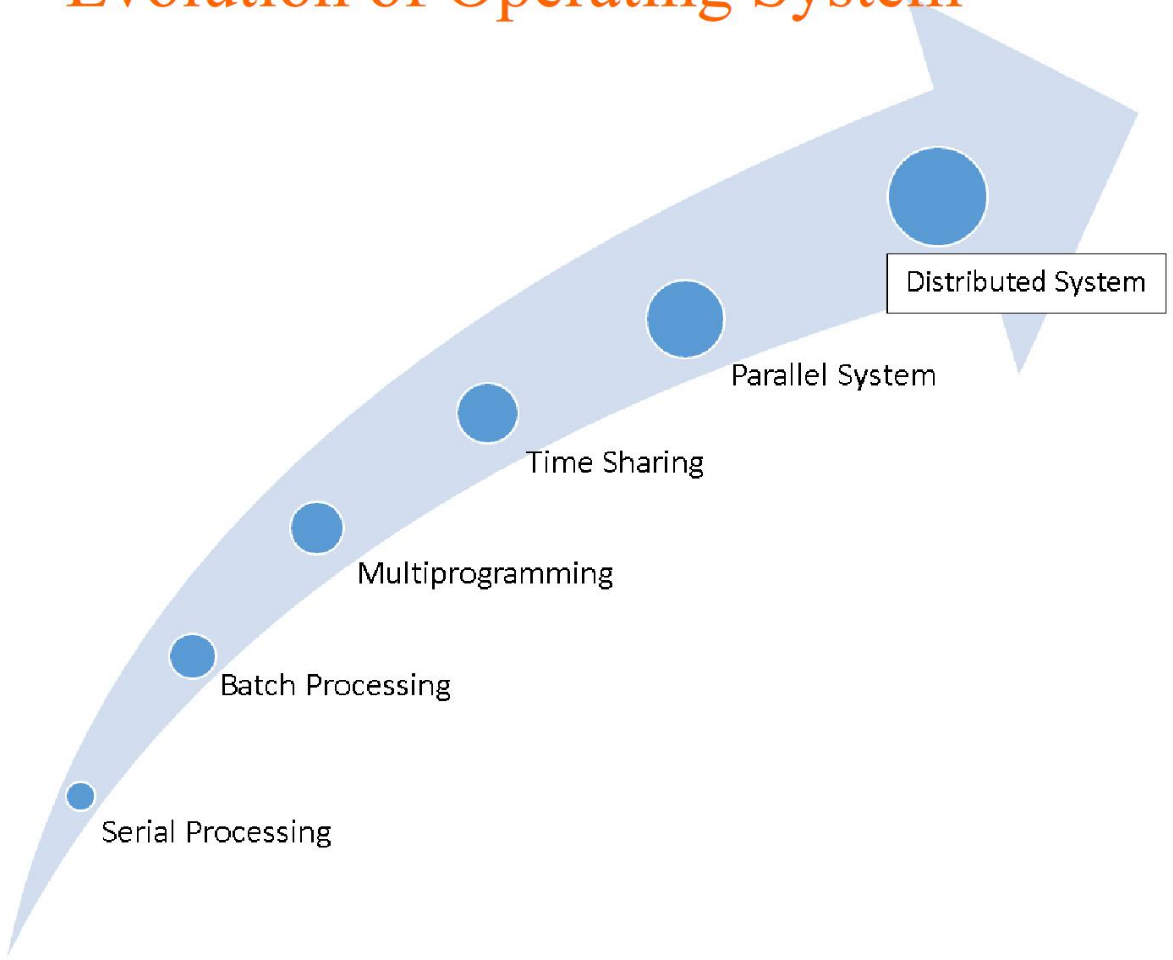
- **First Generation (1945-1955)**
  - In this generation there is no operating system, so the computer system is given instructions which must be done directly.
- **Second Generation (1955-1965)**
  - The Batch processing system was introduced in the second generation.
  - a job or a task that can be done in a series, and then executed sequentially.

