

Operating System (CS - 204)

Topic:	CPU scheduling
---------------	----------------



Course Instructor:	Rao Wakeel Ahmad
Total Marks:	
Issue Date:	15-March-2020
Submission Date:	22-March-2020

Submitted by:	Registration#	Name
	18-CS-33	M Qasim Siddiqui
	18-CS-11	Atif Ali
Obtained Marks:		

Department of Computer Science, University of Engineering
and Technology (UET) Taxila

Question # 01:

In this assignment, you will write a program to implement following CPU scheduling algorithms.

1. First Come First Serve
2. Priority

Your program must read a list of tasks from a file and schedule them based on a desired scheduling algorithm. Each line in the task file describes a single task. Each task has a Name, a Priority, and a CPU Burst and Arrival Time (AT) separated by commas. Priority is represented by an integer number, the higher the number, the higher the priority. CPU burst is also represented by an integer number, which indicates the CPU time required to finish the task. Arrival Time indicates the time the process arrived in the system. Following is the content of an example task file.

```
T1, 4, 20, 0
T2, 3, 25, 6
T3, 3, 25, 2
T4, 5, 15, 0
T5, 5, 20, 4
T6, 1, 10, 1
T7, 3, 30, 5
T8, 10, 25, 8
```

1. You must implement a sorting method which sort the list on Arrival Time.
2. You must implement a linked list to hold the tasks in the memory after reading from the file and to facilitate the scheduling from an in-memory list.
3. You must implement a method to insert a task into the linked list, a method to delete a task from the list, and a method to traverse the list.
4. You must implement scheduling algorithms in separate program files (**schedule_fcfs.c**, **schedule_priority.c**).
5. Your program must compute following things 1) Waiting time of each Process, 2) Turn Around Time of each Process, 3) Draw Gantt Chart and 4) Computer CPU utilization rate?
6. Gantt chart should be animated, it means it should be printed/draw slow and steady showing current process in the CPU.
7. Your program must run from command line, for example, if you want to run **firstcome-first-serve** scheduling algorithm you type the following in the command prompt:

```
C:\> bin/fcfs resource/tasks.txt
```

Here, **tasks.txt** is the name of your task file in **resource** sub folder and the executable **fcfs** is in **bin** sub folder.

Solution:

The “src” folder contains the source of the assignment. The three files contained in the folder are given below:

linkedList.cpp:

```
1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
4. struct node {
5.     string name;
6.     int burstTime;
7.     int arrivalTime;
8.     int priority;
9.     int waitingTime;
10.    int turnaroundTime;
11.    node *next;
12.};
13. class linkedList {
14. private:
15.     node *head = NULL;
16. public:
17.     linkedList();
18.     void insertItem(string, int, int, int);
19.     void deleteItem(string);
20.     void traverse();
21.     node *getHead();
22.};
23. linkedList::linkedList() {
24.}
25. node *linkedList::getHead() {
26.     return head;
27.}
28. void linkedList::traverse() {
29.     node *temp;
30.     temp = head;
31.     cout << "Linked List : ";
32.     if (temp == NULL) {
33.         cout << "Empty Linked List." << endl;
34.     }
35.     else {
36.         cout << "\n_____";
37.         cout << "\n|| Process Name || Priority || Burst Time ||
Arrival Time || Waiting Time || Turnaround Time ||";
38.         cout << "\n||=====||=====||=====||
=====||=====||=====||";
39.         while (temp != NULL) {
40.             cout << "\n||" << setw(9) << temp->name << setw(9) << "||" << setw(8)
<< temp->priority << setw(8) << "||" << setw(9) << temp->burstTime
<< setw(9) << "||" << setw(8) << temp->arrivalTime << setw(10)
<< "||" << setw(9) << temp->waitingTime << setw(9) << "||"
<< setw(10) << temp->turnaroundTime << setw(11) << "||";
41.             temp = temp->next;
42.         }
43.         cout << "\n";
44.     }
45.}
46. node *createNode(string name, int priority, int burstTime, int arrivalTime) {
47.     node *temp;
48.     temp = new struct node;
49.     temp->name = name;
50.     temp->priority = priority;
51.     temp->arrivalTime = arrivalTime;
52.     temp->burstTime = burstTime;
53.     temp->turnaroundTime = 0;
54.     temp->waitingTime = 0;
55.     temp->next = NULL;
56.     return temp;
57.}
```

```

58. void linkedList::insertItem(string name, int priority, int burstTime,
                               int arrivalTime) {
59.     node *temp, *end;
60.     temp = createNode(name, priority, burstTime, arrivalTime);
61.     if (head == NULL) {
62.         head = temp;
63.     }
64.     else {
65.         end = head;
66.         while (end->next != NULL) {
67.             if (end->name == name || end->next->name == name) {
68.                 cout << "\nA process already exists with same name.\n";
69.                 return;
70.             }
71.             end = end->next;
72.         }
73.         end->next = temp;
74.     }
75. }
76. void linkedList::deleteItem(string name) {
77.     node *temp, *previous;
78.     temp = head;
79.     previous = head;
80.     bool deleted = false;
81.     if (temp == NULL) {
82.         cout << "\nEmpty Linked List." << endl;
83.     }
84.     else {
85.         while (temp != NULL) {
86.             if (temp->name == name) {
87.                 previous->next = temp->next;
88.                 deleted = true;
89.                 break;
90.             }
91.             else {
92.                 previous = temp;
93.                 temp = temp->next;
94.             }
95.         }
96.     }
97.     if (deleted)
98.         cout << "\nProcess '\" << name << \"' is deleted.\n";
99.     else
100.        cout << "\nProcess '\" << name << \"' not found.\n";
101. }

