**Programming Assignment 2**

CS450 Spring, 2018

1. This assignment is an individual effort. It is due on 02/28/2018
2. **Requirements:**

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 implement one. This assignment therefore consists of two parts. The first part asks you to trace the execution of the write()system call. The second part asks you to implement a system call that counts the number of a specific system call has been invoked.

* 1. You will trace the write()system call. Assume that a user program executes the statement write(fd,10,n) 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 write(). The story starts in the user space when the system call first gets executed and ends with a feedback to the user that fd is bad.
  2. We ask you to add the system call getCallCount() system call in xv6 such that when a valid system call number (as defined in syscall.h) is passed to it as an argument, it will return the number of times the referenced system call has been invoked in the user program that calls it. You need to design test cases and test programs to prove that your system call works as desired.

1. **Deliverables:**
   1. A 2 page (or less) document describing the execution of write()with a non-existing file descriptor starting from the user level.



* 1. A 3 page (or less) document describing the design of the system call getCallCount() including a manual page. Describe each file in xv6 that you changed and why.
  2. The test program that you use to test the system call and your test cases. We ask that your test program contains calls to printf and strcpy and system calls to read()and write(). Explain your results in a 3 page (or less) document.
  3. Source and executable objects with a README on how to build and execute them.

