



The National Institute of Engineering, Mysuru

Department of Computer Science and Engineering



Operating System tutorials (CS5C02) – 2021-22

To the Course Instructor

Dr JAYASRI BS

(Associate Professor)

Operating System Report

Title: Page replacement and Scheduling Algorithms

Team Details:

Sl. No.	USN	NAME
1.	4NI19CS031	BHAVANI B
2.	4NI19CS075	NIHARIKA
3.	4NI19CS087	PRIYANKA S M

SJF(Shortest job first CPU scheduling algorithm)

Introduction:

- A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. These algorithms are either non-pre-emptive or pre-emptive. Non-pre-emptive algorithms are designed so that once a process enters the running state, it cannot be pre-empted until it completes its allotted time, whereas the pre-emptive scheduling is based on priority where a scheduler may pre-empt a low priority running process anytime when a high priority process enters into a ready state.
- **Shortest Job First (SJF)** is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be pre-emptive or non-pre-emptive. The pre-emptive version of SJF is called Shortest Remaining Time First (SRTF).

Algorithm:

1. Sort all the processes according to their arrival time.
2. Select the process with minimum arrival time as well as minimum burst time.
3. After completion of the process, select from the ready queue the process which has the minimum burst time.
4. If two processes have the same length of CPU burst then FCFS policy is used to solve the tie.
5. Repeat above processes until all processes have finished their execution.

Advantages and Disadvantages:

Advantages:

1. Maximum throughput.
2. Minimum average waiting and turnaround time.

Disadvantages:

1. May suffer with the problem of starvation.
2. It is not implementable because the exact burst time for a process can't be known in advance.

Code:

```
#include <iostream>

using namespace std;

int mat[10][6];

float wt=0,tat=0;

void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void arrangeArrival(int num, int mat[][6])
{
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (mat[j][1] > mat[j + 1][1]) {
```

```
for (int k = 0; k < 5; k++) {  
    swap(mat[j][k], mat[j + 1][k]);  
}  
}  
}  
}  
}  
  
void completionTime(int num, int mat[][6])  
{  
    int temp, val, m=0;  
    int va = mat [0][2];  
    for(int i=1; i<num; i++){  
        if(mat[0][1]==mat[i][1] && va>mat[i][2]){  
            va = mat[i][2];  
            m=i;  
        }  
    }  
}
```

```
if (m!=0){  
    for(int k=0;k<3;k++){  
        swap(mat[m][k], mat[0][k]);  
    }  
}  
mat[0][3] = mat[0][1] + mat[0][2];  
mat[0][5] = mat[0][3] - mat[0][1];  
mat[0][4] = mat[0][5] - mat[0][2];
```

```
for (int i = 1; i < num; i++) {  
    temp = mat[i - 1][3];  
    int low = mat[i][2];  
    for (int j = i; j < num; j++) {  
        if (temp >= mat[j][1] && low >= mat[j][2]) {  
            low = mat[j][2];  
            val = j;  
        }  
    }  
}
```

```
}  
mat[val][3] = temp + mat[val][2];  
mat[val][5] = mat[val][3] - mat[val][1];  
mat[val][4] = mat[val][5] - mat[val][2];  
for (int k = 0; k < 6; k++) {  
    swap(mat[val][k], mat[i][k]);  
}  
}  
}  
  
int main()  
{  
    int num, temp;  
    cout << "Enter number of Process: ";  
    cin >> num;  
    cout << "\n";  
    for (int i = 0; i < num; i++) {  
        cout << "Enter Process Id: ";
```