```

Schedule_fcfs.cpp:

```

1.  #include <iostream>
2.  #include <string>
3.  #include <fstream>
4.  #include <sstream>
5.  using namespace std;
6.  #include "linkedList.cpp"
7.  void swap(node *, node *);
8.  void sortList(linkedList &);
9.  void drawGanttChart(linkedList &);
10. void calculateWaitingTime(linkedList &);
11. void calculateTurnaroundTime(linkedList &);
12. void readFileAndInsertProcess(linkedList &, const char *);
13. int main(int argc, char const *argv[]) {
14.     char ch;
15.     linkedList list;
16.     cout << "First Come, First Serve Scheduler Implementation";
17.     do
18.     {
19.         cout << "\n\nSelect Option:\n"
20.             << "1 >> Insert a New Process\n"
21.             << "2 >> Sort Process\n"
22.             << "3 >> Read file and Insert process into linked list\n"
23.             << "4 >> Delete a Process\n"
24.             << "5 >> Traverse the Linked list\n"

```

```

25.         << "6 >> Calculate Turnaround Time\n"
26.         << "7 >> Calculate waiting Time\n"
27.         << "8 >> Draw Gantt Chart\n"
28.         << "0 >> Exit\n";
29.     cin >> ch;
30.     switch (ch) {
31.     case '0':{
32.         exit(0);
33.         break;
34.     }
35.     case '1':{
36.         string name;
37.         int priority;
38.         int arrival;
39.         int burst;
40.         cout << "Enter name of Process: "; cin >> name;
41.         cout << "Enter priority of Process: "; cin >> priority;
42.         cout << "Enter burst time of Process: "; cin >> burst;
43.         cout << "Enter arrival time of Process: "; cin >> arrival;
44.         list.insertItem(name, priority, burst, arrival);
45.         list.traverse();
46.         break;
47.     }
48.     case '2':{
49.         sortList(list);
50.         list.traverse();
51.         break;
52.     }
53.     case '3':{
54.         readFileAndInsertProcess(list, argv[1]);
55.         list.traverse();
56.         break;
57.     }
58.     case '4':{
59.         string name;
60.         cout << "Enter name of Process to delete: "; cin >> name;
61.         list.deleteItem(name);
62.         break;
63.     }
64.     case '5':{
65.         list.traverse();
66.         break;
67.     }
68.     case '6':{
69.         calculateTurnaroundTime(list);
70.         list.traverse();
71.         break;
72.     }
73.     case '7':{
74.         calculateWaitingTime(list);
75.         list.traverse();
76.         break;
77.     }
78.     case '8':{
79.         drawGanttChart(list);
80.         break;
81.     }
82.     default:{
83.         cout << "Invalid Choice. Try again." << endl;
84.         break;
85.     }
86.     }
87.     } while (ch != '0');
88. }
89. void swap(struct node *a, struct node *b) {
90.     string name = a->name;
91.     int priority = a->priority;
92.     int arrival = a->arrivalTime;
93.     int burst = a->burstTime;

