Introduction to the UNIX Command Line

Mississippi Center for Supercomputing Research
Ben Pharr bnpharr@olemiss.edu

What is UNIX?

Originally an operating system developed by Bell Labs in 1969, the term has now come to mean an operating system that is UNIX-like. Linux is currently the most commonly used UNIX, but several others exist, such as Solaris, FreeBSD, Mac OS X, etc. All UNIX-like operating systems offer a similar command-line environment.

More about Linux

There are hundreds of Linux distributions put together by different companies and non-profit groups. They differ in various ways, but they all use the Linux kernel, which was developed Linus Torvalds, a Finnish college student, in 1991. They also include several GNU software packages and other open source software.

Some distributions you might have heard of are: Red Hat, Fedora, Ubuntu, Mint, and SUSE. Linux runs on everything from supercomputers to cell phones to vending machines.

Connecting to a remote server

Most UNIX and Linux systems that are operating as servers run an SSH daemon. Users with valid accounts on that system can then use an SSH client to connect to the server and run commands on the server remotely. There are a few free SSH clients. We recommend:

- PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/ (Windows)
- Mac OS X and most Linux distributions come with a command-line SSH client (`ssh`) pre-installed.

Today we'll be using the PuTTY to connect to hpcwoods, MCSR's gateway server.

1) Start by opening the PuTTY application.
2) In the "Host Name" field, enter "hpcwoods.olemiss.edu".
3) Click "Open".
4) Depending on whether or not your computer has ever connected to hpcwoods, you may get a dialog asking if you "want to save the new host key to the local database?" Click "Yes". This is so the client can confirm the identity of the server on subsequent connections.
5) If you are an MCSR user, you may use your username in the "User Name" field. Otherwise, use the temporary username given to you by the instructor.
6) Enter your password, or the one given to you by the instructor, in the "Password" field. **Note that nothing will appear as you type. This is a security feature.**
7) If successful, you should see a command prompt. Something like:
   ```
   username@hpcwoods:~>
   ```

Mac and Linux users can open a Terminal window and run:

```
ssh username@hpcwoods.olemiss.edu
```

The Shell

The prompt is displayed by a program called the shell. It allows the user to run other programs.

Almost all UNIX operating systems use `bash` as the default shell. Other popular shells include `tcsh` and `zsh`. Today we will be using `bash`. You can confirm you are using `bash` by typing:

```
echo $SHELL
```

For most users, the output will be:

```
/bin/bash
```

indicating the program that is running as their shell.

Clearing the screen

It is often convenient to clear the screen before you start a new task. This can be done by typing:

```
clear
```

Navigating the filesystem

When you login to a UNIX system, you start off in your home directory. Type:

```
pwd
```

to print your current directory. For most users, it will print something like:

```
/home/username
```

Note the tilde (~) in your shell prompt. This indicates that you are in your home directory. As you move through the filesystem, this section of your prompt will change to indicate your current directory.

To find out what files and directories you have in your home directory, type:

```
ls
```

If your account is new, you may not have any files or directories, so let's create a new directory to hold your files for this workshop:

```
mkdir workshop
```

Now when you run

```
ls
```

you should see the workshop directory listed.

Now we want to move into the newly created directory. Another way of saying that is, we want to make the new directory our current directory. To do this, we type:

```
cd workshop
```

`cd` stands for change directory. Your shell prompt should have changed to indicate that `workshop` is now your current directory.

Creating a text file

Now that we've created a directory to hold our files for this workshop, we can create our first file, using the `nano` text editor.

```
      nano new.txt
```

You can now type as in any normal text editor. Type a few lines, pressing the Enter or Return key to begin a new line. Once you're done, type **Control-o** to save the file. **Press Enter** to confirm the filename. Then type **Control-x** to exit `nano` and return to the shell.

`nano` is the simplest of UNIX text editors. Other options include `vi` and `emacs`. Their learning curves are steeper, but they also have more features.

