CS460 – In Class Kernel Lab
April 22, 2008

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. Change directory to **~/.VirtualBox/VDI** Copy the file /updates/CS460s08.vdi.zip to that directory.

$ cd ~/.VirtualBox/VDI
$ cp /updates/CS460s08.vdi.zip .
$ unzip CS460s08.vdi.zip

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

Next, Finish.

Select the machine and Start. Choose “Linux Original”

Wait for the Desktop to come up. To get the mouse out of the Desktop, press the right control key.

Important Icons:
- edit --- nice source code editor
- console --- command line

Now we are ready to work.

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 bigTable file from the web page.

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/hda1 ro vga=790

changed
kernel /boot/vmlinuz.backup root=/dev/hda1 ro vga=790

Change the second boot option to use the kernel you will make today.
original
kernel /boot/vmlinuz-2.6.21.7-custom root=/dev/hda1 ro vga=790

changed
kernel /boot/ vmlinuz-2.6.21.7-PUNetID root=/dev/hda1 ro vga=790

Change the kernel identifier. Open the file /usr/src/linux-2.6/Makefile with the edit tool.
Change EXTRAVERSION to **.7-PUNetID**

Now, we are ready to build the kernel. In the console, go to the /usr/src/linux-2.6 directory.

Run the commands:

$ make bzImage
$ make install

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 kernel! (Menu button, Shutdown, Reboot).

If it does not restart properly, use the Machine | Reset menu option and choose the Original 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-2.6/kernel

The file should contain:

```
#include <linux/linkage.h>
#include <linux/kernel.h>
asmlinkage int sys_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-2.6/include/asm/unistd.h and look for NR_syscalls. Add a #define to the end of the list

above it:

#define __NR_helloworld 320

Change the value of NR_syscalls to 321

Edit /usr/src/linux-2.6/arch/i386/kernel/syscall_table.S At the bottom add:
.long sys_helloworld

Build and install the kernel as described above.

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

```
#include <sys/syscall.h>
#include <linux/unistd.h>
#include <stdio.h>

#define __NR_helloworld 320

main()
{
        int value = syscall(__NR_helloworld,0);
        printf("return value: %d\n",value);
}
```

build a Makefile

```
all: CS460_Testsyscalls_PUNetID

CS460_Testsyscalls_PUNetID: CS460_Testsyscalls_PUNetID.c
        gcc -o CS460_Testsyscalls_PUNetID CS460_Testsyscalls_PUNetID.c
```

Run your new executable. Be sure you have rebooted 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-2.6 run (WARNING: This will take 30 minutes!):

$ make modules
$ make modules_install

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