

CS 460 Assignment #2

ArchLinux, Kernel, More System Calls

Date assigned: Wednesday, February 12, 2014

Date due: Friday, February 21, 2014

50 points

Goal:

Learn how to add system calls to the Linux kernel using ArchLinux in a virtual environment and deal with serious frustration.

A VirtualBox image ArchLinuxStudent.vdi.gz running ArchLinux exists in the directory /home/CS460/2014 on zeus. On the following pages, there is a lab that will take you through adding a system call (helloworld) to the existing kernel. After recompiling the kernel, you will be able to call your added system call. After you complete this exercise in class, you are to write the following system calls that will add pseudo-random number generation to the kernel. Do not make any assumptions about these system calls!!!!

1) long setRandomSeed (unsigned long) – this system call will seed the pseudo-random number generator with the value passed in. The seed value is to be made available to all functions in this module and is to have an initial value of 123456789. When setRandomSeed is called, the parameter is added to the current seed value which then becomes the new seed. Return the new seed value casted to a long.

2) long getRandom (unsigned long) – this system call will accept a number and return a value in the range [0, number). To implement this function, first compute a new seed value by multiplying the current seed value times 69069 and then adding 362437 to the product. After performing both algebraic computations, the new value becomes the new seed. Without changing the new seed, but using the new seed, return a value in the range [0,number). For instance, if the new seed value is 100 and the value of number is 8, then the value 4 is returned. Return the correct random number casted to a long.

3) long getRandomBetween (unsigned long, unsigned long) – this system call will accept two numbers and return a value in the range [number1, number2] using the pseudo-random number algorithm described in 2). The only difference is that after calculating the new seed value, you are returning a value in the range of number1 to number2 inclusive casted to a long.

Once you have added all three system calls to the kernel (**I WOULD DO ONE AT A TIME**), you are to write a driver `CS460_RandomTest_PUNetID.c` that tests the system calls as follows:

- Set the seed value to 3
- Generate 1,000,000 random numbers in the range of [0,99]
- Output each random number (20 random numbers per line right justified in a field of 4) with a proper heading
- Output the total number of random numbers in the following ranges [0,9], [10,19], [20,29], ... as follows:

RANDOM NUMBER SUMMARY

Randoms in [0, 9]: XXXXXX

Randoms in [10,19]: XXXXXX

Randoms in [20,29]: XXXXXX

and so on

On the day the assignment is due, you are to turn in hard copies of your system calls and the driver. Further, I will ask you to demo your solution and make a few changes to the driver. When you are ready to demo, you must turn in printouts and I will not take any further changes to the assignment at that time. Start early, there could be a fair amount of compile time involved if you don't know what's going on and aren't careful.



Open a Linux console. Start VirtualBox

This was update on April 19, 2012 to work with archlinux-2011.08.19-core-x86_64.iso and ArchLinuxStudent.vdi.

\$ VirtualBox

Edit your existing Virtual Machine:

System:

Motherboard: Enable IO APIC
Base Memory: 4096 MB
Processor: 4

(login as **root**)

Username: root
Password: CS460!!pac

Wait for the Desktop to come up.

Important Icons:

Down desktop | Terminal FileManager Browser Search Geany | Home Folder



Now we are ready to work.

Make sure the date and time are correct. **DO THIS EVERY TIME YOU REBOOT!!!!!!**

In a few cases, there is no way to fix a clock skew problem easily, so if things get out of time, you may have to start over!!!!

```
[root@myhost]# date  
[root@myhost]# date MMDDhhmm[CCYY]
```

For Example:

```
[root@myhost]# date 042016452012
```

Since we will be hacking on the kernel, lots of bad things can happen. Let's backup the kernel. In the console, change directory to /boot. Make a back up of `vmlinuz` and `System.map`

```
$ cd /boot
$ cp vmlinuz-linux vmlinuz-linux.backup
$ cp initramfs-linux.img initramfs-linux.img.backup
```

Let's edit the boot menu to use our backups. Open Geany (via the Desktop icon) and use it to edit the `/boot/grub/menu.lst` file.

```
geany /boot/grub/menu.lst
```

Add these five lines **below the five lines starting with # (0) Arch Linux** to look as follows:

```
# (1) Arch Linux Fallback
title Arch Linux Fallback
root (hd0,0)
kernel /vmlinuz-linux.backup root=/dev/disk/by-uuid.....
initrd /initramfs-linux.img.backup
```

Copy the same 5 lines and paste them below the # (1) Arch Linux Fallback definition

Edit these five lines to look as follows:

```
# (2) Arch Linux CS460 Custom
title Arch Linux CS460 Custom
root (hd0,0)
kernel /vmlinuz-linux-3.1.9-CS460-trad root=/dev/disk..... ro
initrd /kernel-3.1.9-CS460-trad.img
```

Change the kernel identifier. Open the file `~cs460/kernel/linux-3.1.9/Makefile` with the edit tool. Change `EXTRAVERSION` to `-CS460-trad`

Now, we are ready to build the kernel. In the console, go to the `kernel/linux-3.1.9` directory.

Run these next four commands **ONCE (and never again!)**:

```
$ make mrproper
$ ln -s /bin/lsmmod /sbin/lsmmod # already done for you!
$ zcat /proc/config.gz > .config # get current configuration from the kernel
$ make oldconfig # update with any new configuration options
```

Run the following 5 commands each time you want to rebuild and install the custom kernel.

