CS460 – In Class Kernel Lab
May 6, 2010

http://www.puppylinux.com

http://www.virtualbox.org/

Open a Linux console. Start VirtualBox

```
$ VirtualBox
```

Click through the agreement, etc. Create a New machine.

OS Type: Linux 2.6
Base memory: 1024MB

Hard drive: Normally you would create a new hard drive here. We will attach to an existing hard drive. Open another Linux console. Copy the file /updates/CS460s10Clean.vdi.zip to your home directory.

```
$ cp /updates/CS460s10Clean.vdi.zip .
$ unzip CS460s10Clean.vdi.zip
```

Go back to VirtualBox, choose Existing..., choose Add, select the file you just unzipped.

Next, Finish.

Select the machine and Start. Choose “**Backup Linux**”

Wait for the Desktop to come up. To get the mouse out of the Desktop and back to the Lab machine's Desktop, press the right control key.

Important Icons:

edit --- Geany
console --- command line shell

Now we are ready to work.

Enable the network:
Click the Setup Icon
Choose “Connect to Internet or Intranet”
Choose “Internet by network or Wireless LAN”
A dialog box should pop up with eth0, Ethernet, pcnet32 selected. Press the eth0 button the right of the dialog box.
Choose “Auto DHCP”
After a few seconds a Success! dialog box should appear. Press Yes to save your changes. Then click Done. You can close the “Puppy Setup” dialog box now.

Open a console. Checkout your game of life source code.

```
$ svn co svn+ssh://login@zeus.cs.pacificu.edu/home/login/SVNROOT/CS460_Life
```

Download the table files from the web page using the Browse Icon.

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 vmlinuz.backup
$ cp System.map System.backup
```

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

Change the first boot option to use your backup file:
original

```
kernel /boot/vmlinuz root=/dev/sda1
```

changed

```
kernel /boot/vmlinuz.backup root=/dev/sda1
```

Change the second boot option to use the kernel you will make today.
original

```
kernel /boot/vmlinuz-2.6.30.5-custom root=/dev/sda1
```

changed

```
kernel /boot/ vmlinuz-2.6.30.5-PUNetID root=/dev/sda1
```

Change the kernel identifier. Open the file /usr/src/linux/Makefile with the edit tool.
Change EXTRAVERSION to **.5-PUNetID**

Now, we are ready to build the kernel. In the console, go to the /usr/src/linux directory.

Run the commands:

```
$ time make bzImage
$ make install
```

Note: make bzImage may take 15 minutes!

This builds the kernel and installs it in /boot. Run **ls -al /boot** to see your new kernel!
Run **uname -a** to see the running kernel version.

Reboot and select the new (Custom) kernel! (Menu button, Shutdown, Reboot).

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

Is the network available after rebooting? If not, we'll fix that later.

Adding a system call!

Let's add a simple system call that will print “HELLO WORLD” to the logs and return a value of 42 to the user program.

Create a new file (**CS460_Syscalls_PUNetID.c**) in the directory **/usr/src/linux/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 **/usr/src/linux/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 335
```

Edit **/usr/src/linux/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.

Edit **/usr/include/asm/unistd.h** and add the line
#define __NR_helloworld 335
after all the other #define __NR_ lines.

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

```
#include <linux/unistd.h>
#define helloworld() syscall(__NR_helloworld, 0)
```

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**
It should contain:

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

int main()
{
	int value = helloworld();
	printf(“return value: %d\n”,value);
	return 0;
}
```

build a **Makefile** to build the executable.

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
```

Is the network available? If not, you need to rebuild your modules. Remember, Linux uses dynamically loadable modules. Go into /usr/src/linux run (WARNING: This will take 50 minutes!):

```
$ make modules
$ make modules_install
```

You can reboot or use the Setup icon to re-enable the network interface. Load the **pcnet32** module.

Use man, Google, and your book to answer the following questions. Submit these answers as a GoogleDoc (CS460_InClass_PUNetID) by Monday, May 10, 6 pm. These questions are worth a homework grade.

1. Describe what SYSCALL_DEFINE0 does. Where is this defined?
2. Describe what syscall() does. Where is it defined?
3. What does printk do and what KERN_ ??? flags are available besides KERN_EMERG? What does each flag do?
4. What happens if you run your test program using the backup kernel? Why does this happen?

References

http://www.linux.org/docs/ldp/howto/Implement-Sys-Call-Linux-2.6-i386/index.html
http://www.cs.columbia.edu/~hgs/teaching/os/hw3.html
http://lxr.free-electrons.com/source/?v=2.6.30
http://users.sosdg.org/~qiyong/lxr/source/?v=2.6.30
page 74 in your book.