Evolution of Operating System

Operating System is divided into four generations

- **Third Generation (1965-1980)**
  - OS was developed to serve multiple users at once in the third generation.
  - Users can communicate through an online terminal to a computer.
- **Fourth Generation (1980-Now)**
  - OS is used for computer networks where users are aware of the existence of computers that are connected to one another.
  - With the onset of new devices like wearable devices, which includes Smart Watches, Smart Glasses, VR gears etc, the demand for unconventional operating systems is rising.

Evolution of Operating System



Understanding the Evolution of Operating System

Understanding the Evolution of Operating System

- The **Serial Processing** OS are those which Performs all the instructions into a Sequence Manner.
- Program Counter is used for **executing** all the Instructions.
- **Punch Cards** are used where stiff paper holds digital data represented by the presence or absence of holes in predefined positions.

Understanding the Evolution of Operating System

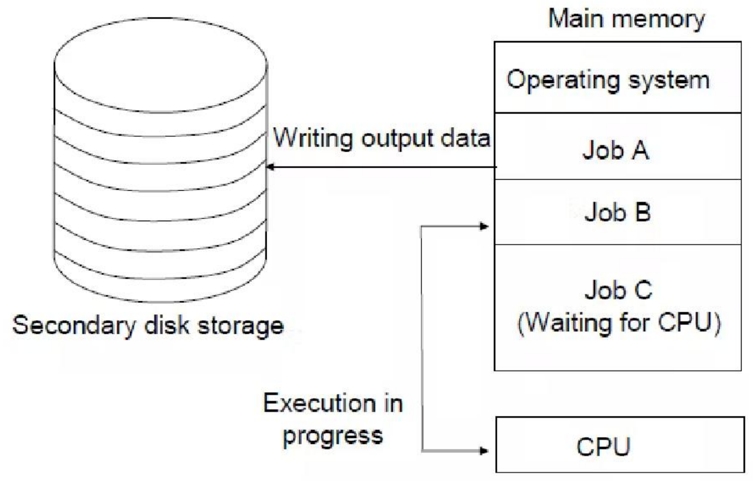
Understanding the Evolution of Operating System

- In **Batch Processing**, similar Types of jobs are first prepared and stored on the Card.
- The System perform all the operations on the Instructions one by one.
- OS will use the LOAD and RUN Operations.
- Jobs prepared for Execution must be the Same Type

Understanding the Evolution of Operating System

Understanding the Evolution of Operating System

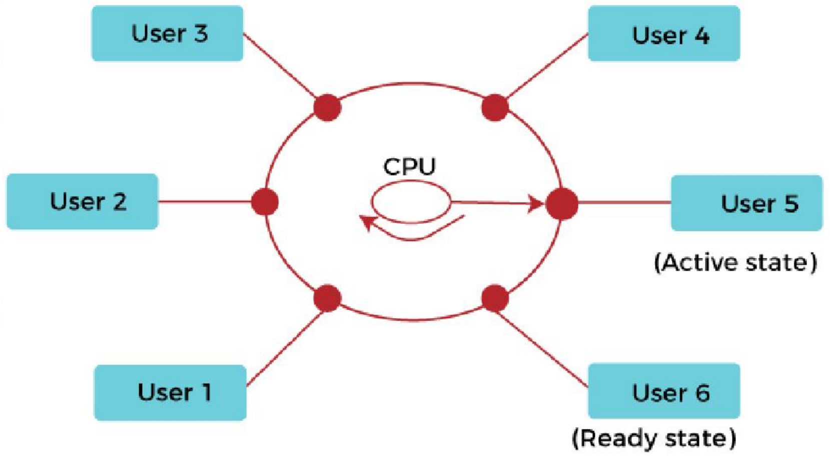
- With the help of **Multi programming** we can execute multiple programs on the System at a time.
- They never use any cards because the Process is entered on the Spot by the user.
- There must be proper management of all the Running Jobs.



