Unix Tutorial

Chapter 1 Unix Command Summary

See the Unix tutorial for a leisurely, self-paced introduction on how to use the commands listed below. For more documentation on a command, consult a good book, or use the man pages. For example, for more information on `grep`, use the command `man grep`.

Contents

---

cat

This is one of the most flexible Unix commands. We can use to create, view and concatenate files. For our first example we create a three-item English-Spanish dictionary in a file called "dict."

```
   % cat >dict
     red rojo
     green verde
     blue azul
<control-D>
   %
```

<control-D> stands for "hold the control key down, then tap 'd'". The symbol > tells the computer that what is typed is to be put into the file dict. To view a file we use cat in a different way:

```
   % cat dict
     red rojo
     green verde
     blue azul
   %
```

If we wish to add text to an existing file we do this:

```
   % cat >>dict
     white blanco
     black negro
     <control-D>
   %
```

Now suppose that we have another file tmp that looks like this:

```
   % cat tmp
     cat gato
     dog perro
   %
```

Then we can join dict and tmp like this:

```
   % cat dict tmp >dict2
```

We could check the number of lines in the new file like this:

```
   % wc -l dict2
8
```

The command wc counts things --- the number of characters, words, and line in a file.

---

chmod

This command is used to change the permissions of a file or directory. For example to make a file essay.001 readable by everyone, we do this:

```
   % chmod a+r essay.001
```

To make a file, e.g., a shell script mycommand executable, we do this

```
% chmod +x mycommand
```
Now we can run `mycommand` as a command.

To check the permissions of a file, use **`ls -l`** . For more information on `chmod`, use **`man chmod`**.

---

cd

Use **cd** to change directory. Use pwd to see what directory you are in.

```
   % cd english
   % pwd
   % /u/ma/jeremy/english
   % ls
novel poems
   % cd novel
   % pwd
   % /u/ma/jeremy/english/novel
   % ls
ch1 ch2 ch3 journal scrapbook
   % cd ..
   % pwd
   % /u/ma/jeremy/english
   % cd poems
   % cd
   % /u/ma/jeremy
```

Jeremy began in his home directory, then went to his english subdirectory. He listed this directory using ls , found that it contained two entries, both of which happen to be diretories. He cd'd to the diretory `novel`, and found that he had gotten only as far as chapter 3 in his writing. Then he used **cd ..** to jump back one level. If had wanted to jump back one level, then go to `poems` he could have said **cd ../poems**. Finally he used **cd** with no argument to jump back to his home directory.

---

cp

Use **cp** to copy files or directories.
```
   % cp foo foo.2
```
This makes a copy of the file foo.
```
   % cp ~/poems/jabber .
```

This copies the file jabber in the directory poems to the current directory. The symbol "." stands for the current directory. The symbol "~" stands for the home directory.

---

date

Use this command to check the date and time.

```
   % date
Fri Jan  6 08:52:42 MST 1995
```

---

echo

The `echo` command echoes its arguments. Here are some examples:

```
   % echo this
     this
   % echo $EDITOR
     /usr/local/bin/emacs
   % echo $PRINTER
     b129lab1
```

Things like `PRINTER` are so-called *environment variables*. This one stores the name of the default printer --- the one that print jobs will go to unless you take some action to change things. The dollar sign before an environment variable is needed to get the value in the variable. Try the following to verify this:

```
   % echo PRINTER
     PRINTER
```

---

ftp

Use `ftp` to connect to a remote machine, then upload or download files. See also: ncftp

Example 1: We'll connect to the machine `fubar.net`, then change director to `mystuff`, then download the file `homework11`:

```
   % ftp solitude
     Connected to fubar.net.
     220 fubar.net FTP server (Version wu-2.4(11) Mon Apr 18 17:26:33
MDT 1994) ready.
   Name (solitude:carlson): jeremy
     331 Password required for jeremy.
   Password:
     230 User jeremy logged in.
   ftp> cd mystuff
     250 CWD command successful.
   ftp> get homework11
   ftp> quit
```

