CHEG 667-013 – CHEMICAL ENGINEERING WITH COMPUTERS
Department of Chemical and Biomolecular Engineering
University of Delaware

Spring 2025

COMMAND LINE INTERFACE PART I

Key idea today:

Installing and using the command line interface

Key goals:

Access a CLI and familiarize ourselves with basic commands and how to navigate the filesystem.

Our goal this week is to learn about the command line interface (CLI). We will start by discussing computer operating systems and the CLI. As we progress, we will learn about various tools for automating tasks, writing and running programs, including shell scripts, and manipulating the filesystem.

Our focus will be on Unix commands. This is the native environment for Linux and macOS. For Windows users, there is a CLI called *PowerShell*, but it will use different commands. While it is worth familiarizing yourself with PowerShell, to follow along, you'll need to install the *Windows Subsystem for Linux*. As of 2019, WSL 2 runs a virtualized full Linux kernel.

1 Running a command line interface

1.1 Windows

Windows 10 and 11 users should have access to the *PowerShell*.

With the PowerShell, use the command

```
wsl --install
```

to install the subsystem. After installing WSL, you will need to create a user account and password for your newly installed Linux distribution. Open the distribution (Ubuntu by default) using the *Start* menu. You will be asked to create a *user name* and *password* for your Linux distribution.

- How to install Linux on Windows with WSL:
  https://learn.microsoft.com/en-us/windows/wsl/install
- Set up a WSL development environment:
  https://learn.microsoft.com/en-us/windows/wsl/setup/environment
- An alternative to WSL is to run a virtual machine (VM) using a hypervisor program like VMWare or VirtualBox. The latter is available at https://www.virtualbox.org

1.2 macOS and Linux

Because the Macintosh OS is a derivative of Unix, users will have access to the Unix command line interface through the *Terminal* program. Just run terminal. Most of the commands we will discuss are more or less identical between Linux and macOS, but there can be some differences.

Most Linux installations will boot to a GUI like Gnome or KDE. From there, you can run a terminal program like *xterm*.

Exercise 1: Open a terminal window. Install WSL, if necessary.

2 Navigating the filesystem

You should see a prompt. The prompt might include user and directory information. Here are some commands we use to navigate the filesystem:

- `pwd` – return working directory name
- `ls` – list directory contents
- `cd` – change directory

The command `pwd` is perhaps the easiest to understand; it shows us our current working directory:

```
ef1j@snaut:~$ pwd
/home/ef1j
```

This tells us that we're in the *home directory* of user `ef1j`. We see that it has the *absolute path* of `/home/ef1j`. We'll look at this more in detail below.

> **NOTE!** The tilde character, `~`, is special. It can be used to refer to our home directory or the home directory of another user, as in `ls ~luser`.

Each command can have a number of options. Let's take a closer look at the `ls` command. I might see something like this when I type it after starting the terminal:

```
ef1j@snaut:~$ ls
Desktop/    Downloads/  Pictures/  Templates/
Documents/  Music/      Public/    Videos/
```

The command shows that there are several *directories* in `/home/ef1j`. (Desktop, Downloads, etc.) I can get more information using the option `-l`:

```
ef1j@snaut:~$ ls -l
total 4
drwxr-xr-x 2 ef1j ef1j 2 Jul 30  2024 Desktop/
drwxr-xr-x 2 ef1j ef1j 2 Jul 30  2024 Documents/
drwxr-xr-x 2 ef1j ef1j 2 Jul 30  2024 Downloads/
drwxr-xr-x 2 ef1j ef1j 2 Jul 30  2024 Music/
drwxr-xr-x 2 ef1j ef1j 2 Jul 30  2024 Pictures/
drwxr-xr-x 2 ef1j ef1j 2 Jul 30  2024 Public/
drwxr-xr-x 2 ef1j ef1j 2 Jul 30  2024 Templates/
```

```
drwxr-xr-x 2 ef1j ef1j 2 Jul 30  2024 Videos/
```

which now shows the owner of each directory (`ef1j`), its group, and the date it was created.

