Help on any Unix command.

`man {command}`	Type **man ls** to read the manual for the **ls** command.
`man {command} > {filename}`	Redirect help to a file to download.
`whatis {command}`	Give short description of command. (Not on RAIN?)
`apropos {keyword}`	Search for all Unix commands that match keyword, eg **apropos file**. (Not on RAIN?)

List a directory

`ls {path}`	It's ok to combine attributes, eg **ls -laF** gets a long listing of all files with types.
`ls {path_1} {path_2}`	List both {path_1} and {path_2}.
`ls -l {path}`	Long listing, with date, size and permisions.
`ls -a {path}`	Show all files, including important .dot files that don't otherwise show.
`ls -F {path}`	Show type of each file. "/" = directory, "*****" = executable.
`ls -R {path}`	Recursive listing, with all subdirs.
`ls {path} > {filename}`	Redirect directory to a file.
`ls {path} | more`	Show listing one screen at a time.
`dir {path}`	Useful alias for DOS people, or use with **ncftp**.

Change to directory

`cd {dirname}`	There must be a space between.
`cd ~`	Go back to home directory, useful if you're lost.
`cd ..`	Go back one directory.
`cdup`	Useful alias, like "cd ..", or use with **ncftp**.

Make a new directory

`mkdir {dirname}`

Remove a directory

`rmdir {dirname}`	Only works if {dirname} is empty.
`rm -r {dirname}`	Remove all files and subdirs. Careful!

Print working directory

`pwd`	Show where you are as full path. Useful if you're lost or exploring.

Copy a file or directory

`cp {file1} {file2}`	
`cp -r {dir1} {dir2}`	Recursive, copy directory and all subdirs.
`cat {newfile} >> {oldfile}`	Append newfile to end of oldfile.

Move (or rename) a file

`mv {oldfile} {newfile}`	Moving a file and renaming it are the same thing.
`mv {oldname} {newname}`	

Delete a file

`rm {filespec}`	**?** and ***** wildcards work like DOS should. "?" is any character; "*" is any string of characters.
`ls {filespec}` `rm {filespec}`	Good strategy: first list a group to make sure it's what's you think... ...then delete it all at once.

View a text file

`more {filename}`	View file one screen at a time.
`less {filename}`	Like **more**, with extra features.
`cat {filename}`	View file, but it scrolls.
`cat {filename} | more`	View file one screen at a time.
`page {filename}`	Very handy with **ncftp**.
`pico {filename}`	Use text editor and don't save.

Edit a text file.

`pico {filename}`	The same editor PINE uses, so you already know it. **vi** and **emacs** are also available.

Create a text file.

`cat > {filename}`	Enter your text (multiple lines with **enter** are ok) and press **control-d** to save.
`pico {filename}`	Create some text and save it.

Compare two files

`diff {file1} {file2}`	Show the differences.
`sdiff {file1} {file2}`	Show files side by side.

Other text commands

`grep '{pattern}' {file}`	Find regular expression in file.
`sort {file1} > {file2}`	Sort file1 and save as file2.
`sort -o {file} {file}`	Replace file with sorted version.
`spell {file}`	Display misspelled words.
`wc {file}`	Count words in file.

Find files on system

`find {filespec}`	Works with wildcards. Handy for snooping.
`find {filespec} > {filename}`	Redirect find list to file. Can be big!

Make an Alias

`alias {name} '{command}'`	Put the command in 'single quotes'. More useful in your **.cshrc** file.

Wildcards and Shortcuts

`*`	Match any string of characters, eg **page*** gets page1, page10, and page.txt.
`?`	Match any single character, eg **page?** gets page1 and page2, but not page10.
`[...]`	Match any characters in a range, eg **page[1-3]** gets page1, page2, and page3.

~	Short for your home directory, eg **cd ~** will take you home, and **rm -r ~** will destroy it.
.	The current directory.
. .	One directory up the tree, eg **ls ..**.

Pipes and Redirection

(You **pipe** a command to another command, and **redirect** it to a file.)

`{command} > {file}`	Redirect output to a file, eg **ls > list.txt** writes directory to file.
`{command} >> {file}`	Append output to an existing file, eg **cat update >> archive** adds update to end of archive.
`{command} < {file}`	Get input from a file, eg **sort < file.txt**
`{command} < {file1} > {file2}`	Get input from `file1`, and write to `file2`, eg **sort < old.txt > new.txt** sorts old.txt and saves as new.txt.
`{command} | {command}`	Pipe one command to another, eg **ls | more** gets directory and sends it to **more** to show it one page at a time.

Permissions, important and tricky!

Unix permissions concern who can **read** a file or directory, **write** to it, and **execute** it. Permissions are granted or withheld with a magic 3-digit number. The three digits correspond to the **owner** (you); the **group** (?); and the **world** (everyone else).

Think of each digit as a sum:

execute permission	= 1
write permission	= 2
write and execute (1+2)	= 3
read permission	= 4
read and execute (4+1)	= 5
read and write (4+2)	= 6
read, write and execute (4+2+1)	= 7

Add the number value of the permissions you want to grant each group to make a three digit number, one digit each for the owner, the group, and the world. Here are some useful combinations. Try to figure them out!

