Code

This section of code initializes the program as follows:

-testfile : string that holds the test file

-outfile : string that holds the name of output file

-p_table : 2D array to store the contents of input file

-processes : array that holds the IDs

-CPU_time : array that holds the CPU brust times

-IO_time : array that holds the IO brust times

-arrival_time : array that holds the arrival times

Note:

If the user enters a wrong name accidentally , a loop will run till he/she inputs the right name and a message saying “Invalid” gets printed out;

```
163  // Driver code
164  int main()
165  {
166      std ::string testfile;
167      std ::string outfile;
168      fstream fp;
169      //get the test case filename from user
170      getline(cin , testfile);
171      while(1){
172          fp.open(testfile , ios::in);
173          if(fp.fail()){
174              cout << "Invalid\n";
175              fp.clear();
176              getline(cin , testfile);
177          }
178          else break;
179      }
180      std ::string unused;
181      int lines = 0;
182      //count the lines of test file,hence the number of processes
183      while(std :: getline(fp , unused))
184          ++lines;
185
186      int p_table[lines][4];
187      int processes[lines] , CPU_time[lines] , IO_time[lines] , arrival_time[lines];
188      //seek the beginning of file
189      fp.clear();
190      fp.seekg(0);
```

```
191     //loop to store the contents of file in 2D array
192     for(int i = 0 ; i < lines ; i++){
193         fp.get();
194         for(int j = 0 ; j < 4 ; j++)
195             fp >> p_table[i][j];
196         std::getline(fp , unused);
197     }
198     for(int i = 0 ; i < lines ; i++){
199         for(int j = 0 ; j < 4 ; j++){
200             processes[i] = p_table[i][0];
201             CPU_time[i] = p_table[i][1];
202             IO_time[i] = p_table[i][2];
203             arrival_time[i] = p_table[i][3];
204         }
205     }
206     int quantumTime ,scheduling_algorithm ;
207     cout << "Enter '0' for FCFS '1' for RR : " ;
208     cin >>scheduling_algorithm ;
209     if(scheduling_algorithm == 0)
210         outfile = "outputFCFS.txt";
211     else outfile = "outputRR.txt";
212     //opening a test file to storing output in it
213     fout.open(outfile , ios::out);
214     if(scheduling_algorithm==1){
215         cout << "Enter The quantumTime : " ;
216         cin >>quantumTime ;
217     }
218     //number of processes
219     n = sizeof processes / sizeof processes[0];
220     scheduling(processes,quantumTime,scheduling_algorithm,n, CPU_time, arrival_time , IO_time);
221     fout.close();
222     fp.close();
223  return 0;
224 }
```

Here we count the files of the input file so we have knowledge of how many processes are in the file and get the scheduling algorithm , quantum time(if necessary) from the user. After that an output file is opened to store the simulation results. Finally we call our major function to perform the simulation task according to the scheduling algorithm.

The below section of code includes the libraries used in the program and the queues being used as follows:
-Blocked_queue : a priority queue to store the blocked processes in a manner such the lowest remaining IO burst are pushed to the front of it.
-Ready : a priority queue to store the ready processes in a manner such that processes of different IDs but arrived -became ready- both at the same time cycle are sorted from lowest to highest ID.
-Ready_queue : a queue that works so normally as FIFO.

