CS 202: Advanced Operating Systems
University of California, Riverside

Lab #1: System Call Implementation

Due: 10/17/2021, Sunday, 11:59 p.m. (Pacific time)

Add a new system call `int info(int param)` that takes as input one integer parameter of value 1, 2 or 3. Depending on the input value, it returns:

(1) A count of the processes in the system;

(2) A count of the total number of system calls that the current process has made so far;

(3) The number of memory pages the current process is using.

What to submit:

You need to submit the following:

(1) The **entire XV6 source code** with your modifications ('make clean' to reduce the size before submission)

(2) A **report** (must be in **PDF**; no other formats will be accepted) including:

   a) The list of all files modified

   b) A detailed explanation on what changes you have made and screenshots showing your work and results

   c) A detailed description of XV6 source code (including your modifications) about how the info system call is processed, from the user-level program into the kernel code, and then back into the user-level program.

(3) A **demo video** showing that all the functionalities you implemented works as expected, as if you were demonstrating your work in-person. For instructions about recording, refer to the *Demo Video Recording Guide*.

How to submit:

Upload the demo video to your own Google Drive (or YouTube) and create a shareable link. Copy this link into a txt file named "video.txt". Along with the other required files above (source code and report), pack them into a **single zip file**, then upload it to eLearn.

Grades breakdown:

- Correct implementation and demo of info() system call: 45 pts
    - count of the processes in the system: 25 pts
    - report the number of system calls this program has invoked: 15 pts
    - report the number of virtual memory pages the current process is using: 15 pts
- Clear and detailed explanation of the changes made: 20 pts
- Clear and detailed explanation of system call workflow: 15 pts

Total: 80 pts

Tips:

This is optional for lab1. To simplify compilation-and-run process, it would be better use preprocessor directives (e.g. #define) to turn on/off some functionalities. This is good for debugging when you are not sure there is anything wrong in your code or in the runtime environment or even in xv6 itself. With awareness of using preprocessor directives, you can turn off (some of) your code to see if everything goes normal. For example:

```
......xv6 code......

#ifdef SYS_CALL_INFO

your code

#endif

......xv6 code......
```

You need to modify the Makefile to let you can toggle it on/off in command line:

After lines 72-73 "`CFLAGS = -fno-pie ...... endif`", insert 2 new lines:

```
LAB1 = -DSYS_CALL_INFO

CFLAGS += $(LAB1)
```

If you want to enable your code, just run make as usual; otherwise, run make LAB1= (leave it blank).

This is quite unnecessary for lab1 but may be mandatory in lab2 so that TA can switch between different functionalities easily.