

**Name : Shivam Pandey**

**Registration No: 11602093**

**Email:**

**[shivamguys@gmail.com](mailto:shivamguys@gmail.com)**

**Github Link:**

**[github.com/shivamguys/prioritypreemptive](https://github.com/shivamguys/prioritypreemptive)**

## “Description”

**“ A Scheduling Algorithm That Takes The Burst Time Of The Processes, and Assigns The Priority To That Process Having The Lowest Burst Time Is Given The Highest Priority ”**

- In Priority Preemptive scheduling we have to schedule processes in such a way that process having higher priority is preempted with process having lower priority, and it should keep checking which process has the rights to execute at that particular point of time.
- The Priority of process is calculated based on the burst time that is higher the burst time lower will be the priority, a keen focus is given to the bursttime.
- A process having short duration of bursttime will be having the highest priority, it's normal that a process requires less time to complete its task, therefore the priority should also be high.

## “Algorithms”

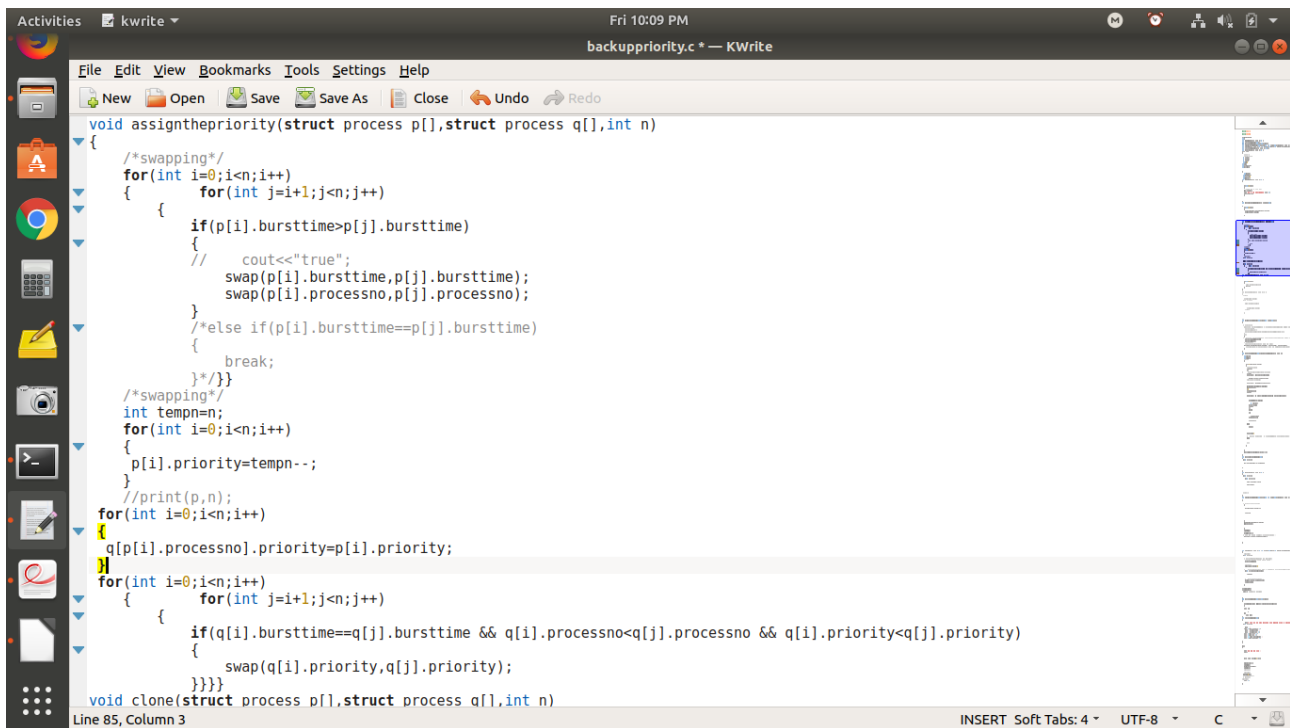
**For Calculating Priority:**

1. The inputs are on the basis of bursttime, we are calculation the priority for the process by considering its bursttime.
2. We will perform a sorting operation on the processes to sort all the process in the increasing order based on the bursttime.
3. Take a variable priority=no of process available;
4. We will assign the priority to each process after performing the sorting that is process 1 will have priority=no of process -- keep on decreasing the variable for the next process.
5. Next we will perform a check whether we have assigned the right priority to the process or not, the reason is that the sorting technique does not guarantee the order it arrived so there might be the case that two process having the same bursttime gets wrong priority.

