Save from: www.uotiq.org/dep-cs

Operating Systems

4th Class

استاذ المادة: د. رنا فريد غني

Lecture 1

Operating Systems

Operating systems are essential part of any computer system. Therefore, a course in operating systems is an essential part of any computer science education. The fundamental concepts of operating systems will be presented in this course.

The syllabus of the operating system course is as follows:

- Operating system overview
- Main frame systems
- Desktop systems
- Multiprocessor systems
- Distributed systems
- Clustered systems
- Real time systems
- Handheld systems
- Computing environment
- Computer system structure
- Hardware protection
- operating system structure
- operating system components
- operating system services
- processes
- process concepts
- operation on processes
- cooperating process
- threads

- CPU scheduling
- Memory Management
- Storage management
- Protection and Security

Objective of the course

- To provide a general explanation of the component of operating systems
- To provide the general organization of the computer systems and the relation between the computer structure and operating systems.

Operating System Concepts – 6th Edition, Silberschatz, Galvin and Gagne 2003

Chapter 1 – Introduction

What are Operating Systems?

A program that manages the computer hardware. Therefore, it acts as an intermediary between a user of a computer and the computer hardware.

Why we need an Operating system?

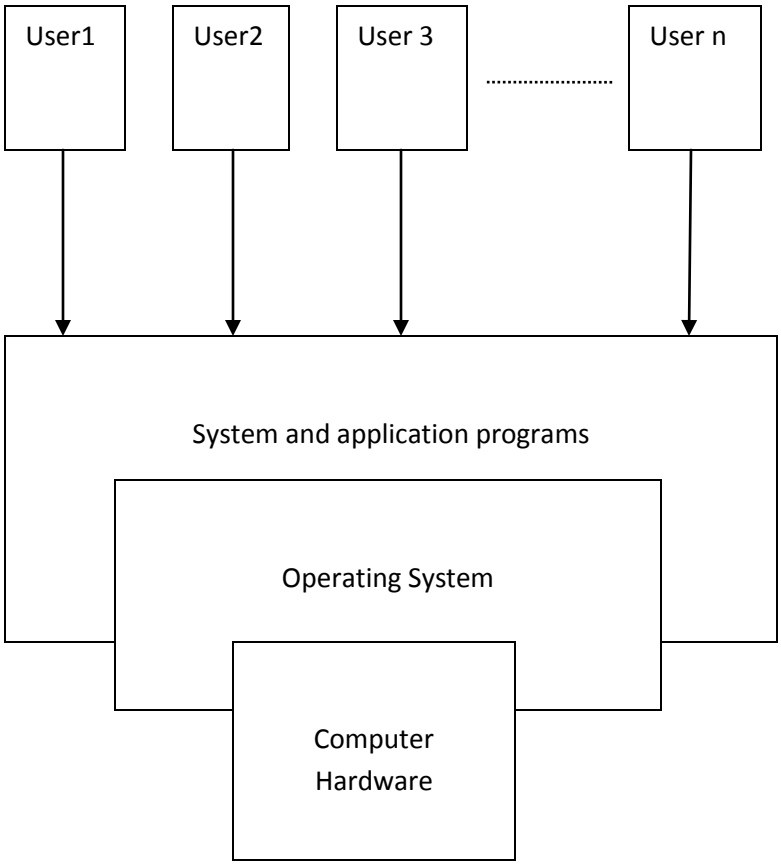
Generally an operating system is needed for the following reasons:

- Execute user programs and make solving user problems easier.
- Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.

Computer systems

Computer systems can be divided into four components

- Hardware –provides basic computing resources CPU, memory, I/O devices
- Operating system-Controls and coordinates use of hardware among various applications and users
- Application programs –Define the ways in which the system resources are used to solve the computing problems of the users
  Word processors, compilers, web browsers, database systems, video games
- Users
  People, machines, other computers



Computer Structure

User View

The user view of computer varies by the interface being used. The operating systems are designed mostly for ease of use. Others are designed to maximize resource utilization. Other operating systems are designed to compromise between individual usability and resource utilization.

System view

From the computer's point of view, the OS is a

- **resource allocator**
  Manages all resources and decides between conflicting requests for efficient and fair resource use
- **control program**
  Controls execution of programs to prevent errors and improper use of the computer

Lecture 2

Operating System Historical Review

Operating systems and computer architecture have influenced each other. To facilitate the use of the hardware, researchers developed operating systems. In the following historical review, we will notice the mutual effect between operating systems and computer hardware which led to developments in both sides.