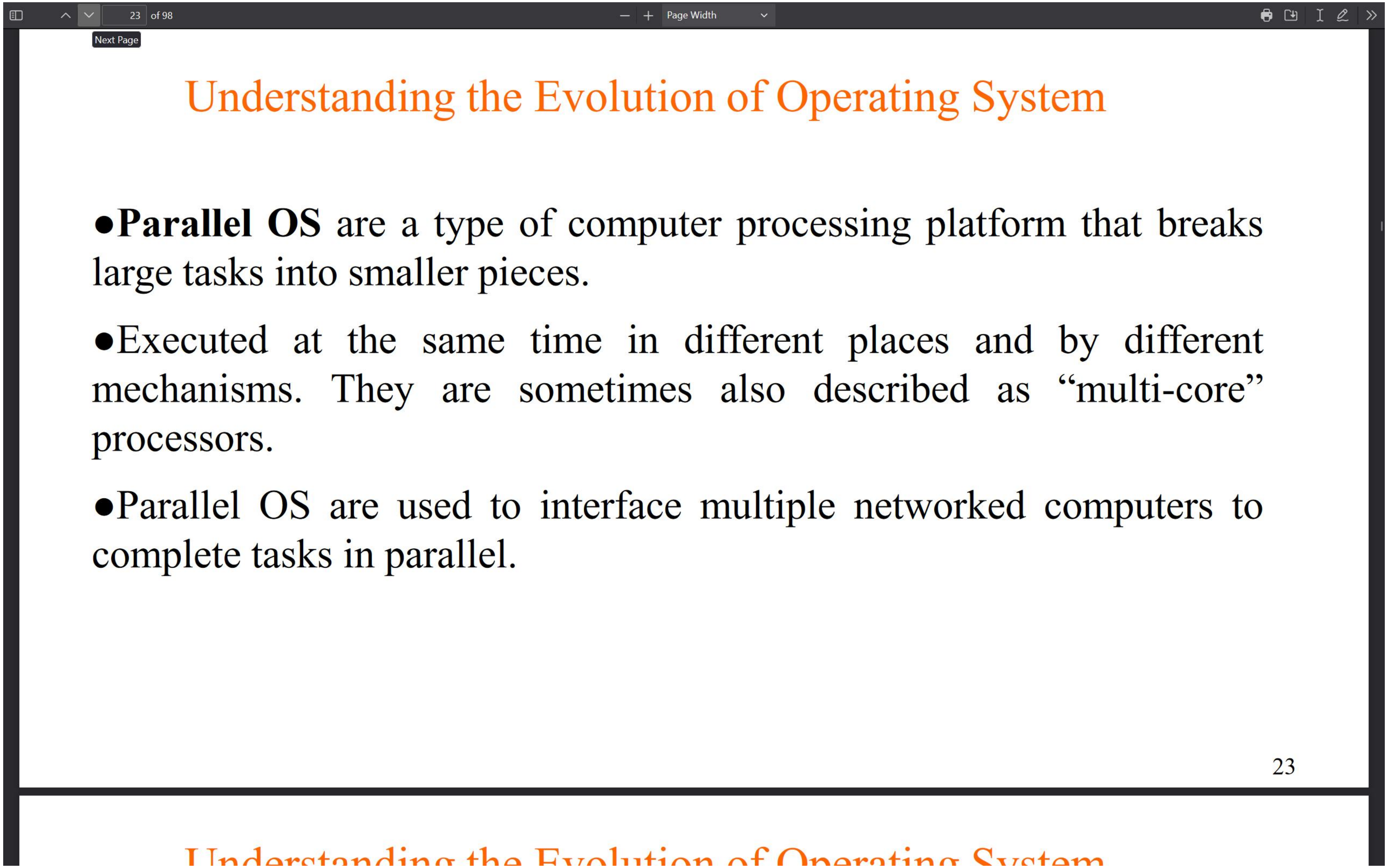
Understanding the Evolution of Operating System

Understanding the Evolution of Operating System

- **Time-sharing OS** enables many peo located at various terminals, to use particular computer system at the same tim
- Multiple jobs are implemented by the C by switching between them, but the switc occur so frequently.
- So, the user can receive an immediate response.



Understanding the Evolution of Operating System



Understanding the Evolution of Operating System

●**Parallel OS** are a type of computer processing platform that breaks large tasks into smaller pieces.

