

Unix® Laboratory 1: COSC 416 Operating Systems, Spring 2007

Goals

Three goals of this lab are (1) to practice the Unix **command line environment**, (2) to draw some conclusions about Unix **processes**, and (3) to work as a team.

Very little that you do here will be special to any one brand of Unix such as Linux, Solaris, HP-UX, FreeBSD, or AIX. Very little that you do here will be special to any particular command line interpreter (shell) such as **cs**, **bash**, **sh**, **bourne**, or **korn**. Take notes on another piece of paper, so that you can write your answers to the numbered questions as a graded assignment due on the date announced in class. You do not need to answer all of the questions correctly to earn an A on this lab. I put some hard ones in along with the easy ones so that those who have had previous exposure to Linux would be as challenged as those who are completely new to it.

Part I: If you never used Unix before, in the Lab

Getting Started

Do the work in the order presented. Some commands assume that the previous ones have been run correctly. Although I have attempted to be generic, Unix is often tailored to different environments. If `<backspace>` doesn't delete for you, try `<ctrl-backspace>` or `<delete>` instead. If you use a different Unix the spelling dictionary may be in `/usr/dict/words`, whereas below you see it's in `/usr/share/dict/words`.

All computers in Frey 366 should boot to SuSe Linux. Don't boot your machine until you have read far enough below to know what to watch for as it boots. Your username is **root**, and your password is as specified on the wall. Since root can do damage, please don't try random commands about which you haven't yet learned.

Powering Up

1. List several of the "system services" that are started as you boot up, especially any that are flagged as not being available. You may use the `<Scroll Lock>` key to freeze the screen, except that doesn't prevent the booting from taking place, so things after a freeze may go by even faster. When you log off at the end (by typing `<CtrlAltDelete>`) for one way, you will get another chance to see the names of some services. By convention, services whose names end in `d` are "daemons" (pronounce it "demons") which is to say "background processes." Question 16 below asks about them.

Linux OS boots to a windowed mode, so open a terminal by finding a menu option that does that.

Getting Help: There are two ways to explore help in Unix. The first is to type the command **apropos** followed by a word or fragment of a word about which you want to know more. Try this right now with

```
apropos whoami
```

The second is with the `man` command. The command **man** means manual, and gives you help on a command, one screen at a time. In a manual page ("man page") you move forward with the *space bar*, back with the *b* key, and quit with *q*. Those ways to move in a manual are also the way to move in any file (such as the on-line dictionary that we will be using) when using the **more** command, since the `man` command is actually another command "piped into" the `more` command. (We will learn about Unix piping later.) You can of course say 'man man' to get help on man pages in general! Read about *apropos* by typing

```
man apropos
```

and about *more* by typing

```
more /usr/share/dict/words
```

(or `more /usr/dict/words` depending on your version of Unix). For some reason the *b* key wasn't working for me in SuSe Linux yesterday for Linux's `more` command, but there is another Linux command with more features called (!) **less** which does work properly. So you might say instead `less /usr/share/dict/words`

2. By exploring the *whatis* and the *apropos* commands tell me a difference between them.

3. By exploring the *whoami* and *who am I* commands tell me a difference between them.

4. By exploring the *who* and *w* commands, tell me a difference between them.

Child and Parent Processes. Type the command `ps -u`

That invokes the process `ps` as a child process of the bash shell that you are using. But since `ps` itself means “show me the processes” you should see them both in response to this command.

5. What are some of the states that a process can be in, in Unix? Hint: skim the `ps` man page for a table of letters like those found under STAT in the response that you saw to the `ps -u` command.

