OS Lab Experiment I

Varun Khadayate

NMIMS

|  |  |  |
| --- | --- | --- |
| SR No. | Topic | Page No. |
|  | Introduction to Operating System | 3 |
|  | Various types of Operating systems | 6 |
|  | Difference between various OS on the basis of Kernels | 13 |
|  | CPU Scheduling Algorithm | 14 |
|  | Virtual memory Management | 22 |
|  | Disk Management policies | 26 |
|  | File management Policy | 26 |

# Introduction to Operating System

An operating system acts as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

An operating system is a software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

**Definition:**

* An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
* A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.
* An operating system is concerned with the allocation of resources and services, such as memory, processors, devices, and information. The operating system correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

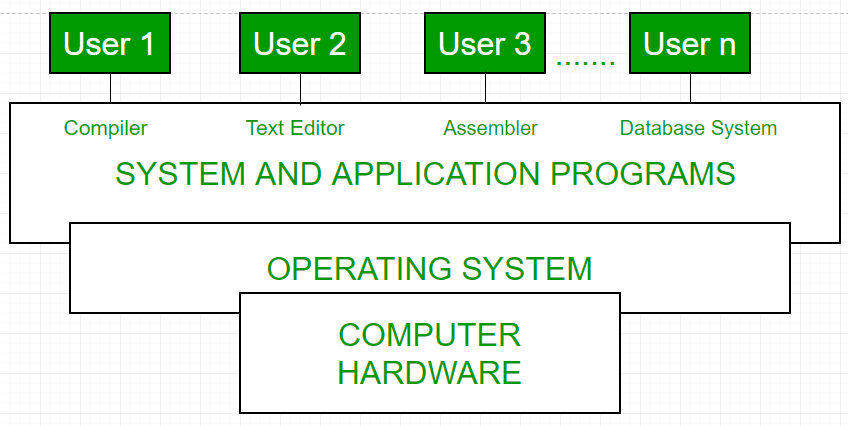
Functions of an OS – The OS performs three functions:

1. **Convenience:** An OS makes a computer more convenient to use.
2. **Efficiency:** An OS allows the computer system resources to be used in an efficient manner.
3. **Ability to Evolve:** An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions at the same time without interfering with service.

OS an UI –

1. User
2. System and application programs
3. Operating system
4. Hardware

Every general-purpose computer consists of the hardware, operating system, system programs, and application programs. The hardware consists of memory, CPU, ALU, and I/O devices, peripheral device, and storage device. System program consists of compilers, loaders, editors, OS, etc. The application program consists of business programs, database programs.



Every computer must have an operating system to run other programs. The operating system coordinates the use of the hardware among the various system programs and application programs for various users. It simply provides an environment within which other programs can do useful work.

The operating system is a set of special programs that run on a computer system that allows it to work properly. It performs basic tasks such as recognizing input from the keyboard, keeping track of files and directories on the disk, sending output to the display screen and controlling peripheral devices.  
OS is designed to serve two basic purposes:

1. It controls the allocation and use of the computing System’s resources among the various user and tasks.
2. It provides an interface between the computer hardware and the programmer that simplifies and makes feasible for coding, creation, debugging of application programs.

The Operating system must support the following tasks. The task is:

1. Provides the facilities to create, modification of programs and data files using an editor.
2. Access to the compiler for translating the user program from high level language to machine language.
3. Provide a loader program to move the compiled program code to the computer’s memory for execution.
4. Provide routines that handle the details of I/O programming.

**History of OS –**Operating system has been evolving through the years. Following Table shows the history of OS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Generation** | **Year** | **Electronic device used** | **Types of OS Device** |
| First | 1945-55 | Vacuum Tubes | Plug Boards |
| Second | 1955-65 | Transistors | Batch Systems |
| Third | 1965-80 | Integrated Circuits (IC) | Multiprogramming |
| Fourth | Since 1980 | Large Scale Integration | PC |