Mainframe systems

Mainframe systems grow on three stages:

- Batch systems

In this type of computer systems, the operator batch together jobs with similar needs and ran through the computer as group.

The operating system was simple and its major task was to transfer control automatically from one job to the next.

- Multiprogrammed systems

The operating system keeps several jobs in memory simultaneously.

Operating systems for the Multiprogrammed is the first one which make a decision for the users. Making this decision is called job scheduling.

- Time_shared systems

The CPU executes multiple jobs by switching among them, but the switches occurred so frequently the users can interact with each program while it is running.

A Time_shared operating systems allows many user programs (processes) to share the computer simultaneously. The CPU executes multiple jobs by switching among them, but the switches occurred so frequently the users can interact with each program while it is running.

Desktop systems

The operating systems of desktop systems were neither multi-user nor multitasking. Operating systems have changed with time; instead of maximizing CPU and peripheral utilization, the systems improved to maximize user convenience and responsiveness.

Multiprocessor Systems (Parallel systems or tightly coupled systems)

Such systems have more than one processor in close communication sharing the computer bus, the clock, and sometimes memory and peripheral devices.

Multiprocessor systems have three main advantages

1- Increase throughput.

2- Economy of scale.

3- Increased reliability.

This ability to continue providing service proportional to the level of surviving hardware is called “graceful degradation” is also called “fault tolerant”.

There are different architectures for multiprocessor systems.

Distributed Systems

A network is a communication path between two or more systems. Distributed systems depend on networking for their functionality. Using communicates, distributed systems are able to share computational tasks, and provide a rich set of set of feature to users.

- client-server systems
- peer-to-peer systems

Some operating system benefits from ideas of networking and distributed systems in build network operating system.

Clustered Systems

Like parallel systems, clustered systems gather together multiple CPUs to accomplish computational work, they composed of two or more individual systems coupled together. The general accepted definition is that clustered computers share storage and is closely linked via LAN networking. Clustering is usually performed to provide high availability.

Real-Time Systems

Special purpose operating system, it is used when there are rigid time requirements on the operation of a processor or the flow of data, thus it is often used as a control device in dedicated application.

Real time system need that the processing must be done within the defined time constraints or the system will fail.

There are two flavours of real time system:

- Hard real-time system
- Soft real time system

Handheld Systems

Handheld systems include personal digital assistants (PDAs). Developers of handheld systems and applications face many challenges (due to the limited size of such devices) such as speed of processor, limited size of memory, and small display screen.

Computing Environments

All above systems are used in verity of computing environments settings.

- Traditional Computing.
- Wed-Based computing.
- Embedded Computing.

Lecture 3

Computer System Structures

Computer System Operation:

A modern, general-purpose computer system consists of CPU and a number of device controllers that connected through a common bus that provides access to shared memory system, CPU other devices can execute concurrently competing for memory cycles.

Booting:

It is the operation of bringing operating system kernel from the secondary storage and put it in main storage to execute it in CPU. There is a program bootstrap which is performing this operation when computer is powered up or rebooted.

Bootstrap software: it is an initial program and simple it is stored in read-only memory (ROM) such as firmware or EEPROM within the computer hardware.

Jobs of Bootstrap program:

1- Initialize all the aspect of the system, from CPU registers to device controllers to memory contents.
2- Locate and load the operating system kernel into memory then the operating system starts executing the first process, such as “init” and waits for some event to occur.

The operating system then waits for some event to occur

Types of events are either software events (system call) or hardware events (signals from the hardware devices to the CPU through the system bus and known as an interrupt).

Note: all modern operating system are “interrupt driven”.

Trap (exception): it is a software-generated interrupt caused either by an error (ex: division by zero or invalid memory access) or by a specific request from a user program that an operating system service be performed.

Interrupt vector (IV): it is a fixed locations (an array) in the low memory area (first 100 locations of RAM) of operating system when the interrupt occur the CPU stops what its doing and transfer execution to a fixed location (IV) contain starting address of the interrupt service routine(ISR), on completion the CPU resumes the interrupted computation.

Interrupt Service Routine: is it a routine provided to be responsible for dealing with the interrupt.