Note :
Here I need to illustrate the difference between the two latter queues suppose I have the following input data formatted as pdf test case :0 1 2 0
1 1 1 5
2 1 1 3
in the above example @ t = 3 : 2 is ready , 0 finished IO and is ready also
so after pushing them to the Ready queue they get arranged ,then pushed to the Ready_queue and when the next cycle comes 0 gets Running .
Now suppose another input data as follows :0 2 3 0
1 1 3 1
2 3 3 0
in the above example @ t = 5 : 1 is ready , 0 finished IO and is ready also But 1 has arrived earlier in the ready queue so it gets Running first and 0 remains Ready.
To sum up I push the processes that became ready at the same cycle in Ready queue which sort them by IDs then at the end of cycle push them in Ready_queue so that I can differentiate between those that became ready at the same cycle or not.

```cpp
// C++ program for implementation of FCFS && Round Robin
// scheduling algorithms with I/O burst included
#include<iostream>
#include <queue>
#include <fstream>
#include <iomanip>

using namespace std;
fstream fout;
typedef pair<int, int> pd;

struct myComp {
    constexpr bool operator()(
        pair<int, int> const& a,
        pair<int, int> const& b)
        const noexcept
    {
        return a.second > b.second;
    }
};

priority_queue<pd, vector<pd>, myComp> Blocked_queue;

priority_queue<int, vector<int>, greater<int> > Ready ;
queue<int> Ready_queue ;
queue<int> Running_queue ;

int n ;
using namespace std;
```

```
31  //functions to show the processes in queues
32  void showq(queue<int> gq)
33  {
34      queue<int> g = gq;
35      while (!g.empty()) {
36          fout  << " " << g.front();
37          g.pop();
38      }
39  }
40
41  void showq2(priority_queue<pd, vector<pd>, myComp> gq)
42  {
43      priority_queue<pd, vector<pd>, myComp> g = gq;
44      while (!g.empty()) {
45          fout << " "  << g.top().first;
46          g.pop();
47      }
48  }
49
50  void scheduling(int processes[],int quantumTime,int scheduling_algorithm, int n, int bt[], int at[],int Iot[]){
51
52      int CPU_times[n] , bt_used[n] ;
53
54      for (int i = 0 ; i < n ; i++){
55          CPU_times[i]=0;
56          bt_used[i]=0;
57      }
58
59      int time_cycle=0 , arrived = 0;
60      int CPU_start_time[n],IO_start_time[n];
61      int  Finishing_time=0 ;
62      float CPU_used=0 ;
63      int quantum=0;
64      int Turnaround_time[n] ;
```

This section of code has two display functions :
-showq : to show the contents of normal queues.
-showq2 : to show the contents of priority queues.

Then a part of the scheduling function comes where some variables are to be initialized :
-CPU_times : array the counts the number of completed CPU brusts for every process.
-bt_used : array that holds the number of cpu cycles finished for every process.
Both are initialized to 0.
-time_cycle : the current simulation time.
-arrived : the number of processes that have been loaded -initiated- by the system.
-CPU_start_time : array that holds the time cycle at which the process began.
-IO_start_time : array that holds the time cycle at which the process got blocked.
-quantum : if round robin is chose this is incremented every time cycle there is running process.
-Turnaround_time : holds the turnaround times of every process.

```
while(time_cycle >= 0){
        if(scheduling_algorithm==1) quantum++;
        //ready
        for (int i = 0 ; i < n ; i++){
            if(at[processes[i]]==time_cycle){
                arrived++;
              if(bt[processes[i]]==0)
                  Blocked_queue.push(make_pair(processes[i],at[processes[i]]+Iot[processes[i]]));
              else
                  Ready.push(processes[i]) ;
            }
        }
```

This section of code checks if a process has arrived if so it's pushed in the ready queue but if its CPU brust time is 0 then it gets blocked directly.

```
//Quantum Ended and IO

if(!Running_queue.empty()){
    if(quantum==quantumTime&&bt_used[Running_queue.front()]<bt[Running_queue.front()]&&scheduling_algorithm==1){
            Ready.push(Running_queue.front()) ;
            Running_queue.pop();
    }
    else if(bt_used[Running_queue.front()]==bt[Running_queue.front()]){
            bt_used[Running_queue.front()]=0;
            CPU_times[Running_queue.front()]++;
        if(CPU_times[Running_queue.front()]>1){
             Turnaround_time[Running_queue.front()]=time_cycle-at[Running_queue.front()];
             Running_queue.pop();
       }
        else
        {
            CPU_times[Running_queue.front()]++;
            IO_start_time[Running_queue.front()]=time_cycle;
            Blocked_queue.push(make_pair(Running_queue.front(),
            IO_start_time[Running_queue.front()]+Iot[Running_queue.front()]));
            Running_queue.pop();
        }
    }

}
```

Here we check if a running process has to be scheduled or still not as follows:

1)Round robin :

The scheduler selects a new process to run on the CPU in the following cases:

a. The process currently running on the CPU finishes its CPU time and it will terminate.

b. The process currently running on the CPU finishes its CPU time and it will block for I/O.

c. The process currently running on the CPU has not finished its CPU time.

However, it has spent number of cycles on the CPU equal to the quantum time.

