

An Introduction to the Linux Command Line



Last updated: September 8, 2021

About this training module

This training module assumes that you have a HiPerGator account. If you do not have a HiPerGator account, you can [follow along using this version of the training](#) which uses a free, web-based Bash shell.

Click the Zoom logo to view a recording of a training session using this material:



Getting Around in Linux

- File paths (directories or folders): `/`, `/home/username/`, `/blue/group/username`

In this documentation, we will use the convention of putting text that needs to be substituted for your specific case in *italic font*. So, you would use **your** username (your GatorLink for UF users), where the directions show *username*.


- `pwd`, `cd`, `ls` (Where am I, change directory, list directory)
- `cp`, `mv`, `rm` (copy, move (also used for rename), delete)
- `more`, `less`, `head`, `tail`, `cat` (examine files)
- `nano`, `vim` (text editors in Linux)

Note that most things in Linux are case-sensitive. This applies to commands and file names. e.g. `CD` `test` will not work to change directories to the "test" folder, while `cd test` will work (assuming there is a folder named "test" in the current folder). Similarly, `Test` and `test` are **different** folders or files.

Making Things Easier

- **Tab completion**- type part of a path or file name and hit tab-key, the shell will auto-complete for you.

Note that this may not work for the group directories in `/blue/` since directories are automounted, meaning that they may not show up, or be able to be tab-completed, until you access the directory.

- **history**: redo something that you did before without retyping (you can also use the  arrow key)
- **man**: getting help. Many built-in Linux applications have **manual** pages that document their use and options that they have. e.g. `man ls` will bring up the page for the `ls` command, documenting the many options that change how the `ls` command functions.
 - Another way to get help information about applications is using the name of the application followed by either the `-h` or `--help` flags. Many applications use this convention to provide

documentation to users.

Learning by Doing

(Note: some of the data and examples are taken from software-carpentry.org):

1. Connect to HiPerGator:

1. *Mac*: `ssh username@hpg.rc.ufl.edu` (Remember *username* should be substituted with **your** username)

For additional help, watch the video tutorial on logging in with from a Mac



2. *Windows*: hostname: `hpg.rc.ufl.edu`

For additional help, watch the video tutorial on logging in with from Windows



3. Note that on both Mac and Windows, when you are typing your password, no characters display while you type. Just keep typing, and hit Enter and you should be logged in.

2. Where are you when you login? `pwd`

3. What files are there? `ls`

4. At Research Computing, we ask that users keep most of their data in the their group's `/blue/group/` folder. Let's change directories there: `cd /blue/group/username`

1. You can see your primary group by typing the `id` command:

```
[userA@login2 ~]$ id
uid=10856(userA) gid=1234(mygroup)
groups=1234(mygroup),1235(othergroup)
```

In the output of the `id` command, you can see your **primary group** after the `gid=`, in this example, `mygroup`. Other groups, referred to as **secondary groups** are listed after that in the `groups=` section, showing that this user is also in the `othergroup`.

5. Let's make a directory to put some data in: `mkdir cli_demo`

6. Now what's there? `ls -l`

1. Linux commands usually have flags to change how they work
2. `man`, `-h` or `--help` often give you help

7. Change into `cli_demo` directory: `cd cli_demo` or `cd cl-Tab-key`

8. Copy some demo data here (`.`):

```
cp -r /data/training/LinuxCLI/molecules .
```

1. Note the `-r` to recursively copy, since `cp` won't copy directories by default
2. Also note the `."` at the end of the command to copy the molecules directory to your current location--in Linux, `."` is short for the current location.

The `molecules` folder of files is also available in this repository at `data/molecules`. This folder originated from the [software-carpentry Shell Training materials](#).

9. Check that the copy worked: `ls`
10. Change directories into the molecules directory: `cd molecules`
11. Look at these files with `more`, `cat`, `head`, `tail`:

1. `more propane.pdb` and `cat propane.pdb`
2. `head propane.pdb` or `head -n2 propane.pdb`
3. `tail propane.pdb` or `tail -n2 propane.pdb`

12. **Redirects:** You can redirect the output of a command to a file with the `>` character (see below for more information about STDIN, STDOUT and STDERR).