Example 2: We'll connect to the machine `fubar.net`, then change director to `mystuff`, then upload the file `collected-letters`:

```
   % ftp solitude
     Connected to fubar.net.
     220 fubar.net FTP server (Version wu-2.4(11) Mon Apr 18 17:26:33
MDT 1994) ready.
   Name (solitude:carlson): jeremy
     331 Password required for jeremy.
   Password:
     230 User jeremy logged in.
   ftp> cd mystuff
     250 CWD command successful.
   ftp> put collected-letters
   ftp> quit
```

The ftp program sends files in ascii (text) format unless you specify binary mode:

```
   ftp> binary
   ftp> put foo
   ftp> ascii
   ftp> get bar
```

The file `foo` was transferred in binary mode, the file `bar` was transferred in ascii mode.

---

grep

Use this command to search for information in a file or files. For example, suppose that we have a file `dict` whose contents are

```
   red rojo
   green verde
   blue azul
   white blanco
   black negro
```

Then we can look up items in our file like this;

```
   % grep red dict
     red rojo
   % grep blanco dict
     white blanco
   % grep brown dict
   %
```

Notice that no output was returned by `grep brown`. This is because "brown" is not in our dictionary file.

Grep can also be combined with other commands. For example, if one had a file of phone numbers named "ph", one entry per line, then the following command would give an alphabetical list of all persons whose name contains the string "Fred".

```
   % grep Fred ph | sort
     Alpha, Fred: 333-6565
     Beta, Freddie: 656-0099
     Frederickson, Molly: 444-0981
     Gamma, Fred-George: 111-7676
```

```
Zeta, Frederick: 431-0987
```
The symbol "|" is called "pipe." It pipes the output of the grep command into the input of the sort command.

For more information on `grep`, consult

```
% man grep
```

---

head

Use this command to look at the head of a file. For example,

```
% head essay.001
```

displays the first 10 lines of the file `essay.001` To see a specific number of lines, do this:

```
% head -20 essay.001
```
This displays the first 20 lines of the file.

---

ls

Use **ls** to see what files you have. Your files are kept in something called a directory.

```
% ls
  foo       letter2
  foobar    letter3
  letter1   maple-assignment1
%
```

Note that you have six files. There are some useful variants of the **ls** command:

```
% ls l*
  letter1 letter2 letter3
%
```

Note what happened: all the files whose name begins with "l" are listed. The asterisk (*) is the " wildcard" character. It matches any string.

---

lpr

This is the standard Unix command for printing a file. It stands for the ancient "line printer." See

```
% man lpr
```

for information on how it works. See print for information on our local intelligent print command.

---

mkdir

Use this command to create a directory.

```
% mkdir essays
```

To get "into" this directory, do

```
% cd essays
```

To see what files are in `essays`, do this:

```
% ls
```

There shouldn't be any files there yet, since you just made it. To create files, see cat or emacs.

---

more

More is a command used to read text files. For example, we could do this:

```
% more poems
```

The effect of this to let you read the file "poems ". It probably will not fit in one screen, so you need to know how to "turn pages". Here are the basic commands:

- **q** --- quit more
- **spacebar** --- read next page
- **return key** --- read next line
- **b** --- go back one page

For still more information, use the command **man more**.

---

mv

Use this command to change the name of file and directories.

```
% mv foo foobar
```

The file that was named `foo` is now named `foobar`

---

ncftp

Use `ncftp` for anonymous ftp --- that means you don't have to have a password.

```
% ncftp ftp.fubar.net
  Connected to ftp.fubar.net
> get jokes.txt
```

The file `jokes.txt` is downloaded from the machine `ftp.fubar.net`.

