

Context Switching

A PROJECT REPORT

Submitted by

S. YAFFIN [Reg No: RA2211032010053]

Under the guidance of

Dr. HEMAMALINI V

(Assistant Professor, Department of Networking and Communications, School of
Computing)

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

with specialization in **INTERNET OF THINGS**



**DEPARTMENT OF NETWORKING AND
COMMUNICATIONS**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,
KATTANKULATHUR- 603 203**

November 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**KATTANKULATHUR - 603203**

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that 21CSC203P project report titled “**Context Switching**” is the bonafide work of “**S. YAFFIN [Reg No: RA2211032010053]**”, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. HEMAMALINI V**GUIDE**

Assistant Professor

Networking and Communications

DR. ANNAPURANI PANAIYAPPAN .K**HEAD OF THE DEPARTMENT**

Professor

Networking and Communications

INTERNAL EXAMINER**EXTERNAL EXAMINER**



Annexure II

Department of Networking and Communications

SRM Institute of Science & Technology

OWN WORK DECLARATION

Degree/ Course: B.Tech/Computer Science Engineering with specialization in Internet Of Things

Student Name: S. YAFFIN

Registration Number: RA2211032010053

Title of Work : Context Switching

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that We have met the following conditions:

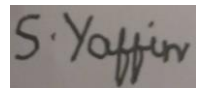
- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own

- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g., fellow students, technicians, statisticians, external sources) .
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my own work, except where indicated by referring, and that I have followed the good academic practices noted above.



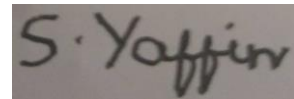
S.YAFFIN

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T.V.Gopal**, for his invaluable support. We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work. We are incredibly grateful to our Head of the Department, **Dr K. Annapurani Panaiyappan**, Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinators **Dr. A. Suresh**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support. We register our immeasurable thanks to our Faculty Advisor, **Dr. Praveena Akki**, Assistant Professor, Networking & Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course. Our inexpressible respect and thanks to my guide, **Dr. Hemamalini V**, Assistant Professor, Networking & Communications, SRM IST, for providing me with an opportunity to pursue my project under his mentorship. He provided me with the freedom and support to explore the research topics of my interest.

His passion for solving problems and making a difference in the world has always been inspiring. We sincerely thank the Networking and Communications Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

A handwritten signature in dark ink on a light background. The signature appears to be "S. Yaffin" written in a cursive, slightly slanted style.

S. YAFFIN

ABSTRACT

Context switching is a fundamental operation in modern operating systems that allows for the smooth execution of multiple processes and threads. It involves saving and restoring the state of a running process, enabling the CPU to seamlessly switch between tasks. Efficient context switching is crucial for the overall system performance and responsiveness. This project focuses on improving context switching mechanisms in operating systems to enhance their efficiency and overall performance.

The primary goal of this project is to optimize context switching by reducing the overhead associated with this critical operation. We aim to achieve this through several key approaches:

1. **Reducing Execution Time:** We will explore ways to minimize the time required to save and restore process states during context switches. This will involve optimizing data structures and algorithms used for context switching.
2. **Enhancing Scheduling Policies:** We will investigate the impact of different scheduling policies on context switching. By implementing more intelligent scheduling algorithms, we can reduce the frequency of context switches and, in turn, improve system performance.
3. **Thread and Process Management:** In addition to process-level context switching, we will also focus on thread-level context switching. This will involve optimizing the management of threads and their associated data structures.
4. **Hardware Acceleration:** Leveraging hardware support, such as CPU registers and memory management units, can significantly improve context switching performance. We will explore the integration of hardware-level optimizations to expedite context switching.
5. **Real-world Testing:** To validate the effectiveness of our optimizations, we will

conduct extensive testing using real-world applications and workloads. This will help ensure that our enhancements have practical benefits for a variety of system usage scenarios.

Efficient context switching is crucial for operating systems in various domains, from desktop computing to cloud servers and embedded systems. This project will contribute to the creation of faster and more responsive operating systems, which, in turn, will lead to improved user experiences and resource utilization.