```

```

94.     a->arrivalTime = b->arrivalTime;
95.     a->burstTime = b->burstTime;
96.     a->name = b->name;
97.     a->priority = b->priority;
98.
99.     b->arrivalTime = arrival;
100.    b->burstTime = burst;
101.    b->priority = priority;
102.    b->name = name;
103.}
104.void sortList(linkedList &list){
105.    node *temp = list.getHead();
106.    while (temp) {
107.        node *min = temp;
108.        node *r = temp->next;
109.        while (r) {
110.            if (min->arrivalTime > r->arrivalTime)
111.                min = r;
112.            r = r->next;
113.        }
114.        swap(temp, min);
115.        temp = temp->next;
116.    }
117.}
118.void readFileAndInsertProcess(linkedList &list, const char *filePath) {
119.    ifstream inFile;
120.    inFile.open(filePath);
121.    if (!inFile){
122.        cout << "File could not be open !! Press any Key...";
123.        return;
124.    }
125.    while (!inFile.eof()) {
126.        int i;
127.        string name;
128.        string subStr;
129.        int val[3];
130.        string line = "";
131.        getline(inFile, line);
132.        stringstream ss(line);
133.        i = -1;
134.        while (!ss.eof()) {
135.            getline(ss, subStr, ',');
136.            if (i == -1) {
137.                name = subStr;
138.            }
139.            else{
140.                val[i] = stoi(subStr);
141.            }
142.            i++;
143.        }
144.        list.insertItem(name, val[0], val[1], val[2]);
145.    }
146.    inFile.close();
147.}
148.void calculateTurnaroundTime(linkedList &list) {
149.    calculateWaitingTime(list);
150.    node *temp;
151.    temp = list.getHead();
152.    if (temp == NULL) {
153.        cout << "Empty Linked List." << endl;
154.    }
155.    else {
156.        while (temp != NULL) {
157.            temp->turnaroundTime = temp->waitingTime + temp->burstTime;
158.            temp = temp->next;
159.        }
160.    }
161.}

```

```

162. void calculateWaitingTime(linkedList &list) {
163.     sortList(list);
164.     node *temp, *p;
165.     temp = list.getHead();
166.     int Wtime = 0;
167.     if (temp == NULL) {
168.         cout << "Empty Linked List." << endl;
169.     }
170.     else {
171.         while (temp != NULL){
172.             p = list.getHead();
173.             Wtime = 0;
174.             while (p->name != temp->name){
175.                 Wtime += p->burstTime;
176.                 p = p->next;
177.             }
178.             temp->waitingTime = Wtime;
179.             temp = temp->next;
180.         }
181.     }
182. }
183. void drawGanttChart(linkedList &list){
184.     calculateTurnaroundTime(list);
185.     node *temp;
186.     temp = list.getHead();
187.     int sign;
188.     int TAtime;
189.     if (temp == NULL){
190.         cout << "\nEmpty Linked List." << endl;
191.     }
192.     else{
193.         cout << "\n\n0";
194.         while (temp != NULL){
195.             cout << setw(temp->burstTime / 2) << temp->turnaroundTime;
196.             if (temp->next == NULL)
197.                 TAtime = temp->turnaroundTime;
198.             temp = temp->next;
199.         }
200.         cout << "\n|";
201.         sign = TAtime;
202.         sign = (sign / 2) - 4;
203.         while (sign >= 0){
204.             cout << "=";
205.             sign--;
206.         }
207.         cout << "| \n|";
208.         temp = list.getHead();
209.         while (temp != NULL){
210.             cout << setw(temp->burstTime / 2) << "|";
211.             temp = temp->next;
212.         }
213.         cout << "\n|";
214.         sign = TAtime;
215.         sign = (sign / 2) - 4;
216.         while (sign >= 0){
217.             cout << "=";
218.             sign--;
219.         }
220.         cout << "| \n";
221.         temp = list.getHead();
222.         while (temp != NULL){
223.             cout << temp->name << setw(temp->burstTime / 2);
224.             temp = temp->next;
225.         }
226.     }
227. }