2)FCFS :

The scheduler selects a new process to run on the CPU in the following cases:

a. The process currently running on the CPU finishes its CPU time and it will terminate.

b. The process currently running on the CPU finishes its CPU time and it will block for I/O.

```
//Blocked
while(!Blocked_queue.empty() &&
((bt[Blocked_queue.top().first]==0&&at[Blocked_queue.top().first]
+Iot[Blocked_queue.top().first]==time_cycle) || (bt[Blocked_queue.top().first]!=0&&
IO_start_time[Blocked_queue.top().first]+Iot[Blocked_queue.top().first]==time_cycle))){

        if(bt[Blocked_queue.top().first]==0&&at[Blocked_queue.top().first]+Iot[Blocked_queue.top().first]==time_cycle){
            Turnaround_time[Blocked_queue.top().first]=Iot[Blocked_queue.top().first];
            Blocked_queue.pop();
        }
        else{
           Ready.push(Blocked_queue.top().first) ;
           Blocked_queue.pop();
        }

}
while(!Ready.empty()){
    Ready_queue.push(Ready.top());
    Ready.pop();
}
```

Here we check if a blocked process has finished its IO burst time. If it's done it's pushed to the ready queue again except for those that their CPU brust is 0 they leave the system.

Also we check the Ready queue (priority queue) if it isn't empty and pop its contents in the normal Ready_queue queue.

```
        //CPU
        if(Running_queue.empty()&&(!Ready_queue.empty())){
            Running_queue.push(Ready_queue.front());
            Ready_queue.pop();
            quantum=0;
        }
        if(!Running_queue.empty()) bt_used[Running_queue.front()]++;

        if(!Ready_queue.empty()||!Running_queue.empty()||!Blocked_queue.empty()||time_cycle<=at[0]){
        fout <<time_cycle << " -->";
        if(!Ready_queue.empty()){

         showq(Ready_queue);
         fout <<" : Ready" ;
        }
         if(!Running_queue.empty()){
             CPU_used++;
           showq(Running_queue);
            fout <<" : Running";
         }
         if(!Blocked_queue.empty()){
              showq2(Blocked_queue);
              fout <<" : Blocked"  ;
         }
          fout <<"\n"  ;
        }
        if (Ready_queue.empty()&&Running_queue.empty()&&Blocked_queue.empty()&&arrived == n)
        break;
        time_cycle++ ,Finishing_time++;;
}
```

Here we check if there are processes Ready if so the front is pushed to the running queue and initializes quantum to 0 again.
Also we increment the bt_used of the currently running process (if any).

Finally we check for the three used queues and print their contents (if any) to the output file.
We break the whole process -i mean the forever loop- when all queues are empty and arrived processes = total number of processes.

```
154     Finishing_time--;
155     float  CPU_Utilization=CPU_used/(Finishing_time+1) ;
156     fout <<"\nFinishing time:"<< Finishing_time <<"\n";
157     fout <<"CPU Utilization:"<< setprecision(2) << CPU_Utilization<<"\n" ;
158      for (int i = 0 ; i < n ; i++)
159   fout <<"Turnaround time of Process " <<i<<":"<<Turnaround_time[i]<<"\n";
160
161  }
```

This section stores the statistics calculated.

