CSE2005 OS – LAB 01

Advait Deochakke
20BCE1143

FCFS w/o Arrival, FCFS with Arrival, SJF(non-primitive), SRTF(primitive)

→ FCFS w/o Arrival

```
#include<iostream>

using namespace std;

struct process
{
  int id=0;
  int wait=0;
  int burst=0;
  int turnaround=0;
};

int main()
{
  cout<<"Advait Deochakke\n20BCE1143\nEnter number of
processes for FCFs: ";
  int n;
  int ttlwait=0, ttlturnaround=0;
  cin>>n;
  process arr[n];
  for(int i=0; i<n; i++)
  {
    cout<<"\nEnter PiD: ";
    cin>>arr[i].id;
    cout<<"\nEnter Burst time: ";
```

```
    cin>>arr[i].burst;
  }
  arr[0].wait=0;
  arr[0].turnaround=arr[0].burst;

  for(int i=1; i<n; i++)
  {
    for(int j=0; j<i; j++)
    {
      arr[i].wait=arr[i].wait+arr[j].burst;
    }
    arr[i].turnaround=arr[i].wait+arr[i].burst;
    ttlturnaround=ttlturnaround+arr[i].turnaround;
    ttlwait=ttlwait+arr[i].wait;

  }

  int avgturnaround, avgwait;
  avgturnaround=ttlturnaround/n;
  avgwait=avgturnaround/n;
  cout<<"Avg waiting time is : "<<avgwait<<endl;
  cout<<"Avg turnaround is : "<<avgturnaround<<endl;

  cout<<"\n\nStats for each process: \n";
  for(int i=0; i<n; i++)
  {
    cout<<"PiD: "<<arr[i].id<<", PWait: "<<arr[i].wait<<",
PBurst: "<<arr[i].burst<<", PTurn: "<<arr[i].turnaround<<endl;
  }

  return 0;
}
```

```
advait@advait-VirtualBox: ~/Desktop/CSE2005/LAB03
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$ g++ fcfs.cpp
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$ ./a.out
Advait Deochakke
20BCE1143
Enter number of processes for FCFs: 3

Enter PiD: 1

Enter Burst time: 2

Enter PiD: 2

Enter Burst time: 4

Enter PiD: 3

Enter Burst time: 5
Avg waiting time is : 1
Avg turnaround is : 5


Stats for each process:
PiD: 1, PWait: 0, PBurst: 2, PTurn: 2
PiD: 2, PWait: 2, PBurst: 4, PTurn: 6
PiD: 3, PWait: 6, PBurst: 5, PTurn: 11
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$
```

<next page>

→ FCFS with Arrival

```
#include<iostream>
#include<algorithm>

using namespace std;

struct process
{
  int id=0;
  int wait=0;
  int burst=0;
  int turnaround=0;
  int arrival=0;
};

bool compareArrival(process p1, process p2)
{
  return (p1.arrival<p2.arrival);
}

int main()
{
  cout<<"Advait Deochakke\n20BCE1143\nEnter number of
processes for FCFs: ";
  int n;
  int ttlwait=0, ttlturnaround=0;
  cin>>n;
  process arr[n];
  for(int i=0; i<n; i++)
  {
    cout<<"\nEnter PiD: ";
    cin>>arr[i].id;
    cout<<"\nEnter Arrival time: ";
    cin>>arr[i].arrival;
    cout<<"\nEnter Burst time: ";
    cin>>arr[i].burst;
  }
  sort(arr, arr+n, compareArrival);
  arr[0].wait=0;
  arr[0].turnaround=arr[0].burst;

  for(int i=1; i<n; i++)
  {
    for(int j=0; j<i; j++)
```

