HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
OFFICE FOR INTERNATIONAL STUDY PROGRAMS

BK
TP.HCM

BK - OISP

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEM

Assignment 1
SYSTEM CALL

Lecturer: Nguyễn Minh Trí

Name: Lê Trung Hiếu
Student Id: 1852365

Table of content

1/ INTRODUCTION

System call is a programmatic way in which a computer program asks for an administration from the part of the working framework executing on and also is a way for programs to interact with the operating system. When a program makes a request to the operating system's kernel, it calls a system call. System call provides the services of the operating system to the user programs via Application Program Interface(API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

2/ METHODOLOGY

a. *Preparation linux kernel*

Preparation

First, I used Parallels Desktop for macOS Catalina (version 11.15) as a virtual machine to install Ubuntu version 12.04 from http://www.osboxes.org/ubuntu/.

Then I got Ubuntu's toolchains (gcc, make, and so forth) by installing the build- essential metapackage with the following command line:

> *$ sudo apt-get update*
> *$ sudo apt-get install build-essential*

And of course, kernel package should also be install:

> *$ sudo apt-get install kernel-package*

QUESTION: Why do we need to install kernel-package?

ANSWER: Kernel-package is a service for building a kernel in linux. This package automates regular steps required in compiling and installing a customized kernel.

Then, I installed the package openssl as required to download the kernel source version 4.4.56 and extracted them to a new folder /kernelbuild.

QUESTION: Why do we have to use another kernel source from the server such as http://www.kernel.org, can we just compile the original kernel (the local kernel on the running OS) directly?

ANSWER: The server above is the allocation for every source code to the kernel working on the Linux operating system. The website stores the source versions that users acquire. One more point, using the kernel source from the website above is easier to make changes in a kernel. We can also remove the kernel to rebuild the new one in case the process fails. Meanwhile, the original one is not affected.

In order to have the most efficient environment for my kernel and computer, I changed some options in .config. First, I copied the configuration file in /boot/ to the source code directory, installed libncurses5 − dev package and ran `$ make nconfig` to change the kernel local version.

b. *System call - procsched*

Kernel module

To save the number of times compiling the kernel, I used the Kernel module to test if the program would run or not.

Prototype

Prototype helped me to understand the meaning and input of a system call.

Implementation

First, I added a new system call by going to arch/x86/entry/syscalls and inserted some definitions:

> *syscall_32.tbl: 377 i386 procsched sys_procsched*
> *syscall_64.tbl: 546 x32 procsched sys_procsched*

I added in both files to make sure the system call will work in any x86 processor.

QUESTION: What is the meaning of other parts, i.e. i386, procsched, and sys procsched?

ANSWER:
- 377, 546: [Number]: This is the identifier of a syscall. We use syscall' id to call a syscall, not the other part such as name, ABI, entry point...
- i386, x32 : [ABI] : Application Binary Interface, contains the value "i386" in 32-bit syscall and "64" or "x32" for 64-bit syscall.
- procsched : Name of the system call.
- sys_procsched : Entry point of the calling function.

The next job is that I defined this system call by adding necessary information to kernel's header files include/linux/syscalls.h
After finishing setting the syscall definition for my new kernel, I determined the file given in the requirement.

QUESTION*: What is the meaning of each line above?*

ANSWER*: Struct proc_segs : Prototype the definition of a struct in header file.
Asmlinkage long sys_procsched (int pid, struct proc_segs * info) : Prototype a function in the
header file. The key work asmlinkage tag tells the compiler that the function is working on the
CPU's stack instead of registers.*

c. *Compiling Linux kernel*

Build the configured kernel

In this working process, I used these command lines to compile and build the kernel:
> *$ make*
> *$ make modules*

QUESTION*: What is the meaning of each line above?*

ANSWER*: The make command is used for compiling all the implementations in the kernel
and links the kernel image into a single file named vmlinuz while make modules will build
all the modules that you configured.*

