CS 441 Project 3

Authors

Ryan Gittins and Phillip Sime

2013-11-12

Summary

This software simulates four process scheduling algorithms including "First-Come, First-Served", "Shortest Job First", "Priority", and "Round Robin". The user supplies a textfile in which the number of processes to be simulated is specified as well as details for each of those processes, namely an identifier, burst length, and priority level. Via a command-line interface the user selects one of the four algorithms, a filename, and (if they chose Round Robin) a quantum. The program then displays information regarding the chosen algorithm, quantum, filename, file contents, scheduling information, and average waiting/turnaround times before exiting.

Build

To build this software, simply navigate to the directory containing the Makefile and all other included files on a unix-based machine or virtual machine. Type the command `make` into the terminal to trigger the build. This will cause the gcc build commands listed in the Makefile to execute, compiling the source code into an executable file.

Usage

To use this software, simply run the command `./scheduler` with a `-s` flag, followed by a single digit from one through four which selects the scheduling algorithm. 1 specifies "First-Come, First-Served", 2 specifies "Shortest Job First", 3 specifies "Priority", and 4 specifies "Round Robin". If "Round Robin" is selected, then the user must also specify the desired quantum with a `-q` flag followed by the number.

The user must also input the filename and path (if applicable) of a textfile where the first line states the number of processes which are to be scheduled and each of the remaining lines specifies an identifier, burst length, and priority for that process, delimited by a single space.

Thus, a complete command for this program might look like `./scheduler -s 4 tests/test1.txt -q 3` or `./scheduler -s 2 tests/test2.txt`. Note that the order of the flags does not matter and the quantum, if specified, will be ignored for all scheduling algorithms except Round Robin. If the user does not specify a proper command, the process will remind the user of the proper syntax and terminate.

Test Cases

`test1.txt` This test is identical to the one given as an example in the Project Overview. Its contents appear on page two and two examples of its use appear on page four. This is a basic sanity check to ensure the results produced by the program are identical to the results on page four.

`test2.txt` This test was built in order to demonstrate that Round Robin can handle many rounds without issue. It is intended to be run with a low quantum (~3) so all processes are split across several rounds. This test also demonstrates that the process identifiers do not need to be in any way consecutive or ordered.

`test3.txt` This test contains 100 processes and was built as a stress test. All tests have the same burst lengths and priorities, so this test also demonstrates the program's maintenance of arrival order in the event of a tie.

`test4.txt` This test demonstrates a graceful failure and controlled termination of the program the in event of execution on a malformed test. This test contains an alphabetic character where a digit is expected.

`test5.txt` This test demonstrates an attempt at handling malformed data. In the event of a missing value, the program assigns a default value of zero. In this test, one of the processes is missing a priority level, and the program handles it as it would handle a process with priority zero.

`test6.txt` This test demonstrates another successful attempt at handling malformed data. If there is extra data beyond the required values on any given line, the program simply ignores it and continues reading the rest of the file without any problems.

Examples

This example shows the output of `./scheduler -s 4 tests/test2.txt -q 3`. Note how the output maintains the order of arrival.

```
shell$ ./scheduler -s 4 tests/test2.txt -q 3
Scheduler    :  4 RR
Quantum      :  3
Sch. File    : tests/test2.txt
------------------------------
Arrival Order: 9, 4, 1, 8, 5
Process Information:
 9      10       1
 4      12       2
 1      14       3
 8      16       4
 5      18       5
------------------------------
Running...
------------------------------
 9      10       1      22      46
 4      12       2      25      49
```

```
 1      14       3      34      60
 8      16       4      41      67
 5      18       5      44      70
Avg. Waiting Time    : 33.20
Avg. Turnaround Time : 58.40
--------------------------------
```

This following example shows the output of `./scheduler tests/test7.txt -s 1`. This output can quickly be checked for accuracy since each CPU burst length is 5 and there are a total of 10 processes.

```
shell3$ ./scheduler -s 1 tests/test7.txt
Scheduler    :  1 FCFS
Quantum      :  0
Sch. File    : tests/test7.txt
--------------------------------
Arrival Order: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Process Information:
 1       5      10
 2       5       9
 3       5       8
 4       5       7
 5       5       6
 6       5       5
 7       5       4
 8       5       3
 9       5       2
10       5       1
--------------------------------
Running...
--------------------------------
 1       5      10       0       5
 2       5       9       5      10
 3       5       8      10      15
 4       5       7      15      20
 5       5       6      20      25
 6       5       5      25      30
 7       5       4      30      35
 8       5       3      35      40
 9       5       2      40      45
10       5       1      45      50
Avg. Waiting Time    : 22.50
Avg. Turnaround Time : 27.50
--------------------------------
```

This following example shows the output of `./scheduler tests/test7.txt -s 2`. This output can quickly be checked for accuracy since each CPU burst length is 5 and there are a total of 10 processes.

```
shell$ ./scheduler -s 2 tests/test7.txt
Scheduler    :  2 SJF
Quantum      :  0
Sch. File    : tests/test7.txt
```

```
------------------------------
Arrival Order: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Process Information:
 1       5      10
 2       5       9
 3       5       8
 4       5       7
 5       5       6
 6       5       5
 7       5       4
 8       5       3
 9       5       2
10       5       1
------------------------------
Running...
------------------------------
 1       5      10       0       5
 2       5       9       5      10
 3       5       8      10      15
 4       5       7      15      20
 5       5       6      20      25
 6       5       5      25      30
 7       5       4      30      35
 8       5       3      35      40
 9       5       2      40      45
10       5       1      45      50
Avg. Waiting Time    : 22.50
Avg. Turnaround Time : 27.50
```

This following example shows the output of `./scheduler tests/test7.txt -s 3`. This output can quickly be checked for accuracy since each CPU burst length is 5 and there are a total of 10 processes.

```
shell$ ./scheduler -s 3 tests/test7.txt
Scheduler    :  3 PRIORITY
Quantum      :  0
Sch. File    : tests/test7.txt
------------------------------
Arrival Order: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Process Information:
 1       5      10
 2       5       9
 3       5       8
 4       5       7
 5       5       6
 6       5       5
 7       5       4
 8       5       3
 9       5       2
10       5       1
------------------------------
Running...
------------------------------
 1       5      10      45      50
 2       5       9      40      45
 3       5       8      35      40
```

```
 4        5        7       30       35
 5        5        6       25       30
 6        5        5       20       25
 7        5        4       15       20
 8        5        3       10       15
 9        5        2        5       10
10        5        1        0        5
Avg. Waiting Time    : 22.50
Avg. Turnaround Time : 27.50
--------------------------------
```

Known Bugs and Problem Areas

- No bugs or problem areas are known at this time.