Hardware protection:

when we have single user any error occur to the system then we could determined that this error must be caused by the user program ,but when we begin to dealing with spooling ,multiprogramming, and sharing disk to hold many users data this sharing both improved utilization and increase problems .

In multiprogramming system, where one erroneous program might modify the program or data of another program, or even the resident monitor itself. MS-DOS and the Macintosh OS both allow this kind of error.

A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other program to execute incorrectly.

Many programming error are detected by the hardware these error are normally handled by the operating system.

Dual-Mode Operation:

To ensure proper operation, we must protect the operating system and all other programs and their data from any malfunctioning program.

The approach taken by many operating systems provides hardware support that allows us to differentiate among various modes of execution.

A bit, called the ***mode bit*** is added to the hardware of the computer to indicates the current mode: monitor (0) or user (1) with mode bit we could distinguish between a task that is executed on behalf of the operating system , and one that is executed on behalf of the user.

I/O Operation Protection:

A use program may disrupt the normal operation of the system by issuing illegal I/O instruction we can use various mechanisms to ensure that such disruption can not take place in the system.

One of them is by defining all I/O instructions to be privileged instructions. Thus users cannot issue I/O instructions directly they must do it through the operating system, by execute a system call to request that the operating system performing I/O in its behalf. The operating system, executing in monitor mode, check that the request is valid, and (if the request is valid) does the I/O requested. The operating system then returns to the user.

Memory Protection:

To insure correct operation, we must protect the interrupt vector and interrupt service routine from modification by a user program. This protection must be provided by the hardware, we need the ability to determine the range of legal addresses that the program may access, and to protect the memory outside that space. We could provided the protection by using two registers a base register and limit register

Base register hold the smallest legal physical memory address.

Limit register: contains the size of the range.

This protection is accomplished by the CPU hardware comparing every address generated in user mode with the registers. Any attempt by a program executing in user mode to access monitor memory or other users' memory results in a trap to the monitor, which treats the attempts as a fatal error.

CPU Protection:

In addition to protecting I/O and memory we must insure that the operating system maintains control. We must prevent the user from getting stuck in an infinite loop or not calling system services, and never returning control to the operating system. To accomplish this goal, we can use ***a timer***.

Timer can be set to interrupt the computer after a specified period. The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second) A variable timer is generally implemented by a fixed rate clock and a counter.

We can use the timer to prevent a user program from running too long Simple technique is to initialize a counter with the mount of time that a program is allowed to run.

Amore common use of timer is to implement time sharing. In the most case, the timer could be set to interrupt every N millisecond, where N is the time slice that each user is allowed to execute before the next user get control of the CPU. The operating system is invoked to perform housekeeping tasks.

This procedure is known as a context switching, following a context switch, the next program continues with its execution from the point at which it left off.

Lecture 4

Operating System Structure

In the following lectures we will consider the components and services that are provided by different operating systems.

System Components

Many modern computer systems share the goal of supporting the following components:

- **Process management**

A process can be thought of a program in execution. A process needs certain resources to accomplish its task. Also the process various initialization values.

A process is the unit of work in a system. Such a system consists of a collection of processes, some of which are system processes others are user processes. All processes execute concurrently by multiplexing the CPU among them.

The OS responsible for the following activities in connection with process management:

- Creation and deletion both user and system processes.
- Suspending and resuming processes.
- Providing mechanisms for process synchronization.

- Providing mechanisms for process communication.
- Providing mechanisms for deadlock handling.

- **Main Memory Management**

The main memory is the central to the operation of a modern computer system. For a program to be executed it must mapped to absolute addresses and loaded to the MM.

The OS responsible for the following activities in connection with MM management:

- Keeping track of which parts of memory are currently being used and by whom.
- Deciding which processes are to be loaded into memory when memory space become available.
- Allocating and deallocating memory space as needed.

- **File Management**

For convenient use of the computer, the OS provides a uniform logical view of information storage. The OS abstracts from the physical properties of its storage device to define the logical storage unit, the file.

A file is acollection of related information defined by its creator. These files are organized in directories to ease their use.

The OS responsible for the following activities in connection with file management:

- Creating and deleting files.
- Creating and deleting directories.
- Supporting primitives for manipulating files and directories.
- Mapping files onto secondary storage.
- Backing up files on stable storage media.

