Programming Assignment 2

CS450 Fall, 2021

1. This assignment is an individual effort. It is due on 10/5/2021

2. **Basic Requirements** (60%)**:**
   The best way to learn how to implement a system call (or the control mechanism of an operating system) is to trace an existing system call and to implement one. This assignment therefore consists of two parts. The first part asks you to trace the execution of the `close()` system call. The second part asks you to implement a system call that will allow a user program to print out how many times the user process has been trapped to the kernel.
   1) (30%) You will trace the `close()` system call. Assume that a user program executes the statement `close(fd)` but `fd` is undefined therefore the result of the execution will be an exception. You are asked to write down in a document which lines in which files of xv6 are executed. Organize the lines of code into blocks of code and explain the purpose of each of block. The result will be a story of what happened when a non-existing file descriptor is given to a `close()`. The story starts in the user space when the system call first get executed and ends with a feedback to the user that `fd` is bad.
   2) (30%) You need to add the `countTraps()` system call in xv6. When a user program calls `countTraps`, it will print out the number of times the user process has been trapped to the OS, and what types of traps has occurred and their corresponding number of occurrences. You may need to implement additional kernel code or system call to make this work. You need to write the user programs which are the test cases of your system call.

3. **Deliverables:**
   1) The document describing the execution of `close()` with a non-existing file descriptor starting from the user level. This document should not be longer than two pages.
   2) (10% 2) and 3))The document describing the design of the system call `countTraps()` and exception handling. You also need to describe how to call it from the user level. This document should not be longer than two pages.
   3) Describe each file that you changed (Hint use:$ diff -uw "$original" "$modified$ > ./diff/"$original.txt") to implement countTraps().
   4) (25%) Describe your test cases in a document.
   5) Source, executables and a README (5%) on how to build and execute.
   6) All your deliverables are in a zip file named: lastname-firstname-PA2.zip
   7) Documents should be in pdf format.