

# Experiment Report

--Adding a System Call to the Linux Kernel

Name:王星艺 StudentID:5140309531

## 1 Introduction

In this project, we 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. By incorporating a new system call into the kernel, we can expand the functionality of the operating system.

## 2 Running environment

- Ubuntu 12.04-i386
- Kernel Linux 2.6.38

## 3 Experimental procedure

1. Download a new Linux kernel from <http://www.kernel.org/>. I'll take linux-2.6.38.tar.bz2 as my new kernel. Then put it in the folder "/usr/src".
2. Decompress the kernel file using the following codes:

```
cd /usr/src
tar -jxvf linux-2.6.38.tar.bz2
```

3. The next step is to modify some files. Remember we should do this in *root* mode.

```
sudo su (then input your password)
gedit /usr/src/linux-2.6.38/kernel/sys.c
```

Then add the following function in the file:

```
asmlinkage int sys_mycall(int number)
{
    printk("Mission Report: December 16, 1991");
    return number;
}
```

This function will be able to test if the system call is successful. The output result is exactly the input parameter. And *printk* will be added to message if it's successful.

Then we need to modify the system call function table.

```
gedit /usr/src/linux-2.6.38/arch/x86/kernel/syscall_table_32.S
```

Add the following sentence in the file, and remember the number of this line (341):

```
.long sys_mycall
```

And the last file to be modified is opened by the following code:

```
gedit /usr/src/linux-2.6.38/arch/x86/include/asm/unistd_32.h
```

Add this sentence:

```
#define __NR_mycall 341
```

4. Then we need to compile the kernel. Before we do this, we need to solve a problem: “.size expression for apf\_page\_fault does not evaluate to a constant”. So we change END(ret\_from\_exception) into END(resume\_userspace) , and change END(apf\_page\_fault) into END(async\_page\_fault).

Now it's time to enter these codes in the terminal to compile the kernel:

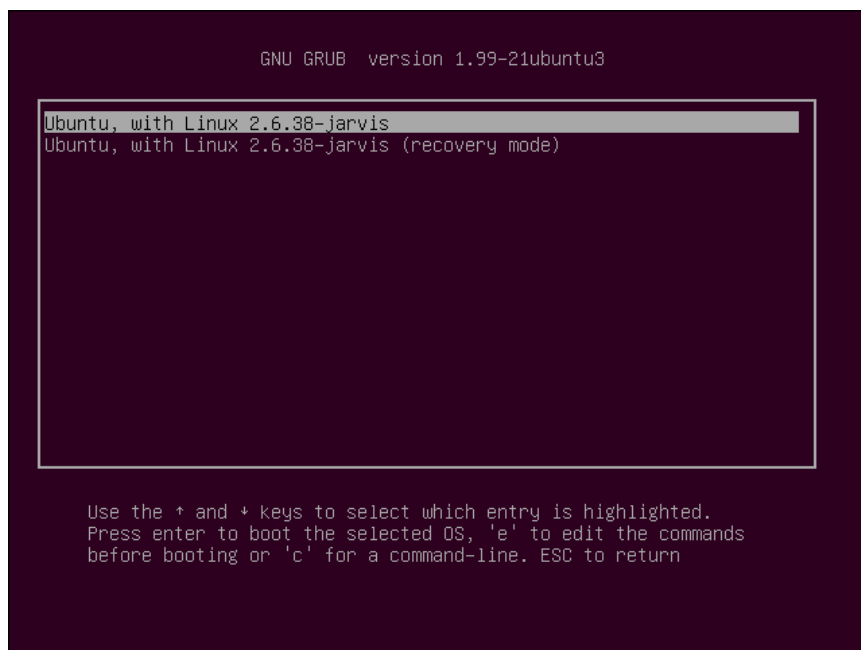
```
cd /usr/src/linux-2.6.38
make mrproper //clean the remaining files
make localmodconfig //simplify the kernel
sudo make-kpkg clean
sudo fakeroot make-kpkg --initrd --append-to-version=-jarvis
kernel_image
```

5. Install the kernel.

```
sudo dpkg -i linux-image-2.6.38-jarvis_2.6.38-jarvis-
10.00.Custom_i386.deb
```

Then we can reboot it to confirm the kernel is installed.

```
sudo reboot
```



6. Use this code to check the version of the kernel:

```
uname -r
```

7. Then we can write a program to use the system call we just wrote.

```
//test.c
#include <linux/unistd.h>
#include <stdio.h>
main(){
    printf("The number is %d\n", syscall(341, 100));
}
```

Compile the program.

```
gcc test.c
```

As the compilation is done, we'll get a file named "*a.out*". We open this file to see the result.

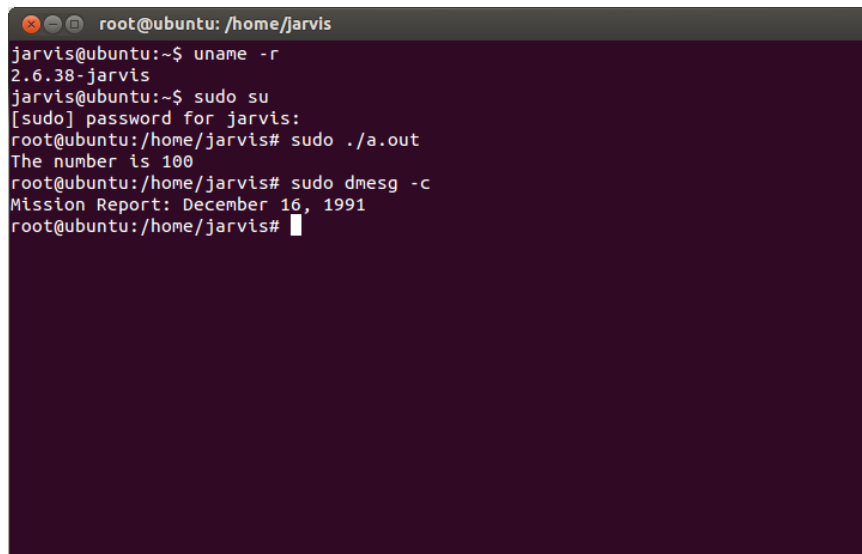
```
./a.out
```

If we get the sentence we wrote on the screen (The number is 100), then we can say the system call is successful.

Finally, we can check the message of the successful system call.

```
sudo dmesg -c
```

And the screen show: Mission Report: December 16, 1991 (LOL, have you seen Captain American 3?)



```
root@ubuntu: /home/jarvis
jarvis@ubuntu:~$ uname -r
2.6.38-jarvis
jarvis@ubuntu:~$ sudo su
[sudo] password for jarvis:
root@ubuntu:/home/jarvis# sudo ./a.out
The number is 100
root@ubuntu:/home/jarvis# sudo dmesg -c
Mission Report: December 16, 1991
root@ubuntu:/home/jarvis#
```

#### 4 Conclusion and Discussion

After successfully building a new Linux kernel in Ubuntu, I have a more specific understanding of the kernel mode. And it also helps me to understand system call. The version of Ubuntu and Linux kernel is a confusing problem, so we've tried a lot of versions and finally decide to use Ubuntu 12.04 and Linux-2.6.38. Online tutorials are not always appropriate either. So these problems cost me a lot of time. But getting used to Ubuntu and Virtual Machine is a good experience for me.