By implementing the proposed optimizations, we expect to reduce the overall system overhead associated with context switching, resulting in improved performance, reduced power consumption, and increased system throughput. These enhancements will be particularly beneficial in scenarios where rapid context switching is essential, such as real-time systems and high-performance computing environments.

.

TABLE OF CONTENTS

CHAPTER NUMBER	CONTENTS	PAGE NO
1	Problem definition	x
2	OS concepts used	xi
3	Model Description of the project	xii
4	Flow Chart for the Project	xiii
5	Process Module and files	xiv
6	Implementation	xiv
7	Results	xv
8	References	xvii

CHAPTER 1

PROBLEM DEFINITION

1.1 Efficient Context Switching for Round-Robin Scheduling in Operating Systems:

We have implemented Context Switching and RR scheduling. We need to do Context Switching whenever Process switch takes place. Here Scheduling is done, so whenever time slice of one process is completed, processor is allocated to next process, for this we need to save the state of process so that next time it should run from where it was left. We have implemented context switch for Scheduling and I/O interrupts.

1.2 Need for Context Switching in OS

A Notes Manager is a vital tool in today's digital age due to the need for efficient Context Switching and RR Scheduling: The project involves the implementation of context switching alongside Round-Robin (RR) scheduling. In RR scheduling, processes are assigned time slices (quantum) during which they can execute. Once a time slice is completed, the processor needs to be allocated to the next process in the queue. This necessitates context switching, where the state of the currently executing process is saved, and the next process's state is loaded.

Saving Process State: To enable the smooth transition between processes, it's crucial to save the state of the currently executing process. This state includes critical information such as the program counter, registers, and other context-specific data. This saved state allows the operating system to resume the process from where it left off when it regains control of the CPU.

In summary, this project aims to implement efficient context switching in an operating system to facilitate Round-Robin scheduling. Context switching is required whenever a process switch occurs, be it due to the completion of a time slice or the initiation of I/O operations. Saving and restoring the state of processes are crucial aspects of this implementation to ensure a seamless and responsive operating system.

CHAPTER 2

OS CONCEPT USED

Five-state model: For handling processes we have considered five-state model. States are Blocked, Running and Ready. Blocked and Ready States are Implemented through Queue. Ready Queue is implemented using circular Queue because process should remain in ready until it terminates or its execution gets completed. If process suffers from I/O interrupt then that Process is dequeued from Ready queue and enqueued to Blocked Queue. Running is not queue because we have considered that one process can run at a time. Whenever Resource is available It is removed from Blocked queue and enqueued to ready queue.

Scheduling: For managing Ready Queue, scheduling is done. We have considered short term scheduling. For this, Round Robin is chosen as scheduling algorithm and quantum=2. Scheduling is done for the process present in ready queue. Quantum is chosen to be 2 to minimise the risk of starvation. Also if process is short than RR provides good response time. It gives fair treatment to all processes.

Context Switching: When context switch occurs, for example if process runs for one time slice, but its execution is not completed, then whenever next time processor is allocated to it process must start from where it has left. For this purpose PCB is used. This stored data is the context of process.

Process Control Block: Process has many elements. Out of which Program and code are essential. PCB contains crucial information needed for a process to execute. We have considered that PCB contains PID (process identifier), State (Describes in which state the process is), PC (Program Counter: it contains address of the next instruction which will be executed), SP (Stack Pointer :it is small register that stores the address of the last program request in a stack).