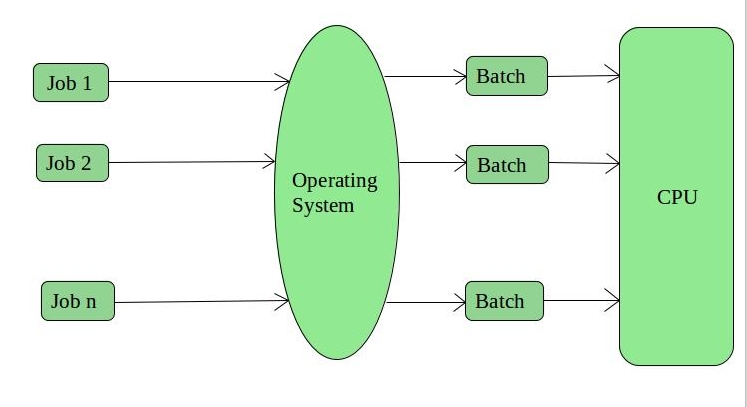
# Various types of Operating systems

An Operating System performs all the basic tasks like managing files, processes, and memory. Thus operating system acts as the manager of all the resources, i.e., **resource manager**. Thus, the operating system becomes an interface between user and machine.

**Types of Operating Systems**: Some widely used operating systems are as follows-

1. **Batch Operating System**

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirement and group them into batches. It is the responsibility of the operator to sort jobs with similar needs.



**Advantages**

* It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in queue
* Multiple users can share the batch systems
* The idle time for the batch system is very less
* It is easy to manage large work repeatedly in batch systems

**Disadvantages**

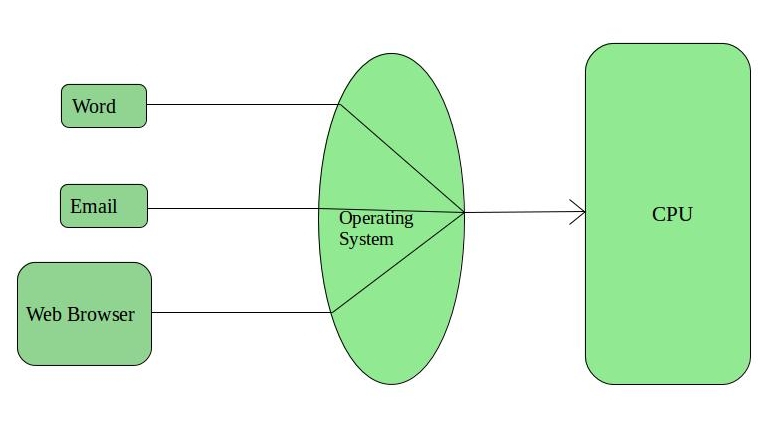
* The computer operators should be well known with batch systems
* Batch systems are hard to debug
* It is sometimes costly
* The other jobs will have to wait for an unknown time if any job fails

**Examples**

Payroll System, Bank Statements, etc.

1. **Time-Sharing Operating System**

Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.



**Advantages**

* Each task gets an equal opportunity
* Fewer chances of duplication of software
* CPU idle time can be reduced

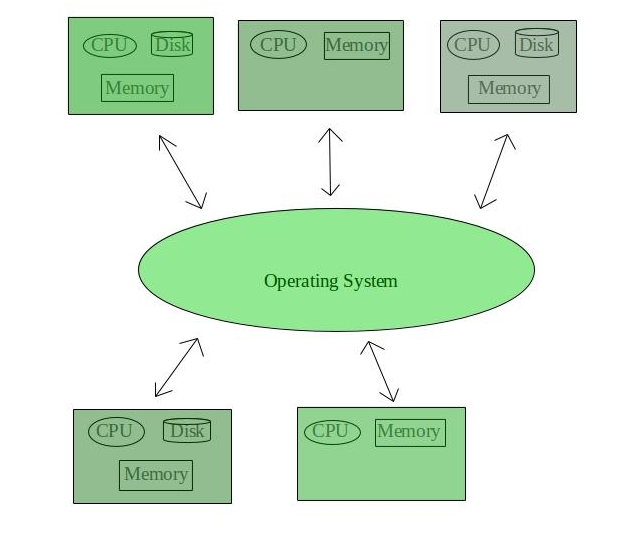
**Disadvantages**

* Reliability problem
* One must have to take care of the security and integrity of user programs and data
* Data communication problem

**Examples**

Multics, Unix, etc.

