Homework 2: Xv6, IPC

Instructor: Professor Sharifi
Operating Systems Fall 2019
Iran University of Science and Technology
https://os-course.github.io/

October 24, 2019

Prerequisite

Create an account on https://git.chillinwars.ir/ and fork the repository https://git.chillinwars.ir/vahid/xv6-public. Make sure that your forked repository is **private**. You can make your repository private via *Settings* → *General* → *Permissions* → *Project visibility* → *Private* from the side panel of the repository. Do not forget to push the **Save changes** button. Add the following accounts as members to your forked repository. You can add members to your repository via *Settings* → *Members*.

- @Vahid
- @EAliakbar
- @Sorouri
- @FShahinfar
- @Hadian

Clone your repository in your laptop and then follow the instruction from this link. Then perform the question 1.

Question 1.

Create a branch named *HW2_Q1* in your repository. Add a new system call to xv6. The system call you add must return information about processes in the RUNNING or RUNNABLE state as an array of struct `proc_info`. This array must be sorted an ascending order according to the memory usage of each process. Structure `proc_info` is defined as follow.

```
struct proc_info {
    int pid;
    int memsize;            // in bytes
};

```

You should write a test program for this system call. The test program may use the `fork()` system call to create some processes and `malloc()` system call to allocate some randomly sized memory for each process. Try to prove that your system call works as described. Note that:

- The process table is in the proc.c file;

- The proc struct is implemented in proc.h file. This struct represents the process control block;
- You should add the test file to UPROGS list in Makefile;
- If you define a new source file, you should add it to Makefile;
- You may need to share the process table so that other source files can access it.

Question 2.

Cross Memory Attach (CMA) is one of the models for Inter-Process Communication. The system calls `process_vm_readv()` and `process_vm_writev()` are based on this model. These two system calls were fully described in the class session. Assume one of the two following scenarios based on your student number and implement two C programs to satisfy the desired scenario. The students with odd student number should consider the first scenario and the others consider second.

1. A producer program should produce 3 Gigabytes data. A consumer should read that data and prints out.
2. A producer produces 3 Gigabytes of data and then writes in the consumer's memory space.

Note that to synchronize the producer with the consumer you can use either `getchar()` or `sleep()`. Also, recall `memset()` from the class session to generate data.

Deadline

The non-extendable deadline for this homework is on **Friday, 1st November, at 23:55** (10th Aban).
The submission after this time will not be accepted with no exception.

Submission

Submit a *zip* file to lms. The file should be named as *your-student-number.zip*. E.g. *96521234.zip*. It is recommended to document what you have done in a pdf file and include it into your zip file.

- For Question 1: Make a directory named *q1* and only copy the files you have modified from Xv6 project.
- For Question 2: Make a directory named *q2* and put the codes you have written into it.

Finally, include these two directories in your zip file and submit it into LMS. Also, you must commit and push your changes for question 1 into *HW2_Q1* branch of your repository before the deadline.