

School of Engineering and Applied Science (SEAS), Ahmedabad University

B.Tech(ICT) Semester V: Operating System Lab(CSE341)

Project Title: Context Switching

- Group No : 04
- Group Members
1741064 Rajvee Kadchha
1741078 Rajvi Patel
- **Problem definition:**

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.

- **OS concepts 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).

- **Model:**

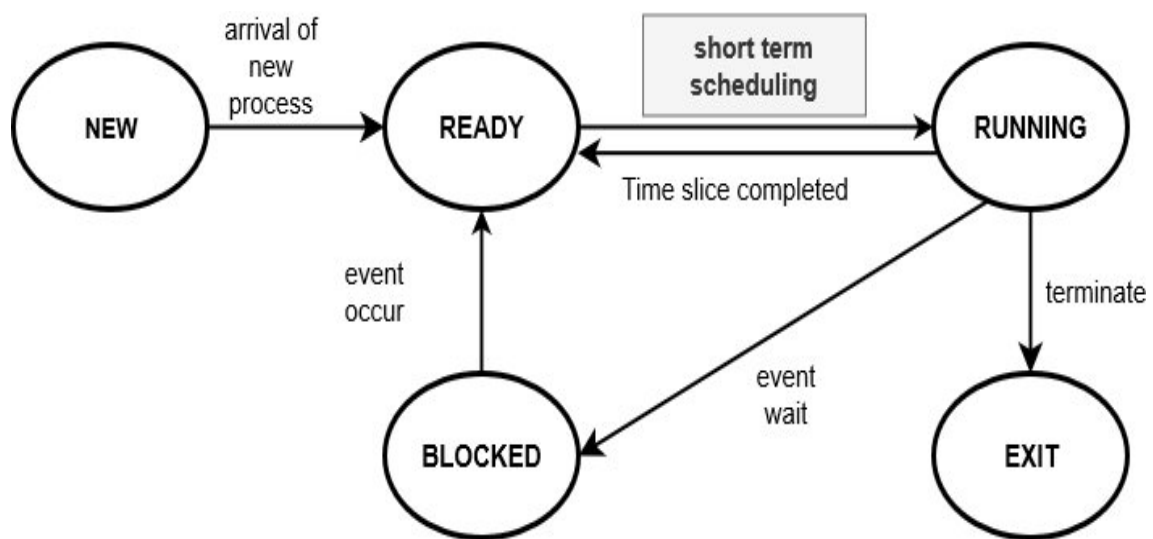


Figure 1: Five state model for context switching

- Flowchart:

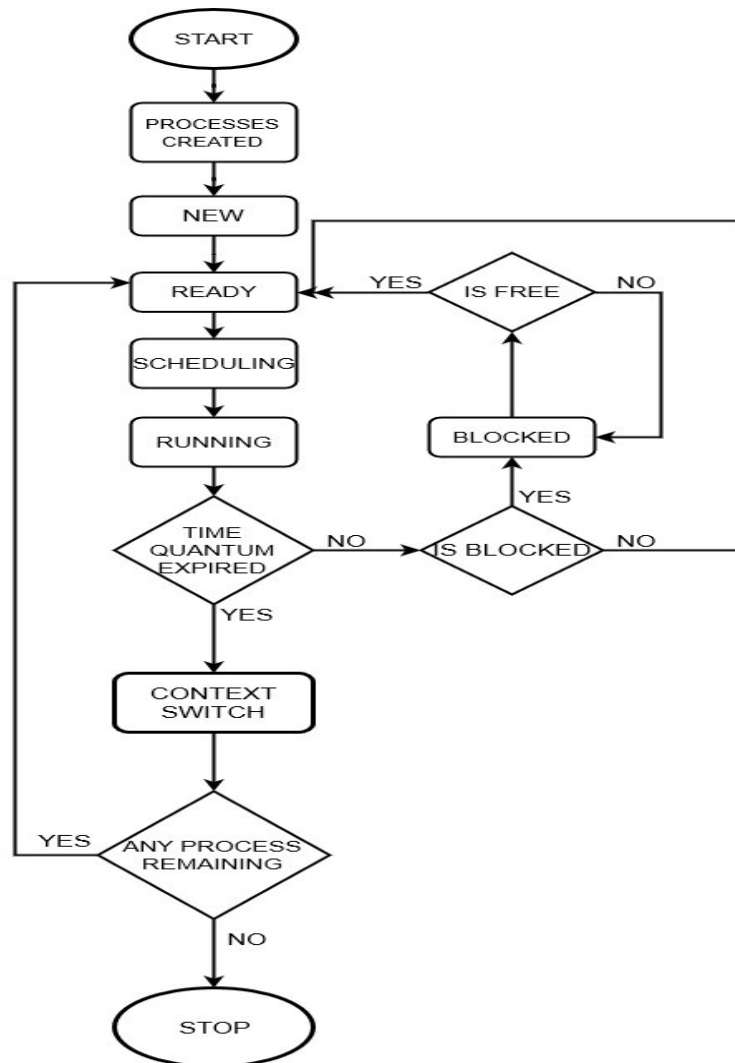


Figure 2: FLOW CHART

- Process Module and files:

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

process*.txt it contains instruction of process.