Caution: Redirection with `>` first erases the file you are redirecting to if it exists, replacing it with the new contents. **You can append to a file with `>>`.**

1. As an example, we can get the lengths, in number of lines, of all the files ending in `.pdb` using the wildcard `*` and redirect the output to a file: `wc -l *.pdb > lengths.txt`
2. Let's see what this file looks like: `cat lengths.txt`

Wildcard expansion (e.g. `*.pdb`) is a convenient way to operate on a list of files matching some characteristic: e.g. having the `.pdb` ending, starting with `p` (`p*`), etc. The wildcard is evaluated and the list of matching files is passed into the command.

13. **Sorting:** We might want the lengths sorted: `sort -n lengths.txt`

1. What happens without the `-n`?

14. **Pipes:** We can connect commands together by piping the output from one command into the input for the next command using the vertical bar character `|`:

The pipe character, `|`, is typically located above the Enter-key on US keyboards with the `\` and is accessed with the Shift-key

1. `wc -l *.pdb | sort -n > lengths.txt`
2. Or if we only want to know the shortest file: `wc -l *.pdb | sort -n | head -n1`

Why this works: Most Linux programs can take input from what is called "standard in", often abbreviated as "STDIN". In addition, they typically have two output streams, "standard out", "STDOUT", and "standard error", "STDERR"--both of which print to the screen by default, but can be redirected as we saw above (in fact we only redirect STDOUT with the `>`; any STDERR would still print to the screen).

So, while you can run a command like `wc -l propane.pdb`, you can also get the same result by running the `cat` command and "piping" the STDOUT of that to the STDIN of `wc`: `cat propane.pdb | wc -l`. Similarly, you can keep piping commands connecting the output of one command to the input for the next command.

15. **grep**: We can search for text using `grep`:

1. `grep ATOM propane.pdb`

16. **awk**: `awk` can do a lot, but one thing it's good it is pulling out columns from files, in this case the 3rd column:

1. `grep ATOM propane.pdb | awk '{print $3}'`

17. **uniq**: `uniq` is a command to find unique entries in a **sorted** list: `grep ATOM propane.pdb | awk '{print $3}' | sort | uniq`

18. **Loops**: One of the great things about the command line is the ability to automate repetitive tasks. Let's say we want to verify that all our molecules are hydrocarbons (made up of only C and H atoms). Below is what the loop looks like neatly spaced out as if it were in a script:

```
for molecule in *.pdb
do
    echo $molecule
    grep ATOM $molecule | awk '{print $3}' | sort | uniq
done
```

As we type this in on the command line, once you hit the Enter-key at the end of the first line, Bash gives you the continuation prompt, ">", essentially telling you that you have started a command (a for loop), but not finished it:

```
[user@login1 molecules]$ for molecule in *.pdb
>
```

You can keep typing the rest of the lines, so that it looks like this:

```
[user@login1 molecules]$ for molecule in *.pdb
> do
> echo $molecule
> grep ATOM $molecule | awk '{print $3}' | sort | uniq
> done
```

Once you hit the Enter-key after the "`done`", that completes the for loop and Bash executes it, showing the output.

If you find yourself stuck in the continuation prompt mode, you can use the key combination **Ctrl-C** to cancel and exit back to the main command prompt. **Ctrl-C** will usually cancel execution of a program in Bash and is a handy key-combination to know!

19. **Deleting files:** Let's get rid of the lengths.txt file: **rm lengths.txt**

1. That file is now gone!! There is no undo, no recycle bin or trash can. As soon as you type the command and hit return, the file is gone!
2. Be careful... but don't keep everything forever either!

Additional exercises

- Which molecule has the most H atoms?
- Make a directory in your **cli_demo** folder and copy the **methane.pdb** file there (preferably without moving from the **molecules** directory)
- From the **molecules** directory, get the **head** of the **methane.pdb** file in the directory you created above.
- Change directories to the directory you made above, rename the **methane.pdb** file to **my_methane.pdb**.
- Edit the **my_methane.pdb** file and put your name as the AUTHOR. I typically recommend the **nano** text editor on the command line for new users: **nano my_methane.pdb** and use the arrow keys to move around. The commands at the bottom of the screen use the Ctrl-key in combination with another key, so **^X** means hold down the Ctrl-key and the X-key at the same time to exit.
- How many ATOMS does methane have?
- Use an SFTP program to download the molecules folder to your computer