



3/29/2020

Academic-Task-3

Operating system



TO
Navneet Kaur

Student Name: Aman Chaturvedi

Student ID: 11801628

Email Address: aman774554@gmail.com

GitHub Username: aman774554

GitHub Link: <https://github.com/aman774554/Academic-Task-3-Operating-System->

Description:

For problem 24:

- This is a scheduling program to implement a Queue with two levels:
- Level 1 : Fixed priority preemptive Scheduling
- Level 2 : Round Robin Scheduling
- For a Fixed priority pre-emptive scheduling if one process P1 is scheduled and running and another process P2 with higher priority comes. The New process with high priority process P2 preempts currently running process P1 and process P1 will go to second level queue. Time for which process will strictly execute must be considered in the multiples of 2.
- All the processes in second level queue will complete their execution according to round robin scheduling.
- In this program Queue 2 will be processed after Queue 1 becomes empty and Priority of Queue 2 has lower priority than in Queue 1.

Algorithm:

- In this program algorithm for round robin scheduling and multilevel queue scheduling is used.

Algorithm For Multilevel Queue:

1. When a process starts executing then it first enters queue 1.
2. In queue 1 process executes for 4 unit and if it completes in this 4 unit or it gives CPU for I/O operation in this 4 unit than the priority of this process does not change and if it again comes in the ready queue than it again starts its execution in Queue 1.

3. If a process in queue 1 does not complete in 4 unit then its priority gets reduced and it shifted to queue 2.
4. Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 unit. In a general case if a process does not complete in a time quantum then it is shifted to the lower priority queue.
5. In the last queue, processes are scheduled in FCFS manner.
6. A process in lower priority queue can only execute only when higher priority queues are empty.
7. A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.

Algorithm for round robin scheduling:

- 1- Create an array **rem_bt[]** to keep track of remaining burst time of processes. This array is initially a copy of **bt[]** (burst times array)
- 2- Create another array **wt[]** to store waiting times of processes. Initialize this array as 0.
- 3- Initialize time : $t = 0$
- 4- Keep traversing the all processes while all processes are not done. Do following for i'th process if it is not done yet.
 - a- If $\text{rem_bt}[i] > \text{quantum}$
 - (i) $t = t + \text{quantum}$
 - (ii) $\text{bt_rem}[i] -= \text{quantum};$
 - c- Else // Last cycle for this process
 - (i) $t = t + \text{bt_rem}[i];$
 - (ii) $\text{wt}[i] = t - \text{bt}[i]$
 - (ii) $\text{bt_rem}[i] = 0;$ // This process is over

Code Snippet :

```
#include<stdio.h>

struct process
{
    int process_name;
    int arrival_time, waiting_time, turn_time, priority, burst_time, burst_timecopy;
} queue1[10], queue2[10];

int main()
{
    struct process temp;

    int
i, time=0, t1, t2, bu_t=0, largest, totalProcess, count=0, k, pf2=0, totalProcess2, n, pos, j, flag=0, y;
    float wait_time=0, turnaround_time= 0, average_waiting_time, average_turnaround_time;
    printf("*****\n");
    printf("*****Lovely Professional University*****\n");
    printf("*****Aman Chaturvedi(11801628)*****\n");
    printf("*****K18CJ*****\n");
    printf("*****\n");
    printf("\n Enter Total Number of Processes:\t");
    scanf("%d", &totalProcess);

    n=totalProcess;
    for(i=0; i<totalProcess; i++)
    {
        printf("-----\n");
        printf("\nEnter Process name:");
        fflush(stdin);
        scanf("%d", &queue1[i].process_name);
        printf("\nEnter Details For processor %d:\n", queue1[i].process_name);
        printf("Enter Arrival Time:");
        fflush(stdin);
        scanf("%d", &queue1[i].arrival_time);
        printf("Enter Burst Time:");
        fflush(stdin);
```

```

scanf("%d",&queue1[i].burst_time);
queue1[i].burst_timecopy=queue1[i].burst_time;
printf("Enter Priority:\t");
fflush(stdin);
scanf("%d",&queue1[i].priority);
}
printf("-----\n");
printf("\nEnter Time Quantum for Fixed priority queue:");
scanf("%d",&t1);
printf("\nEnter Time Quantum for Round Robin queue:");
scanf("%d",&t2);
printf("-----\n");
printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
for(i=0;i<totalProcess;i++)
{
    pos=i;
    for(j=i+1;j<totalProcess;j++)
    {
        if(queue1[j].arrival_time<queue1[pos].arrival_time)
            pos=j;
    }
    temp=queue1[i];
    queue1[i]=queue1[pos];
    queue1[pos]=temp;
}
time=queue1[0].arrival_time;
for(i=0;totalProcess!=0;i++)
{
    while(count!=t1)
    {
        count++;
        if(queue1[i].arrival_time<=time)
        {
            for(j=i+1;j<totalProcess;j++)

