Programming Assignment 3

CS450 Spring, 2020

This assignment is a pair programming effort. It is due on April 5th, 2020

Part 1: Memory leaks and tools to find them (xv6 not required)

Memory leaks degrades system performance over time and may eventually lead to system crash. The problem happens often and is difficult to detect and correct. The purpose of this exercise is to introduce you to some tools that may help you combat this problem.

In this exercise, you will need to use the debugging tools `gdb` and `valgrind`. `valgrind` helps you to find memory leaks and other insidious memory problems. Please find the following link to download and install the tool:

http://valgrind.org/downloads/current.html

Deliverables of Part 1:

1. Write a program that allocates memory using malloc() but forgets to free it before exiting. What happens when this program runs? Can you use `gdb` to find any problems with it? How about `valgrind` (with the command: *valgrind --leak-check=yes null*)?
2. Create other test cases for `valgrind`. Explain why you choose them and the expected results.

Part 2: System calls on memory management

1. Develop the `myV2p()` system call on xv6. The arguments to `myV2p()` are (1) a virtual address and (2) the operation (read or write) to the instruction or data in that address. `myV2p()` returns the corresponding physical address or the appropriate error condition. You will provide a test program that will call `myV2p()` with various test data. Sometimes, people call such a test program an automated test driver. Clearly, xv6 must have code for these functions. Because of the difference in purpose, your code may not be the same as those in xv6, but you should understand the code in xv6.
2. Develop the `hasPages()` system call on xv6. It takes a process id as argument and will display the different kinds of user pages that have been allocated to the process or returns an error code. How do you define the "different kinds"? The objective is to display as much useful information to the user as possible. You should have test drivers that will allocate different kinds of memory and see how they affect the page tables of the process.

Tips for Part 2:

1. Make sure you understand how xv6 uses a page directory and page tables to map a process's virtual memory to physical memory. In particular, understand what the different bits in a Page Directory Entry and Page Table Entry mean. Chapter 2 of the **xv6 textbook** is a useful reference. For example, how does a valid Page Table Entry differ from an invalid one?
2. Have a look at vm.c file of xv6 to understand how page tables are handled in xv6.

What you will submit (only one submission per team):

Part 1:

(1) Source and executables of the test programs. A `readme` on how to build and execute them with the tools.

(2) Screen shoots of test runs. A document (5 pages or less) to describe the results of the test runs and address the deliverables. If you use equivalence partitioning in deliverable 2, explain your partitions.

Part 2:

(3) Source and executables for the system calls and test programs with a `readme` on how to build and execute them.

(4) A document (5 pages) that describes the design of the system calls including a manual page for each. Describe the changes that you made to the xv6 memory management code and why. You do not need to describe xv6 changes to implement the system calls; that was done in PA2.

(5) A document (3 pages or less) that describes your test programs and test data. Explain why do you use only those test cases. If you use the equivalence partitioning method, describe your partitions.

Both Parts:

(6) Upload all files and folders as a **zip** archive as GroupID_PA3.zip. Documents and readme only supports: txt, doc, docx and pdf format.

(7) Write down the names and CWID of team members in all documents and source files.

Grading standard:

(8) In general, we give 65% of the points to working and full featured code, 20% to high quality test data and 15% for well written documents. We give extra credits to very well written code with additional useful and original features, very high quality test data and documents. The test drivers are coding.

(9) When we have a better feel of what is considered difficult (or easy) to the students and what demands more work, we adjust the points allocate to the features.