**System Administration Decal**
Intermediate Lab #2
March 6, 2008

## Introduction

This week, we broadly covered how a Unix filesystem is organized, and how there are many other different types of files than just normal files and directories. We poked around on a Linux system and found many interesting files. Finally, we also talked about how file extensions don't matter in the Unix world, at least not as much as they do on Windows. Since this isn't too fascinating, the subtheme of the following exercises is to prepare you for next week's lecture on file permissions, owners, and groups.

Please send your responses to joshk.decal@triplehelix.org and remember to include your `cs198-XX` username. If you have questions, don't forget to come to my office hours from 3-4P on Tuesdays at the OCF Lab.

## I. Filesystems upon Filesystems

One thing that is special about Unix is that the filesystem is *device agnostic*. This means that you can browse through the filesystem, and /home could be on one hard drive while /boot is on another. Contrast this to Windows where different drives have completely isolated filesystems, differentiated by drive letters. Here, we'll take a look at this concept of *mounting* other filesystems on top of your *root filesystem* in Unix.

First, log in to a Linux machine at the OCF. (You do have an OCF account right?) You can see a list of Linux machines at the OCF at http://www.ocf.berkeley.edu/cgi-bin/ruptime2.pl. Linux machines are the "AMD Athlon" ones. Just append ".ocf.berkeley.edu" to the name of the machine you want to use and use SSH to connect.

1. Run `df -h` at the command line. What does this command do? What can you say about where your home directory is stored? Is it on the machine you're logged into? If not, where is it?
2. How can you tell whether a particular mount is a network share or available from a hard drive connected to the computer? Feel free to guess.
3. Observe the entries that begin with "/stage" and look at the "Disk Used" / "Disk Available" output. What is strange about this?
4. What you saw in #3 are *bind mounts*. Look this up on Google and describe what they are in a sentence or two. How can you tell just by looking if a mount entry in `df -h` is a bind mount?

## II. The /proc Filesystem

A notable filesystem amongst those that are mounted on top of your root filesystem is the /proc filesystem. This is a special filesystem that is provided by the kernel that doesn't actually use any disk space – it is all in memory. Its purpose is to provide information about processes and allow you to twiddle some internal knobs within the kernel.

The proc(5) manpage on a Linux box can assist you throughout this problem.

1. Use the `ps` command to find a process that you own. Try to look it up in /proc and describe what it tells you about that process.
2. Use the `ps` command with an argument that lets you see other people's processes (what is it?) Choose a process that is not yours and try to examine it in /proc. What happens?
3. Use the output of `ls -l` in /proc to explain your answer for #2 more clearly.

   (**Hint:** The two username-looking entries in the output of `ls -l` represent who *owns* the file, and which *group* co-owns the file, respectively. To the left of that is the permission listing for the file, representing whether users, groups, or anyone else can **r**ead, **w**rite or e**x**ecute the file. If this isn't clear, don't worry about it. We will go over it in detail next week.)

4. What is `/proc/kcore`? What happens when you read it? Do you think it's a good idea for the behavior to be this way? Why or why not?
5. Check out `/proc/net/dev`. Try to understand the table it provides – you may need to widen your terminal window to view it correctly. Find a way to use the `watch` command (look it up!) to provide a running update of its output.
6. `/proc/self` is really special. What does it do? How does it accomplish its task? Again, use `ls -l` to examine it.

**III. Device Files – The /dev Directory**

Now we'll take a brief look at device files, which are all located in /dev. Have http://www.lanana.org/docs/device-list/devices.txt open in a browser window for this exercise.

1. Look through /dev and find a way to redirect the output of a process to standard error using a file in /dev.
2. Check out the /dev/zero and /dev/full files. How are these similar to /dev/null? (It's *not* because "full" is only one letter away from "null".)
3. Use `df -h` to figure out the root partition for the machine you're on. Try `cat`ing it. Why can't you do it, and why is it good that you (as a normal user) can't?
4. Use `ls -l` to look through /dev. Why aren't there any file sizes present? What replaces them in the output of `ls`? (**Hint**: Research the `mknod` command.)

You're done!