CS4320 Programming Project 2

Due Wednesday, 10 Nov 1999, 5pm.

1 Overview

The goal is to experimentally evaluate the performance of page replacement algorithms on realistic page request traces. For example, given a set of page requests, how much worse is the least recently used (LRU) algorithm than the optimal algorithm?

2 Specifications

Your job is to create a simulator that, given a list of page requests, a specified page replacement algorithm, and memory size, determines the number of hits and misses necessary to satisfy the requests.

Each page request will be specified as

```
[RW] [0-9a-fA-F]+
```

These usually will occur one per line, but this is not required. The first letter indicates whether the request is for a page to read (`R`) or a page to write (`W`). The following nonnegative hexadecimal number (without a beginning `0x`) indicates the requested page. Your simulator need handle only page requests from one process at one time (so the process ID is not necessary), but it should be able to process a request list having any length. Make no assumptions about the order or frequency of requests for particular pages.

Your simulator should implement the following page replacement algorithms:

1. the optimal page replacement algorithm (§3.4.1),
2. first-in, first-out (§3.4.3),
3. least recently used (§3.4.6),
4. second chance (§3.4.4),
5. least frequently used, i.e., replace the page with the smallest count.

Your simulator can have any user interface, but it should be obvious how to choose among the various choices, and, hopefully, the simulator will not have to be recompiled for each replacement algorithm.

The final input to the simulator should be desired positive number of page frames. Assume all page frames are initially empty.

Given a list of page requests, a page replacement algorithm, and a memory size, your simulator should produce the number of

- read requests for pages already in memory
- write requests for pages already in memory

- read requests for pages not in memory
- write requests for pages not in memory

A few sample page requests will be made available. You are also expected to contribute a reasonably long page request to use in grading the simulators. The request should have at least several hundred requests so it might be easier to either write a program to generate requests or record the page requests of a running program.

Feel free to work on any computer you desire, but grading will occur using a UNIX computer in the computer science domain.

3 Page Request Traces

A few small page request lists will be made available at ftp://atlas03.cs.trinity.edu/pub/-joldham/4320/pp2/samplePgReqs/. You can use them for preliminary testing of your simulator, but you will probably want to also create your own.

Most realistic page request lists show locality of reference. To obtain these realistic page requests, I found a research tool named VMTrace. (See also Scott Kaplan's research page.) To log a program's page references, compile the C++ or C program with the VMTrace library.

Here is my understanding of how to use the tool on a Linux box. (As a research tool, the program is not documented, but I read some of the code.)

1. Copy libvmtrace.a and Makefile to the directory containing your C++/C program. (Be sure to use the your browser's save function so that the tab characters (which look like spaces) are saved. Do not use your mouse to copy.)

2. Modify the blah-linux Makefile entry, substituting your program's name for "blah." If you are using g++, modify the `CC` entry to be `g++`.

3. Compile your program using "`make` blah-`linux`."

4. Set the following environment variables:

`VMT_QSIZE`	1	each request to a page other than this one misses
`VMT_OUT`	vmtrace.out	determines the name of the page request log file
`VMT_DIRTY`	`true`	record whether page request is a read or write
`VMT_STATIC`	`true`	record page requests to the static data segment
`VMT_FLAG`	`true`	record page requests

   If you are using bash or sh, one can set an environment variable using the `declare -x` command:

   ```
   declare -x VMT_QSIZE=1
   ```

   If you are using csh or tcsh, one can set an environment variable using the `setenv` command:

   ```
   setenv VMT_QSIZE 1
   ```

5. Run your program. The page requests are stored in the page request log file.

4 Programming Tips

Use any programming language you desire, remembering that some of the page request lists can have millions of lines so fast processing is desired.

When programming, be sure to turn on all compiler warnings possible and resolve as many issues as possible. This will greatly increase the possibility your program is portable to other computers and operating systems and can also help find programming errors with reduced amount of testing. When using `gcc` and `g++`, use the `-Wall` option. I do not know what the Java options are.

To minimize your workload, use as many library functions as possible. Depending on the language you use, information is available on UNIX manual pages, info pages (use the UNIX command `info`), WWW pages, and books. For more specific information, come see me, telling me for what you desire and what language you are using.

5 Grading and Logistics

You can work in groups of up to two people. Grades will be assigned primarily according to correctness. Adequate documentation of the project and the code will assist in assigning partial credit (but, again, grades will primarily be assigned according to correctness).

Here is a possible point distribution:

compilation, no memory leaks, etc.	5–15 points
page request list submission	5–15 points
easily understood user interface	5–15 points
page replacement algorithms and correctness	remaining points
total	100 points

Grading will be interactive. Each group will make an appointment with me (Jeffrey D. Oldham) to have their simulator graded. Together, we will grade the assignment according to the point distribution, the class's submitted page request lists, and additional page request lists I create.

5.1 Submission

Use anonymous ftp to atlas03 to submit both your program and your page request list. Submit your program's source code and possibly an executable to ftp://atlas03.cs.trinity.edu/pub/joldham/4320/pp2/submitPgms. If you need to package together several files, use tar. Submit your page request list to ftp://atlas03.cs.trinity.edu/pub/joldham/4320/pp2/submitPgReqs. Please name your files using your name.

Here is one way to submit your program and page request list:

1. Package together your program and source code:

   ```
   tar cvf <your-name>.tar program program.cc Makefile
   ```

2. ftp the material.

   ```
   ftp atlas03.cs.trinity.edu
   ```

Use anonymous as your password.

```
cd pub/joldham/4320/pp2/submitPgms
put <your-name>.tar
cd ../submitPgReqs
put <your-page-request-list>
bye
```