6. What are some other “users” (owners of processes) besides root? Hint: type `ps aux | less`

Additional Command, Pipes, Filters, and Redirection

Execute the following commands in order. Turn in only answers to the numbered questions, which continue numbering.

```
alias ls=/bin/ls
The folks who set up our computers wanted
to help us by defining the default behavior
of `ls` differently from what happens when
Linux first began. This will restore the
default behavior of that command for the
rest of this session, so we're standard.
man head
head /usr/share/dict/words
tail /usr/share/dict/words
7. Is there a way to show other than 10
lines of the data?
more /usr/share/dict/words
Here be sure to leave using the q for quit,
or you will be frustrated waiting until the
end of a very long file.
man grep
This is big. Don't read it all, just the
beginning.
grep 'he.*he' /usr/share/dict/words
grep '^he.*he' /usr/share/dict/words
grep 'ookkee' /usr/share/dict/words
mkdir /temporary
mkdir /temporary/junkdir
cd /temporary
man cat
head /usr/share/dict/words > x
tail /usr/share/dict/words > y
The symbol > redirects the standard output
to a file. The symbol < redirects the
standard input from a file.
cat x
cat y
man sort
cat y x | sort | grep '^a' | wc
This is to be typed all on one line. What
do the man pages for sort, uniq, and wc say
that they do? What does ^ mean in grep? Try
this command by leaving off all the pipes,
then adding them one at a time, as in just
cat y z as the first step, and
cat y z | sort as the second. These | are
(unnamed) pipes to allow one process to
communicate with another. They differ from
DOS pipes in that DOS pipes use a temporary
file on the disk which must be completely
written by the first process before the
second process can read it. In Unix, a
pipe is like a queue, and the consumer can
begin to read it even as data continues to
be produced and added to it.
man ls
ls
ls -l
ls -a
ls -la
Give an example of a directory that is
invisible to `ls`, but not to `ls -a`
Tell me what is in one such file.
sort /usr/share/dict/words | wc &
Type the next line (ls) really fast before
this line completes its work.
8. What did the & do?
ls
cp /dev/tty x
This will allow you to type several lines,
which will go into the file named x (cp
means copy, and /dev/tty means your
keyboard). End your typing with a <ctrl-D>
(the Unix equivalent of the DOS <ctrl-Z>
end of file code. You will be prompted
about whether you would like to overwrite
x. Say yes. If you accidentally type
<ctrl-Z> by force of habit, you have "put
the job in the background." You recover
from that by typing the command fg (which
means put the job in the foreground), and
then you can say <ctrl-D>.
cat x
To see if x was created OK, type it.
rm x
To delete x. Beware! Unix may quietly
delete things without asking, even when you
use a wildcard! DOS instead prompts for
confirmation with wildcard deletes. rm
means "remove."
pwd
Means to print the working directory.
cd ..
If you are used to typing cd.. in DOS
without the space, you may get this wrong.
Since Unix allows two dots in
a filename and DOS doesn't, Unix must
enforce a space after the command cd. Some
folks set up their Unix boxes in such a way
that a new verb cd.. has been defined to
accommodate such forgetfulness. You can
see how we modify Unix in such ways by
typing the command alias, which shows all
the verbs we defined for our convenience.
Now we will allow the floppy disk to be
addressed. The name of the floppy disk
device is /dev/fd0 and so if you would like
to put it somewhere in Linux, you must
"mount" it.
mkdir mydisk
This gives a name to what will become the
floppy disk.
mount -t msdos /dev/fd0 mydisk
This tells Linux that the floppy disk uses
the msdos file system type.
cd /mydisk
ls
ls -l
ls -la
You see a file on the disk. Do what the
message in the file says if you can, by
email.
cd /temporary/j*
Unlike DOS, wildcards can be used in place
of words almost everywhere. In DOS, one
cannot wildcard a directory. Notice that
```

```
inside the grep command above we
needed "." to mean multiple
characters, but on the command line,
just as in DOS, only "*" is needed.
Well, not exactly like DOS, for in
DOS, dir *p gives all files whereas
in Unix, ls *p gives only files
ending in p. I think that DOS is
simply broken in this respect. DOS
seems to ignore the p. Likewise, DOS
does not permit * in a directory, as
in
cd /usr*/lib, but Unix does!
```

```
umount /dev/fd0
If you got as far as the bottom of page 2,
then you should type this before you leave.
You are finished with the floppy. This
command disconnects the floppy from the
Linux directory structure. Then type:
exit
Don't type this right away if you are
continuing with Part 2 of this lab;
type it at the end of your session.
Then finally enter the <ControlAlt-Delete>
keychord, and read as much as you can while
the machine is shutting down, just as you
watched the machine boot up. I will ask
you at the end of the lab what you saw.
```

Pipes, filters, and redirection work together. Let me explain some of the symbols above. The symbol | is called a pipe. The symbol > is output redirection. (There is also < for input redirection, and >> for output appending instead of replacing.) The Unix tools *uniq* and *wc* are examples of filters. A filter is a program that connects the output of one program to the input of another program.

9. How does the **top** command differ from the **ps -aux** command? Hints: Don't just say that they give different columns, since **top** can be configured to give the same columns that the **ps -aux** command gives. Hitting the space bar while watching **top** might help answer this question. You exit from **top** using **q** as you did with the **more** command. You enter help options for **top** using **h**.

10. What information is found in the **/etc/passwd** file? You can either look at it by going to the **etc** directory and typing the file out to the screen, or you can look at the `passwd` man page, or both.

11. Here's how to do a wildcard search when using **more**. For example, if you say **more /usr/share/dict/words**, and then you want the first word that has a "b", a "k" and a "p" in that order in them, type the / command followed by a regular expression that does the job. If neither you nor your partner have had Organization of Programming Languages, you may ask someone else about regular expressions. Do regular expressions in **more** follow the grep convention of "." or the command-line globbing convention of "*"? [Historical note: The verb "to glob" probably comes from the noun glob, which means a small round-shaped thing. It was first used as a procedure name in the C part of Unix code to mean the function which replaces wild-card strings with their replacement list of files.]

Part II: What Are Unix Processes Doing?

We are learning about processes in Chapter 3 of your text. Answer the following questions on no more than one typewritten page to show me that you've thought about Unix processes.

- 12.** What shell are you running, from the list under Goals above? You might want to pipe the **set** command through the **more** command as one way to see, if typing **set** scrolls too much off the screen. Since the shell is a process itself, you could find out by listing processes too.
- 13.** How are processes uniquely identified to Unix? Give an example from your session on the machine.
- 14.** What are all of the processes that are running on **your** behalf when you have first successfully logged on, and not any of those that were running before you logged in? What command shows just those?
- 15.** What are the names of **all** the user processes on your system? What command did you type to show those?
- 16.** What are three daemons running on your system? What do you think that they do? (Ask *man* perhaps.)
- 17.** What process is using the most memory? How did you find that out?
- 18.** What processes in the process tree are using input and output from a keyboard and screen? (Hint: "tty" is the old abbreviation for teletypewriter, as you saw above when you used `/dev/tty` for keyboard input.)
- 19.** The **RES** column in the **top** command is the "working set size"—the size of the "pages" of a process currently in memory in kB. The **CODE & DATA** columns gives the total process size, including any parts of it swapped out to the disk, in kB. Unless several of you are on the machine at the same time, all processes seem to be entirely in memory, and none of them have pages swapped out to disk. How do you know? (If SuSe Linux doesn't show the **RSS** and **SIZE** columns that I mention, then **top**'s help screen shows you how to add those columns to the display.)
- 20.** From **top**, how long has the machine been running since it was last booted? How do you know?
- 21.** In the directory **/proc** can be found one entry for each process on the system. Guess (no penalty for wrong guesses) why Unix does that?
- 22.** Write a command that begins with the word **kill** that accomplishes the same function as the **exit** command. Will it be the same every time? (See `man kill`.)