```

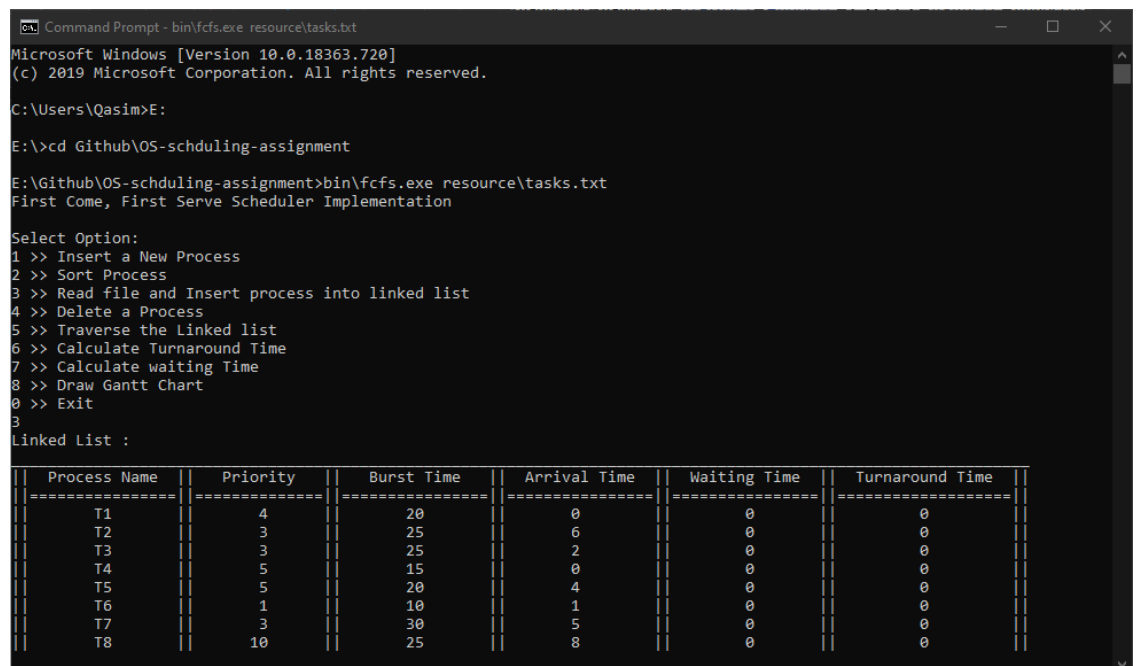
schedule_priority.cpp:

The only difference in this file from the **schedule_fcfs.cpp** above, is the **sortList()** function as we need to sort the list based on priority now.

```
1. void sortList(linkedList &list){
2.     node *temp = list.getHead();
3.     while (temp){
4.         node *min = temp;
5.         node *r = temp->next;
6.         while (r){
7.             if (min->priority > r->priority)
8.                 min = r;
9.             r = r->next;
10.        }
11.        swap(temp, min);
12.        temp = temp->next;
13.    }
14. }
```

Output – “First Come, First Serve”:

