UCS1411 Operating Systems Lab
AY: 2021-22

Name: Krithika Swaminathan
Roll No.: 205001057

Assignment 4 – Implementation of CPU Scheduling Policies: Priority and Round Robin

Date: 04/04/2022

Aim:

To develop a menu driven C program to implement the CPU Scheduling Algorithms Priority (Non-Preemptive and Preemptive) and Round Robin.

Algorithm:

1. Start
2. Declare a storage structure
3. Create a menu with options for the following
    1. Round Robin
        - Get the arrival time and the burst time of each process as input from the user.
        - Set the waiting time for all the processes to be 0.
        - Repeat the following steps until all the processes have been used. Start with the first process on the list.
        - Reduce its remaining type by 1.
        - Check if its remaining time becomes 0. If yes, increment the counter for process completion and move on to the next process.
        - Check if the allotted quantum for the process has expired. If yes, save the remaining time for the process and move on to the next one.
        - If the last process has exhausted its time, lol.

    2. Priority
        - Get the arrival time, the burst time and priority of each process as input from the user.
        - Set the waiting time for all the processes to be 0.
        - Repeat the following steps until all the processes have been used. Start with the first process on the list.
            - Find the process with the maximum priority.
            - Reduce its remaining type by 1.
            - Check if its remaining time becomes 0. If yes, increment the counter for process completion and move on to the next process.
            - Check if the allotted quantum for the process has expired. If yes, save the remaining time for the process and move on to the next one.

- Increase time lapsed by 1.

- Calculate the turnaround time for each process, which is the sum of the respective burst and waiting times.
- Compute the average waiting time and the average turnaround time.