**So we will apply a check as follow:**

```
if(q[i].bursttime==q[j].bursttime && q[i].processno<q[j].processno &&
q[i].priority<q[j].priority)
swap(q[i].priority,q[j].priority);
```

**We have stored process no in the form of index to achieve a complexity of  $O(1)$ , or else it would have taken time to perform searching.**



```
void assignthepriority(struct process p[],struct process q[],int n)
{
    /*swapping*/
    for(int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(p[i].bursttime>p[j].bursttime)
            {
                cout<<"true";
                swap(p[i].bursttime,p[j].bursttime);
                swap(p[i].processno,p[j].processno);
            }
            /*else if(p[i].bursttime==p[j].bursttime)
            {
                break;
            }*/
        }
    }
    /*swapping*/
    int tempn=n;
    for(int i=0;i<n;i++)
    {
        p[i].priority=tempn--;
    }
    //print(p,n);
    for(int i=0;i<n;i++)
    {
        q[p[i].processno].priority=p[i].priority;
    }
    for(int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(q[i].bursttime==q[j].bursttime && q[i].processno<q[j].processno && q[i].priority<q[j].priority)
            {
                swap(q[i].priority,q[j].priority);
            }
        }
    }
}

void clone(struct process p[],struct process q[],int n)
```

## For Scheduling:

- 1) We are running the process till the maximum time ,as our job is to check at each interval of time wether a process at that moment of time has a higher priority than our running process,
- 2) If their is preemption then we will transfer the control from the current process to the preempted process and will save the state and time for that particular process
- 3) After the completion of the process till the last process which arrived to us, after that we will start executing the process based on its priority and its order.

```

void execute(struct process p[],int currently,struct process q[],struct process t[])
{
    int i,alltime;
    int tempburst;
    //while(1)
    {
        for(i=0;i<p[currently].bursttime;)
        {
            //totaltime>=currently
            ++totaltime;
            i++;
            p[currently].bursttime=p[currently].bursttime-i;
            ++alltime;
            if(totaltime<=n-1 && i>=p[currently].bursttime)
            {
                decrement(p,currently,previous,totaltime);
                //p[currently].end=totaltime;
            }
            if(totaltime<=n-1 && check(p,totaltime,currently))
            {
                decrement(p,currently,previous,totaltime);
                previous=currently;
                i=0;
                currently=totaltime;
                continue;
            }
            if(totaltime<=n-1 && !(check(p,totaltime,currently)) && p[currently].bursttime==0)
            {
                if(check(p,previous,totaltime))
                {
                    int t=currently;

```

## For Saving The State:

### 1. Assign the current process start to previous process

```

void decrement(struct process p[],int currently,int previous,int totalp)
{
    /* if(currently==1)
    {cout<<"values of currently and previous are"<<currently<<previous<<"and value of totalp is"<<totalp<<"\n";
    p[currently].start=0;
    p[currently].end=totalp-1;
    p[currently].bursttime=p[currently].bursttime-(p[currently].end-p[currently].start);
    }
    else*/
    {
        cout<<"values of currently and previous are"<<currently<<previous<<"and value of totalp is"<<totalp<<"\n";
        p[currently].start=p[previous].end;
        p[currently].end=totalp;
        // previous=totalp;
        //cout<<p[currently].bursttime<<"this is the burst time\n";
        p[currently].bursttime=(p[currently].bursttime)-( p[currently].end - p[currently].start);
        // cout<<p[currently].end-p[currently].start<<"this is minus of end start for"<<p[currently].processno<<"\n";
    }
}

```

end, and current process end to time at which it is being preempted.

### 2. Also decrement the bursttime for that process as the it will took depends on how many second it had the

control and we can easily calculate that as process  
endtime – starttime

**p[currently].bursttime=(p[currently].bursttime)-  
( p[currently].end - p[currently].start);**

### **For Checking Preemption:**

1. Take a variable as totaltime, and increment it for each second of bursttime and keep calling chek() function to check wether at that particular totaltime we do or don't have a process having higher priority or not.

