Programming Assignment 3

CS450 Fall, 2021

This assignment is a pair programming effort. It is due on 11/08/2021
You can select your own partner otherwise a partner will be assigned to you. Ayush will detail the process in an email. The pairing exercise will finish by 10/18/2021.

Part 1: Memory leaks and tools to find them (xv6 not required) (20%)

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 to share a memory page (40%)

In this assignment, we ask you to write a pair of **xv6** system calls, `GetSharedPage()` and `FreeSharedPage()` that will allow two programs (two processes) to share pages. Linux has similar system calls but they are not the same.

`GetSharedPage()` takes two arguments. The first argument is a key. The second indicates how many pages the user wants. It returns a virtual address to the shared page when successful and the process can start writing and reading from the virtual address.

The first process that calls `GetSharedPage` with a unique key will have the shared pages allocated to the next available virtual pages starting at the high end of the process' address space. The pages should be zero filled initially.

Subsequent calls to `GetSharedPage` with the same key from other processes will be able to read and write to the shared pages.

Calls to `GetSharedPage` with a different key will create a different set of pages.

Calls to `FreeSharedPage` with a key will remove the calling process from accessing the shared pages associated with the key. When no more process can access the pages associated with a key, those pages will be deallocated.

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. How does a Page Table Entry differ for a valid page compared to an invalid page?
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. (2.5%)

(2) Screenshots 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. (15%)

Part 2:

(3) Source and executables for the system calls and test programs with a `readme` on how to build and execute them. (2.5%)

(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. (10%)

(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. (10%)

Both Parts:

(6) Upload all files and folders as a **zip** archive as GroupID_PA3.zip. Documents and readme should only be in pdf format.

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