4. Stop

Code:

```
//Program to understand CPU Scheduling processes
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_CAP 10
#define MAX_LEN 3
#define LIM 999

struct Schedule {
	char pID[3];
	int atime, btime, ttime, wtime;
	int priority;
	};

void inputProcesses (struct Schedule[],int);
void prioritySort (struct Schedule[],int);
void roundrobin (struct Schedule[],int,int);
void printTablePrty (struct Schedule[],int);
void printGantt (struct Schedule[],int);
void printLine();
void printDashLine();
void printShortLine(int);

int main() {
	printf("\n\t\tCPU SCHEDULING ALGORITHMS\n\n");

	printf("_____MENU_____\n\n");
	printf("1. ROUND ROBIN\n2. PRIORITY\n\n");

	int choice = 1;
	while (choice != 0) {
		printf("\nEnter choice: ");
		scanf("%d",&choice);

		switch(choice) {
			case 1: {
				printf("\n__ROUND ROBIN__\n");

				char pID[10][3], porder[MAX_CAP][3]; int ptime[MAX_CAP],
pcount = 0;
```

```
int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, rst=0, at[10], bt[10], temp[10], rt[10], flag[10];
float avg_wt, avg_tat, avg_rt;

printf("Enter no. of processes: ");
scanf("%d",&NOP);

printf("Enter time quantum: ");
scanf("%d",&quant);

//get process details
for (i=0; i<NOP; i++) {
	printf("\nEnter processID: ");
	scanf("%s",pID[i]);

	printf("Enter arrival time: ");
	scanf("%d",&at[i]);

	printf("Enter burst time: ");
	scanf("%d",&bt[i]);

	temp[i] = bt[i];
	flag[i] = 0;
	}

y = NOP;

printf("\n__No. of processes: %d__\n",NOP);
printLine();
printf("| pID\t|A_time\t|B_time\t|W_time\t|T_time\t|R_time\t|\n");
printDashLine();
for (sum=0, i=0; y!=0; ) {
	if (temp[i] <= quant && temp[i] > 0) {
		if (flag[i]==0) {
			rt[i] = sum;
			flag[i] = 1;
			}
		sum = sum + temp[i];
		strcpy(porder[pcount],pID[i]);
		ptime[pcount] = sum;
		pcount++;
		temp[i] = 0;
		count=1;
		}
	else if(temp[i] > 0) {
		if (flag[i]==0) {
			rt[i] = sum;
			flag[i] = 1;
			}
```

```
                                        temp[i] = temp[i] - quant;
                                        sum = sum + quant;
                                        strcpy(porder[pcount],pID[i]);
                                        ptime[pcount] = sum;
                                        pcount++;
                                        }


                                if(temp[i]==0 && count==1) {
                                        y--;
                                        printf("| %s\t| %d\t| %d\t| %d\t| %d\t| %d\t|\
n",pID[i],at[i],bt[i],sum-at[i]-bt[i],sum-at[i],rt[i]);
                                        wt = wt+sum-at[i]-bt[i];
                                        tat = tat+sum-at[i];
                                        rst = rst+rt[i];
                                        count =0;
                                        }

                                if(i==NOP-1)
                                        i=0;
                                else if(at[i+1]<=sum)
                                        i++;
                                else
                                        i=0;
                        }

                        avg_wt = wt * 1.0/NOP;
                        avg_tat = tat * 1.0/NOP;
                        avg_rt = rst * 1.0/NOP;
                        printf("|_____________________________________________________|\
n");
                        printf("| \tAvg. waiting time: %.2f\t\t\t|\n",avg_wt);
                        printf("| \tAvg. turnaround time: %.2f\t\t|\n",avg_tat);
                        printf("| \tAvg. response time: %.2f\t\t|\n",avg_rt);
                        printf("|_____________________________________________________|\
n");
                        printf("\n");

                        printf("\nGantt chart:\n");
                        printShortLine(pcount);
                        printf("| |");
                        for (int i=0; i<pcount; i++) {
                                printf("  %s  | |",porder[i]);
                                }
                        printf("\n");
                        printShortLine(pcount);
                        printf(" 0 ");
                        for (int i=0; i<pcount; i++) {
                                printf("       %d",ptime[i]);
```

```
                        }
                printf("\n\n");

                break;
                }

        case 2: {
                printf("\n__PRIORITY__\n");
                struct Schedule pr[MAX_CAP];

                int numP;
                printf("Enter no. of processes: ");
                scanf("%d",&numP);

                //get process details
                inputProcesses(pr,numP);
                //sort according to priority
                prioritySort(pr,numP);
                //print results
                printTablePrty(pr,numP);

                break;
                }

        case 0: {
                printf("Exiting menu...\n\n");
                exit(0); break;
                }

        default: printf("Invalid choice!\n"); break;
        }
    }

    return 0;
    }

void inputProcesses (struct Schedule sc[], int numP) {
    for (int i=0; i<numP; i++) {
        printf("\nEnter processID: ");
        scanf("%s",sc[i].pID);

        printf("Enter arrival time: ");
        scanf("%d",&sc[i].atime);

        printf("Enter burst time: ");
        scanf("%d",&sc[i].btime);

        printf("Enter priority: ");
        scanf("%d",&sc[i].priority);
```

```
                }
        }

void printTablePrty (struct Schedule sc[], int numP) {
        printf("\n__No. of processes: %d__\n",numP);
        float wsum = 0, tsum = 0;
        printLine();
        printf("| pID\t|A_time\t|B_time\t|Prty\t|W_time\t|T_time\t|\n");
        printDashLine();
        for (int i=0; i<numP; i++) {
                printf("| %s\t| %d\t| %d\t| %d\t| %d\t| %d\t|\
n",sc[i].pID,sc[i].atime,sc[i].btime,sc[i].priority,sc[i].wtime,sc[i].ttime);

                wsum += sc[i].wtime;
                tsum += sc[i].ttime;
                }
        printf("|_____________________________________________________|\n");
        printf("| \tAvg. waiting time: %.2f\t\t\t|\n",wsum/numP);
        printf("| \tAvg. turnaround time: %.2f\t\t|\n",tsum/numP);
        printf("|_____________________________________________________|\n");
        printf("\n");
        }

void prioritySort (struct Schedule sc[], int numP) {
        sc[MAX_CAP-1].priority = LIM;
        int i, smallest, count = 0, time, burst[MAX_CAP];
        char porder[MAX_CAP][3]; int ptime[MAX_CAP], pcount = 0, save = MAX_CAP-1;

        for (i=0; i<numP; i++)
                burst[i] = sc[i].btime;

        for (time=0; count!=numP; time++) {
                smallest = MAX_CAP-1;
                for (i=0; i<numP; i++) {
                        if (sc[i].atime<time && sc[i].priority<sc[smallest].priority && sc[i].btime>0)
{
                                smallest = i;
                                }
                        }
                sc[smallest].btime--;

                if (strcmp(sc[smallest].pID,sc[save].pID) != 0) {
                                        strcpy(porder[pcount],sc[smallest].pID);
                                        ptime[pcount] = time;
                                        pcount++;
                                        }
                save = smallest;

                if (sc[smallest].btime == 0) {
```

```
                        count++;
                        sc[smallest].wtime = time - sc[smallest].atime - burst[smallest];
                        sc[smallest].ttime = time - sc[smallest].atime;
                        }
                }

        for (i=0; i<numP; i++)
                sc[i].btime = burst[i];

        //print Gantt chart;
        printf("\nGantt chart:\n");
        printShortLine(pcount);
        printf("| |");
        for (int i=0; i<pcount; i++) {
                printf("  %s  | |",porder[i]);
                }
        printf("\n");
        printShortLine(pcount);
        printf(" 0 ");
        for (int i=0; i<pcount; i++) {
                printf("       %d",ptime[i]);
                }
        printf("\n\n");

        }

void roundrobin (struct Schedule sc[], int numP, int time_quantum) {
        int i, count, time, remain = numP, flag = 0, rt[MAX_CAP];
        printf("here\n");

        for (i=0; i<numP; i++) {
                rt[i] = sc[i].btime;
                }

        for (time=0, count=0; remain!=0; ) {
                printf("infor\n");
                if (rt[count]<=time_quantum && rt[count]>0) {
                        printf("inif\n");
                        time += rt[count];
                        rt[count] = 0;
                        flag = 1;
                        }
                else if (rt[count]>0) {
                        printf("inelseif\n");
                        rt[count] -= time_quantum;
                        time += time_quantum;
                        }
                else
                        printf("cannot\n");
```

```
                if (rt[count] && flag==1) {
                        printf("inifflag\n");
                        remain--;
                        sc[count].wtime = time - sc[count].atime - sc[count].btime;
                        sc[count].ttime = time - sc[count].atime;
                        flag = 0;
                        }

                if (count == numP-1) {
                        printf("inctif\n");
                        count = 0;
                        }
                else if (sc[count+1].atime <= time) {
                        printf("inctelseif\n");
                        count++;
                        }
                else {
                        printf("inctelse\n");
                        count = 0;
                        }
                printf("therefor\n");
                }

        }

void printLine() {
        printf(" _____\n");
        }

void printDashLine() {
        printf(" ----------\n");
        }

void printShortLine (int n) {
        for (int i=0; i<n; i++)
                printf("----------");
        printf("\n");
        }
```