If I type the command `ls -a`, I might see the following:

```
ef1j@snaut:~$ ls -a
./              .dbus/          .mozilla/                   Videos/
../             Desktop/        Music/                      .vnc/
.bash_aliases   Documents/      Pictures/                   .wget-hsts
.bash_history   Downloads/      .profile                    .Xauthority
.bash_logout    .gnupg/         Public/                     .xsession-errors
.bashrc         .ICEauthority   .ssh/
.cache/         .lesshst        .sudo_as_admin_successful
.config/        .local/         Templates/
```

It shows a number of *hidden files and directories* that are part of the system settings or used for application support.

3 Getting information and help

Now is a good time to introduce where we can get some help with Unix commands. Unix systems have a built-in help system of *manual pages*. The command to access these is `man`.

- Type `man ls` to see the options for the list command.

The `man` command creates a formatted *man page* in the terminal window.[1]

- Use *space* to scroll ahead, *u* to move back up, and / to search for a term.

Exercise 2: Try several `ls` commands and options, like:
- `ls -l` – long format
- `ls -lt` – long format, newest first
- `ls -lh` – long format, "human readable output"
- `ls -F` – append indicator (one of `*/=>@|`) to entries

Note how you can combine options. Use the `man` page to find more options to try. Maybe these will be more interesting once we have more files?

Note the following:

- Command options can be separate or combined: `ls -l -t` is the same as `ls -lt`. Try both!
- Sometimes options have input.

4 Changing directories

When we start a terminal session, usually we're placed in our home directory. This is part of the filesystem structure that you may normally see as folders in a GUI. Try going one step lower in the

[1] Hence the answer to the oft-asked question of how a command works: *"RTFM!"* or *"Read the freakin' man (page)!"*

filesystem:

```
ef1j@snaut:~$ cd ..
```

In my case, I go one directory down to the `/home` directory. If I list the current directory, I see my user folder, and if I use the `pwd` command, it tells me that I'm in `/home`.

```
ef1j@snaut:/home$ ls
ef1j/
ef1j@snaut:/home$ pwd
/home
```

Note the following:

- Two dots, `..` refers to the directory immediately below the current directory.
- One dot, `.` refers to the current directory. It can be important in some commands, including running a program in the current directory.

A few tips:

- Typing `cd` alone will take you to your home directory
- `cd -` will take you back to the directory before the last `cd` command. Try it!

5 Root directory

Go one more level down using `cd ..` or type `cd /`. If you use `ls`, you should see something like this:

```
ef1j@snaut:/$ ls
bin@   dev/  home/  lib32@  libx32@  mnt/  proc/  run/   snap/  sys/  usr/
boot/  etc/  lib@   lib64@  media/   opt/  root/  sbin@  srv/   tmp/  var/
```

This is the *root* directory. It contains directories that hold many of the system files. Here are a few you might see:

`/bin` – executable programs or binaries (in this case, it points to `/usr/bin`)

`/boot` – system startup files

`/dev` – system device files

`/etc` – system configuration files

`/lib` – various *libraries* that the different software uses

`/media` – removable drives and disks will normally show up here

`/mnt` – mount point for manually mounting drives and devices

`/opt` – installed software, like `anaconda` environments

`/proc` – computer info (process information pseudo-filesystem)

`/root` – root user home

`/sbin` – superuser programs

`/sys` – information about devices

`/tmp` – temporary files

`/usr` – system files, including executables and binaries

`/var` – log files, lock files, spool files, and other system info

Exercise 3: Explore! Using `cd` and `ls`, look around the filesystem. Use `ls -l` and `ls -lt` to look at when the files and directories were created or modified. Who is the owner of the file? Be sure to look at the files in `/bin` or `/usr/bin`. You should be able to find the programs we've been using (`ls`, `cd`, `man`) and many others.

Linked files

You might see some files like this:

```
lrwxrwxrwx   1 root   root      7 Apr 24  2022 bin -> usr/bin/
```

The `l` indicates that this is a *linked* file or directory. Notice how it points to a different directory, `/usr/bin`. This allows the filesystem to have more than one name for a file or directory. See `man ln` for more information.

File permissions

In my directory listing, I also see the following for the `/etc` directory:

```
drwxr-xr-x 105 root   root    195 Feb  4 16:12 etc/
```

The letters indicate that the file is a directory. There are three groups of letters after this that indicate permissions:

- `r` – read, `w` – write, and `x` – execute

Each of the three groups corresponds to:

- the file owner, the group, and all or everyone.