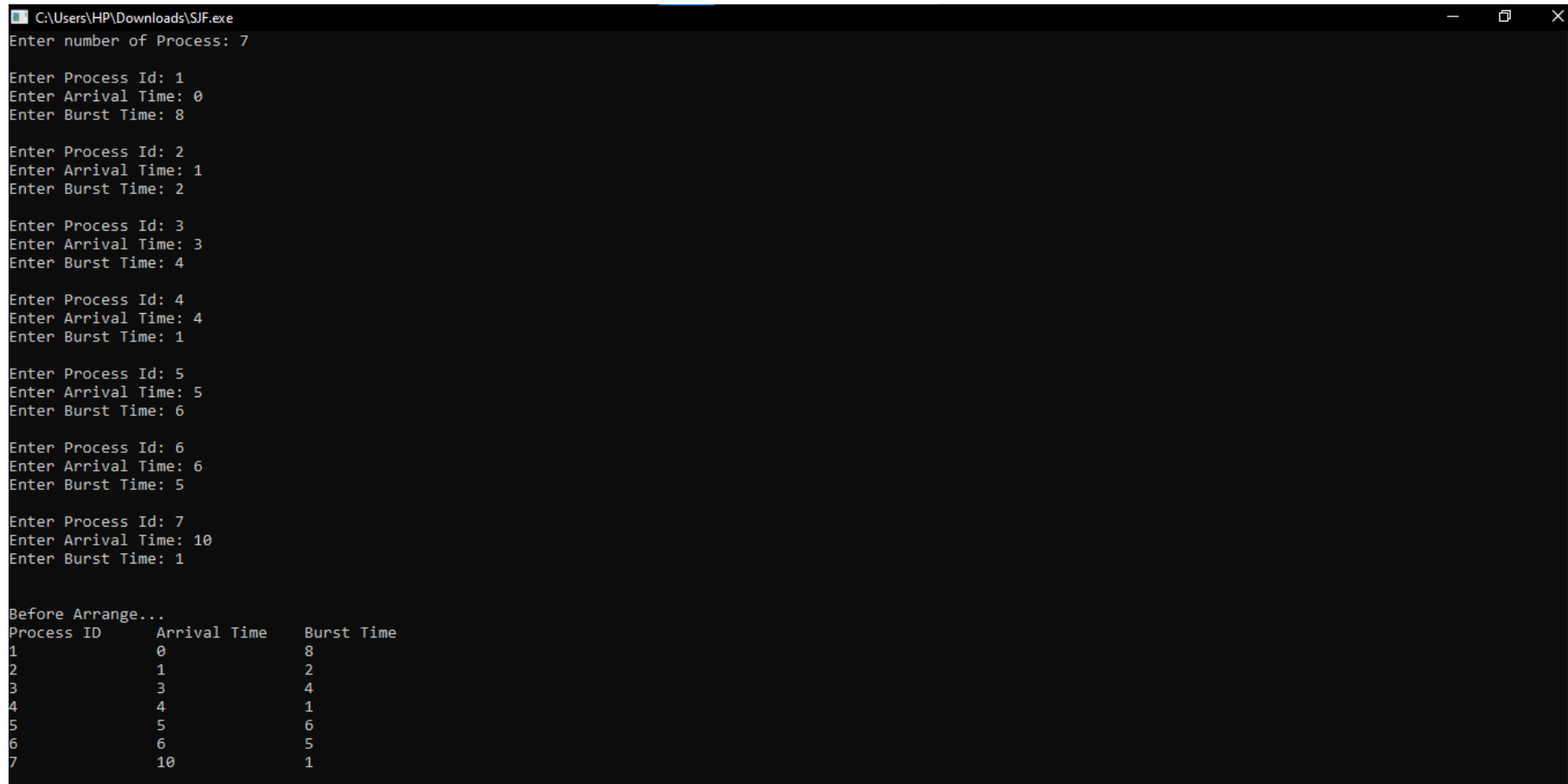


```
cin >> mat[i][0];
cout << "Enter Arrival Time: ";
cin >> mat[i][1];
cout << "Enter Burst Time: ";
cin >> mat[i][2];
cout << "\n";
}
cout << "\n";
cout << "Before Arrange...\n";
cout << "Process ID\tArrival Time\tBurst Time\n";
for (int i = 0; i < num; i++) {
    cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
    << mat[i][2] << "\n";
}
arrangeArrival(num, mat);
completionTime(num, mat);
cout << "\n";
```

```
cout << "Final Result...\n";
cout << "Process ID " << " Arrival Time " << " Burst Time"
<< " Completion Time "<< " Turnaround Time "
<< " Waiting Time \n";
for (int i = 0; i < num; i++) {
cout << " " << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
<< mat[i][2] << "\t\t" << mat[i][3] << "\t\t"<< mat[i][5] <<
"\t\t" << mat[i][4] << endl7;}
for (int i = 0; i < num; i++){
wt=wt+mat[i][4];
}
cout << "\nAverage Waiting Time: ";
cout << wt/num;
for (int i = 0; i < num; i++){
tat=tat+mat[i][5];
}
cout << "\nAverage Turn Around Time: ";
```

```
cout << tat/num;  
}
```

Output:



```
C:\Users\HP\Downloads\SJF.exe  
Enter number of Process: 7  
  
Enter Process Id: 1  
Enter Arrival Time: 0  
Enter Burst Time: 8  
  
Enter Process Id: 2  
Enter Arrival Time: 1  
Enter Burst Time: 2  
  
Enter Process Id: 3  
Enter Arrival Time: 3  
Enter Burst Time: 4  
  
Enter Process Id: 4  
Enter Arrival Time: 4  
Enter Burst Time: 1  
  
Enter Process Id: 5  
Enter Arrival Time: 5  
Enter Burst Time: 6  
  
Enter Process Id: 6  
Enter Arrival Time: 6  
Enter Burst Time: 5  
  
Enter Process Id: 7  
Enter Arrival Time: 10  
Enter Burst Time: 1  
  
Before Arrange...  
Process ID    Arrival Time    Burst Time  
1             0              8  
2             1              2  
3             3              4  
4             4              1  
5             5              6  
6             6              5  
7             10             1
```

C:\Users\HP\Downloads\SJF.exe

Final Result...