**bool check(struct process p[],int total,int currently)**

```
{ if(p[total].bursttime!=0 &&  
p[total].priority>p[currently].priority) {  
    return true;  
    } else  
    return false;
```

## **“Complexities”**

- A)      Sorting the process based on the bursttime for assigning the priority will take  **$O(n^2)$**  as we have used bubble sort technique to sort the process.

- B) Searching for the process while transferring the state is easy as it takes  $O(1)$  because making index numbers as process number will simplify our task.
- C) Scheduling will take  $O(\text{the late arriving process})$ , as after that we don't have to perform a check we just have to execute the process which are left for execution based on their priorities.
- D) Left Over process will take up to the remaining bursttime they are possessing with them, that is  $O(\text{Left over bursttime})$
- E) Calculating WaitingTime and TurnAround time will take  $O(N)$  as we will traverse all the process linearly to calculate the waiting and turnaround time respectively.

**Overall Complexity :** Considering all the statements above it will take  $O(n^2 + 1 + \text{the late arriving process} + \text{left over bursttime})$ .

**If we will use some more efficient sorting technique the complexity will range in terms of  $O(N)$ .**

## “Constraints”

- ➔ Our algorithm is having the support of **backtracking** as well because when a process finishes its task, the algorithm should resume the previous process it was executing and should transfer the control to that process.

→ Let there be following processes:

→ **Example 1**

→ **Process No      BurstTime      AssumedPriority**

**P1                      9                      2**

**P2                      6                      5**

**P3                      6                      4**

**P4                      1                      6**

**P5                      9                      1**

**P6                      6                      3**

P1	P2	P4	P2	P3	P6	P1	P5
0	1	3	4	8	14	20	28 37

→ So in this case as process P4 will get complete , it should transfer the control to process P2 and further on and on.

```
Activities Terminal Thu 1:41 AM shivam@shivamguys: ~
File Edit View Search Terminal Help
Enter total no of process
AC
shivam@shivamguys:~$ g++ backuppriority.c
shivam@shivamguys:~$ ./a.out
Enter total no of process
6
Enter the burst time for Process no 0
9
Enter the burst time for Process no 1
6
Enter the burst time for Process no 2
6
Enter the burst time for Process no 3
1
Enter the burst time for Process no 4
9
Enter the burst time for Process no 5
6
Process no -> 0
Burst time -> 9
Priority is -> 2
Process Start time is -> 20
Process End Time Is -> 28
Turn Around time is -> 28
Waiting Time is -> 19
Process no -> 1
Burst time -> 6
Priority is -> 5
Process Start time is -> 4
Process End Time Is -> 8
Turn Around time is -> 7
Waiting Time is -> 1
Process no -> 2
```



```
Activities Terminal Thu 1:41 AM shivam@shivamguys: ~
File Edit View Search Terminal Help
Waiting Time is -> 1

Process no -> 2
Burst time -> 6
Priority is -> 4
Process Start time is -> 8
Process End Time Is -> 14
Turn Around time is -> 12
Waiting Time is -> 6

Process no -> 3
Burst time -> 1
Priority is -> 6
Process Start time is -> 3
Process End Time Is -> 4
Turn Around time is -> 1
Waiting Time is -> 0

Process no -> 4
Burst time -> 9
Priority is -> 1
Process Start time is -> 28
Process End Time Is -> 37
Turn Around time is -> 33
Waiting Time is -> 24

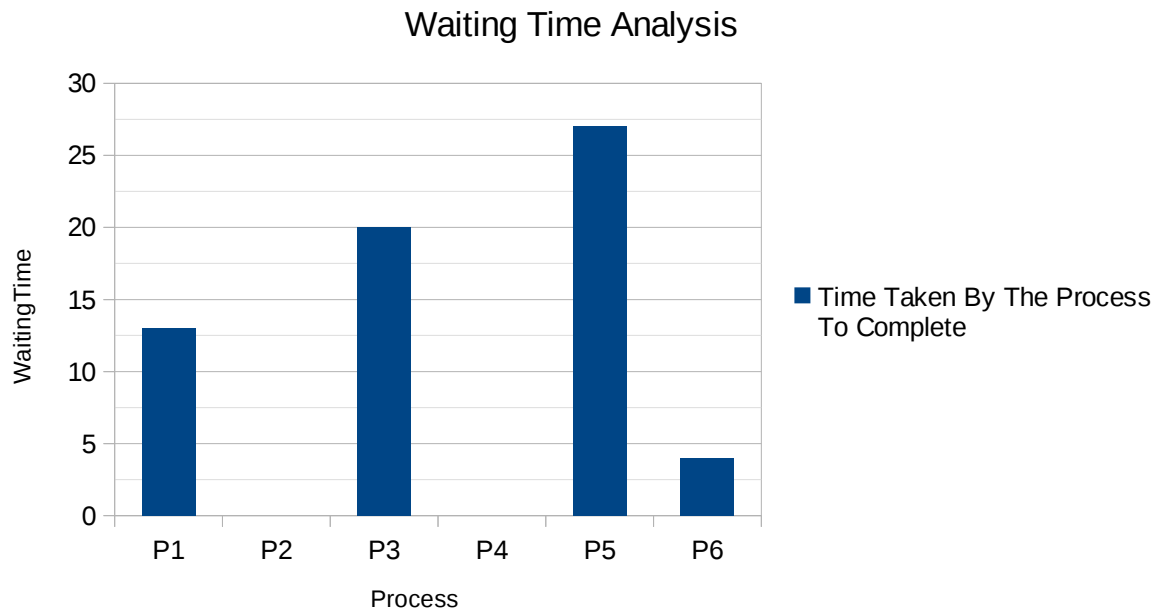
Process no -> 5
Burst time -> 6
Priority is -> 3
Process Start time is -> 14
Process End Time Is -> 20
Turn Around time is -> 15
Waiting Time is -> 9
shivam@shivamguys:~$
```