In this case, `root` is the owner. This is the super-user or system administrator. The user `root` can read, write, or execute files and programs in `/etc`. Members of the `root` group can, too. As a normal user who is not in the `root` group, we can only read and execute files and programs.

6 Making and removing directories

If we want to add or remove folders to our file hierarchy, we use the following commands:

- `mkdir` – make a directory

- `rmdir` – remove a directory (it needs to be empty)

  WARNING! Removing a directory is *permanent*. **There is no trash can or undo!** This will apply to files, too.

Now go back to your home directory (`cd ~`) and make a new folder called `cheg667`:

```
ef1j@snaut:/home/ef1j$ mkdir cheg667
```

Verify that you made the directory by using `ls`. You can remove this directory with the `rmdir` command.

7 Moving, copying, and removing files

Now we'll learn how to move, copy, or remove files.

- `mv` – move (or rename) a file
- `cp` – copy a file
- `rm` – delete or remove a file

  WARNING! All of these file operations are permanent! There is no undo! Moving or copying a file on top of another will erase or *clobber* the original. Be careful!

We need some files to work on. Do this in your home directory:

```
ef1j@snaut:~$ touch foobar
```

The `touch` command is a way of updating the modification time of a file. If the file doesn't exist, `touch` will create one. If we use `ls -lt`, we should see at the top that we created the file `foobar`, which has zero bytes.

```
-rw-rw-r-- 1 ef1j ef1j 0 Mar 30 06:57 foobar
```

Notice the file permissions. The file can be read by the user, group, and other users. Only the user and the group have permission to write to or otherwise modify the file or erase it.

Next, move `foobar` to the directory `~/cheg667`:

```
ef1j@snaut:~$ mv foobar cheg667
```

If you type `ls` now, you will not see it. But if you change directory to `/cheg667` or type `ls cheg667`, you will find that it was moved to that directory.

Change directories to `cheg667` directory and type the following:

```
ef1j@snaut:~/cheg667$ cp foobar foo
```

This copies the file `foobar` to a new file, `foo`:

```
ef1j@snaut:~/cheg667$ ls
foo  foobar
```

Notice that the file modification times are different:

```
ef1j@snaut:~/cheg667$ ls -l
total 1
-rw-rw-r-- 1 ef1j ef1j 0 Mar 30 07:08 foo
-rw-rw-r-- 1 ef1j ef1j 0 Mar 30 06:57 foobar
```

Create another file, but this time with text using the `cat` command,

```
ef1j@snaut:~/cheg667$ cat > bar
Terminal whispers,
cat > file.txt begins
empty lines take shape
^D
```

When you type `cat > bar`, it is telling the terminal to redirect the *standard input* of the terminal to a file. (More on this later.) When you hit enter, the input that you type will be written to the file. You can stop by typing `control-D`. Verify that the text was written to the file, again using the `cat` command:

```
ef1j@snaut:~/cheg667$ cat bar
Terminal whispers,
cat > file.txt begins
empty lines take shape.
```

Using `ls`, you will also see that the file has a non-zero size:

```
-rw-rw-r-- 1 ef1j ef1j 67 Mar 30 07:18 bar
```

You can read back the file by typing

```
ef1j@snaut:~/cheg667$ cat bar
```

You should see the text in the file.

Now copy the file `bar`. Call it `bar2`:

```
ef1j@snaut:~/cheg667$ cp bar bar2
```

Do you see it in the directory listing?

You can remove `bar2` with the command:

```
ef1j@snaut:~/cheg667$ rm bar2
```

Check the directory listing again. The file should be gone.

Exercise 4: Practice making new text files with the `cat` command. Rename a file using `mv`. Practice copying files. Use `man` to learn more about `mv`, `cp`, and `rm`.

8 More information

We learned about the terminal, moving around a filesystem through the command line, and basic file manipulation commands.

Here are a few sources for more information:

- Command line – https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
- Using Linux – https://linuxcommand.org
- Linux filesystem – https://www.linuxfoundation.org/blog/blog/classic-sysadmin-the-linux-filesystem-explained

9 More exercises

Exercise 5: In your home directory, type `ls -R`. What does this command do? What do you see?

Exercise 6: Type `ls /root`. What happens? Why?

Exercise 7: What does the `cat` command stand for? What are some other uses of the command besides creating and printing a file?