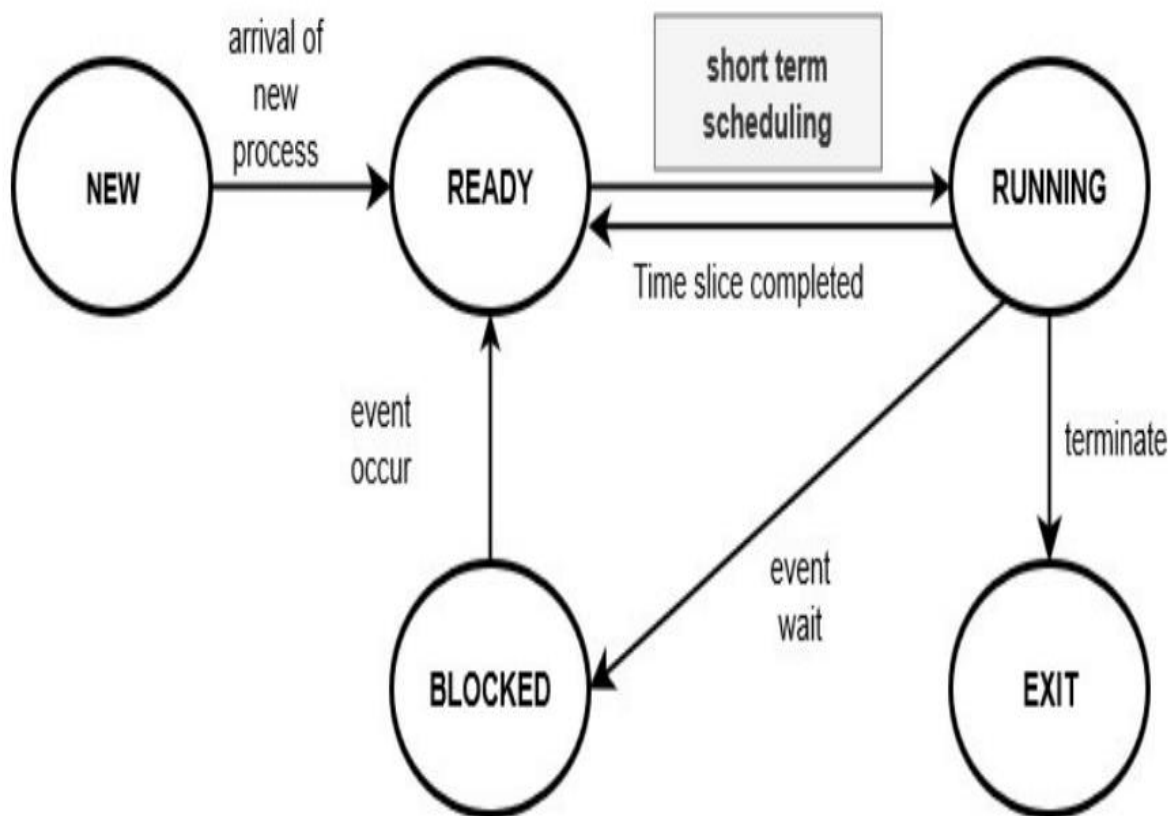
CHAPTER 3**MODEL DESCRIPTION OF THE PROJECT**

