INTRODUCTION

1.1 Introduction of Operating System

1.1.1 Definition of Operating System

The simple definition of Operating System is, it create a platform for providing interface between the hardware and the computer application services. The computer system has operating system for functionality of the system programs. The application programs also need the operating system to interact with the hardware of the computer system. The operating system provides the environment to the application for running their programs in convenient way. It also provides information protection, well manage the resource access control to the user, multiprogramming or multiprocessiong execution of programs, CPU utilization in economic way etc.

1.1.2 Basic Architecture of Operating Systems

Here the Researcher describes the operating system working in the main computer system. The Computer system processing broadly categorizes in the four following categories:

(i) Hardware
(ii) Operating System
(iii) System and Application programs
(iv) Computer Users

These four components are shown in Figure 1.1.

(i) **Hardware**: The major hardware which are used in computer system is containing central processing unit (CPU), memory, and input/output (I/O) devices provides the basic computing resources for the system and used to execute the application program.

(ii) **Operating System**: The Operating System is a system program which performs as an intermediate manager between computer hardware and computer users. The operating system provides services to the hardware and organizes the procedure of its utilization among the dissimilar application programs for the dissimilar users.
(iii) **System and Application Programs:** The system and application programs such as database system, text editor, compilers, and assembler and many more define the platform in which the hardware resources are used to execute users computing problems.

(iv) **Computer Users:** The Computer user is outside environment.

![Diagram of Component of Computer System](image)

**Figure 1.1 Component of Computer System [2]**

1.1.3 **Layered Architecture of Operating System**

The figure 1.2 shows the layered architecture of operating system. The operating system can be defined in two modes which are

(i) **User Mode**

(ii) **Kernel Mode**

The functionality of user mode and kernel mode is described as below.

(i) **User Mode:** In this mode the application program will be run. The user creates their own program using application program like MS Word, MS Excel etc. These programs than interfaced with the computer hardware using operating system services.

(ii) **Kernel Mode:** It provides the basic functionality to the system services, file system, memory and I/O device management, CPU scheduling and the hardware. The system services are used to provide for the convenience of the programmer, to make the programming task easier. The kernel uses the
hardware instruction to create a system calls which used by the application programs. The kernel manages and control all over the operation of operating system. It also manages the memory and I/O operation by creating a sequence to system calls. The CPU scheduling is use to schedule the process. The scheduler selects the processes which are ready to execute and allocate it to the CPU for executing. The scheduler select the process from the ready queue on the basis of the scheduling algorithm like as FCFS (First Come First serve), SJF (Shortest Job First), Priority Scheduling, Round Robin and so on.

Figure1.2 Layered Architecture of Operating System [2]

1.1.4 Types of Operating System

Operating systems are there from the very first computer generation and they keep evolving with time. In this section, there will be discussion of some of the important types of operating systems which are most commonly used.

1.1.4.1 Batch System

Programs were acceded on cards or tape to an operative who batches jobs together sequentially. The program that controls the implementation of the jobs was called monitor - a manageable version of an operating system. The interface to the monitor was exerted through Job Control Language (JCL). For example, a JCL request could be to run the compiler for a specific programming language, then to
link and load the program, then to run the user program. Simple batch system upgrades the utilization of computers. The features of batch system are

a. Memory protection: do not allow the memory area holding the monitor to be altered
b. Timer: prevents a job from monopolizing the system

1.1.4.2 Time-sharing Operating Systems
The IBM, TSO and VM/CMS used a technical term time sharing in which multiple users can share the computer systems which are situated at different terminals. Basically the Time Sharing technical term is not generally used because it is changed by the term multitasking in which the multiple task can be executed simultaneously. The terminal which is also not used in today computer science it is also replaced by desktop computer or workstations. As we know that computer revaluation is become very fast in terms of technology in which the operating system has evolved from batch system to interactive system. The multiprogramming is not only the suitable approach but it also requires some more functionality to enhance the performance to the computer system.