Name: Krithika Swaminathan
Roll No.: 205001057

Output:

Round robin:

```
kri@kri-ubuntu:~/workspace$ gcc -o prr cpu_pr_rr.c
kri@kri-ubuntu:~/workspace$ ./prr

                CPU SCHEDULING ALGORITHMS

_____MENU_____

1. ROUND ROBIN
2. PRIORITY


Enter choice: 1

__ROUND ROBIN__
Enter no. of processes: 5
Enter time quantum: 5

Enter processID: p1
Enter arrival time: 0
Enter burst time: 10

Enter processID: p2
Enter arrival time: 0
Enter burst time: 1

Enter processID: p3
Enter arrival time: 0
Enter burst time: 2

Enter processID: p4
Enter arrival time: 0
Enter burst time: 1

Enter processID: p5
Enter arrival time: 0
Enter burst time: 5
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
__No. of processes: 5__
 _____________________________________________
| pID   |A_time |B_time |W_time |T_time |R_time |
 ---------------------------------------------
| p2    | 0     | 1     | 5     | 6     | 5     |
| p3    | 0     | 2     | 6     | 8     | 6     |
| p4    | 0     | 1     | 8     | 9     | 8     |
| p5    | 0     | 5     | 9     | 14    | 9     |
| p1    | 0     | 10    | 9     | 19    | 0     |
|_____________________________________________|
|       Avg. waiting time: 7.40               |
|       Avg. turnaround time: 11.20           |
|       Avg. response time: 5.60              |
|_____________________________________________|


Gantt chart:
-------------------------------------------------------------
| |  p1  | |  p2  | |  p3  | |  p4  | |  p5  | |  p1  | |
-------------------------------------------------------------
 0        5       6       8       9       14       19
```

Name: Krithika Swaminathan
Roll No.: 205001057

Priority:

```
Enter choice: 2

__PRIORITY__
Enter no. of processes: 3

Enter processID: p1
Enter arrival time: 0
Enter burst time: 5
Enter priority: 3

Enter processID: p2
Enter arrival time: 1
Enter burst time: 3
Enter priority: 1

Enter processID: p3
Enter arrival time: 2
Enter burst time: 2
Enter priority: 2

Gantt chart:
----------------------------------------
| |  p1  | |  p2  | |  p3  | |  p1  | |
----------------------------------------
 0        1       4       6       10


__No. of processes: 3__
 _____________________________________________
| pID   |A_time |B_time |Prty   |W_time |T_time |
 ---------------------------------------------
| p1    | 0     | 5     | 3     | 5     | 10    |
| p2    | 1     | 3     | 1     | 0     | 3     |
| p3    | 2     | 2     | 2     | 2     | 4     |
|_____________________________________________|
|       Avg. waiting time: 2.33               |
|       Avg. turnaround time: 5.67            |
|_____________________________________________|
```

Exiting:

```
Enter choice: 0
Exiting menu...
```

Name: Krithika Swaminathan
Roll No.: 205001057

Learning outcomes:

- The various CPU Scheduling algorithms were understood.
- The CPU Scheduling algorithms, i.e., Round Robin and Priority were implemeted in C.
- Pre-emptive scheduling was understood and implemeted in C.