➔ (Updated Display Of Outputs)

```
Activities Terminal Fri 10:59 PM shivam@shivamguys: ~
File Edit View Search Terminal Help
(evince:9188): Gtk-WARNING **: Attempting to read the recently used resources file at '/home/shivam/.local/share/recently-used.xbel', but the pa
ailed: Failed to open file "/home/shivam/.local/share/recently-used.xbel": Permission denied.
^C
shivam@shivamguys:~$ ./a.out
Enter total no of process
6
Enter the burst time for Process no 0
9
Enter the burst time for Process no 1
6
Enter the burst time for Process no 2
6
Enter the burst time for Process no 3
1
Enter the burst time for Process no 4
9
Enter the burst time for Process no 5
6
|| Process no || Burst time || Priority || Process Start time ||Process End Time||Turn Aroundtime||Waiting Time

1          9          2          20          28          28          19
2          6          5          4          8          7          1
3          6          4          8          14          12          6
4          1          6          3          4          1          0
5          9          1          28          37          33          24
6          6          3          14          20          15          9
shivam@shivamguys:~$
(evince:9188): Gtk-WARNING **: Attempting to read the recently used resources file at '/home/shivam/.local/share/recently-used.xbel', but the pa
ailed: Failed to open file "/home/shivam/.local/share/recently-used.xbel": Permission denied.
```

## ✓ Analysis Of Example 1



**As We can see that the process which will have the highest priority will be completed in 0seconds that is p4, no matter p2 arrived at 2 second but it also get executed within notime.**

➔	Process No	BurstTime	AssumedPriority
	P1	8	1
	P2	1	6
	P3	6	3
	P4	7	2
	P5	1	5
	P6	2	4

P1	P2	P3	P5	P6	P3	P4	P1	
0	1	2	4	5	7	11	18	25

```
Activities Terminal
Fri 11:10 PM
shivam@shivamguys: ~

File Edit View Search Terminal Help
^[[C
|| Process no || Burst time || Priority || Process Start time ||Process End Time||Turn Aroundtime||Waiting Time

shivam@shivamguys:~$ ./a.out
Enter total no of process
6
Enter the burst time for Process no 0
8
Enter the burst time for Process no 1
1
Enter the burst time for Process no 2
6
Enter the burst time for Process no 3
7
Enter the burst time for Process no 4
1
Enter the burst time for Process no 5
2
|| Process no || Burst time || Priority || Process Start time ||Process End Time||Turn Aroundtime||Waiting Time

1 8 1 18 25 25 17
2 1 6 1 2 1 0
3 6 3 7 11 9 3
4 7 2 11 18 15 8
5 1 5 4 5 1 0
6 2 4 6 7 2 0

shivam@shivamguys:~$
(evince:9188): Gtk-WARNING **: Attempting to read the recently used resources file at '/home/shivam/.local/share/recently-used.xbel', but the pa
ailed: Failed to open file "/home/shivam/.local/share/recently-used.xbel": Permission denied.
```