UCS1411 Operating Systems Lab
AY: 2021-22

Name: Krithika Swaminathan
Roll No.: 205001057

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

Date: 21/03/2022

Aim:

To develop a menu-driven C program to implement the CPU Scheduling Algorithms FCFS, SJF and SRTF.

Algorithm:

1. Start
2. Declare a structure with elements such as the process name, arrival time, burst time, waiting time and turnaround time.
3. Create a menu with options for the following:
    a) FCFS
    b) SJF
    c) SRTF

    1. FCFS:
        1. Get the details of the processes as input from the user. This includes the arrival time and the burst time of each process.
        2. As the first process does not need to wait, assign the waiting time for the first process as zero.
        3. For every subsequent, set the current waiting time as the sum of the burst time and the waiting time of the previous process.
        4. For the wording target, set the turnaround time as the sum of the waiting time and the burst time.
        5. Compute the average of the total waiting time and the turnaround time for all the processes.

    2. SJF:
        1. Get the details of the processes as input from the user. This includes the arrival time and the burst time of each process.
        2. Sort the processes in the increasing order of their burst times.
        3. As the first process does not need to wait, assign the waiting time for the first process as zero.

4. For every subsequent project, set the current waiting time as the sum of the burst time and the waiting time of the previous process.
5. For every process, set the turnaround time as the sum of the waiting time and the burst time.
6. Compute the average of the total waiting time and the turnaround time for all the processes.

3. SRTF:
    1. Repeat the following steps until all the processes have been completed:
        1. Find the process with the minimum remaining time at every single time lap.
        2. Reduce its time by 1.
        3. Check if its remaining time becomes 0.
        4. Increment the counter for process completion.
        5. Calculate the waiting time and turnaround time for each completed process.
        6. Increment time lapsed by 1.
    2. Compute the average of the total waiting time and the turnaround time.

4. Stop

Code:

```
/*C Program to understand and implement 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;
        };

void inputProcesses (struct Schedule[],int);
void burstSort (struct Schedule[],int);
void preemptiveSort (struct Schedule[],int);
void printArray (int[],int);
void calcTimes (struct Schedule[],int);
void printTable (struct Schedule[],int);
void printGantt (struct Schedule[],int);
```

```
void printLine();
void printDashLine();
void printShortLine();

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

	printf("_____MENU_____\n\n");
	printf("1. FCFS\n2. SJF\n3. SRTF\n\n");

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

		switch(choice) {
			case 1: {
				printf("\n__FCFS__\n");
				struct Schedule fcfs[MAX_CAP];

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

				//get process details
				inputProcesses(fcfs,numP);
				//calculate waiting time and turnabout time
				calcTimes(fcfs,numP);
				//print results
				printTable(fcfs,numP);
				//print gantt chart
				printGantt(fcfs,numP);

				break;
				}

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

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

				//get process details
				inputProcesses(sjf,numP);
				//sort according to burst time
				burstSort(sjf,numP);
				//calculate wtime and ttime
```

```
                        calcTimes(sjf,numP);
                        //print results
                        printTable(sjf,numP);
                        //print gantt chart
                        printGantt(sjf,numP);

                        break;
                        }

                case 3: {
                        printf("\n__SRTF__\n");
                        struct Schedule srtf[MAX_CAP];

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

                        //get process details
                        inputProcesses(srtf,numP);
                        //sort according to shortest remaining time
                        preemptiveSort(srtf,numP);
                        //print results
                        printTable(srtf,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);
```

```
                }
        }

void calcTimes (struct Schedule sc[], int numP) {
        int wait = 0;
        for (int i=0; i<numP; i++) {
                sc[i].wtime = wait - sc[i].atime;
                if (sc[i].wtime<0) {
                        sc[i].wtime = 0;
                        wait = 0;
                        }
                wait += sc[i].btime;
                sc[i].ttime = sc[i].btime + sc[i].wtime;
                }
        }

void printTable (struct Schedule sc[], int numP) {
        printf("\n__No. of processes: %d__\n",numP);
        float wsum = 0, tsum = 0;
        printLine();
        printf("pID\tA_time\tB_time\tW_time\tT_time\n");
        printDashLine();
        for (int i=0; i<numP; i++) {
                printf("%s\t%d\t%d\t%d\t%d\
n",sc[i].pID,sc[i].atime,sc[i].btime,sc[i].wtime,sc[i].ttime);

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

void printGantt (struct Schedule sc[], int numP) {
        printf("\nGantt chart:\n");
        int time = 0;
        printShortLine();
        printf("|%d|",time);
        for (int i=0; i<numP; i++) {
                time += sc[i].btime;
                printf("%s|%d|",sc[i].pID,time);
                }
        printf("\n");
        printShortLine();
        printf("\n");
```

```
	}

void burstSort (struct Schedule sc[], int n) {
	for (int i=0; i<n-1; i++) {
		int min_idx = i;
		for (int j=i+1; j<n; j++)
			if (sc[j].btime < sc[min_idx].btime)
				min_idx = j;

	struct Schedule temp = sc[min_idx];
		sc[min_idx] = sc[i];
		sc[i] = temp;
		}
	}

void preemptiveSort (struct Schedule sc[], int numP) {
	sc[MAX_CAP-1].btime = LIM;
	int i, smallest, count = 0, newcount = 0, time, context = 0, burst[MAX_CAP];
	//struct Schedule newsc[MAX_CAP];

	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].btime<sc[smallest].btime && sc[i].btime>0) {
				smallest = i;
				/*printf("%d,%d,%d,%s\n",time,i,smallest,sc[smallest].pID);
				newsc[newcount] = sc[smallest];
				newsc[newcount].btime = time - context;
				newcount++;
				context = time;*/
				}
			}
		sc[smallest].btime--;

		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];

	//printGantt(newsc,newcount);
	}
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
void printArray (int arr[], int size) {
        for (int i=0; i < size; i++)
                printf("%d ",arr[i]);
        printf("\n");
        }

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

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

void printShortLine() {
        printf("----------\n");
        }
```