To see your new file, type:

```
      ls
```

You should see new.txt. To find out more about your file, type:

```
      ls -l
```

This will tell you the file's permissions, owner, group, size, and modification date and time. This is an example of modifying a command's behavior by using an option, or switch. The ls command has dozens of options and many commands have several. Most commands have a manual page (often shortened to "man page") that explains what the command does and what options and arguments it takes. You can view the `ls` man page by typing:

```
      man ls
```

You can scroll a line at a time using the **Enter key** or a page at a time using the **space bar**. You can search the man page by typing "/search_term" and pressing Enter. For instance, to search for a way to sort files by size, you might type **"/size" and press Enter**. It will immediately take you to the first occurrence of the word "size." Typing **"/" and pressing Enter** will take you to the next occurrence of the term. Pressing "**q**" will exit back to the shell.

Viewing a file

The simplest way to view a file is to use `cat`:

```
      cat new.txt
```

`cat` simply prints the entire file to the screen. This is handy for short files.

For longer files, you might use less:

```
      less new.txt
```

You'll find that `less` behaves exactly like `man`. This is not a coincidence; `man` actually uses `less` for displaying manual pages.

You can, of course, use any text editor for viewing files as well.

Tab completion

Often it's not necessary to type in the entire name of a command or filename. For instance, using the example above, we can type in:

```
cat n
```

then **press the Tab key**. Since there is only one file in this directory that begins with an n, `bash` can fill out the rest of the name for us.

Command history

You can see commands that you have run in the past by **pressing the up arrow key**. Just press the Enter key to run a command. You can also see a list of previously run commands by typing:

```
history
```

Copying, renaming, and deleting files

To make a copy of your file named copy.txt, type:

```
cp new.txt copy.txt
```

Use the **`ls`** command to see the new file.

You can rename a file using the `mv` command:

```
mv copy.txt new2.txt
```

By default, `ls` shows all files in the current directory. However, you can specify one or more filenames as command line arguments. For example:

```
ls –l new2.txt
```

While we have two similar files sitting around, let's discuss wildcards. Most UNIX commands can accept a pattern rather than a specific filename. For instance, to see all filenames that end in ".txt":

```
ls *.txt
```

or all filenames that begin with "new":

```
ls new*
```

There are other, more advanced wildcards that will be covered in the next workshop.

Finally, you can delete a file by using the `rm` command:

```
rm new2.txt
```

Capturing command output

The w command can be used to see a list of users currently logged into the system:

```
w
```

If we wanted to save the output of the w command in a file named w_output.txt, we could type:

```
w > w_output.txt
```

We can now view or edit this output using the tools discussed earlier. This capability isn't part of the `w` command, but rather the shell, meaning that we can use this for any command.

"Filtering" command output

The output of a command can also be passed to another command for further processing. For instance, the history command often returns several pages of commands. Wouldn't it be nice if we could view the output of the `history` command inside `less`. Of course, `bash` allows us to do this:

```
      history | less
```

Here we are "piping" the output of the `history` command to the input of the `less` command. Almost all UNIX commands can be used in a similar fashion. For instance, we can quickly find our entry in the system's "password" file like so:

```
      cat /etc/passwd | grep bnp
```

Here we are using `cat` to "print out" the password file, and the `grep` command to search for our entry.

Finally, let's say we wanted to know how many users on the system were using the `bash` shell. We can do it like this:

```
      cat /etc/passwd | grep /bin/bash | wc -l
```

Here we are searching for every line in /etc/passwd that contains `/bin/bash` and then counting those lines using `wc -l`.

Copying and deleting directories

Before copying our workshop directory, we want to move back to our home directory. This can be done by typing:

```
      cd
```

`cd` without an argument always returns us to our home directory.

Now, to copy the workshop directory to a new directory called `ws_copy`, type:

```
      cp -r workshop ws_copy
```

To see the contents of the directory, type:

```
      ls ws_copy
```

We can delete this directory by typing:

```
      rm -r ws_copy
```

Disconnecting from the remote server

To disconnect from the remote server, you can type:

```
      exit
```

or just type Control-d.