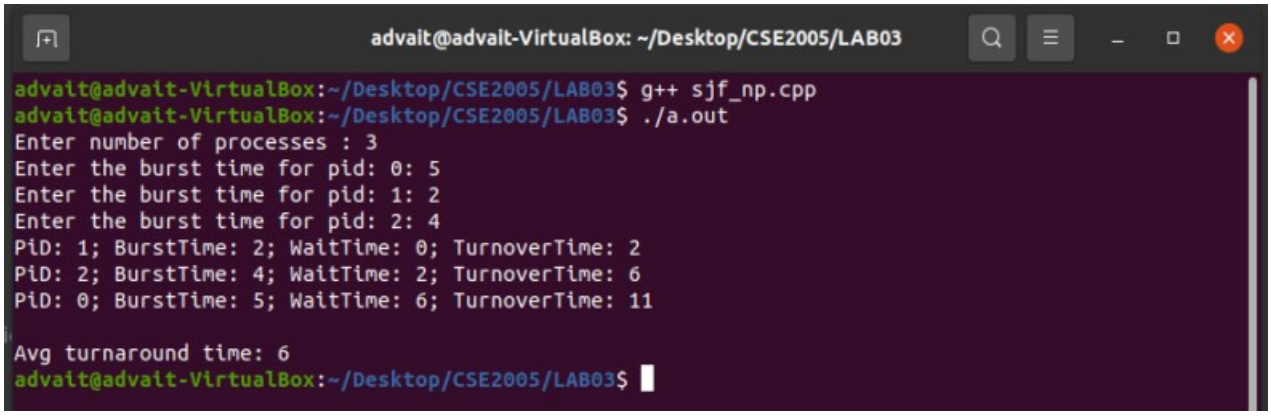
```
    {
      arr[i].wait=arr[i].wait+arr[j].burst;
    }
    arr[i].turnaround=arr[i].wait+arr[i].burst;
    ttlturnaround=ttlturnaround+arr[i].turnaround;
    ttlwait=ttlwait+arr[i].wait;
  }

  int avgturnaround, avgwait;
  avgturnaround=ttlturnaround/n;
  avgwait=avgturnaround/n;
  cout<<"Avg waiting time is : "<<avgwait<<endl;
  cout<<"Avg turnaround is : "<<avgturnaround<<endl;

  cout<<"\n\nStats for each process: \n";
  for(int i=0; i<n; i++)
  {
    cout<<"PiD: "<<arr[i].id<<", PArrival:
"<<arr[i].arrival<<", PWait: "<<arr[i].wait<<", PBurst:
"<<arr[i].burst<<", PTurn: "<<arr[i].turnaround<<endl;
  }
  return 0;
}
```

```
advait@advait-VirtualBox: ~/Desktop/CSE2005/LAB03
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$ g++ fcfs_arrival.cpp
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$ ./a.out
Advait Deochakke
20BCE1143
Enter number of processes for FCFs: 3

Enter PiD: 1

Enter Arrival time: 0

Enter Burst time: 2

Enter PiD: 2

Enter Arrival time: 1

Enter Burst time: 3

Enter PiD: 3

Enter Arrival time: 6

Enter Burst time: 2
Avg waiting time is : 1
Avg turnaround is : 4


Stats for each process:
PiD: 1, PArrival: 0, PWait: 0, PBurst: 2, PTurn: 2
PiD: 2, PArrival: 1, PWait: 2, PBurst: 3, PTurn: 5
PiD: 3, PArrival: 6, PWait: 5, PBurst: 2, PTurn: 7
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$
```

→ SJF (non primitive)

```
#include<iostream>
#include<algorithm>

using namespace std;

struct process
{
int pid;
int btime;
int waitime;
int turnover;
};

bool compareBurst(process p1, process p2)
{
  return(p1.btime < p2.btime);
}

int main()
{
  cout<<"Enter number of processes : ";
  int no;
  cin>>no;
  process arr[no];

  for(int i=0; i<no; i++)
  {
    cout<<"Enter the burst time for pid: "<<i<<": ";
    arr[i].pid=i;
    cin>>arr[i].btime;
  }

  sort(arr, arr+no, compareBurst);
  int w=0;
  int t=0;
  for(int i=0; i<no; i++)
  {
    arr[i].waitime=w;
    w=w+arr[i].btime;
    arr[i].turnover=arr[i].waitime+arr[i].btime;
    t=t+arr[i].turnover;
```

```
    }

    for(int i=0; i<no; i++)
    {
        cout<<"PiD: "<<arr[i].pid<<"; BurstTime: 
"<<arr[i].btime<<"; WaitTime: "<<arr[i].waitime<<"; 
TurnoverTime: "<<arr[i].turnover; 
        cout<<endl;
    }

    cout<<endl;
    cout<<"Avg turnaround time: "<<t/no<<endl;

    return 0;
}
```



<without arrival, as given in lab manual. With arrival is also possible, as shown in next program>