---

print

This is a moderately intelligent print command.

```
% print foo
% print notes.ps
% print manuscript.dvi
```

In each case `print` does the right thing, regardless of whether the file is a text file (like `foo`), a postcript file (like `notes.ps`, or a dvi file (like `manuscript.dvi`. In these examples the file is printed on the default printer. To see what this is, do

```
% print
```

and read the message displayed. To print on a specific printer, do this:

```
% print foo jwb321
% print notes.ps jwb321
% print manuscript.dvi jwb321
```

To change the default printer, do this:

```
% setenv PRINTER jwb321
```

---

pwd

Use this command to find out what directory you are working in.

```
% pwd
/u/ma/jeremy
% cd homework
% pwd
/u/ma/jeremy/homework
% ls
assign-1 assign-2 assign-3
% cd
% pwd
/u/ma/jeremy
%
```

Jeremy began by working in his "home" directory. Then he **cd** 'd into his homework subdirectory. Cd means " change directory". He used pwd to check to make sure he was

in the right place, then used **ls** to see if all his homework files were there. (They were). Then he **cd'd** back to his home directory.

---

rm

Use **rm** to remove files from your directory.

```
% rm foo
  remove foo? y
% rm letter*
  remove letter1? y
  remove letter2? y
  remove letter3? n
%
```

The first command removed a single file. The second command was intended to remove all files beginning with the string "letter." However, our user (Jeremy?) decided not to remove letter3.

---

rmdir

Use this command to remove a directory. For example, to remove a directory called "essays", do this:

```
% rmdir essays
```

A directory must be empty before it can be removed. To empty a directory, use rm.

---

rsh

Use this command if you want to work on a computer different from the one you are currently working on. One reason to do this is that the remote machine might be faster. For example, the command

```
% rsh solitude
```

connects you to the machine `solitude`. This is one of our public workstations and is fairly fast.

See also: telnet

---

setenv

```
% echo $PRINTER
  labprinter
% setenv PRINTER myprinter
% echo $PRINTER
  myprinter
```

---

sort

Use this commmand to sort a file. For example, suppose we have a file `dict` with contents

```
red rojo
green verde
blue azul
white blanco
black negro
```

Then we can do this:

```
% sort dict
  black negro
  blue azul
  green verde
  red rojo
  white blanco
```

Here the output of `sort` went to the screen. To store the output in file we do this:

```
% sort dict >dict.sorted
```

You can check the contents of the file `dict.sorted` using cat , more , or emacs .

---

tail

Use this command to look at the tail of a file. For example,

```
% head essay.001
```

displays the last 10 lines of the file `essay.001` To see a specific number of lines, do this:

```
% head -20 essay.001
```

This displays the last 20 lines of the file.

---

tar

Use create compressed archives of directories and files, and also to extract directories and files from an archive. Example:

```
% tar -tvzf foo.tar.gz
```

displays the file names in the compressed archive `foo.tar.gz` while

```
% tar -xvzf foo.tar.gz
```

extracts the files.

---

telnet

Use this command to log in to another machine from the machine you are currently working on. For example, to log in to the machine "solitude", do this:

```
% telnet solitude
```

See also: rsh.

---

wc

Use this command to count the number of characters, words, and lines in a file. Suppose, for example, that we have a file `dict` with contents

```
red rojo
green verde
blue azul
white blanco
black negro
```

Then we can do this

```
% wc dict
  5      10      56 tmp
```

This shows that `dict` has 5 lines, 10 words, and 56 characters.

The word count command has several options, as illustrated below:

```
% wc -l dict
  5 tmp
% wc -w dict
  10 tmp
% wc -c dict
  56 tmp
```

The goal of these lessons is to learn to "speak Unix", the language in which we shall command our servant.

Let us begin with a simple command: we want the computer to tell us today's date. Here is how to do it:

```
% date
Fri Mar 25 09:24:30 MST 1994
```

Wasn't that easy? Note that your commands --- what you tell the computer to do --- are displayed in **bold**. What the computer replies is displayed `like this`. Note also that you didn't type " `%` ": that is the *prompt* that the computer types. It does this when it is listening for your next command. (Prompts can be different from " `%` ")

Try the "date" command now.

