A Report for **Preemptive dynamic Scheduling** & its Algorithm

Submitted To
Harshpreet Singh

Submitted by
Aditya Singh Sisodiya
Section K1608
Roll number B48
Reg. No. 11614316
Email adity.maxsteel@gmail.com
Github link
https://github.com/adishadow/OperatingSystem-

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my instructor and faculty for Course Harshpreet Singh , Head of Department Operating System, Lovely Professional University, phagwarh, Punjab for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I express my sincere gratitude to Devender Ma'am , for his stimulating guidance, and continuous encouragement throughout the course.

Lastly, I thank almighty, my parents, brother, sisters and friends for their constant encouragement without which this assignment would not be possible.

Content

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Preemptive Dynamic Priority Scheduling

Dynamic priority scheduling is a type of scheduling algorithm in which the priorities are calculated during the execution of the processes. The goal of dynamic priority scheduling is to adapt to dynamically changing progress and form an optimal configuration in self-sustained manner.

(For the given scenario) When a new process enters the Ready Queue its initial priority is set to '0'. When the execution starts, at any instant t, the process available with highest priority in the Queue starts to execute, rest waits in Waiting Queue. During execution the priority of all the process waiting increase by 2, while for process executing, priority increases by 1. After executing for t_1 sec, the Scheduler check for, whether any process sitting in waiting queue have priority equal or greater the priority of currently running process, if so the process running is put to wait state and the process with higher priority starts to execute and so on , till all the process executes.

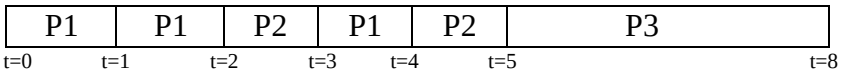
The above scenario can be well understand the following example

Table 1, showing the arrival and execution time of process

Process	Arrival Time(in sec)	Burst Time(in sec)
P1	0	3
P2	1	2
P3	2	3

,

Gantt Chart for process execution.



, showing the status after execution of all the process

Process	Average Wait Time	Final Priority
P1	1 sec	5
P2	2 sec	6
P3	2 sec	9

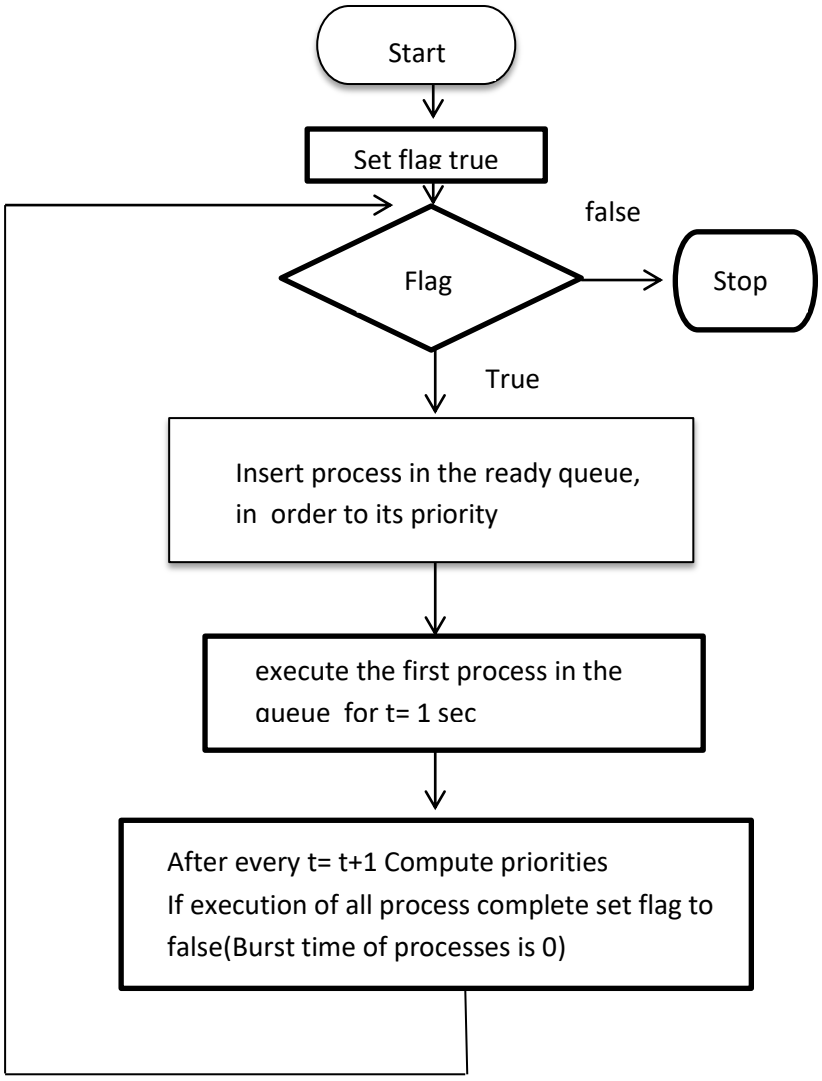
Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Algorithm