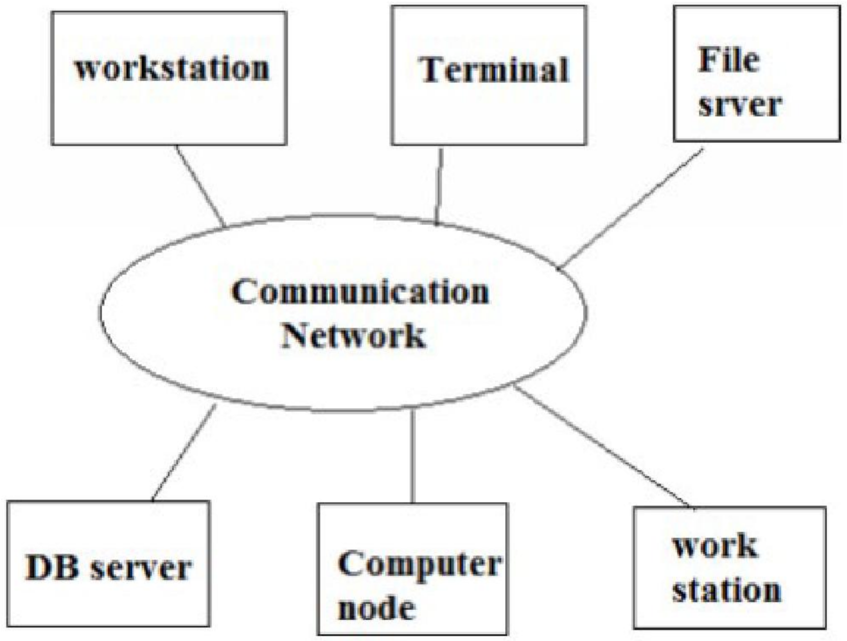
●Executed at the same time in different places and by different mechanisms. They are sometimes also described as “multi-core” processors.

●Parallel OS are used to interface multiple networked computers to complete tasks in parallel.

Understanding the Evolution of Operating System

Understanding the Evolution of Operating System

- A **Distributed OS** is system software over a collection of independent, networked, communicating, and physically separate computational nodes.
- They handle jobs which are serviced by multiple CPUs.
- Each individual node holds a specific software subset of the global aggregate operating system.