So the time sharing processing is included for enhancing the performance of the computer system. The time sharing processing in which the time slice is allocated to each system to execute their programs. The processing will be seems as fast they are not by using the time sharing processing. In the time sharing system the CPU can be surrender by the executable task and this can be completed either by task itself or it can be done by calling a hardware interrupt that is call external event. The time sharing system use the concept of multiprogramming in which the multiple program can be executed simultaneously.

The multitasking system in which the applications are executed is voluntarily transferring the data to each other. This procedure is also called the cooperative multitasking. This technology is adopted by many computer systems. The cooperative multitasking is not frequently used in larger system. This scheme is generally used the windows old version which are developed before the Window 95 and the Window NT. The cooperative multitasking is also used in 16-bit Windows 9x operating system. The NetWare operating system is also use the cooperative multitasking but it
is used up to its version 6.5. The RISC OS system also used this type of operating system.

The cooperative multitasking system in which the processor time is distributed among all the processes in the system. So the programming for that type of system must be fair, a poor program for distributing the processor to each processor may be develop a very serious problem in which a single process can halt all the processing of the system.

The real time embedded application and the time dependent applications are designed using the cooperative multitasking paradigm ignoring the designing complexity of is type of the system.

1.1.4.3 Distributed Operating System

The network provides the secure path for transferring the data between two or more than two terminals or systems. The distributed system provides the functionality to the network in which the data is transferred in economic and efficient way between the terminals. The distributed system provides the functionality of the sharing the resources which are situated at different locations and provides very advanced and fast processing of the user application to the users. The networks have the varying parameters in terms of the protocol used by the network, routing distance of the nodes and the policies used by the communication entities. The operating system supports the variation of the protocol used by the network it is not affected by which protocol is used by the particular network. In most cases the TCP/IP protocol is used by the network. The ATM and other protocol is also used by network but in comparison to TCP/IP they are rarely used. The windows and UNIX operating system support all the properties of TCP/IP protocol in efficient ways. They also provide the extra functionality to support the protocol to provide their needs. For supporting the network protocol the operating system have an interface device called the network adapter for managing and control the operation of network protocol for the communication. The operating system also has the software application to manage the data packaging when it is transferred to other system and unpack aging of data when it receive the data from other system. The network is also established in three way one is local area network (LAN) which is used for short distance network like as network in room or small office which are exist in same building. The wide area network
(WAN) is used for wide area networking purpose. Where the distance is very large like as the network within a cities, countries etc.

1.1.4.4 Network operating System
The network provides the secure path for transferring the data between two or more than two terminals or systems. The distributed system provides the functionality to the network in which the data is transferred in economic and efficient way between the terminals. The distributed system provides the functionality of the sharing the resources which are situated at different locations and provides very advanced and fast processing of the user application to the users. The networks have the varying parameters in terms of the protocol used by the network, routing distance of the nodes and the policies used by the communication entities. The operating system supports the variation of the protocol used by the network it is not affected by which protocol is used by the particular network. In most cases the TCP/IP protocol is used by the network. The ATM and other protocol is also used by network but in comparison to TCP/IP they are rarely used. The windows and UNIX operating system support all the properties of TCP/IP protocol in efficient ways. They also provide the extra functionality to support the protocol to provide their needs. For supporting the network protocol the operating system have an interface device called the network adapter for managing and control the operation of network protocol for the communication. The operating system also has the software application to manage the data packaging when it is transferred to other system and unpack aging of data when it receive the data from other system. The network is also established in three way one is local area network (LAN) which is used for short distance network like as network in room or small office which are exist in same building. The wide area network (WAN) is used for wide area networking purpose. Where the distance is very large like as the network within a cities, countries etc.