1. **Distributed Operating System**

These types of the operating system are a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, with a great pace. Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as **loosely coupled systems** or **distributed systems**. These system’s processors differ in size and function. The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.   
 

**Advantages**

* Failure of one will not affect the other network communication, as all systems are independent from each other
* Electronic mail increases the data exchange speed
* Since resources are being shared, computation is highly fast and durable
* Load on host computer reduces
* These systems are easily scalable as many systems can be easily added to the network
* Delay in data processing reduces

**Disadvantages**

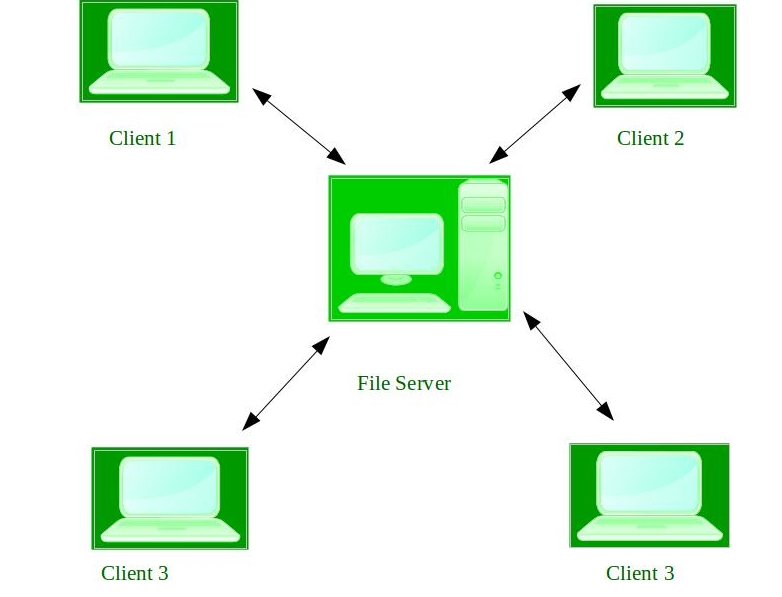
* Failure of the main network will stop the entire communication
* To establish distributed systems the language which is used are not well defined yet
* These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet

**Examples**

LOCUS, etc.

1. **Network Operating System**

These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that’s why these computers are popularly known as **tightly coupled systems**.



**Advantages**

* Highly stable centralized servers
* Security concerns are handled through servers
* New technologies and hardware up-gradation are easily integrated into the system
* Server access is possible remotely from different locations and types of systems

**Disadvantages**

* Servers are costly
* User has to depend on a central location for most operations
* Maintenance and updates are required regularly

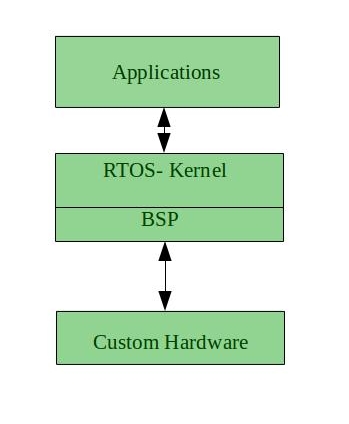
**Examples**

Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

1. **Real-Time Operating System**

These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.

**Real-time systems** are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.



**Two types of Real-Time Operating System which are as follows:**

* **Hard Real-Time Systems:**   
  These OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of any accident. Virtual memory is rarely found in these systems.
* **Soft Real-Time Systems:**   
  These OSs are for applications where for time-constraint is less strict.

**Advantages**

* **Maximum Consumption:** Maximum utilization of devices and system, thus more output from all the resources
* **Task Shifting:** The time assigned for shifting tasks in these systems are very less. For example, in older systems, it takes about 10 microseconds in shifting one task to another, and in the latest systems, it takes 3 microseconds.
* **Focus on Application:** Focus on running applications and less importance to applications which are in the queue.
* **Real-time operating system in the embedded system:** Since the size of programs are small, RTOS can also be used in embedded systems like in transport and others.
* **Error Free:** These types of systems are error-free.
* **Memory Allocation:** Memory allocation is best managed in these types of systems.

