

Lab Exercise 3: Implementation of CPU Scheduling Policies: FCFS and SJF (Non- preemptive and Preemptive)

Mahesh Bharadwaj K - 185001089

February 7, 2020

Program

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Process
{
    int pid;
    float at, bt, st, et, wt, tat, rt, rem_t, pri;
} Process;

#include "MinHeap.h"

int getIndex(Process *const arr, const int size, const Process p)
{
    for (int i = 0; i < size; i++)
        if (arr[i].pid == p.pid)
            return i;

    return -1;
}

Process *getProcesses(const int size)
{
    static Process p[100];

    for (int i = 0; i < size; i++)
    {
        printf("Enter the Arrival Time and Burst Time: ");
        scanf("%f %f", &p[i].at, &p[i].bt);
        getchar();
        p[i].pid = i + 1;
        p[i].rt = -1;
        p[i].rem_t = p[i].bt;
        p[i].st = p[i].et = -1;
        p[i].wt = p[i].tat = -1;
    }

    return p;
}

void gantt_chart(Process arr[], int n, int tot_time)
{
    if (n <= 0)
        return;

    printf("\n\nGANTT CHART");

    int i, j;

    // printing the top bar
    printf("\n\n+");
    for (i = 0; i < n - 1; i++)
    {
        for (j = arr[i].st; j < arr[i + 1].st; j++)
            printf("--");
        printf("+");
    }

    for (j = 0; j < tot_time - arr[n - 1].st; j++)
        printf("--");
    printf("+");
}
```

```

printf("\n|");

// printing the process id in the middle
for (i = 0; i < n - 1; i++)
{
    for (j = arr[i].st; j < arr[i + 1].st - 1; j++)
        printf(" ");
    printf("P%d", arr[i].pid);

    for (j = arr[i].st; j < arr[i + 1].st - 1; j++)
        printf(" ");
    printf("|");
}

for (j = 0; j < tot_time - arr[n - 1].st - 1; j++)
    printf(" ");
printf("P%d", arr[n - 1].pid);
for (j = 0; j < tot_time - arr[n - 1].st - 1; j++)
    printf(" ");
printf("|");

printf("\n+");

// printing the bottom bar
for (i = 0; i < n - 1; i++)
{
    for (j = arr[i].st; j < arr[i + 1].st; j++)
        printf("--");
    printf("+");
}

for (j = 0; j < tot_time - arr[n - 1].st; j++)
    printf("--");
printf("+");

printf("\n");

// printing the time line
for (i = 0; i < n - 1; i++)
{
    printf("%d", (int)arr[i].st);
    for (j = arr[i].st; j < arr[i + 1].st; j++)
        printf(" ");
    if (arr[i].st > 9)
        printf("\b"); // backspace : remove 1 space
}

printf("%d", (int)arr[n - 1].st);
for (j = 0; j < tot_time - arr[n - 1].st; j++)
    printf(" ");

if (tot_time > 9)
    printf("\b%d", tot_time); // backspace : remove space for two digit time instances
printf("\n\n");
}

void FCFS(Process *const arr, const int size)
{
    for (int i = 0; i < size; i++)
        for (int j = i + 1; j < size; j++)
        {
            if (arr[j].at < arr[i].at) //Arrived Earlier
            {
                Process tmp = arr[j];
                arr[j] = arr[i];
                arr[i] = tmp;
            }
        }
    Process gantt[20];
    int count = 0;
    int time = 0;
    int total_time = 0;
    float tot_wt = 0, tot_tat = 0;
    for (int i = 0; i < size; i++)
    {
        arr[i].st = time;
        arr[i].et = arr[i].st + arr[i].bt;
        arr[i].wt = arr[i].st - arr[i].at;
    }
}

```



```

    for (int i = 0; i < size; i++)
        printf("| %3d | %-12.1f | %-10.1f | %-5.1f | %-4.1f | %-9.1f | %-4.1f | %-4.1f |\n",
            p[i].pid, p[i].at, p[i].bt, p[i].st, p[i].et, p[i].wt, p[i].tat, p[i].rt);
    printf("+-----+-----+-----+-----+-----+-----+-----+-----+\n");
    printf("|                                     | Total          | %-9.1f | %-4.1f |                                     |\n", tot_wt,
        tot_tat);
    printf("|                                     | Average        | %-9.1f | %-4.1f |                                     |\n", tot_wt /
        size, tot_tat / size);
    printf("+-----+-----+-----+-----+-----+-----+-----+-----+\n\n");
}

void SRTF(Process *const p, const int size)
{
    int completed = 0;
    int last_process = 0;
    int index = 0;
    int prev_id = -1;
    float tot_tat = 0;
    float tot_wt = 0;