```

```

        {
            if(queue1[j].arrival_time==time                                &&
queue1[j].priority<queue1[i].priority)//pr<
            {
                queue2[pf2]=queue1[i];
                pf2++;
                for(k=i; k<totalProcess-1;k++)
                    queue1[k]=queue1[k+1];
                totalProcess--;
                count=0;
                i=j-1;
                j--;
            }
        }
    }
    time++;
    queue1[i].burst_time--;
    if(queue1[i].burst_time==0)
    {
        queue1[i].turn_time=time-queue1[i].arrival_time;
        queue1[i].waiting_time=queue1[i].turn_time-
queue1[i].burst_timecopy;

        printf("%d\t\t%d\t\t%d\n",queue1[i].process_name,queue1[i].turn_time,queue1[i].wa
iting_time);

        wait_time+=time-queue1[i].waiting_time;
        turnaround_time+=time-queue1[i].turn_time;
        for(k=i;k<totalProcess-1;k++)
            queue1[k]=queue1[k+1];i--;
        totalProcess--;
        count=t1;break;
    }
}
count=0;

```

```

        if(queue1[i].burst_time!=0)
        {
            queue2[pf2]=queue1[i];
            pf2++;
            for(k=i;k<totalProcess-1;k++)
                queue1[k]=queue1[k+1];
            totalProcess--;
        }

        if(i==totalProcess-1)
            i=-1;
    }
    totalProcess2=pf2;
    for(count=0;totalProcess2!=0;)
    {
        if(queue2[count].burst_time<=t2&&queue2[count].burst_time>0)
        {
            time+=queue2[count].burst_time;
            queue2[count].burst_time=0;
            flag=1;
        }
        else if(queue2[count].burst_time>0)
        {
            queue2[count].burst_time-=t2;
            time+=t2;
        }
        if(queue2[count].burst_time==0&&flag==1)
        {
            totalProcess2--;
            queue2[count].turn_time=time-queue2[count].arrival_time;
            queue2[count].waiting_time=queue2[count].turn_time-
            queue2[count].burst_timecopy;

            printf("%d\t\t%d\t\t%d\n",queue2[count].process_name,queue2[count].turn_time,que
            ue2[count].waiting_time);

```

```

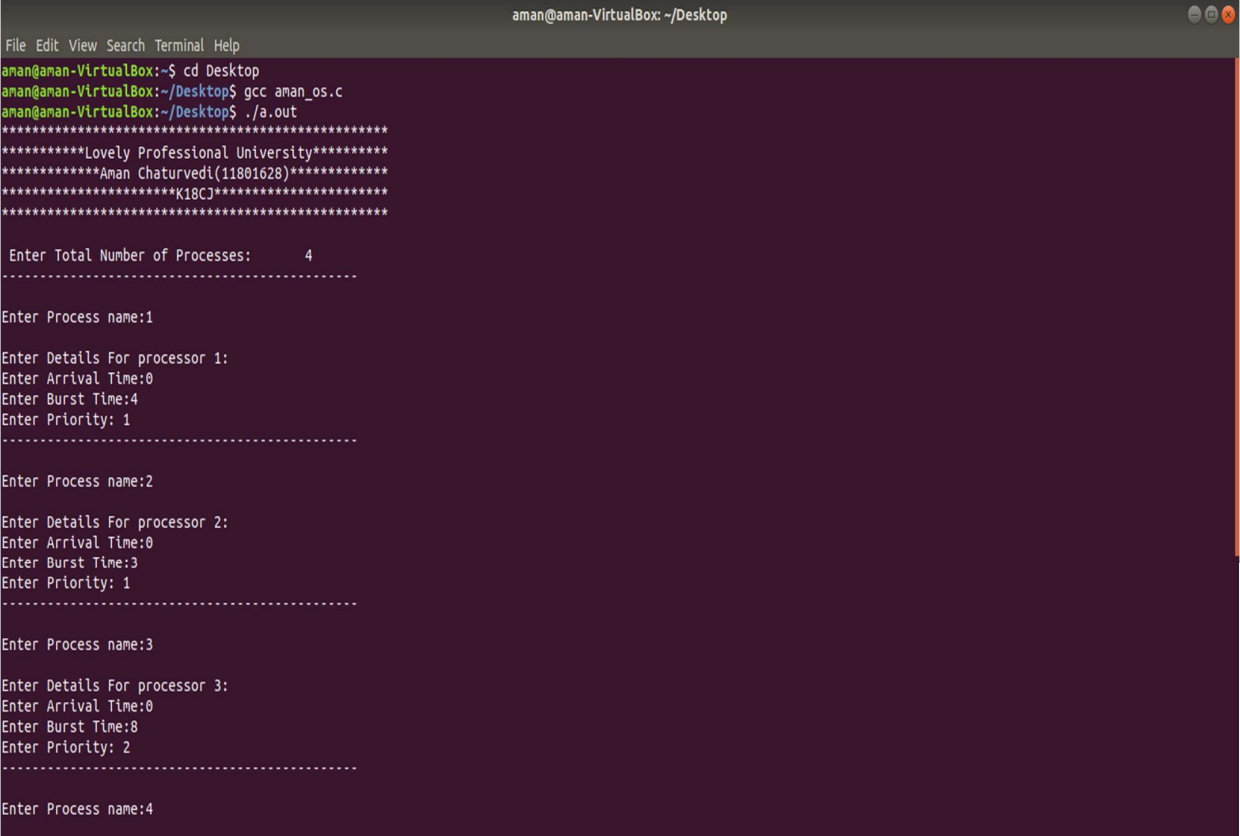
        turnaround_time+=time-queue2[count].arrival_time;
        wait_time+=time-queue2[count].arrival_time-queue2[count].burst_timecopy;
        for(k=count; k<totalProcess2;k++)
            queue2[k]=queue2[k+1];count--;
        flag=0;
    }

    if(count==totalProcess2-1)
        count=0;
    else
        count++;
}

printf("\n Average Waiting Time= %f\n", wait_time/n);
printf("Avg Turnaround Time = %f\n" ,turnaround_time/n);
}

