

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](#) --- for creating and displaying short files
- [chmod](#) --- change permissions
- [cd](#) --- change directory
- [cp](#) --- for copying files
- [date](#) --- display date
- [echo](#) --- echo argument
- [ftp](#) --- connect to a remote machine to download or upload files
- [grep](#) --- search file
- [head](#) --- display first part of file
- [ls](#) --- see what files you have
- [lpr](#) --- standard print command (see also [print](#))
- [more](#) --- use to read files
- [mkdir](#) --- create directory
- [mv](#) --- for moving and renaming files
- [ncftp](#) --- especially good for downloading files via anonymous [ftp](#).
- [print](#) --- custom print command (see also [lpr](#))
- [pwd](#) --- find out what directory you are in
- [rm](#) --- remove a file
- [rmdir](#) --- remove directory
- [rsh](#) --- remote shell
- [setenv](#) --- set an environment variable
- [sort](#) --- sort file
- [tail](#) --- display last part of file
- [tar](#) --- create an archive, add or extract files
- [telnet](#) --- log in to another machine
- [wc](#) --- count characters, words, lines

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 directories. He cd'd to the directory 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 he 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
b1291ab1
```

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 -n 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 postscript 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 command 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,

```
% tail essay.001
```

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

```
% tail -n 20 essay.001
```

This displays the last 20 lines of the file.

tar

Use to 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
```

dummy

Under construction