    Process tmp;
    PQueue processQueue = createPQueue(size);
    int time = 0;
    Process gantt[20];
    int count = 0;

    int total_time = 0;
    for (int i = 0; i < size; i++)
        total_time += p[i].bt;

    while (completed < size)
    {
        for (int i = last_process; i < size; ++i)
            if (p[i].at == time)
            {
                enqueue(processQueue, p[i]);
                last_process = i + 1;
            }

        tmp = dequeue(processQueue);
        if (tmp.rem_t == -1)
        {
            printf("| - ");
            time++;
            continue;
        }
        index = getIndex(p, size, tmp);

        if (p[index].st == -1)
        { //Fresh Process
            p[index].st = time;
            p[index].rt = p[index].st - p[index].at;
            gantt[count++] = p[index];
        }

        tmp.rem_t--;
        p[index].rem_t--;
        if (p[index].pid != gantt[count - 1].pid)
        {
            gantt[count++] = p[index];
            gantt[count - 1].st = time;
        }

        if (tmp.rem_t == 0)
        {
            p[index].et = time + 1;
            p[index].tat = p[index].et - p[index].at;
            tot_tat += p[index].tat;
            p[index].wt = p[index].tat - p[index].bt;
            tot_wt += p[index].wt;
            completed++;
        }
        else
            enqueue(processQueue, p[index]);
        time++;
    }
    gantt_chart(gantt, count, total_time);

    printf("+-----+-----+-----+-----+-----+-----+-----+-----+\n");
}

```

```

printf("| PID | Arrival Time | Burst Time | Start | End | Wait Time | TAT | RT |\n");
printf("+-----+-----+-----+-----+-----+-----+-----+-----+\n");
for (int i = 0; i < size; i++)
    printf("| %3d | %-12.1f | %-10.1f | %-5.1f | %-4.1f | %-9.1f | %-4.1f | %-4.1f |\n",
        p[i].pid, p[i].at, p[i].bt, p[i].st, p[i].et, p[i].wt, p[i].tat, p[i].rt);
printf("+-----+-----+-----+-----+-----+-----+-----+-----+\n");
printf("                                | Total          | %-9.1f | %-4.1f |          |\n", tot_wt,
tot_tat);
printf("                                | Average        | %-9.1f | %-4.1f |          |\n", tot_wt /
size, tot_tat / size);
printf("+-----+-----+-----+-----+-----+-----+-----+-----+\n\n");
}

int main(void)
{
    int size;
    int choice = 5;
    do
    {
        //system("clear");
        printf("1 - FCFS\n");
        printf("2 - SJF\n");
        printf("3 - Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
            {
                printf("Enter the number of processes: ");
                scanf("%d", &size);

                Process *p = getProcesses(size);
                FCFS(p, size);
                printf("Press ENTER to continue...");
                getchar();
            }
            break;
            case 2:
            {
                printf("\n1 - Non Preemptive SJF\n");
                printf("2 - Preemptive SJF[SRTF]\n");
                printf("3 - back\n");
                printf("Enter your choice: ");
                scanf("%d", &choice);
                switch (choice)
                {
                    case 1:
                    {
                        printf("Enter the number of processes: ");
                        scanf("%d", &size);

                        Process *p = getProcesses(size);

                        SJF(p, size);
                        printf("Press ENTER to continue...");
                        getchar();
                    }
                    break;
                    case 2:
                    {
                        printf("Enter the number of processes: ");
                        scanf("%d", &size);

                        Process *p = getProcesses(size);

                        SRTF(p, size);
                        printf("Press ENTER to continue...");
                        getchar();
                    }
                    break;
                    case 3:
                    {
                        choice = 2;
                        break;
                    }
                    default:
                    {
                        printf("\nInvalid Input!\n");
                    }
                }
            }
            break;
        }
    }
    break;
}

```

```

        case 3:
            return 0;
        default:
            printf("Invalid Input\n");
    }
} while (choice != 3);
}

```

Min Heap implementation

Min heap is used to procure the process with least remaining time

```

typedef Process Data;

typedef struct PriorityQueue{
    int capacity;
    int size;
    Data* arr;
}PriorityQueue;

typedef PriorityQueue* PQueue;

int isFull(PQueue Q){
    return Q -> size == Q -> capacity;
}

int isEmpty(PQueue Q){
    return Q -> size == 0;
}

PQueue createPQueue(const int maxsize){
    PQueue tmp = (PQueue)malloc(sizeof(PriorityQueue));

    tmp -> capacity = maxsize;
    tmp -> size = 0;
    tmp -> arr = (Data*)malloc(sizeof(Data) * maxsize);

    tmp -> arr[0].rem_t = -1;
    return tmp;
}

void enqueue(PQueue q,const Data d){
    if(isFull(q)){
        //printf("Queue Full!\n");
        return;
    }
    int i = ++q -> size;
    for(; q -> arr[i/2].rem_t > d.rem_t; i /= 2){
        q -> arr[i] = q -> arr[i/2];
    }