Results

```
Enter file name : Test-file-1.txt
Enter '0' for FCFS '1' for RR : 0
```

```
0 -->
1 -->
2 --> 3 : Ready 1 : Running
3 --> 3 2 : Ready 1 : Running 0 : Blocked
4 --> 2 : Ready 3 : Running 0 1 : Blocked
5 --> 2 : Ready 3 : Running 0 1 : Blocked
6 --> 2 : Ready 3 : Running 0 1 : Blocked
7 --> 2 : Ready 3 : Running 1 : Blocked
8 --> 2 1 : Ready 3 : Running
9 --> 2 1 : Ready 3 : Running
10 --> 2 1 : Ready 3 : Running
11 --> 2 1 : Ready 3 : Running
12 --> 2 1 : Ready 3 : Running
13 --> 1 : Ready 2 : Running 3 : Blocked
14 --> 1 : Ready 2 : Running 3 : Blocked
15 --> 1 : Running 3 2 : Blocked
16 --> 3 : Ready 1 : Running 2 : Blocked
17 --> 3 : Running 2 : Blocked
18 --> 3 : Running 2 : Blocked
19 --> 2 : Ready 3 : Running
20 --> 2 : Ready 3 : Running
21 --> 2 : Ready 3 : Running
22 --> 2 : Ready 3 : Running
23 --> 2 : Ready 3 : Running
24 --> 2 : Ready 3 : Running
25 --> 2 : Ready 3 : Running
26 --> 2 : Running
27 --> 2 : Running

Finishing time:27
CPU Utilization:0.93
Turnaround time of Process 0:4
Turnaround time of Process 1:15
Turnaround time of Process 2:25
Turnaround time of Process 3:24
```

```
Enter file name : Test-file-1.txt
Enter '0' for FCFS '1' for RR : 1
Enter The quantumTime : 2
```

```
0 -->
1 -->
2 --> 3 : Ready 1 : Running
3 --> 3 2 : Ready 1 : Running 0 : Blocked
4 --> 2 : Ready 3 : Running 0 1 : Blocked
5 --> 2 : Ready 3 : Running 0 1 : Blocked
6 --> 3 : Ready 2 : Running 0 1 : Blocked
7 --> 3 : Ready 2 : Running 1 : Blocked
8 --> 1 : Ready 3 : Running 2 : Blocked
9 --> 1 : Ready 3 : Running 2 : Blocked
10 --> 3 : Ready 1 : Running 2 : Blocked
11 --> 3 : Ready 1 : Running 2 : Blocked
12 --> 2 : Ready 3 : Running
13 --> 2 : Ready 3 : Running
14 --> 3 : Ready 2 : Running
15 --> 3 : Ready 2 : Running
16 --> 3 : Running
17 --> 3 : Running
18 --> 3 : Running
19 --> 3 : Blocked
20 --> 3 : Blocked
21 --> 3 : Blocked
22 --> 3 : Running
23 --> 3 : Running
24 --> 3 : Running
25 --> 3 : Running
26 --> 3 : Running
27 --> 3 : Running
28 --> 3 : Running
29 --> 3 : Running
30 --> 3 : Running

Finishing time:30
CPU Utilization:0.84
Turnaround time of Process 0:4
Turnaround time of Process 1:10
Turnaround time of Process 2:13
Turnaround time of Process 3:29
```

```
Enter file name : Test-file-2.txt
Enter '0' for FCFS '1' for RR : 0
0 -->
1 -->
2 --> 2 : Running 1 3 0 : Blocked
3 --> 2 : Running 1 3 0 : Blocked
4 --> 1 3 0 2 : Blocked
5 --> 1 3 0 2 : Blocked
6 --> 0 2 : Blocked
7 --> 0 2 : Blocked
8 --> 2 : Running
9 --> 2 : Running

Finishing time:9
CPU Utilization:0.4
Turnaround time of Process 0:6
Turnaround time of Process 1:4
Turnaround time of Process 2:8
Turnaround time of Process 3:4
```

```
Enter file name : Test-file-2.txt
Enter '0' for FCFS '1' for RR : 1
Enter The quantumTime : 1
0 -->
1 -->
2 --> 2 : Running 1 3 0 : Blocked
3 --> 2 : Running 1 3 0 : Blocked
4 --> 1 3 0 2 : Blocked
5 --> 1 3 0 2 : Blocked
6 --> 0 2 : Blocked
7 --> 0 2 : Blocked
8 --> 2 : Running
9 --> 2 : Running

Finishing time:9
CPU Utilization:0.4
Turnaround time of Process 0:6
Turnaround time of Process 1:4
Turnaround time of Process 2:8
Turnaround time of Process 3:4
```

```
Enter file name : Test-file-3.txt
Enter '0' for FCFS '1' for RR : 0
0 -->
1 --> 6 : Ready 2 : Running
2 --> 6 7 : Ready 2 : Running 3 0 : Blocked
3 --> 6 7 1 5 : Ready 2 : Running 3 0 4 : Blocked
4 --> 6 7 1 5 : Ready 2 : Running 0 4 : Blocked