Figure 1: Five state model for context switching

CHAPTER 4

FLOW CHART FOR THE PROJECT

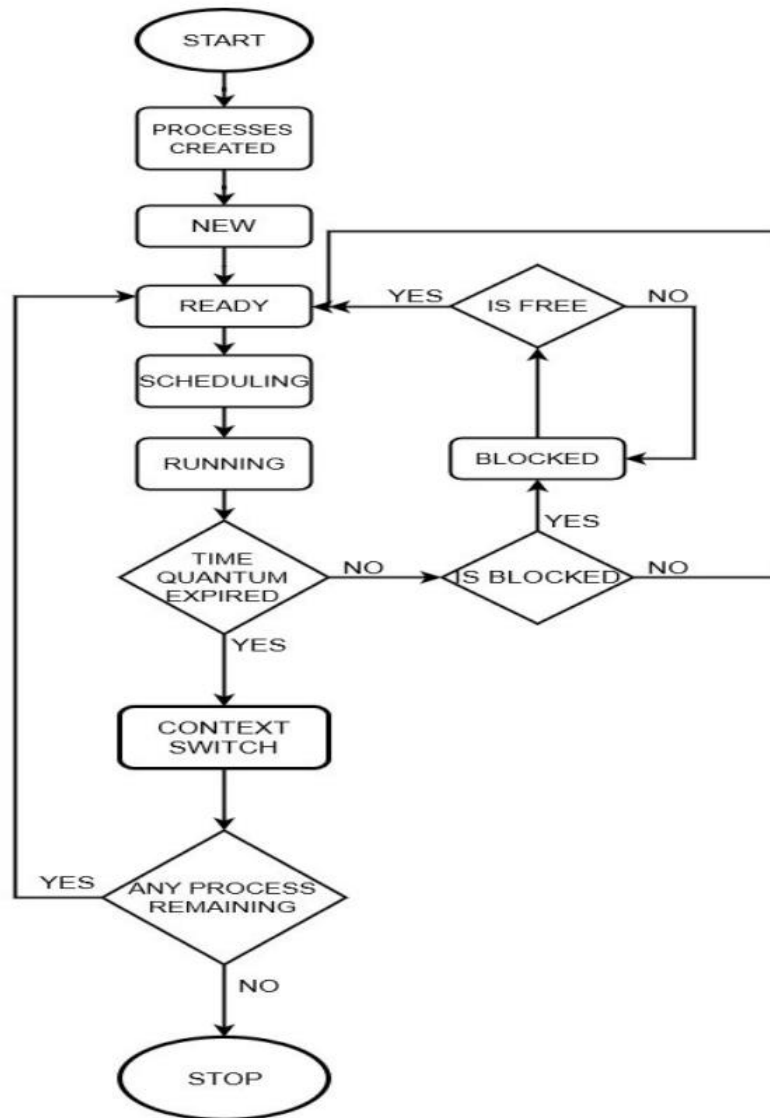


Figure 2: FLOW CHART

CHAPTER 5

PROCESS MODULES AND FILES

MODULE DESCRIPTION

Main.c It contains all operations such as process creation, scheduling , context switch, updating PCB and GUI, which is implemented using GTK.

To run : /gcc 'pkg-config gtk+-3.0 --cflags' main.c stack implementation.c queue implementation.c -o os 'pkg-config gtk+-3.0 --libs' ./os

stack implementation.c It contains stack implementation for push,pop operations

queue implementation.c It contains queue implementation for enqueue, dequeue operations