- **I/O System Management**

One of the purposes of OS is to hide the peculiarities of specific hardware devices. The OS responsible for the following activities in connection with I/O system management:

- A memory management component that includes buffering, caching and spooling.
- A general device driver interface.
- Derivers for specific hardware devices.

- **Secondary Storage Management**

The computer system must provide secondary storage to back up main memory because that are hold by MM are lost when power is switched of f and the MM is too small to accommodate all data programs. The OS responsible for the following activities in connection with disk management:

- Free space management
- Storage allocation
- Disk scheduling

- **Networking**

A distributed system collects physically separate heterogeneous system into a single coherent system, providing the user with the access to various resources that the system maintain. Access to a shared resource allows computation speed up, increase functionality, increase data arability, and enhance reliability.

- **Protection System**

Protection is any mechanism for controlling the access programs, processes, or users to the resources defined by the computer system. This mechanism must provide means for specification of the controls to be imposed and means for enforcement. Protection can improve reliability by detecting latent errors at the interfaces between component subsystems.

- **Command Interpreter System**

Command Interpreter System is the interface between the user and the OS. Some of these Command Interpreter System are user friendly such

as mouse based window and menus. In other shells commands are typed on a keyboard.

Operating System Services

An operating system provides an environment for the execution of programs. It provides certain services to programs and to the users of these programs. The specific services provided differ from one operating system to another but we can identify common classes. These operating system services are provided for the convenience of the programmer, to make the programming task easier.

1. Program execution
2. I/O operation
3. File system manipulation
4. Communications
5. Error detection
6. Resource allocation
7. Accounting
8. Protection

System Calls

System calls provide the interface between a process and the operating system. These calls are generally available as assembly language

instructions and they are usually listed in the various manuals used by assembly language.

System Programs

System programs provide a convenient environment for program development and execution. Some of them are simply user interfaces to system calls others are considerably more complex. They can be divided into these categories:

- File management
- Status information
- File modification
- Programming language support
- Program loading and execution
- Communications

System Structure

A system as large and complex as a modern operating system must be engineered carefully if it is to function properly and to be modified easily. There are three different system structures:

- Simple structure
- Layered Approach
- Microkernal

System Design and Implementation

The problems and steps of system design and implementation are as follows:

- Design Goals
- Mechanisms and Policies
- Implementation

Processes

In the following lectures we will consider the concepts of process.

Process Concepts

A process is a program in execution. A process is more than the program code, which is sometimes known as the text section. It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers.

Process state

The state of a process is defined in part by the current activity of the process. Each process may be in one of the following states:

- New
- Running
- Waiting
- Ready
- Terminated

Process Control Block

Each process is represented by a process control block (PCB). A PCB contains many pieces of information associated with a specific process, such as:

- Process states
- Program counter

- CPU registers
- CPU scheduling information
- Memory management information
- Accounting information
- I/O status information

Process Scheduling

A uniprocessor system can have only one running process. If more processes exist, as in multiprogramming system, there will be only one process running and the rest must wait until the CPU is free and can be rescheduled.

- **Scheduling Queues**

A new process as enter the system is put in a queue called ready queue. It waits in the ready queue until it is selected for execution. Once the process is assigned to the CPU and it is executing, one of the several event could occur:

The process could issue an I/O request, and then be placed in an I/O queue.

The process could create a new subprocess and wait for the termination.

The process could be removed forcibly from the CPU, as a result of an interrupt and be put back in the ready queue.

- **Scheduler**

A process migrates between the various scheduling queues throughout its lifetime. The operating system must select processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler. There are two types of scheduling algorithms categorized according to the frequency of their execution.

- Long term scheduler (job scheduler) which selects a process from the job pool and load them into the MM.
- Short term scheduler (CPU scheduler) which select a process from the ready queue and allocate it to the CPU.

- **Context** Switch

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the process. This task is known as a context switch.

Operation on Processes

The process in the system can execute concurrently, and they must be created and deleted dynamically.

- **Process Creation**

A process may create several new processes during the course of execution. The creating process is called a parent process, whereas the new processes are called the children.

When a process is created it obtains various resources and initialization values that may be passed along from the parent process to the child process.

- **Process Termination**

A process terminates when it finishes executing its final statement and asks the operating system to delete it. At that point the process may return data to its parent process and the OS deallocate all the physical and logical resources that are previously allocated to that process.

