2020-05-18

Doc version 1.0

Computer Organization and Operating Systems: 94210

Assignment 1: system calls

In this assignment you will add a new system call to the Xv6 operating system.

This exercise will show you how function calls are passed between user space and kernel space.

The next assignment will use this one, so make sure you complete it and understand how the internal mechanism of the OS is working.

Before you start: you need to have the Xv6 system installed and make sure it runs.

This is covered in the document “Installing Xv6 and QEMU”

Note: This material was initially taught in MIT and is now used in courses in other universities. You can find a lot of helpful material by searching for *xv6 system call*

Your task is:

Add a command called lsof that will print the list of open files for all running processes.

\* sorted by ascending process id

\* columns: process id, file descriptor number, type: regular or pipe, readable (“R”), writable(“W”)

The output should be like:

$ lsof   
PID      fd      type    read/write   
1        0       regular         RW    
1        1       regular         RW    
1        2       regular         RW    
2        0       regular         RW    
2        1       regular         RW    
2        2       regular         RW    
3        0       regular         RW    
3        1       regular         RW    
3        2       regular         RW  
$ echo "I love python" | lsof   
PID      fd      type    read/write   
1        0       regular         RW    
1        1       regular         RW    
1        2       regular         RW    
2        0       regular         RW    
2        1       regular         RW    
2        2       regular         RW    
4        0       regular         RW    
4        1       regular         RW    
4        2       regular         RW    
5        0       regular         RW    
5        1       pipe     W    
5        2       regular         RW    
6        0       pipe    R     
6        1       regular         RW    
6        2       regular         RW

In the above example, we see that each process has 3 file descriptors (0,1 and 2). process with pid 5 has file descriptor 1 as a pipe open for writing

and process with pid 6, has file descriptor 0 as a pipe open for reading.

Remember that in linux and xv6, stdin uses fd 0, stdout fd 1, stderr fd 2.

How we test:

Your code will be fed into automatic code tester. The output will be compared to a reference output.

* Hints:In Xv6, commands (in user space) are run by the shell. You need to write a user space program that will call the new system function and print the results.
* The new system call you will create bridges the worlds of user and kernel spaces, and therefore involves some ‘magic’ and affects several files. The magic is done by manipulating the stack, triggering an interrupt and changing internal processor state. Luckily we don’t need to understand exactly how it works – just look for existing commands.

## Submission

Create a patch file:

1. Commit your changes to the repository
   1. git add -u
   2. git add your\_new\_files # only if you have new files...
   3. git commit
2. create a patch
   1. git diff HEAD^ > HW1\_my\_id.patch
3. Upload the patch to Moodle.

In a few days I will enable the automatic check at

<http://homework-tester.westeurope.cloudapp.azure.com/94210/submit/hw/1>

You will be able to upload your HW1\_5454545\_7897897.patch file and see your result.

DUE DATE: 28th May 2020, 23:55

Note: This assignment is 10 days so you have more time for the next task which will be give shortly after the above date.

Note: please post any questions regarding technical aspects of the assignment in the Moodle forum

Tip: if you want to see value of variables when debugging with gdb, you have to tell the compiler to “make its optimizations friendly with gdb” by editing the file Makefile in the xv6-public directory:

Find the line

CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer

And replace one letter:

CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -Og-Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer

Then run

make clean qemu-nox-gdb

Tip:

To break out of the xv6 running environment:

If you are connected from Windows using putty: press ALT CTRL a, then c, then q

If you are connected from linux: press ALT a, then c, then q