Iran University of Science & Technology

Operating Systems
Fall 2020

https://os-course.github.io/fall20/

#5 Assignment - Xv6

Download

Prerequisite

We assumed that you have already cloned the Xv6 repository and added `shutdown` system call successfully as described step by step in class.
In these questions of this homework, you are asked to modify the kernel of Xv6 as the questions wishes. After completing the questions, please add the attached script file into the directory of `xv6-public`, and then run `./submission.sh`. Then, a file is generated in the **previous** directory of `xv6-public` named `xv6.tar`. The size of the file should not exceed **1MB**. If exceeds, `make clean` then run again the script.
Put `xv6.tar` beside your other codes in a zip file to submit on LMS.

Question 1

You may experience using `!!` command in terminal of Linux systems such as ubuntu. This command at the execution time returns the most recently used command in the terminal (i.e. previous command).
Add this feature to Xv6 kernel.
Hint: you should keep track of commands which are executed within the terminal.

Example

```
$ wc README
50 314 2246 README
```

```
$ !!
wc README
50 314 2246 README
$
```

Question 2

Challenging!

As hinted in the previous question, implement a mechanism in the console of Xv6 to keep track of commands that are executed on the system from the terminal. In this question, only the top 10 recently executed is enough.
Add a feature to the console which we can switch between previous commands using up and down arrow keys of the keyboard. (e.g. pressing 5 times up arrow key should show the fifth recent executed command)
A file `kbd.h` is attached. You may use the keyboard constants written in that file. However, this file is already included into the Xv6.

Question 3

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 in 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.

Deadline

- **Wednesday** 23rd Dec. 23:00

Submission

Submit just a zip file, containing your codes, in LMS. The file should be named as [`9752xxxx.zip`]. For example `97521234.zip` .