Lab Exercise 4: Implementation of CPU Scheduling Policies: Priority(Preemptive and Non Preemptive) and Round Robin

Mahesh Bharadwaj K - 185001089

February 7, 2020

Program

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

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

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

    for (int i = 0; i < size; i++)
    {
        printf("Enter the Arrival Time, Burst Time & Priority: ");
        scanf("%f %f %d", &p[i].at, &p[i].bt, &p[i].pri);
        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 sortArrival(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;
            }
        }
}

void RR(Process *const p, const int size)
{
    sortArrival(p, size);
    int time = 0;

    float tot_tat = 0;
    float tot_wt = 0;
    int total_time = 0;
    for (int i = 0; i < size; i++)
        total_time += p[i].bt;
    int q = 2;
    int i = 0;
    int completed = 0;
    Process gantt[20];
    int count = 0;

    while (completed != size)
```

```
    {
        if (p[i].rem_t)
        {
            if (p[i].rem_t == p[i].bt)
            {
                p[i].st = time;
                p[i].rt = p[i].st - p[i].at;
                gantt[count++] = p[i];
            }
            if (p[i].rem_t > q)
            {
                p[i].rem_t -= q;
                if (gantt[i].pid != gantt[count - 1].pid)
                {
                    gantt[count++] = p[i];
                    gantt[count - 1].st = time;
                }
                time = time + q;
            }

            else
            {
                if (gantt[i].pid != gantt[count - 1].pid)
                {
                    gantt[count++] = p[i];
                    gantt[count - 1].st = time;
                }
                completed++;
                time += p[i].rem_t;
                p[i].et = time;
                p[i].rem_t = 0;
                p[i].tat = p[i].et - p[i].at;
                p[i].wt = p[i].tat - p[i].bt;
                tot_tat += p[i].tat;
                tot_wt += p[i].wt;
            }
        }

        i = (i + 1) % size;
    }
    gantt_chart(p, size, total_time);

    printf("+-----+-------------+------------+-----+-------+------+-----------+------+------+\n");
    printf("| PID | Arrival Time | Burst Time | PRI | Start | End  | Wait Time | TAT  | RT   |\n");
    printf("+-----+-------------+------------+-----+-------+------+-----------+------+------+\n");
    for (int i = 0; i < size; i++)
        printf("| %3d | %-12.1f | %-10.1f | %3d | %-5.1f | %-4.1f | %-9.1f | %-4.1f | %-4.1f |\n",
               p[i].pid, p[i].at, p[i].bt, p[i].pri, 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 priority_np(Process arr[], const int size)
{
    int time = 0;
    float avgturn = 0, avgwait = 0;
    int min;
    int index;
    Process gantt[20];
    int count = 0;
    float tot_tat = 0, tot_wt = 0;

    time = 0;
    for (int i = 0; i < size; i++)
        time += arr[i].bt;

    for (int t = 0; t < time;)
    {
        min = 9999;
        for (int i = 0; i < size; i++)
        {
            if (arr[i].at <= t && arr[i].pri < min && arr[i].rem_t)
            {
                min = arr[i].pri;
```

```
                index = i;
            }
        }

        arr[index].rem_t = 0;
        arr[index].wt = t - arr[index].at;
        arr[index].st = t;
        arr[index].rt = arr[index].st - arr[index].at;
        t += arr[index].bt;
        arr[index].et = t;
        arr[index].tat = arr[index].et - arr[index].at;
        tot_tat += arr[index].tat;
        tot_wt += arr[index].wt;
        gantt[count++] = arr[index];
    }

    gantt_chart(gantt, size, time);

    printf("+-----+--------------+------------+-----+-------+------+-----------+------+------+\n");
    printf("| PID | Arrival Time | Burst Time | PRI | Start | End  | Wait Time | TAT  | RT   |\n");
    printf("+-----+--------------+------------+-----+-------+------+-----------+------+------+\n");
    for (int i = 0; i < size; i++)
        printf("| %3d | %-12.1f | %-10.1f | %3d | %-5.1f | %-4.1f | %-9.1f | %-4.1f | %-4.1f |\n",
               arr[i].pid, arr[i].at, arr[i].bt, arr[i].pri, arr[i].st, arr[i].et, arr[i].wt, arr[i
    ].tat, arr[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 priority_p(Process *const arr, const int size)
{
    int time = 0;
    float tot_tat = 0, tot_wt = 0;
    int min;
    int index;
    int prev = -1;
    time = 0;
    int completed = 0;

    for (int i = 0; i < size; i++)
        time += arr[i].bt;

    Process gantt[20];
    int count = 0;

    for (int t = 0; completed != size; t++)
    {
        min = 9999;
        for (int i = 0; i < size; i++)
        {
            if (arr[i].at <= t && arr[i].pri < min && arr[i].rem_t > 0)
            {
                min = arr[i].pri;
                index = i;
            }
        }
        if (arr[index].rem_t == arr[index].bt)
        {
            arr[index].st = t;
            arr[index].rt = arr[index].st - arr[index].at;
            gantt[count++] = arr[index];
        }

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

        if (arr[index].rem_t == 0)
        {
            completed++;
            arr[index].et = t + 1;
```

