COMP SCI 3004/7064 - Operating Systems
Practical 2: VM Page replacement

Due by 11:59pm 13th October

Important Information

Submission

- Your implementation of the code is due 11:59pm on Friday 13th October 2017 (Wk 10)
    - Your code will be submitted using SVN and the Web Submission System
    - The SVN directory for your code is 2017/s2/os/assignment2
    - Your code should be written in C++.
    - It will be compiled using `g++ -std=c++11 *.cpp -o memsim`
    - For late code submissions the maximum mark you can obtain will be reduced by 25% per day (or part thereof) past the due date or any extension you are granted.
- Postgraduate students must complete and submit this assignment individually, making individual submissions.
- Undergraduate students may choose to complete and submit in teams of at most two students.

Marking scheme

- This assignment is out of 15:
    - 3 for FIFO implementation (automarked)
    - 5 for ARB implementation (automarked)
    - 5 for Working Set implementation (automarked)
    - 2 marks for code quality (structure and comments; manually marked)

Introduction

Following our discussion of paging in lectures, this practical will allow you to explore how real applications respond to a variety of page replacement schemes. Since modifying a real operating system to use different page replacement algorithms can be quite a technical exercise, your task will be to implement a program that simulates the behaviour of a memory system using a variety of paging schemes.

Memory Traces

We provide you with **three(3)** memory traces to assist you developing your simulator. Each trace is a small portion of real recordings of running programs taken from my PC ;)

Each trace is a series of lines, each listing a hexadecimal memory address preceded by R or W to indicate a read or a write. There are also lines throughout the trace starting with a # followed by a process ID and command. For example, a trace file for gcc might start like this:

```
# 694 /usr/bin/gcc
R 0041f7a0
R 13f5e2c0
R 05e78900
R 004758a0
W 31348900
```

Simulator Requirements

Your task is to write a simulator that reads a memory trace and simulates the action of a virtual memory system with a single level page table.

- Your memory system should keep track of which pages are loaded into memory.
- As it processes each memory event from the trace, it should check to see if the corresponding page is loaded.
- If not, it should choose a page to remove from memory.
- Of course, if the page to be replaced is dirty, it must be saved to disk.
- Finally, the new page is to be loaded into memory from disk, and the page table is updated.

This is just a simulation of the page table, so you do not actually need to read and write data from disk. When a simulated disk read or write must occur, simply increment a counter to keep track of disk reads and writes, respectively.

You must implement three(3) different page replacement algorithms:

- **fifo** replaces the page that has been resident in memory longest.
  See textbook section 9.4.2.
- **arb** (additional reference bits) uses multiple reference bits to approximate a LRU scheme. Your implementation should use an 8-bit shift register.
  See textbook section 9.4.5.1.
- **wsarb** extends the arb page replacement algorithm, by using a working set model to keep track of page access' over a period of time for each process. It prepages in a process' working set when a context switch to that process occurs.
  See textbook section 9.6.2.
  The lines in the trace file beginning with # represent context switches.

Running your code

The simulator should accept arguments as follows:

1. The filename of the trace file
2. The output mode (quiet/debug)
3. The page/frame size in bytes (we recommend you use 4096 bytes when testing).
4. The number of page frames in the simulated memory.
5. The page replacement algorithm to use: fifo / arb / wsarb

If the page replacement algorithm is **arb**, it should accept the following additional argument:

6. The interval, in page references for shifting the reference bits.

If the page replacement algorithm is **wsarb**, it should accept the following additional arguments:

6. The interval, in page references, for shifting the reference bits.
7. The size of the working set window Δ, in page references.

The trace provided should be opened and read as a file, **not** parsed as text input from stdin.

Your solution will be run as shown:

```
./memsim test.trace quiet 4096 32 fifo > output.txt
```

Output

If the mode is "debug", the simulator prints out messages displaying the details of each event in the trace. You may use any format for this output, it is simply there to help you debug and test your code.

If the mode is "quiet", then the simulator should run silently with no output until the very end, at which point it prints out a summary like this:

```
events in trace:     1002050
total disk reads:    1751
total disk writes:   932
page faults:         1751
prefetch faults:     0
```

Where:

- **events in trace** is the number of memory access' in the trace. Should be equal to number of lines in the trace file that start with R or W. Lines starting with # do not count.
- **total disk reads** is the number of times pages have to be read from disk.
- **total disk writes** is the number of times pages have to be written back to disk.
- **page faults** should be equal to the number of disk reads in a demand paging system.
- **prefetch faults** are page faults that occur as part of loading the working set following a context switch. They are not counted in page faults. This + page faults == disk reads.