1.1.4.5 Real Time Operating System
The real time systems are used to execute the real time application. The real time applications are hardly time dependable task. The real time system tasks have a particular deadline in which they should be completed. So these type of task are required the guaranteed amount of time of CPU. If it is not assigned the CPU to the task at particular time at which the task is requiring the CPU then the deadline of real time task will be missed and it will be causes a serious problem. The real time system
are basically designed for like designing of machine which strictly follow the time if the time is followed by that system there will be serious disasters. The multitasking concept is used for developing the real time application for the reason that the multitasking system provides execution of unrelated task on the single processing system. So the multitasking system needs some priority level for executing the real time task. In the multitasking the real time task are given the highest priority than other tasks. The real time task priority is also defined as per their deadline. In normal case the earliest deadline first algorithm is evaluated for assigning the priority to the real time task. The sensor is also the application of the real time system. If the sensor may not be performed in bounded time there may be serious disaster can be happen. Many applications are also there like real time weapons which are used in air craft, the launching of rockets etc. the real time application is also categorize in two parts one is the hard real time system and another one is the soft real time system. The hard real time applications are where the task should be completed within the time frame. The soft real time system in which task can be completed after its deadline but it should be minor. The application of real time system are scientific experiments, industrial control system many more. So the above mentioned statement the real time system application is hardly depends upon the time. So the operating system is designed in such way that it can satisfy all the constraint of real time applications. The real time task should be executed at highest priority. For executing the real time application a operating system is designed which is namely called the real time operating system (RTOS). The tasks are executed in that system as per multitasking models. It has defined its own set of task scheduling algorithm like as earliest deadline first and many more. It also defines the resource allocation strategies in the form of execution of priority inheritance protocol (PIP) and priority ceiling protocol (PCP).

1.1.5 Process Concepts
A process is basically defined by the term active part of the program. The program is passive entity which is stored in the memory. When the program is run by the user than it is turned in to the process, A process may be whole program and part of the program which are currently in execution state. The process can be use multiple treads for execution, it depends upon the properties of the operating system. The threads are then executed in concurrent fashion on the processing unit. The program may contain
the multiple instructions which are stored in the memory. When this program is called for execution then the instruction are executed on the processing unit in the form of process. So the process is active execution of the instructions. A program may contain the multiple instructions. Like for opening of the program one thread is used, memory allocation for data another thread is used, for resource allocation another thread is used. In this way multiple threads can be used for an execution of a program. The process is also defined the state of process like current state, running state, waiting state etc. The figure 1.3 shows a basic architecture of process in memory.

Figure 1.3 Processes in Memory [2]

1.1.5.1 Process State

A process is executed in a number of states. The state of a process shows the in progress action of that process. A process can be resides in one of the following states:

1. **New**: In this state process is being created. It will be kept in to new queue.
2. **Running**: In this state processes are being executed on processor.
3. **Waiting**: In this state process will wait for CPU for its execution. When CPU is busy in execution some event such as I/O request or high priority process then the process will be in waiting queue.
4. **Ready**: In this state the process is ready to execute and waiting for CPU for execution.
5. **Terminated.** The process comes in this state when it finishes its complete execution. The process switches among these states. The process state diagram is shown in figure 1.4. When the process is created it will be kept in to new queue. When the process is ready to execute than job scheduler will move the process from new queue to ready queue. The process which are in ready queue is selected by scheduler (which implements the scheduling algorithm) and dispatched to the processor. Then processor will execute the process. The process will moves in waiting queue if it can be preempted by higher priority queue or for the I/O completion. After the completion of execution of processes on the processor, it will move in to terminate queue.

![Processes State Diagram](image)

**Figure1.4 Processes State Diagram [2]**

1.1.5.2 **Process Control Block**

Each process is associated with some properties in the operating system and these properties are defined in the form of process control block (PCB)—also called a task control block. A PCB is shown in Figure1.5. The following properties of processes are defined in PCB.

(i) **Process state.** This field kept information in which state the process is currently resides. It may be new, ready, running, and waiting, halted state.

(ii) **Program counter.** This field kept information about the address of the next instruction to be executed for this process.

(iii) **CPU registers.** This field kept information about the registers which is associated with that particular process. The register may be accumulators, index registers, stack pointers, general-purpose registers. It also kept the program counter register, it is saved when a process executing when a high
priority process switches that particular process. The all relevant information of registers may be saved in this field.

