Basic Shell Commands

---

1. Shell Basics:

Command	Definition
`.`	a single period refers to the current directory
`..`	a double period refers to the directory immediately above the current directory
`~`	refers to your home directory. *Note:* this command does NOT work on Windows machines (Mac and Linux are okay)
`cd ./dirname`	changes the current directory to the directory `dirname`
`ls -F`	tells you what files and directories are in the current directory
`pwd`	tells you what directory you are in (`pwd` stands for *p*rint *w*orking *d*irectory)
`history`	lists previous commands you have entered. `history | less` lets you page through the list.
`man` *cmd*	displays the *man*ual page for a command.

2. Creating Things:

a) How to create new files and directories..

Command	Definition
`mkdir ./dirname`	makes a new directory called dirname below the current directory. *Note:* Windows users will need to use `\` instead of `/` for the path separator
`nano filename`	if `filename` does not exist, `nano` creates it and opens the `nano` text editor. If the file exists, `nano` opens it. *Note: (i)* You can use a different text editor if you like. In gnome Linux, `gedit` works really well too. *(ii)* `nano` (or `gedit`) create text files. It doesn't matter what the file extension is (or if there is one)

b) How to delete files and directories...

Remember that deleting is forever. There is NO going back

Command	Definition
`rm ./filename`	deletes a file called `filename` from the current directory
`rmdir ./dirname`	deletes the directory `dirname` from the current directory. *Note:* `dirname` must be empty for `rmdir` to run.

c) How to copy and rename files and directories...

Command	Definition
`mv tmp/filename .`	moves the file `filename` from the directory `tmp` to the current directory. *Note: (i)* the original `filename` in `tmp` is deleted. *(ii)* `mv` can also be used to rename files (e.g., `mv filename newname`
`cp tmp/filename .`	copies the file `filename` from the directory `tmp` to the current directory. *Note: (i)* the original file is still there

3. Pipes and Filters

a) How to use wildcards to match filenames...

Wildcards are a shell feature that makes the command line much more powerful than any GUI file managers. Wildcards are particularly useful when you are looking for directories, files, or file content that can vary along a given dimension. These wildcards can be used with any command that accepts file names or text strings as arguments.

Table of commonly used wildcards

Wildcard	Matches
`*`	zero or more characters
`?`	exactly one character
`[abcde]`	exactly one of the characters listed
`[a-e]`	exactly one character in the given range
`[!abcde]`	any character not listed
`[!a-e]`	any character that is not in the given range
`{software,carpentry}`	exactly one entire word from the options given

See the cheatsheet on regular expressions for more "wildcard" shortcuts.

b) How to redirect to a file and get input from a file ...

Redirection operators can be used to redirect the ouput from a program from the display screen to a file where it is saved (or many other places too, like your printer or to another program where it can be used as input).

Command	Description
`>`	write `stdout` to a new file; overwrites any file with that name (e.g., `ls *.md > mardkownfiles.txt`)
`>>`	append `stdout` to a previously existing file; if the file does not exist, it is created

	(e.g., `ls *.md >> markdownfiles.txt`)
`<`	assigns the information in a file to a variable, loop, etc (e.g., `n < markdownfiles.md`)

b.1) How to use the output of one command as the input to another with a pipe...

A special kind of redirection is called a pipe and is denoted by `|`.

Command	Description
`|`	Output from one command line program can be used as input to another one (e.g. ls *.md | head gives you the first 5 `*.md` files in your directory)

Example:

```
ls *.md | head | sed -i `s/markdown/software/g`
```

changes all the instances of the word `markdown` to `software` in the first 5 `*.md` files in your current directory.

4. How to repeat operations using a loop...

Loops assign a value in a list or counter to a variable that takes on a different value each time through the loop. There are 2 primary kinds of loops: `for` loops and `while` loops.

a) For loop

For loops loop through variables in a list

```
for varname in list
do
    command 1
    command 2
done
```

where,

- `for`, `in`, `do`, and `done` are keywords
- `list` contains a list of values separated by spaces. e.g. `list` can be replaced by `1 2 3 4 5 6` or by `Bob Mary Sue Greg`. `list` can also be a variable:

--

```
list[0]=Sam
list[1]=Lynne
list[2]=Dhavide
list[3]=Trevor
.
.
.
list[n]=Mark
```

which is referenced in the loop by:

```
for varname in ${list[@]}
do
    command 1
    command 2
done
```

Note: Bash is zero indexed, so counting always starts at `0`, not `1`.

b) While Loop

While loops loop through the commands until a condition is met. For example

```
COUNTER=0
while [ ${COUNTER} -lt 10 ]; do
    command 1
    command 2
    COUNTER=`expr ${COUNTER} + 1`
done
```

continues the loop as long as the value in the variable COUNTER is less than 10 (incremented by 1 on each iteration of the loop).

- `while`, `do`, and `done` are keywords

b.1) Commonly used conditional operators

Operator	Definition
`-eq`	is equal to
`-ne`	is not equal to

`-gt`	greater than
`-ge`	greater than or equal to
`-lt`	less than
`-le`	less than or equal to

Use `man bash` or `man test` to learn about other operators you can use.

6. Finding Things

a) How to select lines matching patterns in text files...

To find information within files, you use a command called `grep`.

Example command	Description
`grep [options] day haiku.txt`	finds every instance of the string `day` in the file haiku.txt and pipes it to standard output |

a.1) Commonly used `grep` options

`grep` options

`-E`	tells grep you will be using a regular expression. Enclose the regular expression in quotes. *Note:* the power of `grep` comes from using regular expressions. Please see the regular expressions sheet for examples
`-i`	makes matching case-insensitive
`-n`	limits the number of lines that match to the first n matches
`-v`	shows lines that do not match the pattern (inverts the match)
`-w`	outputs instances where the pattern is a whole word

b) How to find files with certain properties...

To find file and directory names, you use a command called `find`

Example command	Description
`find . -type d`	`find` recursively descends the directory tree for each path listed to match the expression given in the command line with file or directory names in the search path

b.1) Commonly used `find` options

`find` options

`-type [df]`	`d` lists directories; `f` lists files
`-maxdepth n`	`find` automatically searches subdirectories. If you don't want that, specify the number of levels below the working directory you would like to search
`-mindepth n`	starts `find`'s search `n` levels below the working directory