Process ID	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
1	0	8	8	8	0
4	4	1	9	5	4
2	1	2	11	10	8
7	10	1	12	2	1
3	3	4	16	13	9
6	6	5	21	15	10
5	5	6	27	22	16

Average Waiting Time: 6.85714

Average Turn Around Time: 10.7143

Process exited after 26.96 seconds with return value 0
Press any key to continue . . .

FIFO (First In First Out page replacement algorithm)

Introduction:

- In operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.
- First In First Out (**FIFO**) page replacement algorithm is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

How it works !

- Operating system keeps track of all the pages in a queue
- If a page is already present in set, it's a hit and there will be no change in the queue. If the page in the string is not present, it's a page fault.
- If set holds less pages than capacity, insert page into the set one by one until all page requests are processed. Simultaneously maintain the pages in the queue to perform FIFO and increment page faults. If current page is already present, it does nothing.
- If the queue is full, remove the first page from the queue, replace that first page with current page in the string and store it back in the queue.
- Then page faults are incremented.

Algorithm:

Let capacity be the number of pages that memory can hold. Let set be the current set of pages in memory.

1. Start traversing the pages.
 - i) If set holds less pages than capacity.
 - a) Insert page into the set one by one until the size of set reaches capacity or all page requests are processed.
 - b) Simultaneously maintain the pages in the queue to perform FIFO.
 - c) Increment page fault.
 - ii) Else If current page is present in set, do nothing. Else
 - a) Remove the first page from the queue as it was the first page to be entered in the memory.
 - b) Replace the first page in the queue with the current page in the string.
 - c) Store current page in the queue and increment page faults.
2. Return page faults.

Advantages and Disadvantages:

Advantages:

- It is easy to understand and implement.
- The number of operations is limited in a queue makes the implementation simple.

Disadvantages:

- When the number of incoming pages is large, it might not provide excellent performance.
- When we increase the number of frames or capacity to store pages in the queue, it should give us less number of page faults.
- Sometimes FIFO may behave abnormally, and it may increase the number of page faults. This behaviour of FIFO is called Belady's anomaly.
- In FIFO, the system should keep track of all the frames. Sometimes it results in slow process execution.

Code:

```
#include<iostream>

using namespace std;

int main(){

int reference_string[10],page_faults=0,m,n,s,pages,frames;

cout<<"Enter the number of pages : ";

cin>>pages;

cout<<"\nEnter the reference string values : \n";

for(m=0;m<pages;m++){

cout<<"Value no.["<<m+1<<"]: \t";

cin>>reference_string[m];

}

cout<<"\nEnter the total number of frames : ";

cin>>frames;

int temp[frames];

for(m=0;m<frames;m++){
```

```
temp[m]=-1;
}
for(m=0;m<pages;m++){
s=0;
for(n=0;n<frames;n++){
if(reference_string[m]==temp[n]){
s++;
page_faults--;
}
}
page_faults++;
if((page_faults<=frames)&&(s==0)){
temp[m]=reference_string[m];
}
else if(s==0){
temp[(page_faults-1)%frames]=reference_string[m];
}
```

```
cout<<"\n";
for(n=0;n<frames;n++){
cout<<temp[n]<<"\t";
}
}

int hits=(pages-page_faults);
int mr=((page_faults*1.0)/pages)*100;
int hr=((hits*1.0)/pages)*100;

cout<<"\n\nTotal page faults : "<<page_faults<<"\n";
cout<<"Total Hits : "<<hits<<"\n";
cout<<"Miss Ratio : "<<mr<<"%\n";
cout<<"Hits Ratio : "<<hr<<"%\n";

return 0;
}
```

Output:

```
C:\Users\HP\Downloads\FIFO.exe
Enter the number of pages : 5

Enter the reference string values :
Value no.[1]: 4
Value no.[2]: 1
Value no.[3]: 2
Value no.[4]: 4
Value no.[5]: 5

Enter the total number of frames : 3

4      -1      -1
4       1      -1
4       1       2
4       1       2
5       1       2

Total page faults : 4
Total Hits : 1
Miss Ratio : 80%
Hits Ratio : 20%

-----
Process exited after 6.061 seconds with return value 0
Press any key to continue . . .
```

GITHUB LINKS:

BHAVANI B - <https://github.com/BHAVANIB29>

NIHARIKA - <https://github.com/NiharikaAcharya/OS.git>

PRIYANKA S M - <https://github.com/priya-082001>