Data Structure :-

- *Process:* A structure to hold various attribute of Process, like Burst Time, Priority, average waiting time, current status (0 if not currently running else 1) along with the name of the process.
- *Queue:* A data Structure, consisting of Process and link to next process, use to represent the Ready Queue/Waiting Queue.
- *Sequence :*Another Queue type data structure to hold the sequence of the execution
- *Execution Time* : An integer type to hold the value of total time process take to complete
- Flag Variables like CheckFlag etc, to lay down the condition

Flow Chart :-



Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Algorithm

Algorithm :-

1. Set Flag=true
2. While(Flag)
3. Construct the ready Queue, following the property Priority Queue i.e. the Process with highest Priority value inserting first in queue followed by process of lower priority.
4. After generation of ready Queue, the process in front/first(at the beginning of Queue)is selected(Since first element has highest priority).
5. Execution of process take for time t (= 1 sec),during this time the priority for all the process change and for the process currently executing status is set 1. (The Priority of the process changes as per the scenario given).
   If status of process is 1
   Increase priority by 1
   Else
   increase priority by 2
6. If burst time for all the process is zero then set flag to FALSE
7. Else Flag is true
8. The average waiting time is calculate with the help of all the process, how long They have to wait to complete its execution

Code Snippet

Code Snippet, is the small region of re-usable/re-locatable source code. Since the code is designed for the special purpose and the given code is constructed without the help of any pre-process provided by the IDE. Hence the given program code consist no Code Snippet part.

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Additional Algorithm

For inserting the element in the Queue, in addition with h previous algorithm mention, algorithm of insertion in priority queue is followed. The algorithm is as mentioned below

// head is the beginning of the queue
// np is the new element coming, that is to be inserted in the queue
// temp is used as an temporary variable

Priority Queue

```
1.  if(Head =NULL)
2.      Head=np
3.  else
4.     temp=Head
5.     while(temp->next!=NULL and temp->priority > np->priority
6.         prev = temp
7.         temp = temp->next
8.  if(prev=NULL) then,
9.        np->next=Head
10.      Head=np
11. else
12.      prev->next=np
13.     np->next=temp
```

With the help of the algorithm mentioned above the process with their respective priority is inserted in the queue waiting for their execution.

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Complexity

The overall complexity for program code as per the given algorithm is as follows

- Loop for taking input of n user process(in main)

```
for(i=0;i<userProcess;i++)
{   ...
}
```
} n

- Loop to display the final status of the code (in main)

```
for(i=0;i<userProcess;i++)
{   ...
}
```
} n

- Loop require to Calculate the average waiting time (in main)

```
for(i=0;i<userProcess;i++)
{  ...
}
```
} n

- For Generating the Queue and processing it(Simulation function)

```
while(flagCheck)
    {       ...
            for(i=0;i<userProcess;i++)
            {       ...
                    while(temp->next!=NULL )
                    {   ...
                      }
                  while(temp!=NULL)
                  { ...
                    }
                 for(i=0;i<userProcess;i++)
                 {
                    }
     }
```
} m*(n*(n+n))

Hence, the final complexity is given as

$O(n+n+n+m(n*(n+n)))$

$=O(3n+2mn^2)$

$=O(mn^2)$

Hence The Overall complexity of the code is $O(mn^2)$, where m is the total burst time of all the process in the Queue(or net execution time) and n is the number of process used.

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Boundary Conditions

The various Boundary condition, for dynamic priority scheduling, are as described.,

- When the priority of two process is/become equal
  In this situation, when the priority of two process becomes equal, can arise if two process arrives at same time or while executing. In case one of the process was in execution, then the other process is given a chance to execute. If none was in execution then on the basis of FCFS the process is inserted in Queue.

- When two or more processes arrives at the same time
  If this condition arises in the beginning of the process execution, then since priority of all the process is same, hence the first in the list is given preference. If the scenario is in between in the execution then the process in this condition wait and once the priority becomes greater than the currently executing process the first in the lit given preference.

NOTE :-

The above boundary condition mentioned are well handle, in the given solution program Code for proper and expected execution of the program.