Name: Krithika Swaminathan
Roll No.: 205001057

Output:

```
kri@kri-ubuntu:~/workspace$ gcc -o cpu cpumenu.c
kri@kri-ubuntu:~/workspace$ ./cpu

                CPU SCHEDULING ALGORITHMS

_____MENU_____

1. FCFS
2. SJF
3. SRTF


Enter choice: 1

__FCFS__
Enter no. of processes: 3

Enter processID: p1
Enter arrival time: 0
Enter burst time: 24

Enter processID: p2
Enter arrival time: 0
Enter burst time: 3

Enter processID: p3
Enter arrival time: 0
Enter burst time: 3

__No. of processes: 3__
_________________________________________
pID     A_time  B_time  W_time  T_time
-----------------------------------------
p1      0       24      0       24
p2      0       3       24      27
p3      0       3       27      30
_________________________________________
        Avg. waiting time: 17.00
        Avg. turnaround time: 27.00
_________________________________________


Gantt chart:
------------------------
|0|p1|24|p2|27|p3|30|
------------------------
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 1

__FCFS__
Enter no. of processes: 0

__No. of processes: 0__
_________________________________________
pID     A_time  B_time  W_time  T_time
-----------------------------------------

_________________________________________
        Avg. waiting time: -nan
        Avg. turnaround time: -nan
_________________________________________


Gantt chart:
------------------------
|0|
------------------------


Enter choice: 1

__FCFS__
Enter no. of processes: 3

Enter processID: p1
Enter arrival time: 0
Enter burst time: 24

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

Enter processID: p3
Enter arrival time: 2
Enter burst time: 3

__No. of processes: 3__
_________________________________________
pID     A_time  B_time  W_time  T_time
-----------------------------------------
p1      0       24      0       24
p2      1       3       23      26
p3      2       3       25      28
_________________________________________
        Avg. waiting time: 16.00
        Avg. turnaround time: 26.00
_________________________________________


Gantt chart:
------------------------
|0|p1|24|p2|27|p3|30|
------------------------
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 2

__SJF__
Enter no. of processes: 4

Enter processID: p1
Enter arrival time: 0
Enter burst time: 6

Enter processID: p2
Enter arrival time: 0
Enter burst time: 8

Enter processID: p3
Enter arrival time: 0
Enter burst time: 7

Enter processID: p4
Enter arrival time: 0
Enter burst time: 3

__No. of processes: 4__
_____________________________________________
pID     A_time  B_time  W_time  T_time
---------------------------------------------
p4      0       3       0       3
p1      0       6       3       9
p3      0       7       9       16
p2      0       8       16      24
_____________________________________________
        Avg. waiting time: 7.00
        Avg. turnaround time: 13.00
_____________________________________________


Gantt chart:
------------------------
|0|p4|3|p1|9|p3|16|p2|24|
------------------------
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 2

__SJF__
Enter no. of processes: 4

Enter processID: p1
Enter arrival time: 0
Enter burst time: 1

Enter processID: p2
Enter arrival time: 1
Enter burst time: 5

Enter processID: p3
Enter arrival time: 2
Enter burst time: 9

Enter processID: p4
Enter arrival time: 3
Enter burst time: 6

__No. of processes: 4__
_____________________________________________
pID     A_time  B_time  W_time  T_time
---------------------------------------------
p1      0       1       0       1
p2      1       5       0       5
p4      3       6       3       9
p3      2       9       10      19
_____________________________________________
        Avg. waiting time: 3.25
        Avg. turnaround time: 8.50
_____________________________________________


Gantt chart:
------------------------
|0|p1|1|p2|6|p4|12|p3|21|
------------------------
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 3

__SRTF__
Enter no. of processes: 4

Enter processID: p1
Enter arrival time: 0
Enter burst time: 8

Enter processID: p2
Enter arrival time: 1
Enter burst time: 4

Enter processID: p3
Enter arrival time: 2
Enter burst time: 9

Enter processID: p4
Enter arrival time: 3
Enter burst time: 5

__No. of processes: 4__
_________________________________________
pID     A_time  B_time  W_time  T_time
-----------------------------------------
p1      0       8       9       17
p2      1       4       0       4
p3      2       9       15      24
p4      3       5       2       7
_________________________________________
        Avg. waiting time: 6.50
        Avg. turnaround time: 13.00
_________________________________________
```

Name: Krithika Swaminathan
Roll No.: 205001057

Learning outcomes:

- The various CPU Scheduling algorithms were understood.
- The CPU Scheduling algorithms, i.e., FCFS, SJF and SRTF were implemeted in C.
- Non-preemptive scheduling was understood.
