

CS370

FCFS, Shortest Job First and Round Robin
Spring 2022

Homework-4 Review

Write a C program to demonstrate the following scheduling algorithms

- First Come First Serve
- Shortest job first without preemption
- Round Robin with time quantum

Evaluation:

- Sequence of execution in (Gantt chart)
- Individual turnaround time and Average turnaround time.
- Individual waiting time and Average waiting time.
- Throughput.

Homework-4 Review

A CSV file is provided which contains all the processes.

processes.csv

Header	➡	ProcessID	Arrival Time	Burst Time
Process 1	➡	2	0	5
Process 2	➡	3	0	3
•		1	9	8
•		4	10	6
•				

- You can expect a maximum of 9 processes existing in the processes.csv file

First Come First Serve

- Non preemptive.
- Schedules with respect to arrival time.
- Process that arrived first will get the CPU burst until it completes.
- If multiple processes arrive at the same time, execute lower PID first

Let's work on an example.

First Come First Serve

Proc ID	Arrival Time	Burst Time
2	0	5
1	0	3
4	3	2
3	12	6

Sort by
Arrival time



Process ID	Arrival Time	Burst Time
1	0	3
2	0	5
4	3	2
3	12	6

First Come First Serve

Process ID	Arrival Time	Burst Time
1	0	3
2	0	5
4	3	2
3	12	6

Gantt Chart



Pseudo Code to proceed with the Algorithm

Sort by PID

Sort by ArrivalTime

#No process to run --> IDLE

if($cur_time < arr$) :

gantt<-----((cur_time , arr , 'IDLE'))

#Process arrived and has waited

if($cur_time > arr$) :

gantt<-----((cur_time , $cur_time+burst$, pid))

#Get total wait time (no preemption)

#Advance to end of burst

#Process arrived and has not waited

else :

gantt<-----(arr , $arr+burst$, pid))

#Advance to end of burst

Shortest Job First without Preemption

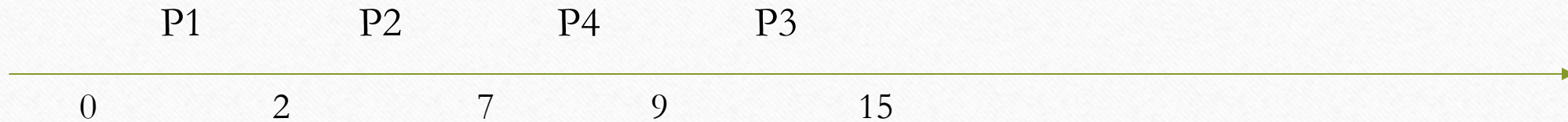
- The process with the least burst time will execute first
- Preemption - if a job comes in with a lower burst rate, it gets to execute right away

Again, let's work with an example.

Priority Scheduling without Preemption

Process ID	Arrival Time	Burst Time	
1	0	2	
2	1	5	
4	5	2	
3	3	6	

Gantt Chart



Round Robin

- Round robin – Everyone gets a chance.
- The quantum (integer) is used to determine the time quantum for round robin. (Command line argument)
- Ready Queue is First come First served.
- For this assignment, if a new process arrives the same instant when a process is switched out, the new process gets in the ready queue first.

Round Robin (quantum 2) – (1)

Process ID	Arrival Time	Burst Time	Remaining Burst time
1	0	3	3
2	1	5	5
4	3	2	2
3	12	6	6

Time elapsed = 0

Ready Queue

P1									
----	--	--	--	--	--	--	--	--	--

Round Robin (quantum 2) – (2)

Process ID	Arrival Time	Burst Time	Remaining Burst time
1	0	3	1
2	1	5	5
4	3	2	2
3	12	6	6

Time elapsed = 2

Ready Queue

P2	P1								
----	----	--	--	--	--	--	--	--	--

P1

0

2

Round Robin (quantum 2) – (3)

Process ID	Arrival Time	Burst Time	Remaining Burst time
1	0	3	1
2	1	5	3
4	3	2	2
3	12	6	6

Time elapsed = 4

Ready Queue

P1	P4	P2							
----	----	----	--	--	--	--	--	--	--

P1 P2

0 2 4

Round Robin (quantum 2) – (4)

Process ID	Arrival Time	Burst Time	Remaining Burst time
1	0	3	0
2	1	5	3
4	3	2	2
3	12	6	6

Time elapsed = 5

Ready Queue

P4	P2								
----	----	--	--	--	--	--	--	--	--

P1 P2 P1



Round Robin (quantum 2) – (5)

Process ID	Arrival Time	Burst Time	Remaining Burst time
1	0	3	0
2	1	5	3
4	3	2	0
3	12	6	6

Time elapsed = 7

Ready Queue

P2									
----	--	--	--	--	--	--	--	--	--

P1

P2

P1

P4

0

2

4

5

7

Round Robin (quantum 2) – (6)

Process ID	Arrival Time	Burst Time	Remaining Burst time
1	0	3	0
2	1	5	0
4	3	2	0
3	12	6	6

Time elapsed = 10

Ready Queue



P1

P2

P1

P4

P2

0

2

4

5

7

10

Round Robin (quantum 2) – (7)

Process ID	Arrival Time	Burst Time	Remaining Burst time
1	0	3	0
2	1	5	0
4	3	2	0
3	12	6	6

Time elapsed = 12

Ready Queue



P1

P2

P1

P4

P2

IDLE

0

2

4

5

7

10

12

Round Robin (quantum 2) – (8)

Process ID	Arrival Time	Burst Time	Remaining Burst time
1	0	3	0
2	1	5	0
4	3	2	0
3	12	6	0

Time elapsed = 18

Ready Queue



Method accepts (data, Time Quantum)

create Ready Queue

Sort by pid

Sort by arrival time

Add all processes arriving at time 0 to ready queue

Remove them from unarrived queue

Hold objects once finished in finished_queue

while **ready_queue** is! **empty** or **not all process arrived** or **prev_run_process** is not None :

Now in this while Loop implement the following

Check if every process has arrived to prevent early termination upon IDLE.

Add new arrivals to ready queue and remove from unarrived queue

Add most recently run process to ready queue after new arrivals

Checks if IDLE and sets process to correct time while updating Gantt Chart

Get next process in ready queue

Add waiting time

If process has more than time quantum remaining. Burst and store to put back on ready queue.

Else burst remaining amount and set to finished. Do not add back on queue

Notes:

-
- The maximum length of the Gantt chart will not exceed 100 intervals.
 - Processes may not appear in the file in order
 - The processID is not related to arrival time, priority, or anything else

The processID's may not always be consecutive numbers e.x. {1,3,6}

- There may be multiple processes with the same arrival time.
 - Break ties by processID
 - processID's are always unique

Other Requirements

- Language: C or Python.
- Must run on department machines.
- Use Canvas to submit a single .tar file named HW4.tar that contains:
 - All files related to the assignment. (Please document your code)
 - A README.txt file containing a description of each file and any information you feel the grader may need.
 - If your code exists in multiple files then there must exist one driver program which runs all the three scheduling algorithms. We execute only one file.

Thank you - Questions?

Acknowledgements

- These slides are based on contributions of current and past CS370 instructors and TAs, including Phil.Sharp ,J. Applin, S. R. Chowdhury, K. Bruhwiler, Y. K. Malaiya, S. Pallickara, K. Drago