`chmod 600 {filespec}`	You can read and write; the world can't. Good for files.
`chmod 700 {filespec}`	You can read, write, and execute; the world can't. Good for scripts.
`chmod 644 {filespec}`	You can read and write; the world can only read. Good for web pages.
`chmod 755 {filespec}`	You can read, write, and execute; the world can read and execute. Good for programs you want to share, and your `public_html` directory.

Permissions, another way

You can also change file permissions with letters:

u = user (yourself) **g** = group **a** = everyone

r = read **w** = write **x** = execute

`chmod u+rw {filespec}`	Give yourself read and write permission
`chmod u+x {filespec}`	Give yourself execute permission.
`chmod a+rw {filespec}`	Give read and write permission to everyone.

System info

`date`	Show date and time.
`df`	Check system disk capacity.
`du`	Check your disk usage and show bytes in each directory.
`more /etc/motd`	Read message of the day, "motd" is a useful alias..
`printenv`	Show all environmental variables (in C-shell% - use **set** in Korn shell$).
`quota -v`	Check your total disk use.
`uptime`	Find out system load.
`w`	Who's online and what are they doing?

Unix Directory Format

Long listings (**ls -l**) have this format:

```
- file
```

```
d directory,                                                  * executable
^   symbolic links (?)  file size (bytes)   file name   / directory
^           ^                 ^                 ^         ^
drwxr-xr-x 11 mkummel      2560 Mar  7 23:25 public_html/
-rw-r--r--  1 mkummel     10297 Mar  8 23:42 index.html
                                        ^
 ^^^        user permission  (rwx)      date and time last modified
    ^^^     group permission (rwx)
       ^^^  world permission (rwx)
```

How to Make an Alias

An alias lets you type something simple and do something complex. It's a shorthand for a command. If you want to type "dir" instead of "ls -l" then type **alias dir 'ls -l'**. The single quotes tell Unix that the enclosed text is one command.

Aliases are more useful if they're permanent so you don't have to think about them. You can do this by adding the alias to your **.cshrc** file so they're automatically loaded when you start. Type **pico .cshrc** and look for the alias section and add what you want. It will be effective when you start. Just remember that if you make an alias with the name of a Unix command, that command will become unavailable.

Here are a few aliases from my **.cshrc** file:

```
# enter your aliases here in the form:
# alias     this    means this

alias       h       history
alias       m       more
alias      q       quota -v
alias       bye     exit
alias      ls      ls -F
alias       dir     ls
alias      cdup    cd ..
alias      motd    more /etc/motd
```

How to Make a Script

A Unix script is a text file of commands that can be executed, like a **.bat** file in DOS. Unix contains a powerful programming language with loops and variables that I don't really understand. Here's a useful example.

Unix can't rename a bunch of files at once the way DOS can. This is a problem if you develop Web pages on a DOS machine and then upload them to your Unix Server. You might have a bunch of **.htm** files that you want to rename as **.html** files, but Unix makes you do it one by one. This is actually not a defect. (It's a feature!) Unix is just being more consistent than DOS. So make a script!

Make a text file (eg with **pico**) with the following lines. The first line is special. It tells Unix what program or shell should execute the script. Other # lines are comments.

```
#! /bin/csh
# htm2html converts *.htm files to *.html
foreach f ( *.htm )
  set base=`basename $f .htm`
  mv $f $base.html
end
```

Save this in your home directory as **htm2html** (or whatever). Then make it user-executable by typing **chmod 700 htm2html**. After this a ***** will appear by the file name when you **ls -F**, to show that it's executable. Change to a directory with **.htm** files and type **~/htm2html**, and it will do its stuff.

Think about scripts whenever you find yourself doing the same tedious thing over and over.

Dotfiles (aka Hidden Files)

Dotfile names begin with a "." These files and directories don't show up when you list a directory unless you use the **-a** option, so they are also called **hidden files**. Type **ls -la** in your home directory to see what you have.

Some of these dotfiles are crucial. They initialize your shell and the programs you use, like **autoexec.bat** in DOS and **.ini** files in Windows. **rc** means "run commands". These are all text files that can be edited, but change them at your peril. Make backups first!

Here's some of what I get when I type **ls -laF**:

.cshrc	my C-shell startup info, important!
.history	list of past commands.
.login	login init, important!
.plan	text appears when I'm **fingered**, ok to edit.
.profile	Korn shell startup info, important!
.project	text appears when I'm **fingered**, ok to edit.

DOS and UNIX commands

Action	DOS	UNIX
change directory	cd	cd
change file protection	attrib	chmod
compare files	comp	diff
copy file	copy	cp
delete file	del	rm
delete directory	rd	rmdir
directory list	dir	ls
edit a file	edit	pico
environment	set	printenv

find string in file	find	grep
help	help	man
make directory	md	mkdir
move file	move	mv
rename file	ren	mv
show date and time	date, time	date
show disk space	chkdsk	df
show file	type	cat
show file by screens	type filename | more	more
sort data	sort	sort