TP n°3: Task scheduling

1. Objectives

This practical work has the following objectives:

- Understand how the main scheduling algorithms work,
- Familiarize yourself with the programming environment for scheduling algorithms,
- Know the performance of scheduling algorithms.

2. Scheduling

The operating system must allow all applications and all users to work at the same time, in other words give everyone the impression that they are the only ones using the computer and its physical resources. This complex management of processes is carried out by a specific part of the kernel: the scheduler.

In operating systems, the scheduler refers to the component of the operating system kernel that chooses the order in which processes are executed on a computer's processors.

Switching from one process to another is called context switching.

2.1 Scheduling concept

As a resource (the processor or a peripheral) cannot be shared, it is its time of use that will be shared; the time of use of a resource is divided into very short intervals, during which the scheduler allocates it to a single user.

The scheduler enables to:

- minimize the processing time of a user's process,
- guarantee fairness between different users,
- optimize resource use,
- avoid blockages.

2.2 Types of scheduling algorithms

Several scheduling algorithms are possible, the most common being:

- **Round Robin**: the resource is allocated to each process in turn. For simultaneous execution of processes, it is the speed of this round robin that will give each user the impression that their process is the only one using the processor. This old method has the advantages of simplicity, rapid management and robustness.
- **Priority**: the order in which the resource is allocated will then depend on the priority of the task. This method is very fair, but defining the priority level of the task must be objective. In UNIX this concept is called niceness.
- **FCFS** "First Come, First Served" management. The most obvious example of this algorithm is the document print queue on a printer.

- **SJF** “Shortened Job First” algorithm: very effective in satisfying users, but it is not always easy to assess the execution time of a task before it begins. Rather than measuring execution times, we can limit ourselves to the number of instructions to be carried out.
- **SRTF** (Shortest Remaining Time First) scheduling: can be seen as the preemptive version of SJF scheduling. As soon as the processor is requisitioned, the process is elected.

2.3 Performance criteria

The performance of a scheduling algorithm can be measured according to criteria such as:

- The turnaround time of a process, or dwell time or overall execution time, corresponds to the difference between the arrival time and the termination time (or the sum of the dwell time and the execution time).
- The waiting time of a process or waiting time is the difference between the turnaround time and the execution time per process.

Exercises

Exercise n°1

1) In this exercise you will be requested to implement the FCFS scheduling algorithm and calculate the waiting time and turnaround time for each process. It is assumed that all processes have arrived at the same time and the execution order of the processes is as follow: P_1, P_2, P_3, …..

   For the FCFS scheduling algorithm, read the number of processes in the system and their CPU execution times. Scheduling is carried out on the basis of the arrival time of the processes, independently of their other parameters. Each process will be executed according to its arrival time.

   FCFS is executed using the following algorithm:

```
1. Begin
2. Declare the array size wt[10], bt[10] and tt[10] where:
   - wt: is an array containing the waiting time of each process,
   - bt: is an array containing the execution time of each process,
   - tt: is an array containing the turnaround time of each process,
3. Read the number of processes to be inserted;
4. Read the burst times (execution times) of processes;
5. Calculate the waiting time of each process:
   - wt[i+1]=bt[i]+wt[i]
6. Calculate the turnaround time of each process
   - tt[i+1]=tt[i]+bt[i+1]
7. Calculate the average waiting time and average turnaround time;
8. Display the values;
9. End
```

1) Perform a trial run on the following example to check your program:

INPUT

Enter number of processes
3
Enter burst time (execution time)
12
8
20

OUTPUT : bt wt tt

12 0 12
8 12 20
20 20 40

aw=10.666670 /* Average waiting time */
at=24.00000 /* Average turnaround time */

Exercise n°2

1) In this exercise you will be asked to implement the SJF scheduling algorithm and calculate the waiting time and turnaround time for each process in two versions: y
   a) non-preemptive SJF,
   b) preemptive SJF.

For the SJF scheduling algorithm, read the number of processes in the system, their CPU execution times. Rank all processes in order according to their execution time. There may be two processes in the queue with the same execution time, and the FCFS approach should then be applied to break the tie. Each process will be executed according to its runtime.

2) Perform a trial run on the following example to check your program:

INPUT:

Enter no of processes
3
Enter burst time
12
8
20

OUTPUT: bt wt tt
12 8 20
8 0 8
20 20 40

aw=9.33

at=22.64

Exercise n°3

2) You are requested to implement the round robin scheduling algorithm and calculate the waiting time and turnaround time of each of the processes. It is assumed that all processes have arrived at the same time.

   For round robin scheduling algorithm, read the number of processes/jobs in the system, their CPU execution times, and the size of the time slice (quantum). Time slices are assigned to each process in equal portions and in circular order, handling all processes execution. This allows every process to get an equal chance.

3) Perform a trial run on the following example to check your program:

Input:

Enter no of processes
4
Enter burst time
5
12
8
20

Output: bt wt tt
5 0 5
12 5 13
8 13 25
20 25 45
aw=10.75000
at=22.000000

Exercise n° 4

1) You are requested to implement the priority scheduling algorithm and calculate the waiting time and turnaround time of each of the processes.

For priority scheduling algorithm, read the number of processes/jobs in the system, their CPU execution times, and their arrival times. There may be two processes in the queue with the same execution time, and the FCFS approach should then be applied to break the tie.

2) Perform a trial run on the following example to check your program:

INPUT:

Enter number of processes

4

Enter burst time

10

2

4

7

Enter priority values

4

2

1

3

OUTPUT:

bt wt tt

4 1 0 4

2 2 4 6

7 3 6 13

10 4 13 23

aw=5.750000

at=12.500000