CS107 Unix Guide

Unix Guide

Written by Chris Gregg and Nick Troccoli

There are walkthrough videos for many of the commands listed below. You can view all the walkthrough videos here.

The Filesystem

When logged in to myth, or any other machine, we can navigate the filesystem, - go in and out of folders ("directories"), make files, etc. We can think of the filesystem as a tree, with the root of the tree labeled "/" (forward slash). Files and directories (also called "folders") make up the nodes of the tree, and directories can contain more files and directories.

For example, you may have a directory on your computer called "CS107", which has some files and directories in that directory. It turns out there's a fun `tree` command that can show you a list of the files and directories in a tree format (names that end with `/` are directories, names that end with `*` are executable - files):

```
myth$ tree cs107 -F
cs107
├──
│   ├── main*
│   ├── main.c
│   ├── main.cpp~
│   └── readme.txt
├── assign1/
│   ├── file1.txt
│   ├── file2.txt
│   ├── folder1/
│   │   └── myProgram.c
│   └── folder2/
└── assign2/
    ├── docs/
    └── readme.txt

6 directories, 8 files
```

There are two special files in every directory: `.` (the current directory) and `..` (the parent directory). `.` is useful to refer to files in the current directory, like running a program by saying `./myprogram`. `..` is useful for navigating up the filesystem, like going up a level by saying `cd ..`.

If you want to refer to a file or folder, you can specify its location in two ways:

- an **absolute path**: how to get there from the root of the filesystem (`/`), or

- a **relative path**: how to get there from where you currently are

For example, lets say we want to open `file1.txt` above in `emacs`. Regardless of where we are in the filesystem, we can always refer to it like this: `emacs /cs107/assign1/file1.txt`. However, it may be easier to refer to it from where we currently are. For instance, let's say we're in the `cs107` folder. We can also say `emacs assign1/file1.txt`. This is using a relative path; we are specifying how to get there from where we currently are. Note that there is no leading `/`, because we aren't starting from the filesystem root; we start from our current location. Another example: let's say we're in the `/cs107/assign1/folder1` folder. We could say `emacs ../file1.txt`.

When you log into the myth computers, you are automatically put in your home directory (`~`), your personal file space on myth. This is not `/`, as it turns out - it is a path like `/afs/ir/users/t/r/troccoli`. But you can refer to it via the shorthand `~`.

Autocomplete

Your terminal remembers unix commands you enter, and can autocomplete commands for you. For example, if you press the up arrow key, the terminal will auto-fill your most-recently-typed command. You can continue pressing the up arrow key to go through commands you executed in order of decreasing recency. You can use the down arrow key to go through commands in order of increasing recency. If you want to execute that command, just press ENTER.

Second, if you type an exclamation point (also known as *bang*) followed by a character or a string and then *enter*, you will run the last command that starts with that character or string. E.g.,

```
myth$ ls -a
.  ..  hello  hello.c  hello.c~  innerFolder  readme.txt  samples
myth$ cd ..
myth$ !l
ls -a
.   assign0  assign2  .do_not_look.txt  file2.txt
..  assign1  assign3  file1.txt     .this_is_hidden.txt
myth$
```

Typing `!l` runs `ls -a` again, because the previous command that started with `l` was `ls -a`.

Third, if you press `Ctl-r`, you can then start typing characters and the line will start being populated with the last command that starts with the characters you have typed. This is easier to see in the video linked at the top of the page; also try it yourself!

Fourth, hit the `tab` key while you are typing a command, and the shell will automatically finish the command for you. Or, if it is ambiguous, it will provide options (you might have to type `tab` again). For example, if you want to type the `history` command, you can type `his` - `tab` and the rest of the command will be filled in:

```
myth$ his (then hit tab, at which point the entire history command will show)
myth$ history
...
```

If, on the other hand, you typed `hi` and then `tab` twice, you would see this:

```
myth$ hi (then the tab key twice)
hipercdecode hipsopgm history
myth$ hi
```

The shell is telling you that there are three commands that start with `hi`, and you have to continue with the letter you know is next. Then, if you type `tab` again, you can finish the completion.

Tab completion is extra-handy when you want to type file names or directories. Instead of typing a long file name, you can simply start typing a name and then type `tab` to have the name autocomplete. If only part of the name completes, that means there are multiple options, and typing `tab` again will give you the options that the shell is considering.

Commands To Navigate The Filesystem

Commands To Create and Delete Files and Directories

Commands To Print, Search and Compare Files

Manipulating Input and Output