Here are some more commands to try, with samples of how the computer might respond.

```
 % whoami
jeremy
```

whoami displays the login name of the current user, who (for the purpose of these lessons) is "jeremy. " *Try this now. The computer should respond with your login name.*

```
%  echo This is a test
This is a test.
```

echo does just that: it tells the computer to retype the string "This is a test". Here is another use of echo:

```
% echo $PRINTER
b129lab1
```

This time echo tells us what is stored in the PRINTER variable --- the name of the printer the computer will use if you print something. Capitalization is important in Unix, so be sure to say "PRINTER", not " printer" or "Printer". The dollar sign in front of the variable name is also important. Note what happens if we forget to use it:

```
% echo PRINTER
PRINTER
```

Try all the examples above. Also: know where your printer is!

This is as good a time as any to introduce a little computer jargon. The words of the string "This is a test" are *arguments* to the command `echo`. The results of a command depend on what the arguments are.

Chapter 2 Files

Your home directory

Computers store information in things called *files*. A file is like a sheet of paper (maybe a very long one) with something written on it. Files are stored in things called *directories*. We are now going to learn how make, use, and manage files and directories. We will begin by figuring out what our home directory is:

```
% cd
% pwd
/u/c/jeremy
% echo $HOME
/u/c/jeremy
```

The **cd** command (**c** hange **d** irectory) used with no arguments takes us from wherever we might be to our home directory. The **pwd** (**p** rint **w** orking **d** irectory) tells in which directory we find ourselves for the moment. In the case at hand it is `/u/c/jeremy`. Don't be concerned for the moment about the `/u/c/` part. It is a **path**, but that is irrelevant for now. Note that **echo $HOME** has exactly the same effect as **pwd**.

Exercise: Figure out what your home directory is.

Creating short files

Now let us create a short file. For this we use the **cat** command. Follow the example below carefully:

```
% cat >dict
red: rojo
yellow: amarillo
black: negro
white: blanco
blue: azul
green: verde
<control-d>
%
```

By **<control-d>** we mean: hold the control key down; while it is down press "d". We have just used `cat` to create a short English-Spanish dictionary. This dictionary resides in the file `dict` . We told `cat` to put what we typed in `dict` using the "into" symbol, namely `>` . To tell `cat` that we were done typing we typed **control-d** ("d" for "done"). To check that the dictionary is really there and that it was correctly entered we do this:

```
% ls
dict
% cat dict
red: rojo
yellow: amarillo
black: negro
white: blanco
blue: azul
green: verde
```

```
%
```

The **ls** command **l** ist **s** the files in the current directory. For the moment there is only one, namely `dict`. The command **cat** shows us what is in `dict`.

Printing files

Now that we know how to make small files and view them, let's learn how to print them. Here's how:

```
% print dict

Printing dict2 (text) on jwb129lab1

%
```

Print is *not* a standard Unix command. For this see lpr. The print command will try to figure out what kind of file you are trying to print and use the method it deems best. For more information on what it does, type the command with no arguments:

```
% print
... displays info on print ...
%
```

If you need a list of printers, use **print -l**. The "-l" is an *option* for the print command. Many Unix commands have options.

Note for those who need it: The print command understands dvi and postcript (ps) files. For example, **print foo.dvi** correctly prints foo.dvi, and **print bar.ps** correctly prints the postscript file bar.ps. If you need to force a file to be printed as text use **print -t**, e.g., **print -t weirdfile**.

Examining files

Unix has some useful commands for examining files, e.g., counting the number of words, seeing whether a particular word is in the file, sorting the file, etc. We will learn about a few of these commands now:

```
% wc dict
6       12      78
% grep white dict
white: blanco
% sort dict
black: negro
blue: azul
```

```
green: verde
red: rojo
white: blanco
yellow: amarillo
%
```

The **wc** command **c** ounts **w** ords (and more). In the case at hand it tells us that `dict` contains 6 lines, 12 words, and 78 characters ("letters "). The **grep** command looks for the word **white** in the file **dict** and displays the lines in which this word appears. It gives us a way to search through files. The **sort** command does just what it says.

Before going on, let's see how to save a copy of our sorted dictionary. We'll put in a file called `dict2`.

```
% sort dict >dict2
% ls
dict dict2
% cat dict2
black: negro
blue: azul
green: verde
red: rojo
white: blanco
yellow: amarillo
%
```

Notice once again the use of the "into" symbol ">". In our example it had the effect of directing the output of the sort command from the screen to the file `dict2`. Just to be sure that everything went according to plan, we used **ls** to be sure that `dict2` was there, and we used **cat dict2** to be sure that it contained what we thought it should.

