Operating Systems–2: CS3523
January 2022

Programming Assignment 2: Syscall Implementation

Submission Date: 30th January 2022 (Sunday), 9:00 pm

This assignment requires downloading the xv6 learning OS and implementing a system call in it.

Setup: Please install the following libraries on your Ubuntu system:
sudo apt-get install qemu qemu-system g++-multilib git-all grub2 grub-pc-bin libsdl-console-dev

$ Download file from:
https://drive.google.com/file/d/1sZv1_TCwpd2yrg3IT3t6wjUpyTtFf5rG/view?usp=sharing
$ tar -zxvf xv6.tar.gz
$ cd xv6

Booting xv6:
$ make
$ make qemu-nox

It will open a shell in your terminal where you can run some basic commands just like a unix shell. Try echo or ls or cat commands and see that they work.

Part-1: Understanding and tracing system call

Your first task is to modify the xv6 kernel to print out a line for each system call invocation. It is enough to print the name of the system call and the return value; you don't need to print the system call arguments.
When you're done, you should see output like this when booting xv6:

...
fork -> 2
exec -> 0
open -> 3
close -> 0
write -> 1
write -> 1

That's init forking and execing sh, sh making sure only two file descriptors are open, and sh writing the $ prompt.

Hint: modify the syscall() function in syscall.c.

Part-2: Implementing your own system call: date

Your second task is to add a new system call to xv6. The main point of the exercise is for you to see some of the different pieces of the system call machinery. Your new system call will get the current time and return it to the user program. You may want to use the helper function, cmostime() (defined in lapic.c), to read the real time clock. date.h contains the definition of the struct rtcdate struct, which you will provide as an argument to cmostime() as a pointer.

You should create a user-level program that calls your new date system call; here's some source you should put in mydate.c:

```
#include "types.h"
#include "user.h"
#include "date.h"

int main(int argc, char *argv[])
{
  struct rtcdate r;

  if (date(&r)) {
    printf(2, "date failed\n");
    exit();
  }

  // your code to print the time in any format you like...
  /* example output format should be like this:
          Year: 2016
           Month: 1 or January
           Date: 26
           Hour: 15
           Minute: 12
           Second: 11

  */
  exit();
}
```

In order to make your new date program available to run from the xv6 shell, add _mydate to the UPROGS definition in Makefile.

Your strategy for making a date system call should be to clone all of the pieces of code that are specific to some existing system call, for example the "uptime" system call. You should grep for uptime in all the source files, using grep -n uptime *.[chS].

When you're done, typing mydate to an xv6 shell prompt should print the current time/date.

Submission Instructions

Submission is to be done at the appropriate link. Just bundle the modified files as per below instructions.

1. Go inside the xv6 directory
2. make clean
3. tar -zcvf xv6.tar.gz * --exclude .git
4. Create a small report (report.pdf) of max. 1 page explaining your understanding of how system call works and what you learnt from this assignment.
5. Create a zip file containing xv6.tar.gz and report.pdf and upload it. The zip file should follow the name: Assgn2-<RollNo>.zip

Grading Policy

1. Part-1 working: 30%
2. Part-2 working: 45%
3. Report: 25%

Required background concepts:

You can go through the following document to learn about concepts related to this assignment. https://docs.google.com/document/d/1UOsRR5hO7SyiTjLVHg5qE0qrKawjsGOohq0G6zb7C5c/edit?usp=sharing