(iii) **CPU-scheduling information.** This field kept the information about the scheduling algorithm properties of the process like process priority, CPU burst time, arrival time etc.

(vi) **Memory-management information.** This field kept information about the value of the base and limit registers and the page tables, or the segment tables, which are the memory management scheme which are used by the operating system.

(vii) **Accounting information.** This field kept information about the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

(viii) **I/O status information.** This field kept information related to the, number of I/O devices are used by the process, a list of open files, and so on.

Figure 1.5 Process Control Box (PCB) [2]

The process switches on CPU is shown in figure 1.6. The process P0 is switched by process P1. First the P0 will save its all relevant information in to its PCB. The operating system then reloads the state of P1 from its PCB. This can be accomplished by a calling of certain system calls. When the P1 finishes its execution then the operating system save the state of P1 in its PCB and reload the P0 form it’s PCB.
1.1.5 Operation on Processes

The process is very important term in operating system which is executed in the system concurrently, and they are created and deleted by operating system dynamically. The operating system defines the procedure for creating and terminating the process.

1.1.5.3 Process Creation

An operating system creates several new processes by calling a system call create-process during execution. The creating process is called a parent process, whereas are called the children of that process.

When an operating system is booted, typically several processes are created. Some of these processes interact with the users and perform work for them. While some are not associated with particular user but instead have some specific function.

In UNIX system there is only one system call to create a new process i.e. fork.
1.1.5.3.2 Process Termination

After a process has been created, it starts running and does whatever its job is. A process terminates when it finishes executing its last statement and ask the operating system to delete it by using the exit system call or exit process. At that point the process may return data in the form of output to its parent process, via wait system call. All the resources allocated to the process including physical and virtual memory, I/O buffers, open files, are deallocated by the operating system. These are generally four conditions, which may result in process termination, these are.

(a) **Normal Exist**: Most processes terminates because they have done their jobs. This call is exist in UNIX.

(b) **Error Exist**: When processes found a fatal error.

(c) **Fetal Error**: An error caused by process often due to a bug in program for example, executing an illegal instruction, referring non-existing memory or dividing by zero.

(d) **Killed by Another Process**: A process executes a system call telling the operating system to terminate some other process.

1.1.5.4 Cooperative Process:

There are two types of processes are created by operating system: cooperating and independent processes. A process is independent if it does not use the other process or used by other processes. A process is called to be a cooperating process if it may use the other process or used by other processes in the system for its complete execution. There are following advantages of cooperating processes:

1. **Information Sharing**: by allowing this feature the users can share the same information in multiple processes e.g. a shared file. The O/S needs to provide a way of allowing concurrent access.

2. **Computation Speedup**: sharing of data by process speedup the computation by the bigger task is divided in to the smaller task that can be executed parallel on multiple processors.

3. **Convenience**: the cooperative process running on multiple processes to get computational speedup, or where a multiple components are interconnected via a pipe structure that attaches the stdout of one stage to stdin of the next etc.
If we allow processes to execute concurrently and share data, then we must either provide some procedure to grip conflicts e.g. writing and reading the same piece of data. We must also be prepared to handle inconsistent or corrupted data.

**Example:** here the researcher included simple example of two cooperating processes. The problem is called the Producer Consumer problem and it uses two processes, the producer and the consumer.

(a) **Producer Process:** It produces information that will be consumed by the consumer.

(b) **Consumer Process:** It consumes information produced by the producer. Both processes run concurrently. Initially the producer has no information to produce and consumer has no information to consume. There are two verity of the producer depending on the size of buffer. If the buffer size is taken infinite then it is called the Unbounded Buffer Producer Consumer Problem. If there is fixed length of buffer than it is called bounded buffer Producer and Consumer Problem. When the buffer is full, the producer will wait until there is some information consumed by consumer or buffer space may have the space for storing the produced information.