```

Screenshot:-



```

aman@aman-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
aman@aman-VirtualBox:~$ cd Desktop
aman@aman-VirtualBox:~/Desktop$ gcc aman_os.c
aman@aman-VirtualBox:~/Desktop$ ./a.out
*****Lovely Professional University*****
*****Aman Chaturvedi(11801628)*****
*****K18CJ*****
*****

Enter Total Number of Processes:      4
-----

Enter Process name:1
Enter Details For processor 1:
Enter Arrival Time:0
Enter Burst Time:4
Enter Priority: 1
-----

Enter Process name:2
Enter Details For processor 2:
Enter Arrival Time:0
Enter Burst Time:3
Enter Priority: 1
-----

Enter Process name:3
Enter Details For processor 3:
Enter Arrival Time:0
Enter Burst Time:8
Enter Priority: 2
-----

Enter Process name:4

```



```

aman@aman-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
Enter Details For processor 2:
Enter Arrival Time:0
Enter Burst Time:3
Enter Priority: 1
-----
Enter Process name:3

Enter Details For processor 3:
Enter Arrival Time:0
Enter Burst Time:8
Enter Priority: 2
-----
Enter Process name:4

Enter Details For processor 4:
Enter Arrival Time:10
Enter Burst Time:5
Enter Priority: 1
-----
Enter Time Quantum for Fixed priority queue:2
Enter Time Quantum for Round Robin queue:2
-----

Process |Turnaround Time|Waiting Time
1       |      10      |      6
2       |      13      |     10
4       |      8       |      3
3       |      20      |     12

Average Waiting Time= 7.750000
Avg Turnaround Time = 12.750000
aman@aman-VirtualBox:~/Desktop$

```

Complexity: $O(n^3)$

Boundary Conditions:

- | |
|---|
| • Level 1 : Fixed priority preemptive Scheduling |
| • Level 2 : Round Robin Scheduling |
| • Consider: 1. Queue 2 will be processed after Queue 1 becomes empty. |
| • Consider 2. Priority of Queue 2 has lower priority than in Queue 1. |

Test Cases:

- Time Quantum for Fixed priority queue- 2
- Time Quantum for Round Robin queue- 2

Process	Arrival Time	Burst Time	Priority	Turnaround Time	Waiting Time
1	0	4	1	10	6

2	0	3	1	13	10
3	0	8	2	8	3
4	10	5	1	20	12

Process	Arrival Time	Burst Time	Priority	Turnaround Time	Waiting Time
1	0	4	1	9	5
2	1	3	2	15	9
3	2	6	1	17	14
4	4	6	1	15	9

Have you made minimum 5 revisions of solution on GitHub?

✓ Yes

GitHub Link: <https://github.com/aman774554/Academic-Task-3-Operating-System->

aman774554 / Academic-Task-3-Operating-System-
Unwatch 1
Unstar 1
Fork 0

Code
Issues 0
Pull requests 0
Actions
Projects 0
Wiki
Security
Insights
Settings

Design a scheduling program to implements a Queue with two levels: Level 1 : Fixed priority preemptive Scheduling Level 2 : Round Robin Scheduling
Edit

Manage topics

15 commits
1 branch
0 packages
0 releases
1 contributor

Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

aman774554 Add files via upload
Latest commit 1b8bd2b 27 seconds ago

.gitignore Initial commit 17 days ago

README.md Update README.md 8 hours ago

Screenshot from 2020-03-28 19-59-52.png Add files via upload 27 seconds ago

Screenshot from 2020-03-28 19-59-58.png Add files via upload 27 seconds ago

aman_os.c Update aman_os.c 2 minutes ago

README.md

aman_os_project

Design a scheduling program to implements a Queue with two levels: Level 1 : Fixed priority preemptive Scheduling Level 2 : Round Robin Scheduling For a Fixed priority preemptive Scheduling (Queue 1), the Priority 0 is highest priority. If one process P1 is scheduled and running, another process P2 with higher priority comes. The New process (high priority) process P2 preempts currently running process P1 and process P1 will go to second level queue. Time for which process will strictly execute must be considered in the multiples of 2. All the processes in second level queue will complete their execution according to round robin scheduling. Consider: 1. Queue 2 will be processed after Queue 1 becomes empty. 2. Priority of Queue 2 has lower priority than in Queue 1.