**Disadvantages**

* **Limited Tasks:** Very few tasks run at the same time and their concentration is very less on few applications to avoid errors.
* **Use heavy system resources:** Sometimes the system resources are not so good and they are expensive as well.
* **Complex Algorithms:** The algorithms are very complex and difficult for the designer to write on.
* **Device driver and interrupt signals:** It needs specific device drivers and interrupts signals to respond earliest to interrupts.
* **Thread Priority:** It is not good to set thread priority as these systems are very less prone to switching tasks.

**Examples**

Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

1. **Mobile Operating System**

Mobile operating systems are those OS which is especially that are designed to power smartphones, tablets, and wearables devices.

Some most famous mobile operating systems are Android and iOS, but others include BlackBerry, Web, and watchOS.



# Difference between various OS on the basis of Kernels

| **Sr. No.** | **Key** | **Operating System** | **Kernel** |
| --- | --- | --- | --- |
| 1 | Type | Operating system is a system software. | Kernel is a part of operating system. |
| 2 | Work | Operating system acts as an interface between user and hardware. | Kernel acts as an interface between applications and hardware. |
| 3 | Main tasks | Ease of doing system operations, security etc. | Memory management, space management, process management and task management. |
| 4 | Basis | A computer need Operating System to run. | An Operating System needs Kernel to run. |
| 5 | Types | Operating Systems types are multiuser, multitasking, multiprocessor, realtime, distributed etc. | Kernel types are monolithic kernel and micro kernel. |
| 6 | Boot | Operating System is the first program to load when computer boots up. | Kernel is the first program to load when operating system loads. |
| 7 | Purpose | Kernel memory management, process management, task management, disk management. | In addition to the responsibilities of Kernel, Operating System is responsible for protection and security of the computer. |

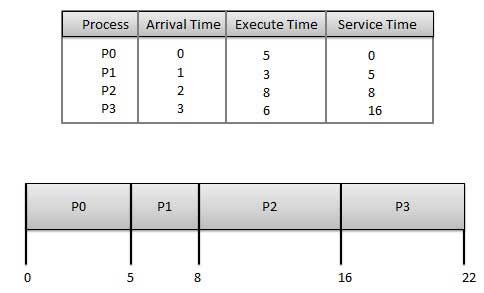
# CPU Scheduling Algorithm

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms −

These algorithms are either **non-preemptive** or **preemptive**. **Non-preemptive** algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the **preemptive** scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

1. **First Come First Serve (FCFS)**

* Jobs are executed on first come, first serve basis.
* It is a non-preemptive, pre-emptive scheduling algorithm.
* Easy to understand and implement.
* Its implementation is based on FIFO queue.
* Poor in performance as average wait time is high.



Wait time of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Wait Time: Service Time - Arrival Time** |
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 8 - 2 = 6 |
| P3 | 16 - 3 = 13 |

Average Wait Time: (0+4+6+13) / 4 = 5.75

1. **Shortest Job Next (SJN)**

* This is also known as shortest job first, or SJF
* This is a non-preemptive, pre-emptive scheduling algorithm.
* Best approach to minimize waiting time.
* Easy to implement in Batch systems where required CPU time is known in advance.
* Impossible to implement in interactive systems where required CPU time is not known.
* The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Execution Time** | **Service Time** |
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 14 |
| P3 | 3 | 6 | 8 |

Waiting time of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Waiting Time** |
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 14 - 2 = 12 |
| P3 | 8 - 3 = 5 |

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

1. **Priority Based Scheduling**

* Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
* Each process is assigned a priority. Process with highest priority is to be executed first and so on.
* Processes with same priority are executed on first come first served basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Execution Time** | **Priority** | **Service Time** |
| P0 | 0 | 5 | 1 | 0 |
| P1 | 1 | 3 | 2 | 11 |
| P2 | 2 | 8 | 1 | 14 |
| P3 | 3 | 6 | 3 | 5 |

Waiting time of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Waiting Time** |
| P0 | 0 - 0 = 0 |
| P1 | 11 - 1 = 10 |
| P2 | 14 - 2 = 12 |
| P3 | 5 - 3 = 2 |

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

1. **Shortest Remaining Time**

Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.