**Bounded Buffer - Shared Memory Solution:**

Here is a shared memory solution for Bounded Buffer Producer Consumer problem. Both processes have some shared data that can be accessed and updated. The shared data is as follows:

```c
// Shared data:
    int n1 = 10, item1, in1, out1;
    // number of items in buffer is at most 10
    int buffer1[n];

    // initially consumer and producer are currently at the position of buffer element 0.
    in1 = 0;
    out1 = 0;

    // Producer Process .
    while (true)
    {
```
while (in1 + 1 % n1 == out1)
{
    // Do nothing
}
// produce a random item buffer1[in] = nextProduction; in1 = in1 + 1 % n1;
}
// Consumer process
while (true)
{
    while (in1 == out1)
    {
        //Do nothing
    }
    nextConsumed = buffer1 [out1];
    out1 = out1 + 1 % n1;
}
The producer will inspect to see if there is any space in which to keep a newly produced item (outer while loop). If there is no space, then it waits, until there is some space. The consumer waits while the buffer is empty. If there is something/ it grabs the next item and consumes it. One drawback with this solution is that there is one element of the buffer that is wasted.

1.1.5.5 Inter-Process Communication

An operating system provides inter-process communication in which processes need to communicate with the other processes, while they are running and for that, the output of the first process must be passed to the second processes and so on. Thus there is need for communication between the processes, preferably in a well structured way not using interrupts.

There are several mechanisms for inter-process communication. We discuss some of these below.

1. **Shared Memory**: Processes that share memory can exchange data by writing and reading shared variables. As an example, suppose there are two processes which use the shared memory future say p and q. Suppose the process p
writing a new data s in share memory space or can say it will be read by the process q. Here an important question is comes in mind. How does q can get the information about the new information is written in to s by process p? In some cases, q does not need to know or instance, if it is a load balancing program that simply looks at the current load in p’s machine stored in s. When it does need to know, it will use the polling mechanism but this polling scheme may affect the performance of CPU. Another solution is that it can use software interrupt, which we discuss below. A simple alternative is using semaphores for locking the common variable. Process q will wait till p changes s and sends a signal that allows q to enter in critical section.

2. **Software Interrupt:** this feature provide a process can interrupt another process. Software interrupt have two parameters which are process id and interrupt number. The process also issues a handler to handle that interrupt which has two parameters interrupt number and handler. Software interrupts allow only one bit information to be communicated - that an event associated with the interrupt number has occurred.

3. **Message Passing:** In computer science, **message passing** sends a message to a process (which may be an actor or object) and relies on the process and the supporting infrastructure to select and invoke the actual code to run. Message passing differs from conventional programming where a process, subroutine, or function is directly invoked by name. Message passing is key to some models of concurrency and object-oriented programming.

Message passing is used ubiquitously in modern computer software. It is used as a way for the objects that make up a program to work with each other and as a way for objects and systems running on different computers (e.g., the Internet) to interact. Message passing may be implemented by various mechanisms, including channels.

1.1.6 **Threads**

Process executes the program by using the thread. The thread controls the whole execution of process. Practically all current operating systems provide features facilitate a process to include multiple threads of control.

The CPU utilization is depends upon the how many threads are executed to execute a process. Every thread is defined by some properties which are thread ID, program counter, register set, and stack. The threads also shares code section, data section, and
other operating-system resources, such as open files and signals of other threads which are executing for same process. In older scenario each process uses only a single thread. The modern operating system uses multiple threads of control it can executes multiple task simultaneously. Figure 1.7 shows the difference between single-threaded process and multithreaded process.

Figure 1.7 Single-Threaded and Multithreaded Processes [2]

Modern computer system software applications executed in multithreaded environment. An application has implementation of separate process with several threads of control. A web browser also uses the multithreaded architecture in which one thread show the images or text while another thread repossess data from the network, another example. When writing a word document then word processor may use a thread for showing pictures, another thread for responding to keystrokes from the user, and a third thread for performing spelling and grammar checking in the background. Applications can also be designed to support the processing in multi-core systems. Such applications executed in parallel manner on the multiple computing cores which achieve high degree of thread utilization.

