CS 441 Project 5

Authors

Phillip Sime and Ryan Gittins

2013-12-10

Summary

This software simulates five different page replacement algorithms and displays the number of page faults produced by each for a given number of frames. The number of frames can be specified by the user and must range from one to seven, inclusive. If the number of frames is not specified, then all algorithms are against the full range of frame sizes. The user is required to specify the name a text file containing the reference string in a specific format.

The following page replacement algorithms are simulated by this program:

- Optimal
- First-Come, First Served (FCFS)
- Least Recently Used (LRU)
- Least Recently Used, Second Chance (LRU SC)
- Least Recently Used, Enhanced Second Chance (LRU ESC)

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 the filename and path (if applicable) of a text file where the first line states the length of the reference string and each of the remaining lines specifies an element of that string, with both the page number and read/write bit, delimited by a single space. The page number is a single digit and the read/write bit is an 'R', 'W', or their lowercase versions.

Optionally, the number of frames to be simulated can be specified with a `-f` flag followed by a number from one to seven, inclusive. If the number of frames is not specified, the simulations will be run against the full range of frame numbers with the reference string from the specified file.

Thus, a complete command for this program might look like `./scheduler -f 4 test/level1.txt` or `./scheduler test/level2.txt`. Note that the order of the parameters

does not matter. If incorrect syntax is specified for this command, an error will be displayed along with usage information before graceful termination.

Test Cases

- Test 1 - `test/level1.txt` - This test is a default test case from the Project Description PDF.
- Test 2 - `test/level2.txt` - This test is a default test case from the Project Description PDF.
- Test 3 - `test/level3.txt` - This test is a default test case from the Project Description PDF.
- Test 4 - `test/level4.txt` - This test is a default test case from the Project Description PDF.
- Test 5 - `test/level5.txt` - This test is 40 page references to page 1. This allows for a quick check of logic where we expect all of the results to be 1.
- Test 6 - `test/level6.txt` - This test contains 10 page references with all possible page values (0-9) being tested. Since we never repeat any page numbers, each algorithm should produce 10 page faults in every case.
- Test 7 - `test/level7.txt` - This test contains 98 page references that iterate over the values 0-6 in order. This test demonstrates a large number of page references. This test also demonstrates the accuracy and correctness of the Optimal algorithm.
- Test 8 - `test/level8.txt` - This test was created with 20 page references. This test allows a sudo-random case to be used to verify the algorithms are working properly by manually checking the results. For example: LRU with 4 frames should produce 15 hits according to the program. After manually working through LRU the results were 15 hits with the ending frames containing, in order, [7,6,4,2].
- Test 9 - `test/level9.txt` - This test contains 1000 sudo-randomly generated page references. This test is purely to demonstrate that a large number of pages can be processed by all algorithms. Note: one quick check to ensure accuracy is that all page faults are equal for 1 frame.
- Test 10 - `test/level10.txt` - This test contains a value declaring a greater number of reference strings than actually exist in the file. This test ensures that files that are improperly formatted are handled correctly.
- Test 11 - `test/level11.txt` - This test contains 15 page references where all page references are reads. By executing this test it is a quick check to ensure that LRU SC and LRU ESC are working properly. Since there are no writes in this test file, both algorithms should produce the exact same number of page faults.
- Test 12 - `test/level12.txt` - This test contains 15 page references with a mixture of reads and writes. By executing this test we can verify that LRU SC and LRU ESC behave differently when there are writes involved.

Examples

This example shows the command `./scheduler test/level8.txt -f 4`. This demonstrates the ability to supply a frame number to print only one frame.

```
shell$ ./scheduler test/level8.txt -f 4
Num. Frames : 4
Ref. File   : test/level8.txt
------------------------------
Reference String:
 1:R,  0:W,  3:R,  5:W,  5:W,  9:W,  4:R,  9:R,  8:W,  3:R,  8:W,  2:W,  6:W,
2:W,  9:W,  4:W,  4:R,  7:R,  6:W,  2:R
------------------------------
Frames  Opt.    FIFO    LRU     SC      ESC
 4        10     14     15     14     14
```

This example shows the command `./scheduler test/level8.txt`. This demonstrates the same reference string as above, but printing all frames.

```
shell$ ./scheduler test/level8.txt
Num. Frames : All
Ref. File   : test/level8.txt
------------------------------
Reference String:
 1:R,  0:W,  3:R,  5:W,  5:W,  9:W,  4:R,  9:R,  8:W,  3:R,  8:W,  2:W,  6:W,
2:W,  9:W,  4:W,  4:R,  7:R,  6:W,  2:R
------------------------------
Frames  Opt.    FIFO    LRU     SC      ESC
 1        18     18     18     18     18
 2        14     15     15     15     15
 3        11     15     15     15     15
 4        10     14     15     14     14
 5        10     10     11     10     12
 6        10     10     10     10     11
 7        10     10     10     10     10
```

This example shows the command `./scheduler test/level10.txt`. This demonstrates how the program will handle improperly formatted files without crashing.

```
shell$ ./scheduler test/level10.txt
Num. Frames : All
Ref. File   : test/level10.txt
------------------------------
Ensure that the file specified exists and is properly formatted.
```

Known Bugs and Problem Areas

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