SYSTEM CALL

Juhi Maheshwari (121020)

Kashyap Patel (121021)

Index

1. Project Description

The basic problem statement is to write a system call. The approaching step is to compile and install the latest version of kernel. Followed by, writing a system call and adding it to kernel. Then a kernel needs to be recompiled to test the system call function. A simple C program has to be written to invoke our system call.

2. The project is mainly divided into 3 part

1. Downloading and initialization subversion repository with a copy of recent stable version
2. Creation of new branch in that repository within that branch and then write a system call and register it with kernel
3. This part is testing part. In this part we compile and install our modified kernel and then write a simple C code to invoke system call.

3. What is System call?

System call is the fundamental interface between application and the Linux kernel. System call provides an essential interface between a process and the operating system.

A system call is an entry point into the Linux kernel. Usually, system calls are not invoked directly: instead, most system calls have corresponding C library wrapper functions which perform the steps required in order to invoke the system call. Thus, making a system call looks the same as invoking a normal library function.

System calls can be invoked from userspace process as well as other system call can also call the required system call.

There are 5 different categories of system calls:

Process control, file manipulation, device manipulation, information maintenance and communication.

a. Process Control

Process control is basically changing the state of process. Running process, if invokes an I/O request must be blocked or put in a waiting state.

b. File Manipulation

Some common system calls are create, delete, read, write and close.

c. Device Manipulation

Process usually requires several resources to execute. If these resources are available, they will be granted and control returns to the user process. Resources are released by process as soon as it terminates or has finished executing.

d. Information Maintenance

Some system calls exist only to transfer information between user program and kernel/OS.

e. Communication

It refers to inter-process communication. It can be done by message passing technique. Message-passing uses a common mailbox to pass messages between processes.

4. Why do we need system calls?

System calls acts as entry point to OS kernel. There are certain tasks that can only be done if a process is running in kernel mode. Examples of these tasks can be interacting with hardware etc. So if a process wants to do such kind of task then it would require itself to be running in kernel mode which is made possible by system calls.

System calls are also used for switching the execution mode. Example: systenter and sysexit

5. Difference between system call and library function

- A library function is linked to the user program and executes in user space while a system call is not linked to a user program and executes in kernel space.
- Library function's execution time is counted in user level time while a system call execution time is counted as a part of system time.
- Library functions can be debugged easily using a debugger while System calls cannot be debugged as they are executed by the kernel.

6. How to add System call in kernel

6.1Download Kernel from kernel.org

http://www.kernel.org/pub.linux/kernel/v3.x/linux3-16.tar.xz

6.2 Extract the kernel source code

Change directory from terminal to downloads by command
cd Download
Then extract that file to /usr/src directory by command
Sudo tar –xvf linux-3.16.tar.xz –C/usr/src

6.3 Go to Kernel directory

Go to directory where we extracted kernel source file. Go into directory (linux3-16)
Command 1: cd
Command 2: cd /usr/src/linux3-16

6.4 Define new system call

6.4.1 For that, make new directory called with any name. (Here Systemcall)

Command: sudo mkdir Systemcall

6.4.2 Then go to newly created directory

Command: cd Systemcall

6.4.3 Add Systemcall Definition

Create a “Syscall.c” file in this folder and add the definition of the system call.
Command: gedit Syscall.c
Definition contains c code of systemcall.

6.4.4 Create a Makefile into Systemcall folder

Obj-y: = Syscall.o
This is to ensure that the Syscall.c file is compiled and included in the kernel source code.

6.4.5 Add the Systemcall directory into Kernel’s Makefile

Change back into the linux-3.16 folder and open Makefile
Command: gedit Makefile
Go to line which says: - “core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/”
Change this to “core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ Systemcall/”
This is to tell the compiler that the source files of our new system call (sys_buffer()) are in present in the hello directory.

6.4.5 Add the new system call (sys_buffer()) into the system call table (syscall_64.tbl file)

If your system is a 64 bit system you will need to alter the syscall_64.tbl file else in syscall_32.tbl file. For that the command would be as follows

```
cd arch/x86/syscalls
gedit syscall_64.tbl
```

Add system call at the end of the file. The table would be containing four arguments. First argument tells about the number of system call. (In our system it was 318). Simply write common as the second argument. The third argument should be name of the directly where source code of system call lies. And fourth argument would be the name of system call.

6.4.6 Add the new system call(sys_hello()) in the system call header file using commands

```
cd include/linux/
gedit syscalls.h
```

Add the following line to the end of the file just before the #endif statement at the bottom.
asmlinkage long sys_buffer(void);

It defines the prototype of the function of our system call.

7. Kernel Compilation

Once you add system call to kernel then compilation of kernel is necessary thing.

7.1 Requirement for Kernel compilation

To compile Linux Kernel, there are some necessary packages which need to be installed or should be up-to date. The packages are gcc latest version and ncurses development package.

7.1.1 Kernel Configuration

To configure the kernel, the following command is used:

- sudo make menuconfig

Once the above command is executed, the configuration menu of kernel opens up. Any necessary changes can be made in kernel configuration. (Here we did not change anything).
Compile the changed kernel using command

- 'make' in kernel directory.

7.2 Install /Update the kernel.

Now the edited kernel needs to be installed. Hence for that following command has to be executed.

- Sudo make modules install

Now to update the kernel in your system reboot the system.

8. Test System Call.

To execute a system call, a c code has to written which calls system call. If system call has run successfully it should return 0 else it will return 1. The output of successful system call can be checked from dmesg. In our case it returns the head pointer of the buffer.

9. Work done till date

As mentioned before, the project was divided into 3 stages. All of those stages were completed long before. Then the task was to implement a buffer of size 5 as a system call. So whenever any other process needs a buffer, the only task would be to call that system call and buffer would be assigned. The system call returns the head position of the buffer. Hence a permanent buffer is made in system. So whenever needed you don't to make it explicitly.

10. Future work

Implementing producer consumer problem using the buffer implemented in system call. As producer produces the item, it gets stored into buffer and consumer can consume from that buffer.

11. Learnings

We learned how to interact with kernel, from user mode. i.e., writing a system call and calling it from application layer through c code. We tried to understand the kernel, kernel libraries, kernel functions and its structure. We learnt how to add system call to kernel and how to compile it afterwards. In all it was a complete project to learn system calls and we gained lots of learning from it.

12. Reference

1. http://www.tldp.org/LDP/lkmpg/2.4/html/x939.html
2. http://en.wikipedia.org/wiki/System_call
3. http://www.thegeekstuff.com/2012/07/system-calls-library-functions/
4. http://man7.org/linux/man-pages/man2/intro.2.html
5. http://faculty.salina.k-state.edu/tim/ossg/Introduction/sys_calls.html
6. https://tssurya.wordpress.com/2014/08/19/adding-a-hello-world-system-call-to-linux-kernel-3-16-0/
7. ftp://www.cs.uregina.ca/pub/class/330/SystemCall_IO/SystemCall_IO.html
8. http://www.csee.umbc.edu/~chettri/421/projects/hello_syscall.html
9. http://www.bottomupcs.com/system_calls.html
10. http://www.linuxjournal.com/article/3326