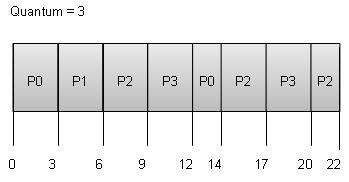
The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.

Impossible to implement in interactive systems where required CPU time is not known.

It is often used in batch environments where short jobs need to give preference.

1. **Round Robin Scheduling**

* Round Robin is the preemptive process scheduling algorithm.
* Each process is provided a fix time to execute, it is called a quantum.
* Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
* Context switching is used to save states of preempted processes.



Wait time of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time - Arrival Time** |
| P0 | (0 - 0) + (12 - 3) = 9 |
| P1 | (3 - 1) = 2 |
| P2 | (6 - 2) + (14 - 9) + (20 - 17) = 12 |
| P3 | (9 - 3) + (17 - 12) = 11 |

Average Wait Time: (9+2+12+11) / 4 = 8.5

1. **Multiple-Level Queues Scheduling**

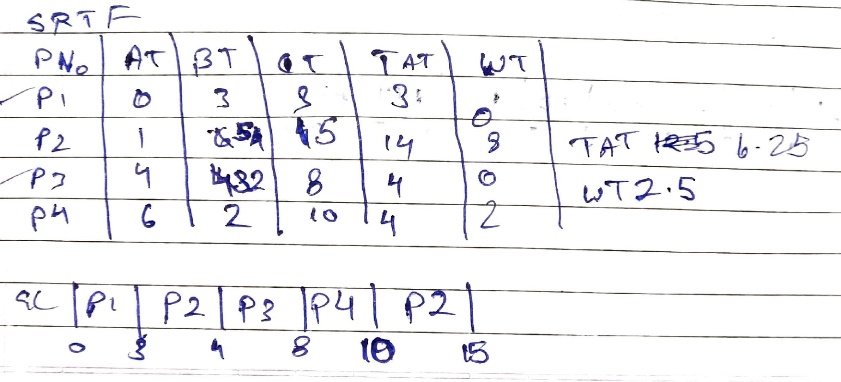
Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

* Multiple queues are maintained for processes with common characteristics.
* Each queue can have its own scheduling algorithms.
* Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

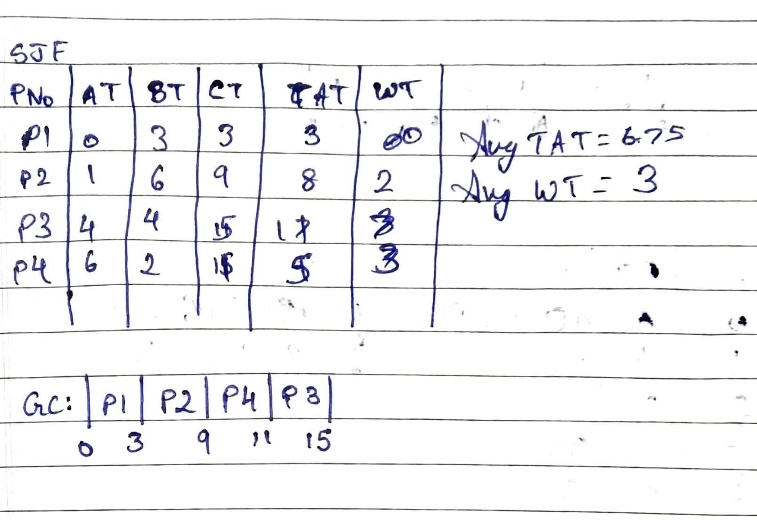
1. **Preemptive SJF**

|  |  |  |
| --- | --- | --- |
| **Process No.** | **Arrival Time** | **Burst Time** |
| A | 0 | 3 |
| B | 1 | 6 |
| C | 4 | 4 |
| D | 6 | 2 |