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Sample Test Cases

The various Test Case use to test the given Simulation code and to verify it correctness are as mentioned below

Input

- The First line of input consist of N (1<N<10), which is the Number of Process used.
- The Next N line Consist of the input for Process Name, its arrival time and the process's burst time.

Output

The first N line of Output tells about, the final status for all the process related to it priority, waiting time, etc.,

The Last Line of the output Tells the Average Waiting for the all the Process.

Explanation

The Processes are inserted into the Ready Queue Once they arrive, i.e. as the Arrival Time becomes greater than 0. All the Process are inserted in the queue with the help of implementation of the Priority Queue. Once inserted The First element or Head is chosen from the Queue(since as per the priority queue the first element is one with the highest priority), and hence status change to 1 and burst time reduce by 1. This is repeated until all the processes burst time doesn't turn to zero. The priority of the process changes as per the condition given in scenario and waiting time updated once in the queue waiting. With the help of waiting of the processes the Average is calculated

Test Case 1 :-

Input :-

Number of Process 3

Process Name P1 Arrival Time 0 Burst Time 3

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Process Name P2 Arrival Time 1 Burst Time 2
Process Name P3 Arrival Time 2 Burst Time 3

Output :-

Process Name ***P1*** with Arrival Time ***0*** and Wait time ***1*** and Priority ***5***
Process Name ***P2*** with Arrival Time ***1*** and Wait time ***2*** and Priority ***6***
Process Name ***P3*** with Arrival Time ***2*** and Wait time ***3*** and Priority ***9***
P1->P1->P2->P1->P2->P2->P3->P3->P3
Total execution time require :- 8 sec
Average Wait Time :- 2 sec

Test Case 2 :-

Input :-

Number of Process 3
Process Name P1 Arrival Time 0 Burst Time 4
Process Name P2 Arrival Time 1 Burst Time 1
Process Name P3 Arrival Time 2 Burst Time 2
Process Name P4 Arrival Time 3 Burst Time 1

Output :-

Process Name ***P1*** with Arrival Time ***0*** and Wait time ***1*** and Priority ***6***
Process Name ***P2*** with Arrival Time ***1*** and Wait time ***1*** and Priority ***3***
Process Name ***P3*** with Arrival Time ***2*** and Wait time ***3*** and Priority ***8***
Process Name ***P3*** with Arrival Time ***2*** and Wait time ***4*** and Priority ***9***
P1->P1->P2->P1->P1->P3->P3->P4
Total execution time require :- 8 sec
Average Wait Time :- 1 sec

Test Case 3 :-

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Input :-

Number of Process 3
Process Name P1 Arrival Time 0 Burst Time 4
Process Name P2 Arrival Time 0 Burst Time 1
Process Name P4 Arrival Time 1 Burst Time 1

Output :-

Process Name ***P1*** with Arrival Time ***0*** and Wait time ***1*** and Priority ***6***
Process Name ***P2*** with Arrival Time ***1*** and Wait time ***1*** and Priority ***3***
Process Name ***P3*** with Arrival Time ***2*** and Wait time ***3*** and Priority ***7***
P1->P2->P1->P1->P1->P3->
Total execution time require :- 6 sec
Average Wait Time :- 1 sec

Hence, from the above test cases we can conclude that,
All the test case provided, generates the required and expected result for every input given and helps in verification of correctness of the program code.

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

Solution Code

```
#include<stdio.h>
#include<conio.h>

struct process
{
	int arrivalTime;
	int burstTime;
	int priority;
	char processName[3];
	int averageTime;
	int status; // 0-waiting, 1-running
};

struct queue{
	struct process *p;
	struct queue *next;
}*readyQueue=NULL,*temp,*prev=NULL,*Sequence=NULL;

struct process* createProcess(int userProcess);
void Stimulate(struct process *p,int userProcess,int executionTime);
void CreateSequence(struct process *np);

 main()
{
	int userProcess;
	int i,sum=0;
	struct process *p;
	printf("Enter the number of the process\t");
	scanf("%d",&userProcess);
	p=createProcess(userProcess);
	printf("\nThe Initial condition set are \n");
	for(i=0;i<userProcess;i++)
	{
		printf("\nProcess Name  %s",(p+i)->processName);
		printf("\t with Arrival Time  %d",(p+i)->arrivalTime);
		printf("\tand Burst Time %d",(p+i)->burstTime);
	}
	Stimulate(p,userProcess,0);
	for(i=0;i<userProcess;i++)
	{
		sum+=(p+1)->averageTime;
	}
```