1.1.6.1 Advantages of Threads

The advantages of multithreaded programming can be defined by four most important categories:

1. **Responsiveness.** The Multithreading provide the feature that allow a program to continue running even if it some portion of it is not responding or is taking much more time to complete the execution of some very big operation. So it increases responsiveness to the end user. This feature is especially helpful in designing user interfaces. For example, suppose when a user clicks a button that results in the
performance of a lengthy calculus operation. A single-threaded architecture will not provide response to the user until the operation had completed. While if it uses the multithreaded architecture than lengthy calculus operation will be executed by separate thread and the application will be executed by another thread so it may be responsive to the user.

2. **Resource sharing.** In computer system the processes can share resources using shared memory and message passing architecture. These architectures must be explicitly programmed by the software developer. However, in threading architecture the threads can share the memory and the resources of the process to which they belong by default. The advantage of sharing code and data is that the application can use the same address space to execute different threads.

3. **Economy.** Allocating memory and resources for process creation is costly. Because threads share the resources of the process to which they belong by default no need of extra programming done by software developer. The threads are created in very economic way due to less code section it is containing.

4. **Scalability.** The advantages of multithreading can enhance in multiprocessor architecture, where threads may be executed in parallel fashion on different processing cores. While A single-threaded process can be executed on one processor, if there are n processor are available.

1.1.6.2 Types of Thread

There are two types of Threads are existed which are following

a. **User Level Threads** – theses threads are managed by user.

b. **Kernel Level Threads** – theses threads are managed by operating system by executing a sequence of system calls.

(a) **User Level Threads**

In this model, the user manages thread management. The kernel is not conscious of the creation or deletion of those threads. The application user define a thread library which store programming code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application begins with a single thread and begins running in that thread. The model of user level threads is shown in figure 1.8. The advantages and disadvantages of this model are as follows

**Advantages**
(i) Thread creation and deletion does not require Kernel attention.
(ii) User level thread can be executed on any operating system.
(iii) Scheduling of user level thread is also defined by the user which may be varying application wise.
(iv) User level threads are very less code so it is easy and use less time to create and manage.

Disadvantages
(i) In a complex operating system, there is most system calls are blocking.
(ii) The Multithreaded application may not take the benefit of multiprocessing system.

![Diagram showing User Level Threads](image)

Figure 1.8 Diagram showing User Level Threads

(b) Kernel Level Threads
These types of threads controlled and managed by kernel. The thread management code is separated from application area. These threads are working in the supervision of the operating system. The kernel may be programmed for supporting multithreaded. The threads created for an application will be managed and controlled single process.

The Kernel stores all relevant information of process and also stores information of threads which are created for that process. The kernel defines scheduling on the thread basis. The thread creation, thread scheduling and thread management all are done in kernel space. Kernel threads are overall take much time to create and manage than the user threads.

Advantages
(i) Multiple threads of the same process can be simultaneously scheduled or executed by kernel on multiple processes.

(ii) The Kernel can execute other thread of the same process if one thread of that process is blocked.

(iii) Kernel routines can also be programmed as multithreaded model.

Disadvantages

(i) Kernel threads have more code section than user threads so in general they take much more time to create and manage than the user threads.

(ii) When one thread replaced by another thread within same process than it is requires a mode switch to the Kernel.

1.1.6.3 Multithreading Models

Some operating system are comes in the combination of user level thread and Kernel level thread ability. Solaris operating system is a best example of this approach. In this type of system, multiple threads can be executed in parallel fashion on multiple processors within the same application. The blocking system cannot block the entire process only it can block particular thread which consume more time in complex calculation. Multithreading models relationship can be defined in three following ways.

1. Many to Many Relationship.
2. Many to One Relationship.
3. One to One Relationship.

1.1.6.3.1 Many to Many Model

In this model, there many user level threads are combined with many Kernel thread. The Kernel threads are designed for a specific machine or specific application.

