Programming Assignment 2

CS450 Fall, 2017

1. This assignment is an individual effort. It is due on 10/7/2017

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 `read()` system call. The second part asks you to implement a system call that will allow a user program to print out some status information of a user process.

   1) You will trace the `read()` system call. Assume that a user program executes the statement `read(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 `read()`. 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) You need to add `procState()` system call such that you can list all process's information: name, process state, process id, and memory usage. You need to design test cases and test programs to prove that your system call works as desired.

      Example (ps is the user program that will call `procState()`):

      $ ps

Name	State	ID	Memory
shell	Running	1	X KBytes
wc	Block	2	Y KBytes

3. **Deliverables:**
   1) The document describing the execution of `read()` with a non-existing file descriptor starting from the user level.
   2) The document describing the system call `procState()` and exception handling. You also need to describe how to call it from user level.
   3) Describe each file that you changed (Notice using:$ diff -uw "$orginal" "$modified$ > ./diff/"$orginal.txt").
   4) Source and executable objects with a README on how to build and execute them.