- Running the command to execute the program via command line.
- Reading and inserting the **tasks.txt** file in the linked list.



```
Command Prompt - bin\fcfs.exe resource\tasks.txt
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Qasim>E:

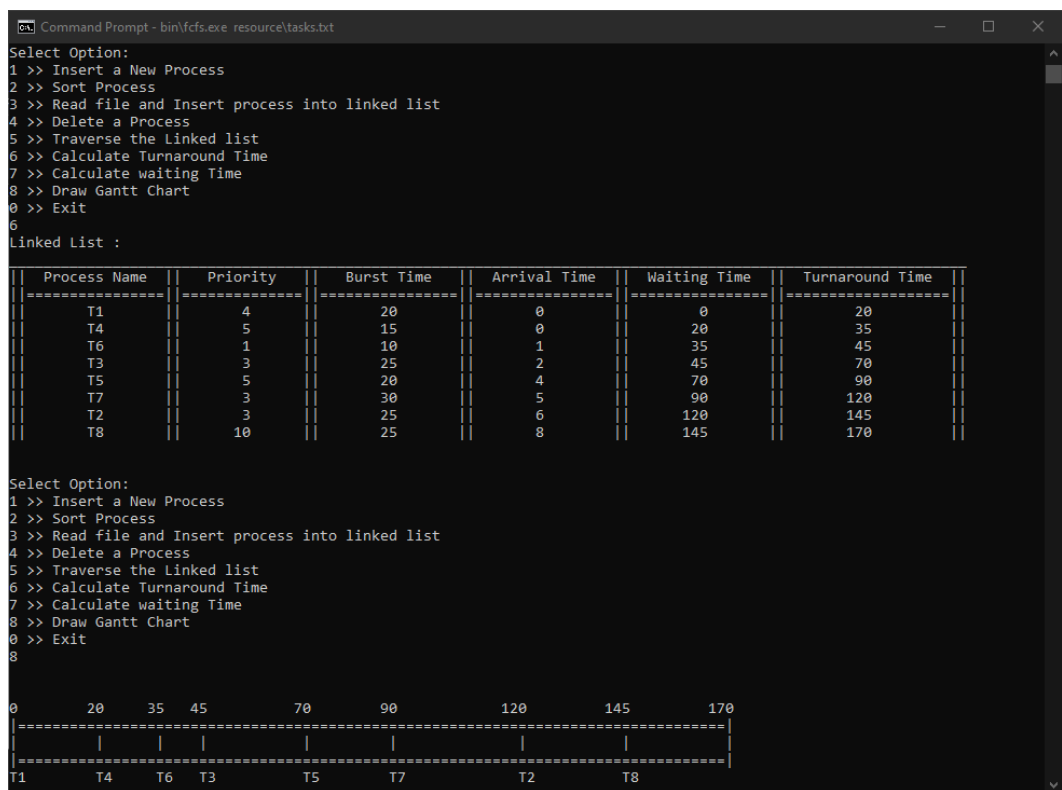
E:\>cd Github\OS-scheduling-assignment

E:\Github\OS-scheduling-assignment>bin\fcfs.exe resource\tasks.txt
First Come, First Serve Scheduler Implementation

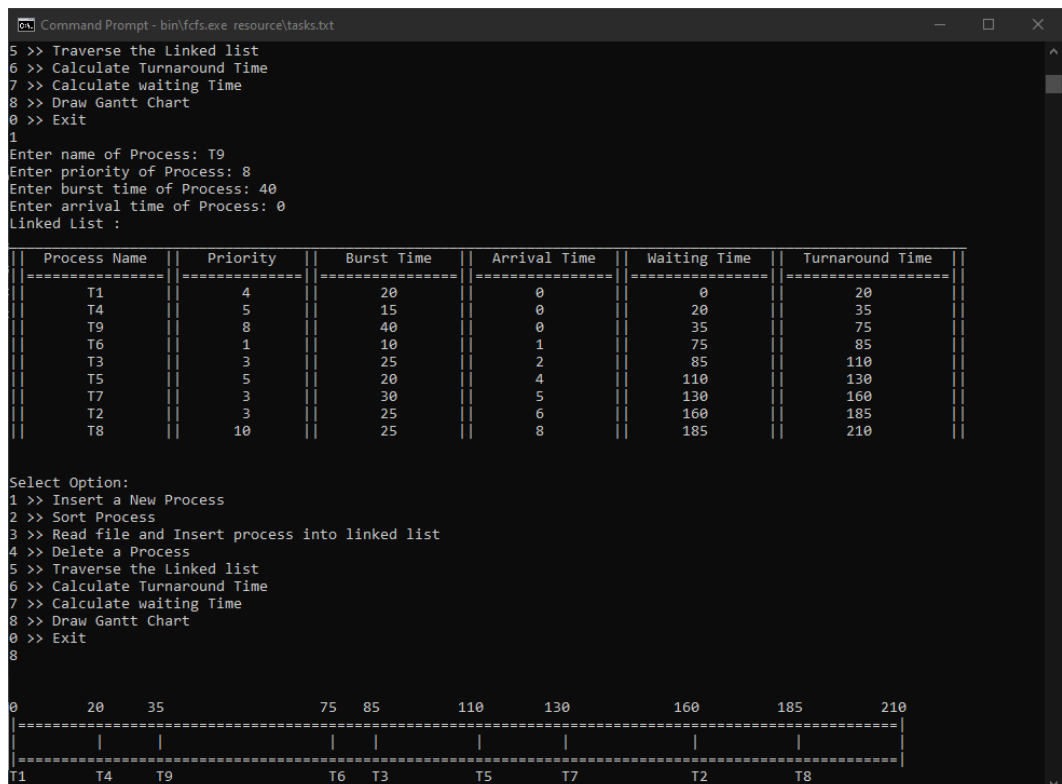
Select Option:
1 >> Insert a New Process
2 >> Sort Process
3 >> Read file and Insert process into linked list
4 >> Delete a Process
5 >> Traverse the Linked list
6 >> Calculate Turnaround Time
7 >> Calculate waiting Time
8 >> Draw Gantt Chart
0 >> Exit
3
Linked List :
```

Process Name	Priority	Burst Time	Arrival Time	Waiting Time	Turnaround Time
T1	4	20	0	0	0
T2	3	25	6	0	0
T3	3	25	2	0	0
T4	5	15	0	0	0
T5	5	20	4	0	0
T6	1	10	1	0	0
T7	3	30	5	0	0
T8	10	25	8	0	0

- Waiting time and Turnaround time is calculated for each process according to the scheduling method via **arrival time**.
- Gantt chart is created from the linked list.



- Inserting a new Process “T9” and drawing the Gantt Chart again.



The Priority scheduler works in a similar manner and sorts the processes respective of their priority and performs all these respective tasks.