The Figure 1.9 shows the many to many models. In this model, programmer can create n numbers of user threads as requirement and the resultant Kernel threads can executed in parallels on a multiprocessor.
1.1.6.3.2 Many to One Model

In this model there is n numbers of user level threads and one Kernel level thread will be combined. The user space is used for Thread management. If a single thread of any process executed a blocking system call than the entire process will be blocked. Multiple threads cannot be executed in parallel manner because only one thread can access the Kernel at a time.

If the user level thread libraries are not supported by system due to its implementation or development environment of thread in the operating system then Kernel threads are used to execute the user level thread by using kernel thread library. So in this level the operating system uses the Many to one relationship modes. The Figure 1.10 shows the model.

Figure 1.9 Many to Many Model of Multithreading

Figure 1.10 Many to One Model of Multithreading
1.1.6.3.3 One to One Model

In this model each user level thread combined with the kernel level thread. This model can executed multiple threads in parallel fashion on multiple processors. This model is more responsive because if one thread is blocked than other thread of same process may be executed on different core of processor.

Disadvantage of this relationship that each user thread will requires the corresponding Kernel thread. OS/2, window NT and windows 2000 use one to one relationship model.

![One to One Model of Multithreading](image)

**Figure1.11 One to One Model of Multithreading**

1.2 Need for Proposed Research Study

Most of Round Robin (RR) CPU scheduling techniques levered in past few years has evolved to optimize the scheduling criteria in terms of minimizing average waiting time, average turnaround time, response time and number of context switches and maximizing throughput and CPU utilization and thus CPU scheduling has gained more in last decade or so.

1.2.1 Issue with Existing Round Robin (RR) CPU Scheduling Technique

The existing Round Robin (RR) CPU scheduling techniques suffer from following Problems.

1. **Selection of Time Quantum**: The simple Round Robin scheduling techniques suffer from the selection of time quantum. If the time taken is
too high then the performance of system may be decreases and if time quantum is taken too low than context switches may be increases which may leads to decreasing system performance. It is also define static time quantum.

2. **Priority of Process**: The simple Round Robin scheduling technique may not include any user priority and system priority. It only schedules the process in first come first basis. It defines equal priority of all process.

3. **Average Waiting Time**: The Simple Round Robin scheduling technique is works on FCFS scheduling so the waiting time for high priority process may becomes high.

4. **Average Turnaround Time**: The turnaround time of high priority process may also high if the selected time quantum is high.

5. **Number of Context Switch**: In the Round Robin scheduling if selected time quantum is too short than the number of context switching is increases which may leads degradation of performance of system.

6. **Response Time**: In the Round Robin Scheduling if the selected time quantum is too high than response time of the processes may be increases which may lead degradation of performance of system.

Above discussions and references prove that there is lot to be done in the field of Round Robin CPU scheduling, which can be evolved and performed to get good way of CPU scheduling of process and meet the good performance criteria of CPU scheduling. The works which have been done by various researchers and technocrats are individually very strong but they lack the compilation and are not a complete solution.

1.3 **Research Motivation**

In the previous section it has been found that there is no system exists for scheduling algorithm which is smart enough to prioritize the jobs and calculate dynamic time quantum in tendon. So the researcher is motivated to coin a new term called Best Job.

In this thesis the researcher has aims to identify a Best Job available in ready queue, which is smart enough to have **smart priority factor** weight age and **optimal time quantum** in order to optimize the context switching and maximize the throughput.
• Best Job can be defined as, the job in the queue which deserves to be executed first because; it can calculate the
  o **Smart Priority Factor (SPF)** which is the first property of best job. It is calculated on the basis of the User Priority or System Priority (it is defined as shortest burst time process has assigned highest priority). The researcher also has given 55% weight age to user priority and 45% weight age to system priority. The process which has lowest value of SPF can be called as Best Job in the queue and will be schedule first.

  So here in this work the researcher has defined the best job which has minimum value of Smart Priority Factor.