```
            arr[index].tat = arr[index].et - arr[index].at;
            arr[index].wt = arr[index].tat - arr[index].bt;
            tot_tat += arr[index].tat;
            tot_wt += arr[index].wt;
        }
    }
    gantt_chart(gantt,size,time);

    printf("+-----+--------------+------------+-----+-------+------+-----------+------+------+\n");
    printf("| PID | Arrival Time | Burst Time | PRI | Start | End  | Wait Time | TAT  | RT   |\n");
    printf("+-----+--------------+------------+-----+-------+------+-----------+------+------+\n");
    for (int i = 0; i < size; i++)
        printf("| %3d | %-12.1f | %-10.1f | %3d | %-5.1f | %-4.1f | %-9.1f | %-4.1f | %-4.1f |\n",
               arr[i].pid, arr[i].at, arr[i].bt, arr[i].pri, arr[i].st, arr[i].et, arr[i].wt, arr[i
    ].tat, arr[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
    {
        printf("1 - Round Robin\n");
        printf("2 - Priority\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);
            RR(p, size);
            printf("Press ENTER to continue...");
            getchar();
        }
        break;
        case 2:
        {

            printf("\n1 - Non Preemptive Priority\n");
            printf("2 - Preemptive Priority\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);

                priority_np(p, size);
                printf("Press ENTER to continue...");
                getchar();
            }
            break;
            case 2:
            {

                printf("Enter the number of processes: ");
                scanf("%d", &size);

                Process *p = getProcesses(size);
```

```
                priority_p(p, size);
                printf("Press ENTER to continue...");
                getchar();
            }
            case 3:
                choice = 2;
                break;
            default:
                printf("\nInvalid Input!\n");
            }
        }
        break;
        case 3:
            return 0;
        default:
            printf("Invalid Input\n");
        }
    } while (choice != 3);
}
```

Output

```
1 - Round Robin
2 - Priority
3 - Exit
Enter your choice: 1
Enter the number of processes: 5
Enter the Arrival Time, Burst Time & Priority: 0 6 2
Enter the Arrival Time, Burst Time & Priority: 1 2 2
Enter the Arrival Time, Burst Time & Priority: 1 3 4
Enter the Arrival Time, Burst Time & Priority: 2 1 1
Enter the Arrival Time, Burst Time & Priority: 2 2 3


GANTT CHART

+----+----+----+--+--------------+
| P1 | P2 | P3 |P4|      P5      |
+----+----+----+--+--------------+
0    2    4    6  7             14

+-----+--------------+------------+-----+-------+------+-----------+------+------+
| PID | Arrival Time | Burst Time | PRI | Start | End  | Wait Time | TAT  | RT   |
+-----+--------------+------------+-----+-------+------+-----------+------+------+
|   1 | 0.0          | 6.0        |   2 | 0.0   | 14.0 | 8.0       | 14.0 | 0.0  |
|   2 | 1.0          | 2.0        |   2 | 2.0   | 4.0  | 1.0       | 3.0  | 1.0  |
|   3 | 1.0          | 3.0        |   4 | 4.0   | 12.0 | 8.0       | 11.0 | 3.0  |
|   4 | 2.0          | 1.0        |   1 | 6.0   | 7.0  | 4.0       | 5.0  | 4.0  |
|   5 | 2.0          | 2.0        |   3 | 7.0   | 9.0  | 5.0       | 7.0  | 5.0  |
+-----+--------------+------------+-----+-------+------+-----------+------+------+
|                                    | Total        | 26.0      | 40.0 |      |
|                                    | Average      | 5.2       | 8.0  |      |
+-----+--------------+------------+-----+-------+------+-----------+------+------+

Press ENTER to continue...
1 - Round Robin
2 - Priority
3 - Exit
Enter your choice: 2

1 - Non Preemptive Priority
2 - Preemptive Priority
3 - back
Enter your choice: 1
Enter the number of processes: 5
```

```
Enter the Arrival Time, Burst Time & Priority: 0 6 2
Enter the Arrival Time, Burst Time & Priority: 1 2 2
Enter the Arrival Time, Burst Time & Priority: 1 3 4
Enter the Arrival Time, Burst Time & Priority: 2 1 1
Enter the Arrival Time, Burst Time & Priority: 2 2 3

GANTT CHART

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

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

Press ENTER to continue...
1 - Round Robin
2 - Priority
3 - Exit
Enter your choice: 2

1 - Non Preemptive Priority
2 - Preemptive Priority
3 - back
Enter your choice: 2
Enter the number of processes: 5
Enter the Arrival Time, Burst Time & Priority: 0 6 2
Enter the Arrival Time, Burst Time & Priority: 1 2 2
Enter the Arrival Time, Burst Time & Priority: 1 3 4
Enter the Arrival Time, Burst Time & Priority: 2 1 1
Enter the Arrival Time, Burst Time & Priority: 2 2 3


GANTT CHART

+----+--+--------+----+----------+
| P1 |P4|   P1   | P2 |    P5    |
+----+--+--------+----+----------+
0    2  3        7    9          14

+-----+--------------+------------+-----+-------+------+-----------+------+------+
| PID | Arrival Time | Burst Time | PRI | Start | End  | Wait Time | TAT  | RT   |
+-----+--------------+------------+-----+-------+------+-----------+------+------+
|   1 | 0.0          | 6.0        |   2 | 0.0   | 7.0  | 1.0       | 7.0  | 0.0  |
|   2 | 1.0          | 2.0        |   2 | 7.0   | 9.0  | 6.0       | 8.0  | 6.0  |
|   3 | 1.0          | 3.0        |   4 | 11.0  | 14.0 | 10.0      | 13.0 | 10.0 |
|   4 | 2.0          | 1.0        |   1 | 2.0   | 3.0  | 0.0       | 1.0  | 0.0  |
|   5 | 2.0          | 2.0        |   3 | 9.0   | 11.0 | 7.0       | 9.0  | 7.0  |
+-----+--------------+------------+-----+-------+------+-----------+------+------+
|                                    | Total        | 24.0      | 38.0 |      |
|                                    | Average      | 4.8       | 7.6  |      |
+-----+--------------+------------+-----+-------+------+-----------+------+------+
```