5 --> 6 7 1 5 : Ready 2 : Running 0 4 : Blocked
6 --> 7 1 5 : Ready 6 : Running 4 2 : Blocked
7 --> 7 1 5 : Ready 6 : Running 2 : Blocked
8 --> 1 5 : Ready 7 : Running 2 6 : Blocked
9 --> 1 5 : Ready 7 : Running 2 6 : Blocked
10 --> 5 : Ready 1 : Running 2 6 7 : Blocked
11 --> 5 2 6 : Ready 1 : Running 7 : Blocked
12 --> 2 6 : Ready 5 : Running 7 1 : Blocked
13 --> 2 6 7 : Ready 5 : Running 1 : Blocked
14 --> 6 7 : Ready 2 : Running 1 5 : Blocked
15 --> 6 7 : Ready 2 : Running 1 5 : Blocked
16 --> 6 7 1 : Ready 2 : Running 5 : Blocked
17 --> 6 7 1 : Ready 2 : Running 5 : Blocked
18 --> 6 7 1 5 : Ready 2 : Running
19 --> 7 1 5 : Ready 6 : Running
20 --> 7 1 5 : Ready 6 : Running
21 --> 1 5 : Ready 7 : Running
22 --> 1 5 : Ready 7 : Running
23 --> 5 : Ready 1 : Running
24 --> 5 : Ready 1 : Running
25 --> 5 : Running
26 --> 5 : Running

Finishing time:26
CPU Utilization:0.96
Turnaround time of Process 0:4
Turnaround time of Process 1:22
Turnaround time of Process 2:18
Turnaround time of Process 3:2
Turnaround time of Process 4:4
Turnaround time of Process 5:24
Turnaround time of Process 6:20
Turnaround time of Process 7:21
```

```
Enter file name : Test-file-3.txt
Enter '0' for FCFS '1' for RR : 1
Enter The quantumTime : 3
```

```
0 -->
1 --> 6 : Ready 2 : Running
2 --> 6 7 : Ready 2 : Running 3 0 : Blocked
3 --> 6 7 1 5 : Ready 2 : Running 3 0 4 : Blocked
4 --> 7 1 5 2 : Ready 6 : Running 0 4 : Blocked
5 --> 7 1 5 2 : Ready 6 : Running 0 4 : Blocked
6 --> 1 5 2 : Ready 7 : Running 4 6 : Blocked
7 --> 1 5 2 : Ready 7 : Running 6 : Blocked
8 --> 5 2 : Ready 1 : Running 6 7 : Blocked
9 --> 5 2 6 : Ready 1 : Running 7 : Blocked
10 --> 2 6 : Ready 5 : Running 7 1 : Blocked
11 --> 2 6 7 : Ready 5 : Running 1 : Blocked
12 --> 6 7 : Ready 2 : Running 1 5 : Blocked
13 --> 6 7 : Ready 2 : Running 1 5 : Blocked
14 --> 7 1 : Ready 6 : Running 5 2 : Blocked
15 --> 7 1 : Ready 6 : Running 5 2 : Blocked
16 --> 1 5 : Ready 7 : Running 2 : Blocked
17 --> 1 5 : Ready 7 : Running 2 : Blocked
18 --> 5 : Ready 1 : Running 2 : Blocked
19 --> 5 2 : Ready 1 : Running
20 --> 2 : Ready 5 : Running
21 --> 2 : Ready 5 : Running
22 --> 2 : Running
23 --> 2 : Running
24 --> 2 : Running
25 --> 2 : Running
26 --> 2 : Running

Finishing time:26
CPU Utilization:0.96
Turnaround time of Process 0:4
Turnaround time of Process 1:17
Turnaround time of Process 2:26
Turnaround time of Process 3:2
Turnaround time of Process 4:4
Turnaround time of Process 5:19
Turnaround time of Process 6:15
Turnaround time of Process 7:16
```

1

```
Enter file name : Test-file-pdf.txt
Enter '0' for FCFS '1' for RR : 0
0 --> 0 : Running
1 --> 0 : Blocked
2 --> 0 : Blocked
3 --> 2 : Ready 0 : Running
4 --> 2 : Running
5 --> 1 : Running 2 : Blocked
6 --> 2 : Running 1 : Blocked
7 --> 1 : Running

Finishing time:7
CPU Utilization:0.75
Turnaround time of Process 0:4
Turnaround time of Process 1:3
Turnaround time of Process 2:4
```

```
Enter file name : Test-file-pdf.txt
Enter '0' for FCFS '1' for RR : 1
Enter The quantumTime : 1
0 --> 0 : Running
1 --> 0 : Blocked
2 --> 0 : Blocked
3 --> 2 : Ready 0 : Running
4 --> 2 : Running
5 --> 1 : Running 2 : Blocked
6 --> 2 : Running 1 : Blocked
7 --> 1 : Running

Finishing time:7
CPU Utilization:0.75
Turnaround time of Process 0:4
Turnaround time of Process 1:3
Turnaround time of Process 2:4
```