    q -> arr[i] = d;
}

Data dequeue(PQueue q){
    if(isEmpty(q)){
        //printf("Queue Empty!\n");
        return q -> arr[0];
    }
    int i,child;
    Data min,last;

    min = q -> arr[1];
    last = q -> arr[q -> size--];

    for(i = 1; i * 2 <= q -> size ; i = child){
        child = i * 2;

        if(child != q -> size && q -> arr[child + 1].rem_t < q -> arr[child].rem_t)
            child ++;
        if(last.rem_t >= q -> arr[child].rem_t)
            q -> arr[i] = q -> arr[child];
        else
            break;
    }
}

```

```
    q -> arr[i] = last;
    return min;
}

void display(PQueue Q){
    for(int i = 1 ; i <= Q -> size ; i++)
        printf("PID :%d rem_t: %.2f\n",Q -> arr[i].pid,Q -> arr[i].rem_t);
}
```

Output

```
1 - FCFS
2 - SJF
3 - Exit
Enter your choice: 1
Enter the number of processes: 5
Enter the Arrival Time and Burst Time: 0 6
Enter the Arrival Time and Burst Time: 1 2
Enter the Arrival Time and Burst Time: 1 3
Enter the Arrival Time and Burst Time: 2 1
Enter the Arrival Time and Burst Time: 2 2
```

GANTT CHART

```
+-----+-----+-----+-----+
|      P1      | P2 |  P3  |P4| P5 |
+-----+-----+-----+-----+
0              6    8      11 12   14
```

PID	Arrival Time	Burst Time	Start	End	Wait Time	TAT	RT
1	0.0	6.0	0.0	6.0	0.0	6.0	0.0
2	1.0	2.0	6.0	8.0	5.0	7.0	5.0
3	1.0	3.0	8.0	11.0	7.0	10.0	7.0
4	2.0	1.0	11.0	12.0	9.0	10.0	9.0
5	2.0	2.0	12.0	14.0	10.0	12.0	10.0
Total					31.0	45.0	
Average					6.2	9.0	

Press ENTER to continue...

```
1 - FCFS
2 - SJF
3 - Exit
Enter your choice: 2

1 - Non Preemptive SJF
2 - Preemptive SJF[SRTF]
3 - back
```

```
Enter your choice: 1
Enter the number of processes: 5
Enter the Arrival Time and Burst Time: 0 6
Enter the Arrival Time and Burst Time: 1 2
Enter the Arrival Time and Burst Time: 1 3
Enter the Arrival Time and Burst Time: 2 1
Enter the Arrival Time and Burst Time: 2 2
```

GANTT CHART

```
+-----+-----+-----+-----+
|      P1      |P4| P2 | P5 | P3 |
+-----+-----+-----+-----+
0              6  7    9    11   14
```


PID	Arrival Time	Burst Time	Start	End	Wait Time	TAT	RT
1	0.0	6.0	0.0	6.0	0.0	6.0	0.0
2	1.0	2.0	7.0	9.0	8.0	10.0	8.0
3	1.0	3.0	11.0	14.0	10.0	13.0	10.0
4	2.0	1.0	6.0	7.0	4.0	5.0	4.0
5	2.0	2.0	9.0	11.0	5.0	7.0	5.0
			Total		27.0	41.0	
			Average		5.4	8.2	

Press ENTER to continue...

1 - FCFS

2 - SJF

3 - Exit

Enter your choice: 2

1 - Non Preemptive SJF

2 - Preemptive SJF[SRTF]

3 - back

Enter your choice: 2

Enter the number of processes: 5

Enter the Arrival Time and Burst Time: 0 6

Enter the Arrival Time and Burst Time: 1 2

Enter the Arrival Time and Burst Time: 1 3

Enter the Arrival Time and Burst Time: 2 1

Enter the Arrival Time and Burst Time: 2 2

GANTT CHART

P1	P2	P4	P5		P3		P1	
0	1	3	4	6	9			14

PID	Arrival Time	Burst Time	Start	End	Wait Time	TAT	RT
1	0.0	6.0	0.0	14.0	8.0	14.0	0.0
2	1.0	2.0	1.0	3.0	0.0	2.0	0.0
3	1.0	3.0	6.0	9.0	5.0	8.0	5.0
4	2.0	1.0	3.0	4.0	1.0	2.0	1.0
5	2.0	2.0	4.0	6.0	2.0	4.0	2.0
			Total		16.0	30.0	
			Average		3.2	6.0	

Press ENTER to continue...

1 - FCFS

2 - SJF

3 - Exit

Enter your choice: 3