<optionally remove the CC=".." from each command line>

```
$ time make CC="ccache gcc" -j 6 # ~13 minutes with 4 CPUs
$ time make CC="ccache gcc" modules_install # ~60 seconds
$ cp -v arch/x86_64/boot/bzImage /boot/vmlinuz-linux-3.1.9-CS460-trad
$ cp -v System.map /boot/System.map-3.1.9-CS460-trad
$ mkinitcpio -k 3.1.9-CS460-trad-ARCH -g /boot/kernel-3.1.9-CS460-trad.img
```

If you get time errors (clock skew), run:

```
find . -name '*' | xargs touch
```

Reboot and select Arch Linux CS460 Custom!

Guest Additions

DO NOT INSTALL GUEST ADDITIONS!!!!!!!!!!

Devices | CD/DVD | Remove Disk | Force Unmount

Reboot (in Custom kernel) (ArchLinux *may* abort the reboot. Just restart the VirtualMachine)

Run `ls -al /boot` to see your new kernel! Run `uname -a` to see the running kernel version.

If it does not restart properly, use the Machine | Reset menu option and choose the Fallback Kernel.

Adding a system call!

Let's add a simple system call that will print "HELLO WORLD" to the logs (/var/log/messages) and return a value of 42 to the user program.

Create a new file (CS460_Syscalls_PUNetID.c) in the directory
~cs460/kernel/linux-3.1.9/kernel

The file should contain:

```
#include <linux/linkage.h>
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(helloworld)
{
    printk(KERN_EMERG "HELLO WORLD!");
    return 42;
}
```

Edit the **Makefile** in that directory and add `CS460_Syscalls_PUNetID.o` to the end of the `obj-y` list.

Edit `~cs460/kernel/linux-3.1.9/arch/x86/include/asm/unistd_32.h` and look for the list of `__NR_???`. Add a `#define` to the end of the list:

```
#define __NR_helloworld 347
Update NR_Syscalls to 348
```

Edit `~cs460/kernel/linux-3.1.9/arch/x86/include/asm/unistd_64.h`
Immediately before `#ifndef __NO_STUBS`
add the following:

```
#define __NR_helloworld 310
__SYSCALL(__NR_helloworld, sys_helloworld)
```

Add :

Edit `~cs460/kernel/linux-3.1.9/include/linux/syscalls.h` (this is the non-architecture specific version of the file). Add the following line:

```
asmlinkage long sys_helloworld(void);
```

before the final `#endif`

Edit `~cs460/kernel/linux-3.1.9/arch/x86/kernel/syscall_table_32.S` At the bottom add:

```
.long sys_helloworld
```

Build and install the kernel as described above. Reboot into the new (Custom) kernel.
<warning future time stamp???

Add the file `/usr/include/CS460.h` which contains:

```
#include <linux/unistd.h>
#include <sys/syscall.h>
#define sys_helloworld 310
#define helloworld() syscall(sys_helloworld)
```

This header file exposes the `helloworld` system call to the user. Normally, this code would be in a shared library (such as `glibc`) but for simplicity we will just put it in a header file.

Write a test case. In your home directory, create the file `CS460_TestSyscalls_PUNetID.c`
This file should contain:

```
#include <stdio.h>
#include <CS460.h>

int main(void)
{
    int c;
    c = helloworld();
}
```

```
printf("System call returned %d\n", c);  
  
return 0;  
  
}
```

```
gcc -o CS460_TestSyscalls_PUNetID CS460_TestSyscalls_PUNetID.c -g  
./CS460_TestSyscalls_PUNetID
```

Run your new executable. **Be sure you have rebooted to the Custom kernel since installing the new kernel!** To see the hello world message in the logs run

```
$ dmesg | tail
```

NOTES:

You may need to resize the screen after a reboot.

You may need to reinstall the VirtualBox Guest Additions after a kernel rebuild: See above.

Make sure "Use host I/O cache" is checked for both the SATA Controller and IDE Controller. (Settings | Storage)

Key Files For Repeated Editing

```
~cs460/kernel/linux-3.1.9/Makefile  
~cs460/kernel/linux-3.1.9/arch/x86/include/asm/unistd_32.h  
~cs460/kernel/linux-3.1.9/arch/x86/include/asm/unistd_64.h  
~cs460/kernel/linux-3.1.9/include/linux/syscalls.h  
~cs460/kernel/linux-3.1.9/arch/x86/kernel/syscall_table_32.S  
/usr/include/CS460.h
```

QUESTIONS

Before starting this assignment, you need to be able to answer the following 2. questions. Trust me!!!!

1. Describe what SYSCALL_DEFINE0 does. Where is this defined?
2. Describe what syscall() does. Where is it defined?

References

<http://www.cs.columbia.edu/~hgs/teaching/os/hw3.html>

<http://lxr.free-electrons.com/source/?v=3.1>

<http://users.sosdg.org/~qiyong/lxr/source/?v=3.x>

page 93 in your book.

<http://enzam.wordpress.com/2011/03/26/how-to-add-a-system-call-in-linux-kernel-ubuntu-os/>

http://tldp.org/HOWTO/html_single/Implement-Sys-Call-Linux-2.6-i386/

https://wiki.archlinux.org/index.php/Kernel_Compilation_From_Source

https://wiki.archlinux.org/index.php/Arch_Linux_VirtualBox_Guest