Cooperating Processes

The concurrent process executing in the operating system may be either independent processes that does not share any data or cooperating that affects each others.

We may provide an environment that allows process cooperation for several reasons:

- Information sharing
- Computation speedup
- Modularity
- Convenience

Interprocess Communication

The cooperating processes can communicate in a shared memory environment. The scheme requires that these processes

share a common buffer pool. Another way to achieve the same effect for the operating system is provided via an interprocess communication (IPC).

IPC provides a mechanism to allow processes to communicate and synchronize their actions without sharing the same address space. This technique is useful for distributed systems. IPC is provided by a message passing system.

CPU Scheduling

In the following lectures we will introduce the basic scheduling concepts and present several different CPU scheduling algorithms.

Scheduling Concepts

Scheduling is a fundamental operating system function. Almost all computer resources are scheduled before use. The CPU scheduling is central to operating systems.

CPU-I/O Burst Cycle

The success of CPU scheduling depends on the following observed property of processes: process execution consists of a cycle of CPU execution and I/O wait. Processes alternate between these two states. Process execution begins with a CPU burst. That is followed by I/O burst, then another CPU burst and so on. The last CPU burst will end with a system request to terminate execution.

CPU Scheduler

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short term scheduler(CPU scheduler). The scheduler selects from among

the processes in in memory that are ready to execute and allocates the CPU to one of them.

Scheduling Schemes

There are two scheduling schemes can be recognized:

- Preemptive scheduling
- Nonpreemptive scheduling

Under the nonpreemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it release the CPU either by terminating or by switching to the waiting state. On the other hand Preemptive scheduling occure when the CPU has been allocted to a process and this process is interrupted by higher priority process. At this moment the executing process is stopped and returned back to the ready queue, the CPU is allocated to the higher priority process.

Dispatcher

It is the module that gives control of the CPU to the process selected by the CPU scheduler. This function involves:

- Switching Context
- Switching to user mode
- Jumping to the proper location in the user program to restart the program.

Scheduling Criteria

Many criteria have been suggested for comparing CPU scheduling algorithms. The criteria include the following:

- CPU Utilization
- Throughput
- Turnedaround Time
- Waiting time
- Response time

Scheduling Algorithm

Here we will mention soime of the CPU scheduling algorithms that are used in different operating systems

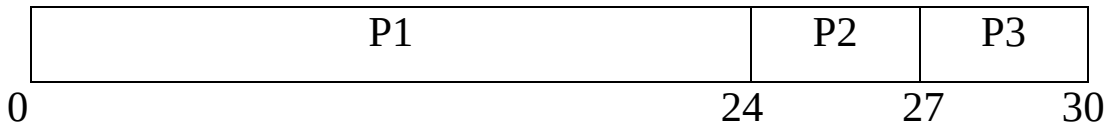
- **First Come First Served (FCFS)**

With this algorithm the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue.

The average waiting time under the FCFS policy is often quite long. Consider the following set of processes that arrive at time 0, with the length of CPU burst time given in millisecond:

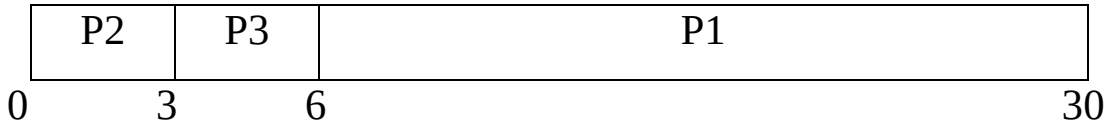
Process	Burst time
P1	24
P2	3
P3	3

The Gantt Chart is as follows:



The average waiting time = (0+24+27)/ 3=17 millisecond

If the processes arrive in the order P2, P3, P1 the result will be shown in the following Gantt Chart:



The average waiting time = (0+3+6)/ 3=3 millisecond

Thus the average waiting time under FCFS policy is not the minimal.

