Programming Assignment 1
CPU scheduling algorithms in GO

Michael McAlpin

September 5, 2018

The goal of this assignment is to write a program in *GO* that reads an input file containing several configuration parameters to simulate *First-Come First-Served*, *preemptive Shortest Job First*, and *Round-Robin* CPU scheduling algorithms. The output will reflect the results of the configured CPU scheduling algorithm to a specified output file.

1 Objectives

The objectives of this assignment are to simulate the CPU scheduline of several processes for the configured *First-Come First-Served*, *preemptive Shortest Job First*, and *Round-Robin* CPU scheduling algorithms. Therefore the algorithm will need to be selected/configured as will the processes and their durations and burst cycles, as appropriate.

1.1 Algorithms

1.1.1 First-Come First-Served

The *First-Come First-Served* algorithm is just that: processes are allocated the CPU in the order in which they arrive and run until *completion* or *termination*.

1.1.2 Preemptive Shortest Job First

The *Preemptive Shortest Job First* selects the process for execution which has the smallest amount of time remaining until completion. Therefore, a process with a budget or expected time of 2 time units will be scheduled **ahead** of a process with 6 time units.

1.1.3 Round-Robin

Round-Robin was originally built for time-sharing systems. It works on the premise of providing a rigorous enforcement of an interrupt every **configured** time interval, then swapping to the next task in the process list at the moment of the interrupt. This scheduling algorithm treats the process list as a circular list until each process is complete. New processes can be added *infinitum*.
Note that the *Round-Robin* algorithm requires a configuration of the time quantum for the period of time used for the process to be in the CPU. Once the period expires the next process in the list will have CPU resources until its time quantum expires.

2 Specifications

2.1 Inputs

The input file will be passed as the first command line argument. Also the program should ignore everything on a line after a # mark and ignore additional spaces in the input file.

```
processcount 2 # Read 2 processes
runfor 15 # Run for 15 time units
use rr # Can be fcfs, sjf, or rr
quantum 2 # Time quantum, only if using rr
process name P1 arrival 3 burst 5
process name P2 arrival 0 burst 9
end
```

There is no requirement to process blank lines.
The input files are as follows:

2.1.1 Command Line Inputs

The *command line inputs* or *cli* are the input file name as the first parameter and the output file name as the second parameter. For example:
`./pa1 c5-fcfs.in c5-fcfs.stu`
where the files' extensions correspond to **input** and **student**. Review the *pa1Test.sh* file in the assignment ZIP file for more examples.

Table 2.1: Input Test Files

Filename	Description
c2-fcfs.in	2 processes scheduled *First-Come First-Served*
c2-rr.in	2 processes scheduled *Round-Robin*
c2-sjf.in	2 processes scheduled *preemptive Shortest Job First*
c5-fcfs.in	5 processes scheduled *First-Come First-Served*
c5-rr.in	5 processes scheduled *Round-Robin*
c5-sjf.in	5 processes scheduled *preemptive Shortest Job First*
c10-fcfs.in	10 processes scheduled *First-Come First-Served*
c10-rr.in	10 processes scheduled *Round-Robin*
c10-sjf.in	10 processes scheduled *preemptive Shortest Job First*

2.2 Outputs

The output of each schedule shall be formatted as shown below. *Note that the output below corresponds to the input file shown above.* You can also review the files that have the ***base*** extension for each test cases' expected outputs.

```
  2 processes
Using Round-Robin
Quantum   2

Time   0 : P2 arrived
Time   0 : P2 selected (burst   9)
Time   2 : P2 selected (burst   7)
Time   3 : P1 arrived
Time   4 : P1 selected (burst   5)
Time   6 : P2 selected (burst   5)
Time   8 : P1 selected (burst   3)
Time  10 : P2 selected (burst   3)
Time  12 : P1 selected (burst   1)
Time  13 : P1 finished
Time  13 : P2 selected (burst   1)
Time  14 : P2 finished
Time  14 : Idle
Finished at time  15

P1 wait   5 turnaround  10
P2 wait   5 turnaround  14
```

Notes:

1. All integer numeric data is formatted using %3d.
2. Process names are specified in the input file.

2.3 Sample input with matching output

The text files below ar the corresponding output for the inputs shown.

```
# Begin input file
processcount 2 # Read 2 processes
runfor 10 # Run for 10 time units
use fcfs
process name P1 arrival 0 burst 3
process name P2 arrival 5 burst 2
end

#begin corresponding output file

  2 processes
Using First-Come First-Served
Time   0 : P1 arrived
Time   0 : P1 selected (burst   3)
Time   3 : P1 finished
Time   3 : Idle
Time   4 : Idle
Time   5 : P2 arrived
Time   5 : P2 selected (burst   2)
Time   7 : P2 finished
Time   7 : Idle
Time   8 : Idle
Time   9 : Idle
Finished at time  10

P1 wait   0 turnaround   3
P2 wait   0 turnaround   2
```

3 Resources

There are many, many resources on **Go** available online. The list below should help you get quickly started.

- Basic building blocks: Go by example (Very useful for command line interfaces, files, and so much more.
- A cookbook with examples: Go Language CookBook With Examples Nice seminar summary with extensive examples and descriptions.
- What not to do: 50 Shades of Go: Traps, Gotchas, and Common Mistakes for New Golang Devs Hint: *If you are new to Go the information here will save you hours debugging your code.*

There are many different links that were provided during lecture and are in the *Intro to GO* slide deck.
Note that the links above work in most Acrobat Readers - with **one** notable exception - the Webcourses PDF reader in the browser. To take advantage of the links, download this assignment PDF and use Adobe's Acrobat reader.

4 Submission Instructions

The assignment shall be submitted via *WebCourses*. The only file submitted should be the **pa1.go** source file.
Make sure to include a comment at the top of your main source file that contains the following *Academic Integrity* statement (substitute your name and NID) - *"I [name] ([NID]) affirm that this program is entirely my own work and that I have neither developed my code together with any another person, nor copied any code from any other person, nor permitted my code to be copied or otherwise used by any other person, nor have I copied, modified, or otherwise used programs created by others. I acknowledge that any violation of the above terms will be treated as academic dishonesty."*

5 Grading

Scoring will be based on the following rubric:

Table 5.1: Grading Rubric

Deduction	Description
-100	Cannot compile on *eustis*
-100	Cannot read input file specified on command line
-100	Cannot write output file specified on command line
- 50	Produces **no meaningful** output
- 20	Fails to produce appropriate *Round-Robin* schedule
- 20	Fails to produce appropriate *Shortest Job First* schedule
- 20	Fails to produce appropriate *preemptive First-Come First-Served* schedule
- 20	Fails to produce appropriate schedule for *more than* 5 processes
- 25	Does not have *Academic Integrity* statement