Shanghai Jiao Tong University

EI 338

Computer System Engineering

Adding A System Call To The Linux Kernel

Author:
Hongru Huang

Student Number:
515030910549

October 15, 2017

Contents

1 Project Assignment

In this project, you will study the system call interface provided by the Linux operating system and how user programs communicate with the operating system kernel via this interface. Your task is to incorporate a new system call into the kernel, thereby expanding the functionality of the operating system.

2 Project Implementation

This project is based on Linux Ubuntu 16.04.3 and the kernel version is 4.10.13. For other version of Ubuntu and kernel, this procedure is somewhat similar.

2.1 Preparing

Type the following commands to make sure the compiling process successfully:

- *sudo apt-get install gcc*
- *sudo apt-get install libncurses5-dev*
- *sudo apt-get update*
- *sudo apt-get upgrade*

2.2 Download the kernel

At first I plan to complete this project just on the kernel provided by Ubuntu. It turns out that I can't do that because the kernel provided by Ubuntu is well-compiled and I can't find the source code, the *syscall_64.tbl* file and so on. The most important thing is that you can't compile the modified kernel without source code. Therefore the first step is to download a kernel with source code:

wget https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.10.13.tar.gz

After successfully downloading it, extract the kernel source code in **/usr/src/** directory using the following command:

sudo tar -xvf linux-4.10.13.tar.gz -C /usr/src/

Then we can implement the project in directory **/usr/src/linux-4.10.13**.

2.3 Define a new system call sys_helloworld()

Create a new directory **hello** in the kernel folder. And create a **helloworld.c** file in **hello** folder. This **.c** file is the source code of our new system call. You can check the source code in the project file. Note that We can't use the **printf** function here because this program runs in kernel mode and the **printf** function only works in user mode.

Create a **Makefile** in the **hello** folder add the following line:

obj-y := helloworld.o

A makefile is a special file, containing shell commands, that you create and name makefile (or Makefile depending upon the system). While in the directory containing this makefile, typing *make* and the commands in the makefile will be executed. This line is to ensure that the **helloworld.o** will be compiled and included in the kernel source code.

2.4 Modify the kernel

2.4.1 Add the hello directory to the kernel's Makefile

Edit Makefile in the kernel folder. Find the line which says:

core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/

and change it as follows:

core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ hello/

We can find these files(**kernel**, **fs**, etc) in the kernel file. This is to tell the compiler that the source files of our new system call are in present in the **hello** directory.

2.4.2 Add the new system call into the system call table

My Ubuntu OS is 64bit. So firstly turn the directory **/arch/x86/entry/syscalls** and edit the file **syscall_64.tbl**. Add the following line after 331_th system call

332 64 helloworld sys_helloworld

Note that the blank is a tab not a space. 332 is the number of the system call. This has to be noted down to make the system call in the userspace program.

2.4.3 Add the new system call in the system call header file

Turn to directory **/include/linux/** and edit file **syscalls.h**. Add the following line to the end of the file just before the #endif statement at the very bottom:

asmlinkage long sys_hello(void);

This defines the prototype of the function of our system call. **asmlinkage** is a key word used to indicate that all parameters of the function would be available on the stack.

2.5 Update the kernel

2.5.1 Compile the kernel

To configure the kernel we need to use the following command:

sudo make menuconfig

This command is used to configure the Linux kernel and it will pop up a window with lists of menus. In this project we just click **save** and **exit**.

sudo make oldconfig

Then turn to the kernel file and type *make* to compile the modified kernel. It will take us several hours of time.

2.5.2 Install the kernel

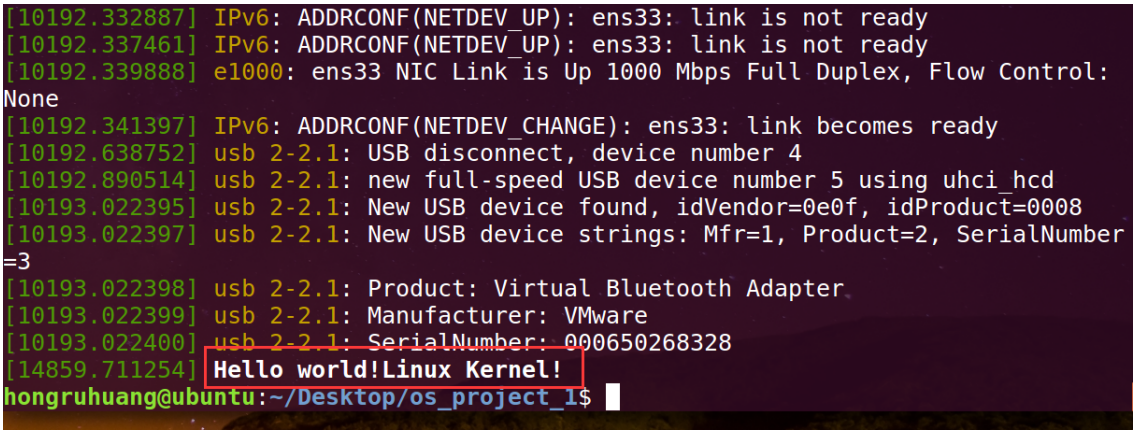
After successfully compiling the kernel, type the following command:

sudo make modules_install install

The command will install the Linux kernel 4.10.13 with our new system call **sys_helloworld**. After installation restart Ubuntu. If you type *uname -r* after rebooting you will see the output kernel version is 4.10.13.

2.6 Test the system call

Recall that we use the **printk** function in the source file of our new system call, which will not display any information on the terminal but will display information



in kernel's message. So we need another testing program to use the system call. You can check the test program in the project file.

Type *gcc testprog.c* to compile the test program. We will get a **a.out** file. Then type *./a.out* will display the information in the test program. Finally we can type *dmesg* to scan the kernel's message. If all steps are done correctly we will see **"Hello World! Linux Kernel!"** at the end of the message just like this:

3 Conclusion

This project gives me a deeper understanding about the kernel of Ubuntu Operating System. During this implementation procedure, I found many familiar files, like **stdio.h**, is visuable to us. This is very amazing because I think it is very secret before. And this makes me look at hello world program from another angle.

But there are many things I can't understand during this process, for example, the kernel configure window, which needs further study in this course.