  Also a best possible job shall perform better than other scheduling methods in all the possible scenarios, in order to quantify the same, in this research work the Best Job Technique using SPF over SJF &RR has been proposed for below given three architecture of Operating Systems

  a. Uniprocessor Architecture
  b. Multilevel Feedback Queue
  c. Multi-Core Processing System

  o **Optimal Time Quantum** which is another property of best job. It calculates the Smart Time Quantum (STQ) dynamically. The STQ is calculated as first median of the burst time of the process and then the calculating mean average of Median and highest burst time among the processes left in the queue.

  So here in this work the researcher has defined the Optimal Time Quantum in terms of Smart Time Quantum (STQ).

1.4 **Objectives of Study**

The researcher proposes to design a new CPU scheduling which may enhanced the following performance matrices of scheduling.

  a. The researcher proposes to design and analyze a new CPU scheduling which combine the feature of SJF with RR scheduling.

  b. The researcher proposes to design and analyze a new CPU scheduling which generate the Time quantum dynamically using priority of process.
c. The researcher proposes to design and analyze a new CPU scheduling in Uniprocessor, multilevel feedback and Multi-core processing system which minimize the Average waiting time of processes.

d. The researcher proposes to design and analyze a new CPU scheduling in Uniprocessor, multilevel feedback and Multi-core processing system which minimize the Average turnaround time of processes.

e. The researcher proposes to design and analyze a new CPU scheduling in Uniprocessor, multilevel feedback and Multi-core processing system which minimize the number of context switches of processes.

1.5 Contributions of Study

1. The proposed technique Smart Job First Dynamic Round Robin (SJFDRR) shall overcome the following issue prevalent in older techniques.
   a. It defines a Smart Priority Factor (SPF) which is calculated on the basis of user priority and system priority. So the priority is included in the selection of process for execution.
   b. In this technique the user priority has been given more weight age than system priority. So the response time may be decreases for high priority process.

2. The proposed technique has defined dynamically calculated time quantum which is called Smart Time Quantum (STQ), shall overcome the following issue prevalent in older techniques.
   a. The STQ is calculated on the basis of remaining burst time of the processes. So the selection of time quantum is removed which is major problem in older techniques.
   b. The STQ calculation is depends upon the burst time of process which may lead to improve the performances of system.

3. The proposed techniques is implemented on Uniprocessor system, multilevel feedback queue system and Multi-core System and found better result.

1.6 Chapter Plan of Thesis

1. Chapter 1 is Introduction, as described above it gives the overview of Operating System, Process, Process Threading, issues of existing scheduling
techniques, research motivation and details out objectives and contribution of research.

2. Chapter 2 is Review of Literature, here the researcher has discussed more than 50 research papers on related technologies and developments in past few years, which helped the researcher to identify the research objectives, discussed in chapter 1 and also helped to understand the CPU scheduling technique as discussed in this thesis and need of improving them.

3. Chapter 3 is Materials and Methods, in this chapter researcher discusses all the Round Robin CPU scheduling algorithm on Uniprocessor, Multilevel Feedback Queue Scheduling algorithm and Multi-core CPU scheduling algorithm and analyze their performance matrices in given set of situations and methodology behind implementation of proposed work.

4. Chapter 4 Observations and Results is the proposed work, which is divided into three major components first implementation of Smart Job First Dynamic Round Robin (SJFDRR) with smart time quantum in Uniprocessor processor system, second implementation of Smart Job First Dynamic Round Robin (SJFDRR) with smart time quantum in Multilevel Feedback Queue scheduling and third implementation of Smart Job First Dynamic Round Robin (SJFDRR) with smart time quantum in Multi-core system. Here researcher also analyzes the performance of given proposed work both technically and theoretically and compare the same with existing techniques of previous work as discussed in chapter 3.

5. Chapter 5 Conclusions gives the final concluding summary on the outcomes of the proposed work justifying its viability and techno social benefit of the effort made, at the same time it also discusses the future possibilities and limitations of the proposed work.