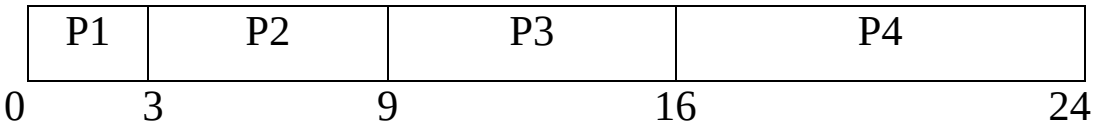
Shortest Job First Scheduling (SJF)

This algorithm associate with each process the length of the latter's next CPU burst. When the CPU is available, it is assigned to the process has the smallest next CPU burst. If two processes have the same length , FCFS scheduling is used to break this tie.

As an example consider the following set of processes with the length of the CPU burst given in millisecond:

Process	Burst time
P1	6
P2	8
P3	7
P4	3

The Gantt Chart is as follows:



The average waiting time = (0+3+8+16)/ 4=7 millisecond

The average waiting time in SJF is the optimal that it gives the minimum average waiting time.

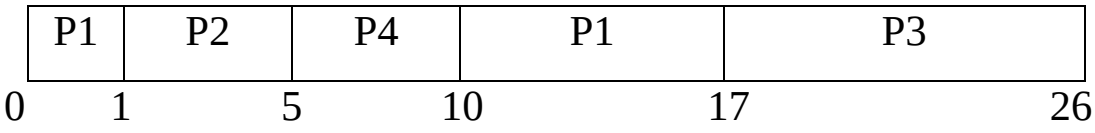
The SJF is either preemptive or nonpreemptive. The choice arises when a new process arrives at the ready queue while a

previous process is executing. The new process may have a shorter next CPU burst than what is left of the currently executing process. A preemptive SJF will preempt the currently executing process whereas a nonpreemptive SJF algorithm will allow the currently running process to finish its CPU burst.

As an example consider the following set of processes with the length of the CPU burst given in millisecond:

Process	Arrival Time	Burst time
P1	0	6
P2	1	8
P3	2	7
P4	3	3

The Gantt Chart is as follows:



AWT =((10-1)+(1-1)+(17-2)+(5-3))/ 4=6.5millisecond

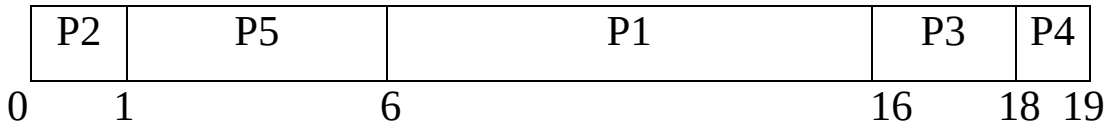
Priority Scheduling Algorithm

In this algorithm a priority is associated with each process and the CPU is allocated to the process of the highest priority. We use the low numbers to represent high priority.

As an example consider the following set of processes with the length of the CPU burst given in millisecond:

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

The Gantt Chart is as follows:



The average waiting time = 8.2 millisecond

Priority scheduling can be either Preemptive or nonpreemptive, when a process arrives the ready queue, its priority is compared with the priority of the currently running process. A preemptive priority will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process. A nonpreemptive priority scheduling will put the new

process with the higher priority than the priority of the currently running process at the head of the ready queue.

Round Robin Scheduling Algorithm

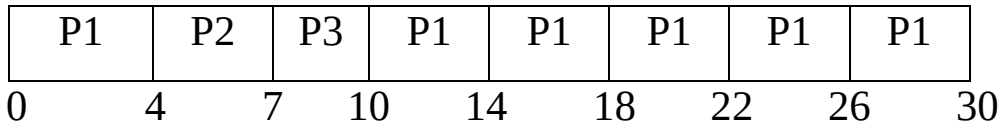
The Round Robin algorithm is designed especially for time sharing system. It is similar to FCFS but preemption is added switch between processes. A small unit of time called time quantum (or time slice) is defined. A time quantum is generally from 10 to 100 milliseconds. The ready queue is treated as a circular queue, allocated the CPU to each process for a time interval of up to 1 time quantum.

The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatch the processes. One of two things will then happen. The processes may have a CPU burst of less than 1 time quantum. In this case, the processes itself will release the CPU. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the currently running processes is longer than 1 time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.

As an example consider the following set of processes with the length of the CPU burst given in millisecond:

Process	Burst time
P1	24
P2	3
P3	3

The Gantt Chart is as follows:



The average waiting time = (17)/ 3=5.66 millisecond

The average waiting time under RR policy is quite long.