****

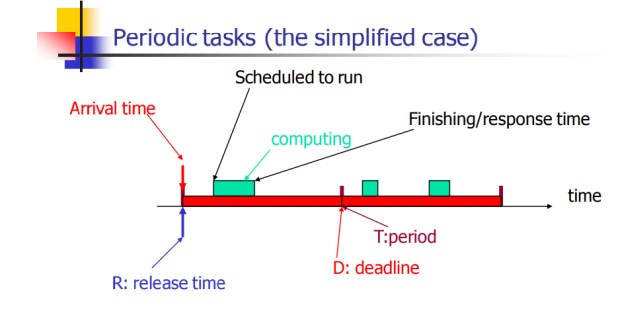
1. **Non-Preemptive SJF**

|  |  |  |
| --- | --- | --- |
| **Process No.** | **Arrival Time** | **Burst Time** |
| A | 0 | 3 |
| B | 1 | 6 |
| C | 4 | 4 |
| D | 6 | 2 |

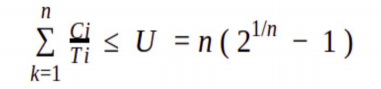


1. **Rate Monotonic Scheduling**

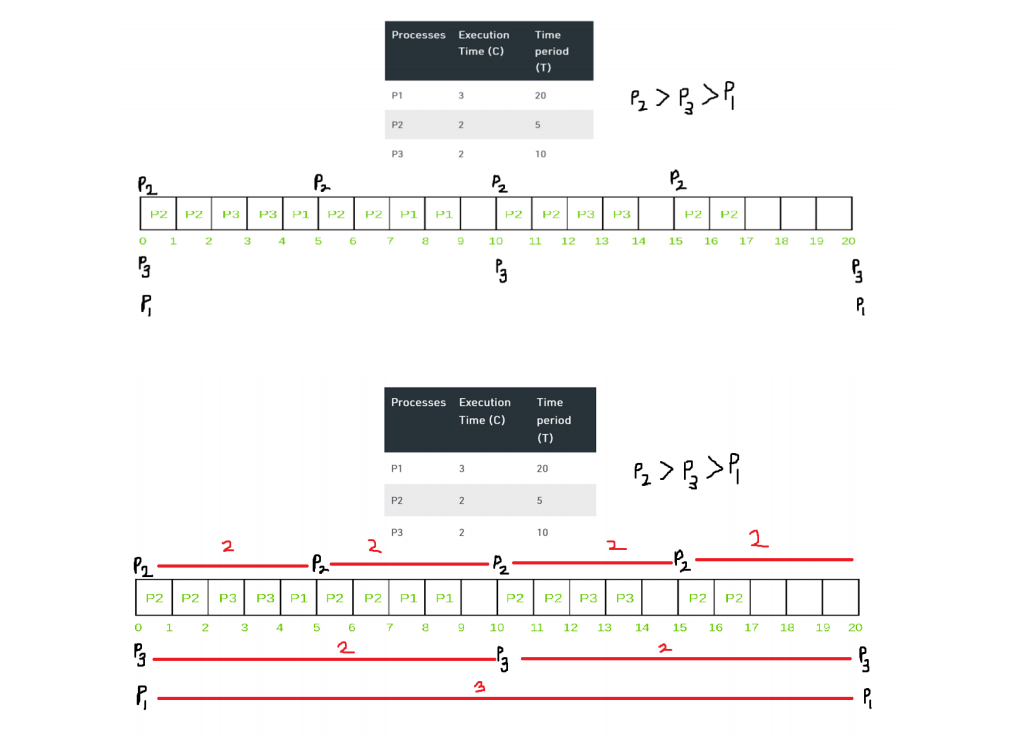
* Rate monotonic scheduling is a priority algorithm that belongs to the static priority scheduling category of Real Time Operating Systems.
* It is pre-emptive in nature.
* The priority is decided according to the cycle time of the processes that are involved.
* If the process has small job duration, then it has the highest priority.
* Thus, if a process with highest priority starts execution, it will pre-empt the other running processes. The priority of a process is inversely proportional to the period it will run for.



