MEMORY MANAGEMENT

OVERVIEW

This assignment will explore the use and misuse of large arrays and the effects of virtual memory. You will write a single program to compare the time required to read and write elements of large arrays in both row-major and column-major order.

THE PROGRAM

In a C program, `matrix.c`, create a two-dimensional matrix where each element is a character (char) and the size of each row is exactly one page (4096B). Your matrix should contain 20480 rows. You will notice that the size of the array is large. In fact, it is too large to be placed on the stack. Therefore, it cannot be declared locally inside of any function. It may also be too large to be dynamically created with a call to malloc(). The only remaining option is to make it global. You may want to experiment with the other options to see the results. There is no need to initialize the array elements.

Access the elements of the array four different times:

1. Read Row – read the value of each element in row-major order
2. Read Col – read the value of each element in column-major order
3. Write Row – write a new value, other than zero, to each element in row-major order
4. Write Col – write a new value, other than zero, to each element in column-major order

Do not print the values of the elements. Any I/O will distort the timing results. Repeat each experiment 10 times and get an average execution time. For example, for each experiment you may want to do something like this:

```
Start timer
Repeat 10 times
         Access all fields in array in desired order
Stop timer
Display average time result
```

Illustrate the average run times from the different operations above in a bar chart. Use this bar chart to write up a short report explaining your experiment, the results, and any implications you may draw from the experiment. Also describe in your report any problems you had with implementation and testing. Keep in mind that you may have to explore additional compiler options to ensure that the compiler is not optimizing your code, thereby eliminating the bottlenecks you want to test.

IMPLEMENTATION SUGGESTIONS

Look back at your previous programming projects for hints on using timers in C.

PAGE-FAULT ANALYSIS

Consider the above matrix along with memory requirements and layout specification and two processes *A* and *B* implementing *Read Row* and *Read Column* respectively. Compute the total number of page faults for *A* and *B*

assuming that A) each process is given 10 frames, B) the array is a global data element, C) 2 of the 10 frames are used for code and stack, and D) a LRU page replacement algorithm is applied to evict pages from memory. Hint: Draw a diagram that shows the layout of the process in memory – it is the same for *A* and *B*.

DELIVERABLES

Your project submission should follow the instructions below. Any submissions that do not follow the stated requirements will not be graded.

1. Follow the submission requirements of your instructor as published on *eLearning* under the Content area.
2. You should have at a minimum the following files for this assignment:
    a. `matrix.c`
    b. *analysis.doc* (the results from the runtime experiment and issues you may have encountered)
    c. *reportPF.doc* (the solution of the page-fault analysis problem)
    d. Makefile

The report *analysis.doc* describes A) the four different runtime results from accessing the matrix in memory, B) your interpretation of the results, C) implications on system performance, and D) issues you may have encountered. The document *reportPF.doc* describes the solution of the page-fault analysis problem with illustrations on memory layout and how you arrived at the solution. If you do not include comments in your source code and refactor your code adequately, points will be deducted.

GRADING

This project is worth 100 points total. The points will be given based on the following criteria:

- 40 points for correct implementation of code with proper documentation and refactoring,
- 30 points for the report describing your experiment and your analysis of it,
- 30 points for the memory problem.

DUE DATE

The project is due Wednesday, November 6th by 3:00 pm to the Dropbox of project 5 in *eLearning*. I will not accept submissions emailed to me or the grader. Upload ahead of time, as last minute uploads may fail. Please review the policy in the syllabus regarding late work.