Portfolio 3: Introduction to Linux

Ed Davis

04/03/2021

Directories

Using the terminal, you can navigate directories (i.e. folders) through the use of the `cd` (change directory) command with the path of the directory to be selected. In the directory path, parent directories can be chained together using the `/` divider. For example, the following command will navigate through the `Documents` → `PhD` → `SC2` directories. To check the new current working directory, the command, `pwd` (print working directory) can be used.

```
$ cd Documents/PhD/SC2
$ pwd

/home/user/Documents/PhD/SC2
```

Directories can also be selected using symbols:

- `.`: current directory,
- `..`: previous directory,
- `~`: home directory.

To view the files and directories in the current working directory, the `ls` (list) command can be used, and the `mkdir` (make directory) command can be used to create a new directory.

```
$ ls

dir_1    dir_2    dir_3

$ mkdir new_dir
$ ls

dir_1    dir_2    dir_3   new_dir
```

File Manipulation

The following commands are used for file manipulation:

- `touch`: create file,
- `cp`: copy file,
- `mv`: move/rename file,
- `rm`: remove file (permanently!),

- `rmdir`: remove directory.

An example use of these commands is below.

```
$ touch my_file.txt           # Create file
$ cp my_file.txt my_copy.txt  # Copy file
$ ls

my_file.txt    my_copy.txt

$ mkdir dir_for_copy                        # Create directory for copy
$ mv my_copy.txt dir_for_copy/my_copy.txt   # Move copy to new directory
$ cd dir_for_copy
$ ls

## my_copy.txt

$ rm my_copy.txt  # Delete file
$ cd ..
$ rmdir dir_for_copy  # Delete the new directory
$ ls

my_file.txt
```

The following commands can be used to edit simple files.

- `cat`: reads a file,
- `echo`: prints string argument,
- `>>`: appends command,
- `>`: overwrites file with command.

```
$ touch hello.txt
$ echo "hello " >> hello.txt  # append strings to text file (note newline added)
$ echo "world" >> hello.txt
$ cat hello.txt

hello
world

$ echo "hello world!" > hello.txt  # Overwrite the file with a single line
$ cat hello.txt

hello world!
```

To write a file which is more than one line, use a text editor such as `nano`. The command,

```
$ nano names.txt
```

will create a text file and open it for editing in the terminal. Once written, `^O` can be used to save the file and `^X` to exit.

Examine file contents

These commands can be used to display or search file contents.

- `less`: read file one page a ta time (`u` = page up, `space` = page down, `q` = quit),
- `head`: reads the first 10 lines (`head -n` reads first `n` lines),
- `tail`: reads the last 10 lines (`tail -n` reads last `n` lines),
- `grep`: display lines that mention argument,
- `grep -c`: counts the lines that mention argument,
- `grep -i`: displays the lines that mention the case insensitive argument,
- `sort`: sort lines of text line by line (does not modify file),
- `uniq`: searches for and removes duplicate lines in a file (does not modify file).

An example use of the `grep` command is shown in the following.

```
$ grep GNU example.txt

in their name. The Free Software Foundation uses the name GNU/Linux to
from the GNU project. This has led to some controversy.
licenses, such as the GNU General Public License.
which are provided by the GNU Project, and usually a large amount of
```

Wildcards

Wildcards can be used to reference objects in the terminal. The `?` wildcard will select all objects which have a single character, and `ab?` could select the objects `aba`, `abb`, `abc` and so on. The `*` wildcard works similar to `?`, except `*` can reference zero or many characters. For example `ab*` could select `ab`, `abaa`, `ab21398fh` and so on. Crucially, `*` by itself will select all objects available. This means that the command,

```
$ rm *.txt
```

will delete all available text files.

Pipes and Pipelines

Pipes, `|`, can be used to chain multiple commands together (to make a pipeline). For example, the following text file,

```
$ cat names.txt

Ed
Jack
Shannon
Sam
Sam
Ed
Annie
Dan
Ed
```

can have its contents organised using a pipeline including the `sort` and `uniq` commands.

```
$ sort names.txt | uniq -c

    1 Annie
    1 Dan
    3 Ed
    1 Jack
    2 Sam
    1 Shannon
```