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

```
        printf("Average Waiting Time = %d",(sum/userProcess));
                getch();

}

//Function  to input the n process with its arrival time and burst time

struct process* createProcess(int userProcess)
{
        struct process *p;
        int i;
        p=calloc(userProcess,(sizeof(struct process)));
        for(i=0;i<userProcess;i++)
        {
                printf("Enter the process name\t");
                scanf("%s",(p+i)->processName);
                printf("Enter the Arrivl time of the Process\t");
                scanf("%d",&(p+i)->arrivalTime);
                printf("Enter the burst time\t");
                scanf("%d",&(p+i)->burstTime);
                (p+i)->priority=0;
                (p+i)->averageTime=0;
                (p+i)->status=0;
        }
        return p;
}

//Simulation function



void Stimulate(struct process *pr,int userProcess,int executionTime)
{

        struct process *runingProcess=NULL;
        struct queue *ready,*temp2;
        int flagCheck=1,flag=0;
        Sequence=NULL;
        int i;
                while(flagCheck)
                {
                        flag=0;
                        readyQueue=NULL;

                        for(i=0;i<userProcess;i++)
                        {

                                if(((pr+i)->arrivalTime-executionTime)<=0 && (pr+i)->burstTime>0)
                                {
                                        ready=malloc(sizeof(struct queue));
                                        ready->p = (pr+i);
                                        ready->next=NULL;
```

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

```
if(readyQueue==NULL)
{
        readyQueue=ready;

}
else
{

        temp=readyQueue;
        while(temp->next!=NULL )
        {
                if(ready->p->priority == temp->p->priority)
                {
                        if(temp->p->status==1)
                        {
                                break;
                        }
                        else
                        {
                                prev=temp;
                                temp=temp->next;
                        }


                 }
                else if( ready->p->priority < temp->p->priority )
                {
                prev=temp;
                temp=temp->next;
                }
                else
                {
                        break;
                }
        }
        if(prev==NULL )
        {
                if(( ready->p->priority==temp->p->priority ))
                {
                        if(temp->p->status==1)
                        {

                                ready->next=readyQueue;
                                readyQueue=ready;
                        }else
                        {
                                temp->next=ready;
                        }
                }else if(ready->p->priority<temp->p->priority)
                {
                        temp->next=ready;
                }
                else if(ready->p->priority>temp->p->priority)
```

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

```
                        {
                                ready->next=readyQueue;
                                readyQueue=ready;

                        }

                } else
                if(temp->next==NULL)
                {
                        if( ready->p->priority <= temp->p->priority )
                        temp->next=ready;
                }
                else
                {
                        ready->next=temp;
                        prev->next=ready;
                }

        }
}}


        runingProcess=readyQueue->p;
        CreateSequence(runingProcess);
        readyQueue->p->burstTime-=1;
        readyQueue->p->status=1;
        readyQueue->p->priority+=1;
  temp=readyQueue;
        while(temp!=NULL)
        {
                if(temp->p == runingProcess)  //changing here
                {

                }
           else
                {
                        temp->p->priority+=2;
                        temp->p->status=0;
                        temp->p->averageTime+=1;
                }
                temp=temp->next;
        }

        for(i=0;i<userProcess;i++)
        {
                if((pr+i)->burstTime!=0)
                {
                        flag=1;
                        break;
                }
        }
        if(flag==1)
        {
```

Name :Aditya Singh Sisodiya
Registration No. :11614316
Email : aditya.maxsteel@gmail.com
GitHub link:
https://github.com/adishadow/OperatingSystem-

```
                                        executionTime++;
                                }
                                else
                                {
                                        flagCheck=0;
                                }
                        }

        printf("\nThe final status for all the Process are :- \n");
        for(i=0;i<userProcess;i++)
        {
                printf("\nProcess Name  %s",(pr+i)->processName);
           printf("\t with Arrival Time  %d",(pr+i)->arrivalTime);
           printf("  and  Wait Time %d",(pr+i)->averageTime);
                printf("  and priority %d",(pr+i)->priority);
        }


        printf("\nThe Sequnce for Execution is given as \n\t\t");

        temp=Sequence;
        while(temp!=NULL)
        {
                printf("%s  - > ",(temp->p->processName));
                temp=temp->next;
        }
        printf("\nTotal execution Time Require :- %d\n",executionTime+1);


}



void CreateSequence(struct process *np)
{
        struct queue *temp,*temp2;
        temp=malloc(sizeof(struct queue));
        temp->p=np;
        temp->next=NULL;
        if(Sequence==NULL)
        {
                Sequence=temp;
        }
        else
        {
                temp2=Sequence;
                while(temp2->next!=NULL)
                {
                        temp2=temp2->next;
                }
                temp2->next=temp;
        }
}
```