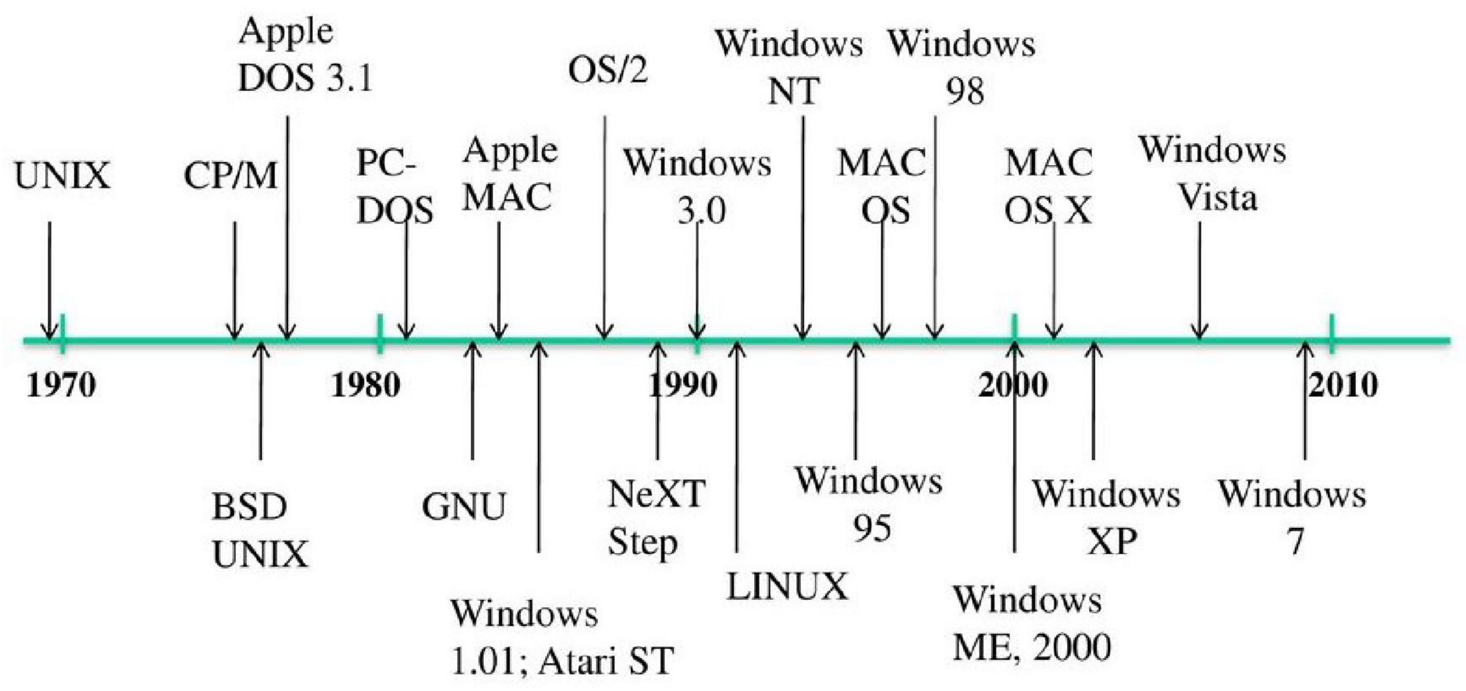


Understanding the Evolution of Operating System

25 of 98

Page Width

Understanding the Evolution of Operating System



Major Achievements of OS

Operating System - Process Scheduling

- The process scheduling is the activity of the process manager
- that handles the removal of the running process from the CPU
- and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Categories of Scheduling

There are two categories of scheduling:

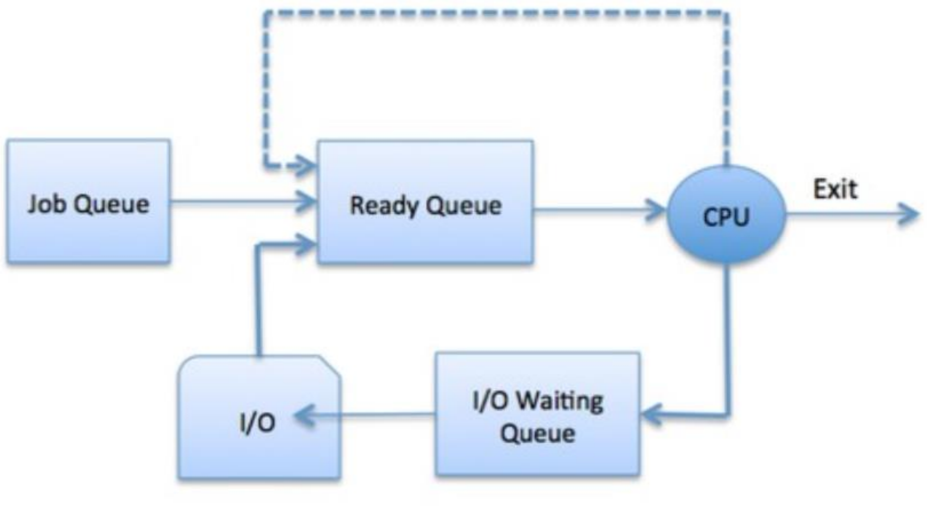
1. **Non-preemptive: Here the resource can't be taken from a process until the process completes execution**. The switching of resources occurs when the running process terminates and moves to a waiting state.
2. **Preemptive: Here the OS allocates the resources to a process for a fixed amount of time.** During resource allocation, the process switches from running state to ready state or from waiting state to ready state. This switching occurs as the CPU may give priority to other processes and replace the process with higher priority with the running process.