CHAPTER 6

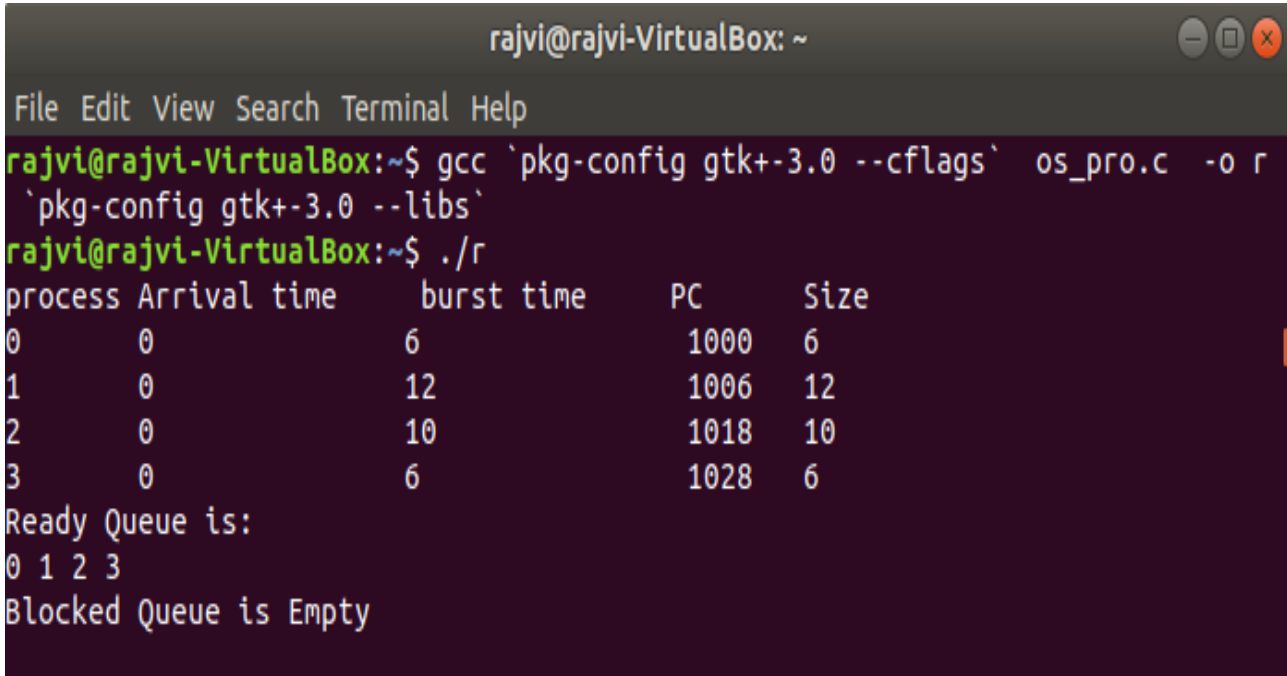
IMPLEMENTATION

IMPLEMENTATION:

Four processes are added to four different text files and in which instruction for process is given. Ready queue is formed using circular queue. Now scheduling is done,for every quantum when process runs, its PC is incremented,completed time for particular running process increases by quantum, 2 instructions are executed in one quantum and value of variables are PUSH-ed in stack of that process.Whenever that process again gets processor to execute, value of this registers is used. After this, next process which is in ready queue gets turn and execute instructions in similar way. This will continue until any of the process gets blocked. Whenever any process gets blocked, it is added to block queue and processor is given to next process. When needed resource for blocked process is free/available, it is again added o ready queue. As a result we are showing before and updated PCB of each process. Resources are blocked and released through GUI.

CHAPTER 7

RESULTS



```
rajvi@rajvi-VirtualBox: ~
File Edit View Search Terminal Help
rajvi@rajvi-VirtualBox:~$ gcc `pkg-config gtk+-3.0 --cflags` os_pro.c -o r
`pkg-config gtk+-3.0 --libs`
rajvi@rajvi-VirtualBox:~$ ./r
process Arrival time    burst time    PC    Size
0         0             6         1000   6
1         0            12         1006  12
2         0            10         1018  10
3         0             6         1028   6
Ready Queue is:
0 1 2 3
Blocked Queue is Empty
```

Figure 3: process block

```

rajvi@rajvi-VirtualBox: ~
File Edit View Search Terminal Help
Ready Queue is:
2 3 0 1
Blocked Queue is Empty

-----
Before execution
-----
Process      PC      State      SP
0            1004     Ready     0x5639466ef7b0
1            1010     Ready     0x5639466e0430
2            1020     Running   0x563946557e20
3            1030     Ready     0x5639466d9bb0
-----
After execution
-----
Process      PC      State      SP
0            1004     Ready     0x5639466ef7b0
1            1010     Ready     0x5639466e0430
2            1022     Ready     0x5639464e81e0
3            1030     Ready     0x5639466d9bb0
Ready Queue is:
3 0 1 2
Blocked Queue is Empty

```

Figure 4: Normal execution without any process blocked

```

rajvi@rajvi-VirtualBox: ~
File Edit View Search Terminal Help
Ready Queue is:
2 3 0 1
Blocked Queue is Empty
process 2 is blocked
Ready Queue is:
3 0 1
Blocked Queue is:
2

-----
Before execution
-----
Process      PC      State      SP
0            1004     Ready     0x55cfefb2d7170
1            1010     Ready     0x55cfefb0fd630
2            1020     Blocked   0x55cfefb2c02b0
3            1030     Running   0x55cfefb2ab840
-----
After execution
-----
Process      PC      State      SP
0            1004     Ready     0x55cfefb2d7170
1            1010     Ready     0x55cfefb0fd630
2            1020     Blocked   0x55cfefb2c02b0
3            1032     Ready     0x55cfefb2c0290
Resource 2 released!