<next page>

→ SRTF (Primitive)

```
#include<iostream>
#include<vector>
#include<algorithm>


using namespace std;

struct process
{
  int pid;
  int arrtime;
  int burstime;
  int remaintime;
  int wtime=999;
  int turnaroundtime;
  int exitime;

};

bool compareArr(process p1, process p2)
{
  return(p1.arrtime > p2.arrtime);
}

bool compareRem(process p1, process p2)
{
  return(p1.remaintime > p2.remaintime);
}


int main()
{
  int n;
  int avgturnaround=0, avgwait=0;
  cout<<"Enter no. of processes: ";
  cin>>n;
  int curtime=0, ttlexec=0;
  vector<process> notinqueue, inqueue, completed;

  for(int i=1; i<n+1; i++)
  {
```

```
    process a;
    cout<<"Enter the arrival time for pid "<<i<<": ";
    cin>>a.arrtime;
    cout<<"Enter the burst time for pid "<<i<<": ";
    a.pid=i;
    cin>>a.burstime;
    a.remaintime=a.burstime;
    ttlexec=ttlexec+a.burstime;

    notinqueue.push_back(a);

  }
  cout<<endl<<endl;

  sort(notinqueue.begin(), notinqueue.end(), compareArr);

  for(curtime; curtime<=ttlexec; curtime++)
  {
    int k1=999;
    if(!notinqueue.empty())
      k1=notinqueue.back().arrtime;

    if(inqueue.empty())
    {
      //only takes one process per time cause im lazy sry,
could use like while loop and update k1 inside
      if(k1==curtime)
      {
        process a=notinqueue.back();
        notinqueue.pop_back();
        inqueue.push_back(a);
        inqueue.back().remaintime--;
        inqueue.back().wtime=0;
        avgwait=avgwait+curtime;
        if(inqueue.back().remaintime==0)
        {
          inqueue.back().exitime=curtime+1;
          inqueue.back().turnaroundtime=inqueue.back().exitime-
inqueue.back().arrtime;

avgturnaround=avgturnaround+inqueue.back().turnaroundtime;
          process a=inqueue.back();
          completed.push_back(a);
```

```
            inqueue.pop_back();

        }
      }
    }
    else
    {
      if(k1==curtime)
      {
        process a=notinqueue.back();
        notinqueue.pop_back();
        inqueue.push_back(a);
      }

      sort(inqueue.begin(), inqueue.end(), compareRem);

      if(!inqueue.empty())
      {
        inqueue.back().remaintime--;

      }
      if(inqueue.back().wtime!=999)
      {
        inqueue.back().wtime=curtime;
        avgwait=avgwait+curtime;
      }

      if(inqueue.back().remaintime==0)
      {
        inqueue.back().exitime=curtime+1;
        inqueue.back().turnaroundtime=inqueue.back().exitime-
inqueue.back().arrtime;

avgturnaround=avgturnaround+inqueue.back().turnaroundtime;
        process a=inqueue.back();
        completed.push_back(a);
        inqueue.pop_back();

      }
```

```
    }

  }

  avgturnaround=avgturnaround/n;
  avgwait=avgwait/n;

  cout<<"avg turnaround is: "<<avgturnaround;
  cout<<"\navg wait is: "<<avgwait<<endl;

  cout<<"final results: "<<endl;

  sort(completed.begin(), completed.end(), compareArr);

  for(int i=0; i<n; i++)
  {
    cout<<"Pid\tArrival Time\tBurst Time\tTurnaround Time\t\
tExit Time";
    cout<<completed.back().pid<<"\t\
t"<<completed.back().arrtime<<"\t\
t"<<completed.back().burstime<<"\t\
t"<<completed.back().turnaroundtime<<"\t\
t"<<completed.back().exitime;
    completed.pop_back();

  }



  return 0;
}
```

<next page>

```
advait@advait-VirtualBox: ~/Desktop/CSE2005/LAB03
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$ g++ sjf_p.cpp
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$ ./a.out
Enter no. of processes: 5
Enter the arrival time for pid 1: 0
Enter the burst time for pid 1: 2
Enter the arrival time for pid 2: 1
Enter the burst time for pid 2: 4
Enter the arrival time for pid 3: 3
Enter the burst time for pid 3: 1
Enter the arrival time for pid 4: 2
Enter the burst time for pid 4: 5
Enter the arrival time for pid 5: 4
Enter the burst time for pid 5: 3


avg turnaround is: 5
avg wait is: 0
final results:
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time
1               0               2               2               2
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time
2               1               4               9               10
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time
4               2               5               13              15
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time
3               3               1               1               4
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time
5               4               3               3               7
advait@advait-VirtualBox:~/Desktop/CSE2005/LAB03$
```

<to test the code, simply copy paste in a file, give it the .cpp extension, and run>
<copy pasted as easier to verify>