Process Scheduling Queues

The OS maintains all Process Control Blocks (PCBs) in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues −

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

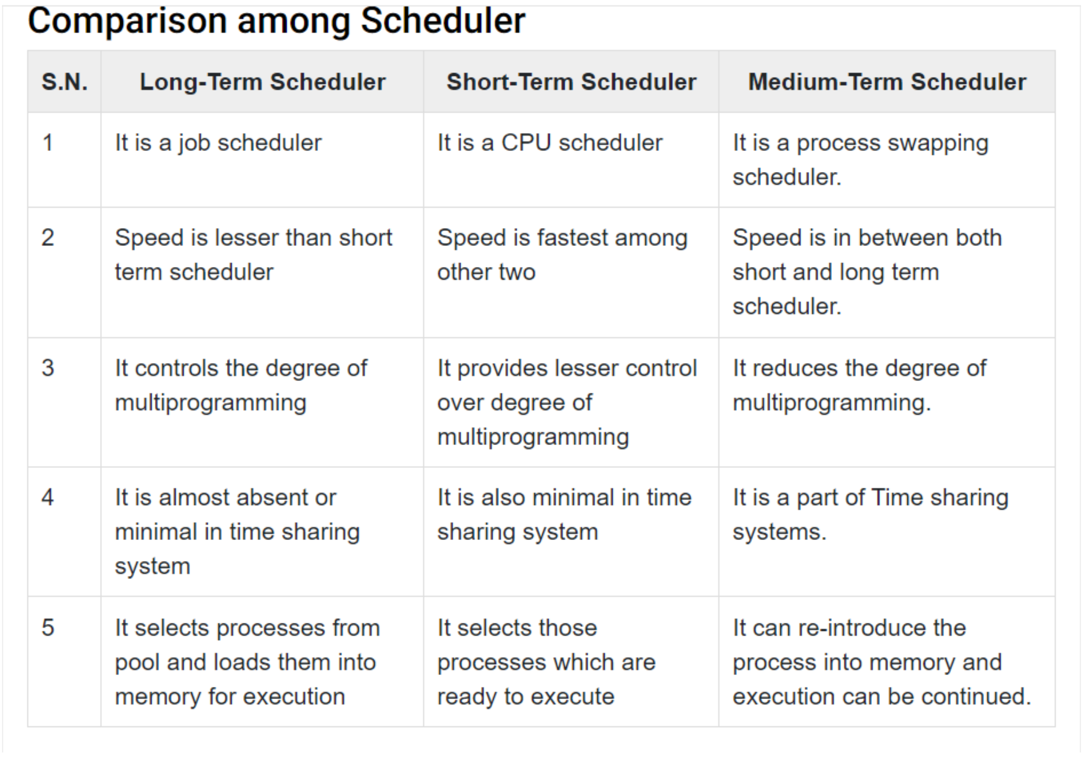
Two-state process model refers to running and non-running states which are described below −

S.N.	State & Description
1	**Running** When a new process is created, it enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler



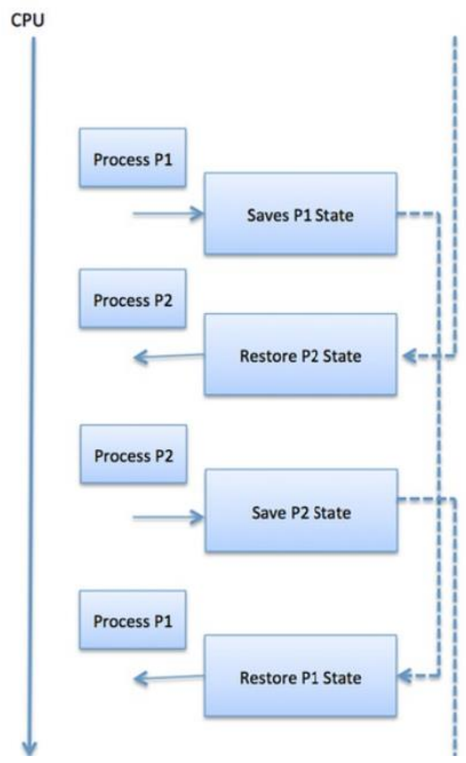
Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switching

A context switching is the mechanism to store and restore the state or context of a **CPU in Process Control block** so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.

When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.



Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

- Program Counter
- Scheduling information
- Base and limit register value
- Currently used register
- Changed State
- I/O State information
- Accounting information