Main & Virtual Memory

Deadline: Monday, February 3, 2020

Project Goals

- To familiarize the main memory and the virtual memory.
- To develop programming skills in C.
- To implement algorithms for page replacement.

The Assignment

This assignment consists of two parts, a page-replacement simulator and a concurrency problem.

Page-replacement simulator

Develop a program that simulates the page-replacement algorithms of a given setup. The program will load a configuration file from the command line. An example of the configuration file is as follows:

```
# System setup
LAS=4GB
PAS=1GB
PAGESIZE=128B
# Process setup
PROCESS_NAME=P0
PROCESS_SIZE=7300B
PROCESS_MEMREF=898,100,150,260,127,400,110,515,180,420,120,425,256,175,270
ALGORITHM=LIFO
PAGE_FRAMES=4
```

where LAS is the Logical Address Space, PAS is the Physical Address Space, PAGESIZE is the page size, PROCESS_NAME is a label of the process name, PROCESS_ZISE is the size of the process, PROCESS_MEMREF is a list of address sequence for a particular process, ALGORITHM is a page-replacement algorithm (possible values: FIFO, LRU, OPTIMAL), and PAGE_FRAMES is the number of page frames available. Additionally, a word or line beginning with # causes that word and all remaining characters on that line to be ignored, i.e. it is comment.

The program should output the following:

1. Logical address structure: page bits and offset bits
2. Physical address structure: frame bits and offset bits
3. Max number of Page Table Entries (PTE)
4. Max number of frames
5. Size in bytes required to represent the Page Table (Hint: memory is byte addressable)
6. Number of pages marked as valid for the process
7. Reference string based on the address sequence
8. Show the output (table) of the page-replacement algorithm (be creative)
9. Summarise number of hits and page fault

The program should validate that the address sequence is valid, i.e. it refers to valid pages. Otherwise, it should issue an error indicating the address that has problems. Assume the valid pages are allocated consecutively.

Concurrency problem

A linked list is accessed by threads to do these operations:

- Search: it searches an element in the list
- Insert: it inserts an element at the end of the list
- Update: it updates an element in the list
- Delete: it deletes an element in the list

Search operations can be executed concurrently. Insert operations are mutually exclusive among them to avoid overwriting the last item. However, insert and search operations can happen concurrently. Update operations are mutually exclusive. No search can be performed over a node that is being updated. Delete operations are mutually exclusive among them and also to the other operations (search, insert and update).

Define a file with a set of operations to test your program.

Make sure no deadlock occurs during the execution.

Recommendations

Defensive programming is an important concept in operating systems: an OS cannot simply fail when it encounters an error. It must check all parameters before it trusts them. In general, there should be no circumstances in which your C program will core dump, hang indefinitely, or prematurely terminate. Therefore, your program must respond to all input in a reasonable manner. By "reasonable", this means that you should print a meaningful and understandable error message and either continue processing or exit, depending upon the situation.

Grading

This assignment must be submitted via SidWeb with the following elements:

- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, course section, date, and brief description), comments for each variable, and commented blocks of code.

- A Makefile for compiling your source code, including a clean directive.

- A diagram(s) that explains how your strategy works.

- A README file with some basic documentation about your code. This file should contain the following three components:
  - Design Overview: A few paragraphs describing the overall structure of your code and any important structures.
  - Complete Specification: Describe how you handled any ambiguities in the specification. For example, for this project, explain how your shell will handle lines that have no commands, double spaces between arguments, tabs, etc.
  - Known bugs or problems: A list of any features that you did not implement or that you know are not working correctly.

Submit a compressed file (*.zip) to the SidWeb and name it as follows:

OS.P1.2019_2.*<Lastname><Firstname>*.zip

For example: **OS.P1.2019_2.LopezAngel.zip**

The grade on the project will be calculated as follows:

- Page-replacement: 40 points
- Concurrency problem: 30 points
- Documentation: 10 points
- Strategy: 20 points