A set of processes can be scheduled only if they satisfy the following equation:

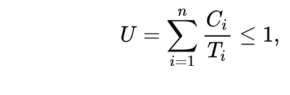


Where n is the number of processes in the process set, Ci is the computation time of the process, Ti is the Time period for the process to run and U is the processor utilization.

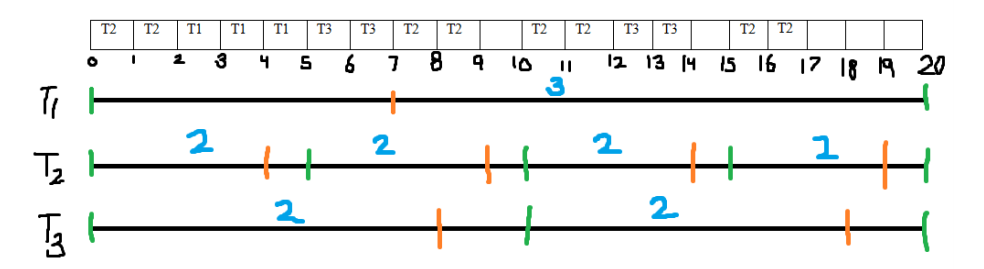


1. **Earliest Deadline First**

* Earliest deadline first (EDF) or least time to go is priority-based preemptive scheduling policy
* EDF is a dynamic priority scheduling algorithm used in real-time operating systems to place processes in a priority queue.
* Whenever a scheduling event occurs (task finishes, new task released, etc.) the queue will be searched for the process closest to its deadline.
* Job with earliest (absolute) deadline has highest priority
* Does not require knowledge of execution times
* Is known to be an optimal policy for a single processor (?)
* With scheduling periodic processes that have deadlines equal to their periods, EDF has a utilization bound of 100%. Thus, the schedulable test for EDF is:



|  |  |  |  |
| --- | --- | --- | --- |
| Task | Execution Time (C) | Deadline | Time Period (T) |
| T1 | 3 | 7 | 20 |
| T2 | 2 | 4 | 5 |
| T3 | 2 | 8 | 10 |



# Virtual memory Management

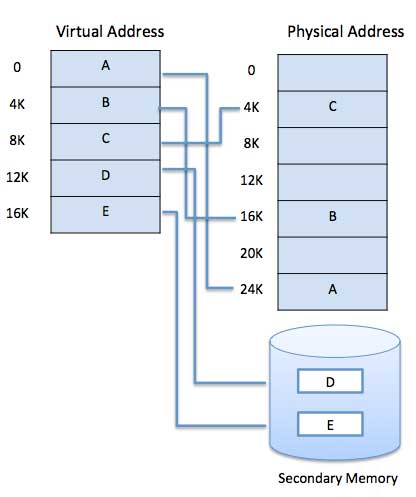
A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.

The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

* User written error handling routines are used only when an error occurred in the data or computation.
* Certain options and features of a program may be used rarely.
* Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
* The ability to execute a program that is only partially in memory would counter many benefits.
* Less number of I/O would be needed to load or swap each user program into memory.
* A program would no longer be constrained by the amount of physical memory that is available.
* Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below −



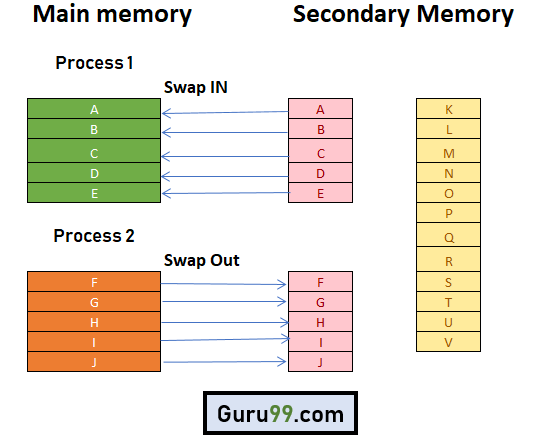
Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

## Demand Paging

A demand paging mechanism is very much similar to a paging system with swapping where processes stored in the secondary memory and pages are loaded only on demand, not in advance.