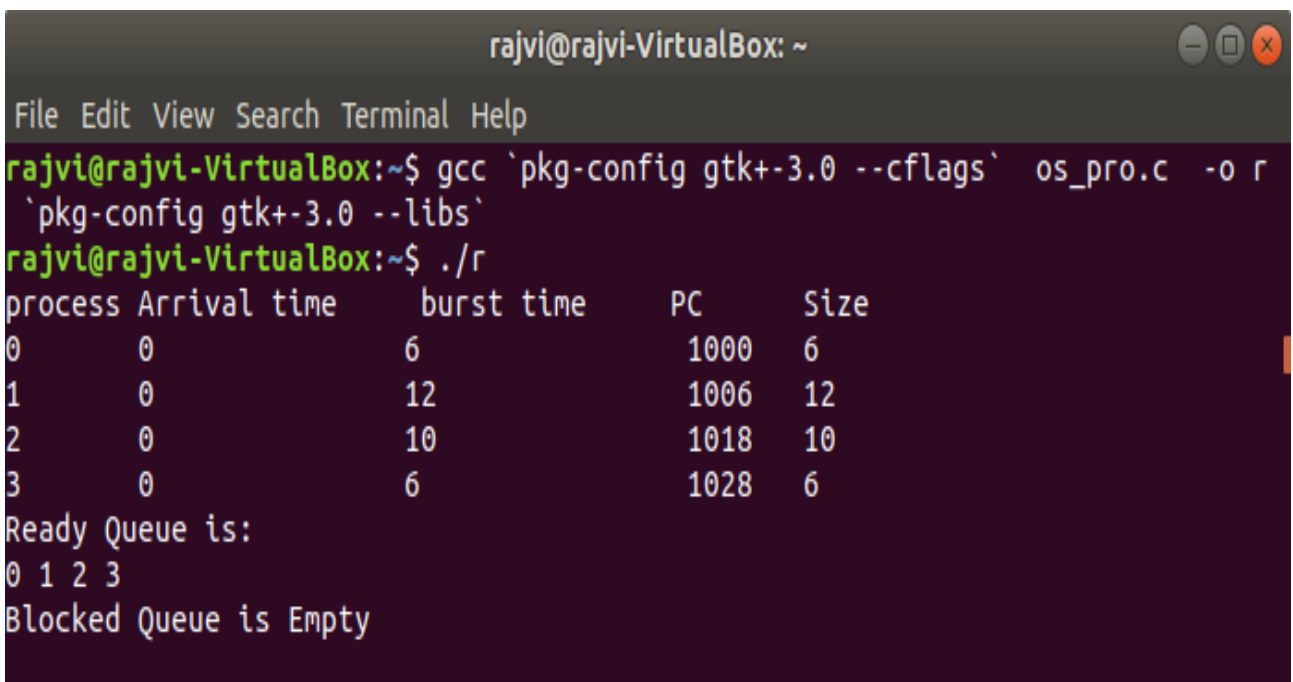
This code can run in linux. Only main.c needs to be runned. You need to install GTK3.0 to run this.

- **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 to ready queue.

As a result we are showing before and updated PCB of each process. Resources are blocked and released through GUI.

- **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      0x55cfeb2d7170
1            1010     Ready      0x55cfeb0fd630
2            1020     Blocked    0x55cfeb2c02b0
3            1030     Running    0x55cfeb2ab840
-----
After execution
-----
Process      PC      State      SP
0            1004     Ready      0x55cfeb2d7170
1            1010     Ready      0x55cfeb0fd630
2            1020     Blocked    0x55cfeb2c02b0
3            1032     Ready      0x55cfeb2c0290
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

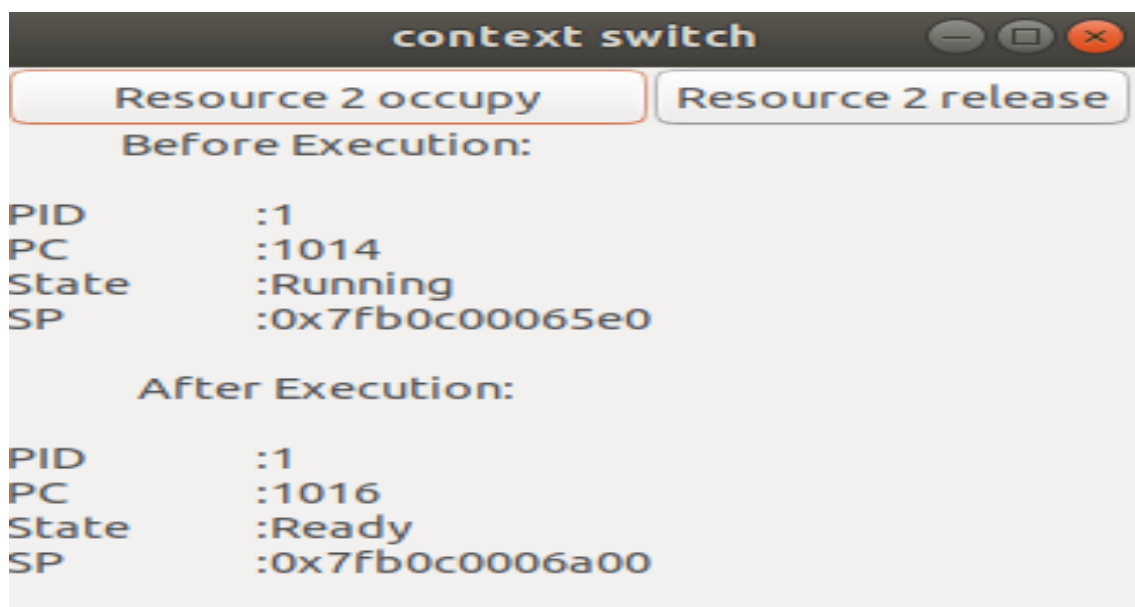


Figure 7: GUI

- **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 service time. 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
- [4] <https://www.sciencedirect.com/topics/engineering/process-stack-pointer>