cs154: Introduction to Computer Systems
Autumn 2019

Homework 6
(Assigned Nov 7)
Due Nov 12 (Tuesday) 11:59pm

Submit answers by committing a file into the `hw6` sub-directory of your `CNETID-cs154-aut-19` svn repository. Do not create this directory yourself: an "`svn update`" at the top level of your checkout will create it. The filename should be either `hw6.txt` or `hw6.pdf` for answers in plain ASCII text, or PDF, respectively. PDFs of scanned hand-written pages must not exceed 6 megabytes. No other file formats, or filenames are acceptable, and no files besides `hw6.txt` or `hw6.pdf` will be graded. Not following directions will result in losing points.

(1) (16 points)

The textbook Sections 8.1-8.3 cover the difference between things like `open`, a **system call**, versus `fopen`, a **function** in C's standard **library**. From a C programmer's standpoint, the distinction is blurred by how the C standard library also includes thin wrappers (the book calls these "system-level functions") around the system calls. For example, compiling a C program on CSIL that calls `open()` will produce assembly that includes the instruction "`call open`", where that "`open`" is a system-level function. Good systems programmers understand, however, that there is also a distinct underlying `open` system call.

A. Based on what you learned in Section 8.1, describe (with about 40 of your own words) **two aspects** of how in general the execution of system calls differs from execution of library functions (e.g. execution mode, or what assembly instructions are used to start them).

B. Choosing from the following mix of system calls (system-level functions) and library functions,

```
accept   dup2   exit   fclose     fread   fseek   fstat
fork     log    lseek  mallinfo   mblen   mmap    pause
sprintf  raise  read   sbrk       shmget  signal  strpbrk
```

list four system calls and then list four library functions. For each of your eight choices give a brief (roughly 10 of your own words) description of its purpose. Feel free to consult `man` pages or whatever other online resources you find.

(2) (15 points)

This is a question about how the CPU responds to a divide-by-zero error, which builds on the discussion of exception handling from lecture.

Suppose that an instruction representing "`idivl %ebx`" is at address `0x08031000`, and that the address of the exception handler `divideByZero` is `0xC0015200`.

A. Consistent with how it was shown in lecture os1-Exceptions, show the relevant entry in the exception table built into the hardware.

B. Suppose that `%ebx` is zero, that `%eip` is `0x08031000`, and the CPU is about to start executing the current instruction. Describe the subsequent sequence of actions happening on the CPU up until and including the execution of the first instruction of `divideByZero`. Include the specific addresses given above, and how they are used.