Installing the new kernel

I used these command lines to install the new kernel:
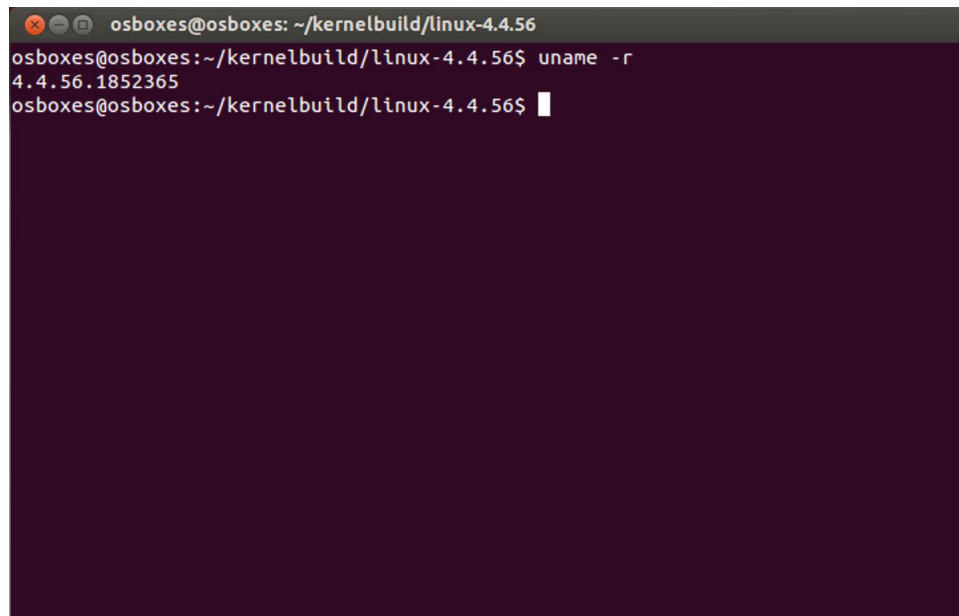> *$ sudo make modules_install*
> *$ sudo make install*

And to verify the work, I rebooted the virtual machine:

> *$ sudo reboot*

After logging into the computer again, I ran the following command to view the modified config:

> *$ uname -r*

The output string contains my student ID which means it has been compiled and installed successfully.

Testing

After booting to the new kernel, I created a small C program to check if the system call has been integrated into the kernel.

QUESTION: Why could this program indicate whether our system works or not?

ANSWER: In this program, the command syscall is included in <sys/syscall.h>. If the kernel was built successfully, the syscall would return the value in the info pointer, else the program would print out the killed message and kill the process. So even if the system does not work, this program still indicates.

d. *Wrapper*

In this process, I create another directory to store my source code for the wrapper. This will be done outside the kernel to avoid recompiling the whole system again.

QUESTION: Why do we have to re-define proc_segs struct while we have already defined it inside the kernel?

ANSWER: We re-define the proc_segs struct in the header file and put it outside of the kernel to avoid recompiling the whole kernel again. By doing that, we can leave the kernel directory and start the wrapper process in another directory.

After that, I made the header file available in GCC to make sure that the function could be accessible. So, I copied the procsched.h into /usr/include.

QUESTION: Why root privilege (e.g. adding sudo before the cp command) is required to copy the header file to /usr/include?

ANSWER: Sudo command is used to execute commands as a different user, such as user root. And the folder /usr/ is administered by the user root. The copy process is later saved in the folder that we need permission by sudo command.

Lastly, I only need to compile my source code as a shared object to allow user to integrate my system call to their application by the following command:
        $ gcc -shared -fpic procsched.c -o libprocsched.so

QUESTION: Why must we put -share and -fpic option into gcc command?

ANSWER: The option -shared is used to make a shared library and the option -fpic is used to make PIC (position independent code) which is compatible with the usage of shared libraries. We must include those options to create a shared object libprocsched.so.

3/ RESULT

After compiling and executing the source file. This is the final result that I have gained:

```
osboxes@osboxes: ~/Desktop
osboxes@osboxes:~$ cd Desktop/
osboxes@osboxes:~/Desktop$ gcc -g test.c -lprocsched
osboxes@osboxes:~/Desktop$ ls
a.out  syscall_module  syscall_test  test.c  wrapper
osboxes@osboxes:~/Desktop$ ./a.out
PID: 2657
Student ID: 1852365
pcount: 2
run_delay: 1715014801031168
last_arrival: 14204675179770019840
last_queued: 47
osboxes@osboxes:~/Desktop$ ./a.out
PID: 2658
Student ID: 1852365
pcount: 1
run_delay: 1021549381419008
last_arrival: 5310388025375588352
last_queued: 48
osboxes@osboxes:~/Desktop$
```

4/ REFERENCES

Samit Mandal, (2020). Introduction of System Call. GeeksforGeeks.
https://www.geeksforgeeks.org/introduction-of-system-call/

Shuah Khan, (July 10, 2014). Linux Kernel Testing and Debugging: Configuring Development and Test System. Linux Journal.
https://www.linuxjournal.com/content/linux-kernel-testing-and-debugging

Dr. Song Li, (June 14, 2018). A Note on Linux Directory Structure & Users & Permissions: Linux Group & User Associations, The Sudoers. Code Project.
https://www.codeproject.com/Articles/1246574/A-Note-on-Linux-Directory-Structure-Users-Permission

Le Dinh Tri, (2017). Operating Systems. Github.
https://github.com/ledinhtri97/OperaSystem_HK161_Assignment1