Timeout: after working through the last example, stand up and stretch. Then congrutalate yourself for having made so much progress learning Unix.

Getting rid of files

As you continue to work you will create more and more files. Eventually you will want to get rid of some of them. For this we use the **rm** command (for **r** e **m** ove):

```
% ls
dict dict2
% rm dict2
rm: remove dict2? y
% ls
dict
%
```

Chapter 3 Creating and using directories

After working on your unix system for a while you will accumulate many files. Just like an unorganized desk, this creates a mess in which it is hard to work. The solution is to create *directories* in which to store related items. A directory is like a file folder which contains related documents (your files). As an example, suppose that when you list your files you see this:

```
% ls
fred1 fred2 fred3 ch1 ch2 ch3 foo.c bar.c
%
```

This is really not badly organized: the files fred1, etc. are letters to fred, the files ch1, etc. chapters of a book, and foo.c, bar.c are C programs. Nonetheless, we decide that it is time to get organized, with one directory per project.

The 'mdkir' command.

We create a new directory using the **mkdir** command (**m** ake **d** irectory).

```
% mkdir letters
% ls
fred1 fred2 fred3 ch1 ch2 ch3 foo.c bar.c letters
```

Notice that the directory `letters` shows up in the listing. If you are not sure what is a file and what is a directory, try this:

```
% ls -F
fred1 fred2 fred3 ch1 ch2 ch3 foo.c bar.c letters/
```

Notice that `letters` is displayed somewhat differently.

The 'mv' command

Now we move the letters into the directory letters using the **mv** command (**m** o **v** e).

```
% mv fred1 fred2 fred3 letters
% ls
ch1 ch2 ch3 foo.c bar.c letters
```

If we want to check that `letters` really contains the files it should, we do this:

```
% ls letters
fred1 fred2 fred3
```

There is, by the way, a useful shortcut:

```
% mv fred* letters
```

Here the character `*` matches any sequence of characters, including the null string. Thus files named `fred`, `fred101`, and `freddy` would all be moved into `letters`.

Paths

You can deal directly with files in a directory like this:

```
% cat letters/fred1
```

This command displays the contents of the file `fred1` , which is in the directory `letters`. Here are some other ways of doing the same thing:

```
% cat letters/fred1
% more letters/fred1
% emacs letters/fred1
```

We could even do this:

```
% cat l*f*1
```

Changing directories with 'cd'

Sometimes it is better to work inside the directory `letters`. To do it we use the `cd` command (**ch** hange **d** irectory).

```
% cd letters
% ls
fred1 fred2 fred3
```

The letters are there, as they chould be. To go back to our home directory we do this:

```
% cd
```

We check that our home directory contains what it should.

```
% ls
ch1 ch2 ch3 foo.c bar.c letters
```

Now we make directories for the other files and move them into the right places:

```
% mkdir book; mv ch* book
% mkdir cprogs; mv *.c cprogs
% ls -F
book/ cprogs/ letters/
% ls book
ch1 ch2 ch3
%
```

Where are we?

Sometimes in moving from one directory to another we lose track of where we are. To find out what the current directory is, use the **pwd** command (**p** rint **w** orking **d** irectory).

```
% pwd
jeremy
% cd book
```

```
% pwd
jeremy/book
%
```

Removing directories

To remove a directory we first remove all the file in it, then remove the directory with **rmdir** (**r** emove **d** irectory).

```
% pwd
jeremy
% cd letters
% pwd
jeremy/letters
% rm *
% cd ..
% rmdir letters
```

The command **rm *** removes all files in the current directory. The command **cd ..** changes the current directory to the parent of the current one. In this case, it changes us from `jeremy/letters` to `jeremy .` Remember that `jeremy/letters` is a *path* , as is `jeremy/letters/fred1`. The latter is the path which starts with Jeremy's home directory and ends with the file `fred1`.