So, when a context switch occurs, the OS never copy any of the old program's pages from the disk or any of the new program's pages into the main memory. Instead, it will start executing the new program after loading the first page and fetches the program's pages, which are referenced.

During the program execution, if the program references a page that may not be available in the main memory because it was swapped, then the processor considers it as an invalid memory reference. That's because the page fault and transfers send control back from the program to the OS, which demands to store page back into the memory.



## Types of Page Replacement Methods

1. **FIFO Page Replacement**

FIFO (First-in-first-out) is a simple implementation method. In this method, memory selects the page for a replacement that has been in the virtual address of the memory for the longest time.

**Features:**

* Whenever a new page loaded, the page recently comes in the memory is removed. So, it is easy to decide which page requires to be removed as its identification number is always at the FIFO stack.
* The oldest page in the main memory is one that should be selected for replacement first.

1. Optimal Algorithm

The optimal page replacement method selects that page for a replacement for which the time to the next reference is the longest.

**Features:**

* Optimal algorithm results in the fewest number of page faults. This algorithm is difficult to implement.
* An optimal page-replacement algorithm method has the lowest page-fault rate of all algorithms. This algorithm exists and which should be called MIN or OPT.
* Replace the page which unlike to use for a longer period of time. It only uses the time when a page needs to be used.

1. LRU Page Replacement

The full form of LRU is the Least Recently Used page. This method helps OS to find page usage over a short period of time. This algorithm should be implemented by associating a counter with an even- page.

**Features:**

* The LRU replacement method has the highest count. This counter is also called aging registers, which specify their age and how much their associated pages should also be referenced.
* The page which hasn't been used for the longest time in the main memory is the one that should be selected for replacement.
* It also keeps a list and replaces pages by looking back into time.

## Advantages of Virtual Memory

* Virtual memory helps to gain speed when only a particular segment of the program is required for the execution of the program.
* It is very helpful in implementing a multiprogramming environment.
* It allows you to run more applications at once.
* It helps you to fit many large programs into smaller programs.
* Common data or code may be shared between memory.
* Process may become even larger than all of the physical memory.
* Data / code should be read from disk whenever required.
* The code can be placed anywhere in physical memory without requiring relocation.
* More processes should be maintained in the main memory, which increases the effective use of CPU.
* Each page is stored on a disk until it is required after that, it will be removed.
* It allows more applications to be run at the same time.
* There is no specific limit on the degree of multiprogramming.
* Large programs should be written, as virtual address space available is more compared to physical memory.

## Disadvantages of Virtual Memory

* Applications may run slower if the system is using virtual memory.
* Likely takes more time to switch between applications.
* Offers lesser hard drive space for your use.
* It reduces system stability.
* It allows larger applications to run in systems that don't offer enough physical RAM alone to run them.
* It doesn't offer the same performance as RAM.
* It negatively affects the overall performance of a system.
* Occupy the storage space, which may be used otherwise for long term data storage.

# Device Management Policies

A device management policy determines what a process may do to a device, and how the device behaves.

## Disk Management Policy

The disk management policy presents the operating system's view of the disk to the user process. A user process' view of the disk is the same as the operating system's view of the disk; that is, the disk appears to a user process as an C-style array of disk blocks.

This policy is known as **raw disk management**, and is important for those application, such as data-base applications, that need to control all aspects of data movement between disk and primary store.

# File Management Policies

A file is collection of specific information stored in the memory of computer system. **File management is defined as the process of manipulating files in computer system, it management includes the process of creating, modifying and deleting the files**.

The following are some of the **tasks performed by file management of operating system of any computer system**:

1. It helps to create new files in computer system and placing them at the specific locations.
2. It helps in easily and quickly locating these files in computer system.
3. It makes the process of sharing of the files among different users very easy and user friendly.
4. It helps to stores the files in separate folders known as directories. These directories help users to search file quickly or to manage the files according to their types or uses.
5. It helps the user to modify the data of files or to modify the name of the file in the directories.