```

Figure 5: when process is blocked


```

rajvi@rajvi-VirtualBox: ~
File Edit View Search Terminal Help
Resource 2 released!
Ready Queue is:
0 1 3 2
Blocked Queue is Empty

-----
Before execution
-----
Process      PC      State      SP
0            1004    Running    0x55cfeb2d7170
1            1010    Ready      0x55cfeb0fd630
2            1020    Ready      0x55cfeb2c02b0
3            1032    Ready      0x55cfeb2c0290
-----
After execution
-----
Process      PC      State      SP
0            1006    Ready      0x7f1990003b50
1            1010    Ready      0x55cfeb0fd630
2            1020    Ready      0x55cfeb2c02b0
3            1032    Ready      0x55cfeb2c0290
process 0 is completed
Ready Queue is:
1 3 2
Blocked Queue is Empty

```

Figure 6: when process is released

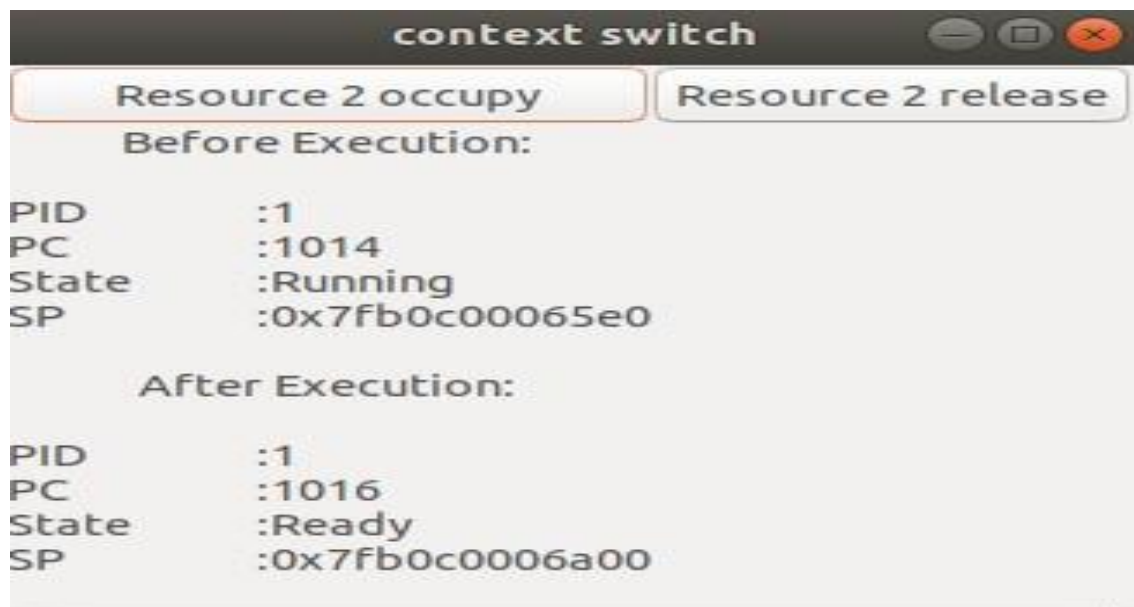


Figure 7: GUI

CHAPTER 8

REFERENCES

• **Conclusion:**

Context switching is important part of OS. As without context switching there is no use of different scheduling algorithms. If concept of context switch is not implemented then forcibly we have to use FCFS(first come first server) scheduling. RR(round robin) and other scheduling algorithms are not possible to implement without switching. And SRT(shortest remaining time) and SPN(shortest process next) and HRRN (highest response ratio next) are not applicable in real life as we do not know servicetime. In FCFS no need of context switching as once process enters it gets executed. For large value of quantum RR will behave like FCFS. So, to show context switching in better way RR with small quantum value is preferred.

• **References:**

- [1] Operating System :Internal and System Design, 9th edition, William Stallings, Pearson Publication.
- [2] https://www.tutorialspoint.com/operating_system/os_processes.html
- [3] https://www.tutorialspoint.com/operating_system/os_process_scheduling.html