Basics of UNIX Walkthrough

Getting Started

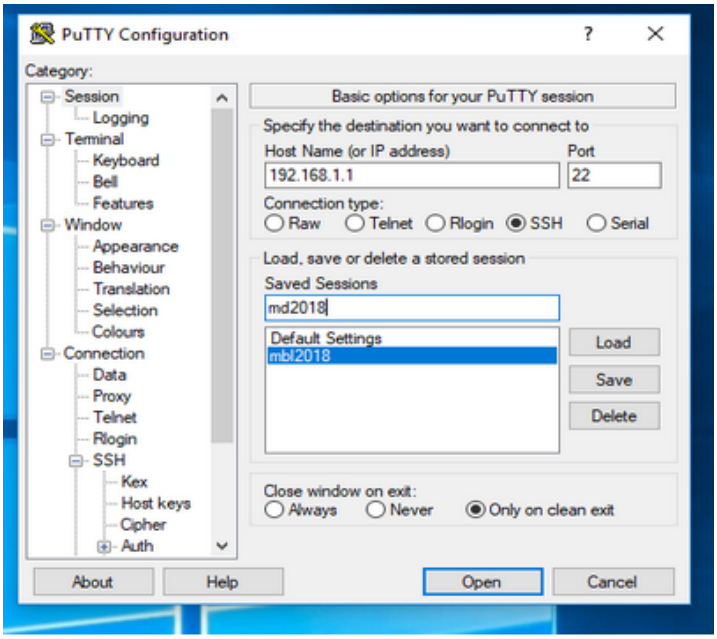
Mac/ Linux

Login using Terminal on a MAC or UNIX. Your username will be your first name. Use the key private key from the keypair you generated at the beginning of the course.

```
ssh -Y -i ~/.ssh/MyPrivateKey.pem your_username@18.219.176.206
```

Windows (using PuTTY)

1. Open PuTTY
2. Under Category, click on SSH > Auth
3. Click browse
4. Find your private key (keyname.pem) and select it
5. Under Category, click on SSH > Auth and check the "Enable X11 forwarding" box.
6. Under Category, click Session and input the server IP address (18.219.176.206) in the "host name" box
7. Type "md2022" in the box under saved sessions and click save.
8. Double-click on the "md2022" that appears under saved sessions.
9. Log in with your username. Your key should be used automatically.
10. For future logins, just double-click the "md2022" saved session.



Where am I?

By default, you'll be in your home directory. The `pwd` command will **print** the current **working directory**, telling you where you are.

In [ ]:
```
pwd
```

You can also establish who you are.

In [ ]:
```
whoami
```

To see the files and directories present in your current directory, use the `ls` command.

In [ ]:
```
ls
```

You can get more detail on the files using `ls -l`.

In [ ]:
```
ls -l
```

There are lots of other ways to customize the output of `ls`. You can use the `man` command to get a **man**ual for another command.

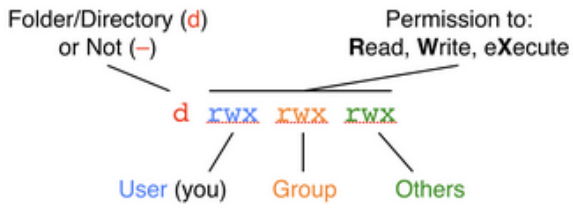
In [ ]:
```
man ls
```

Now you can try something more complex.

In [ ]:
```
ls -lah
```

File permissions

So what's all this `drwxrwxr-x`? These are permissions, which tell you what you (and others) can do to a file or directory.



Files and directories

You can create new directories, move to them, and see what's inside.

Avoid using these characters in file or directory names: `| , ! @ # $ % ^ & * ( ) { } [ ] + = ; : \ ` ? > < space tab`. `_` and `-` are good substitutes for spaces.

In [ ]:
```
mkdir my_new_directory
```

In [ ]:
```
ls
```

In [ ]:
```
cd my_new_directory
```

In [ ]:
```
ls
```

You can move to a directory above your current one using a double dot (`..`).

In [ ]:
```
cd ..
```

In [ ]:
```
pwd
```

You can chain together as many double dots as you like. This would take you up three directory levels: `../../..`.

Another useful shortcut is `~`, which represents your home directory.

In [ ]:
```
cd ~
```

In [ ]:
```
pwd
```

We can also create simple text files.

In [ ]:
```
echo "some fun text" > myfile.txt
```

In [ ]:
```
ls
```

We can show the entire file with `cat`, scroll through it with `less`, or edit it with `nano`.

In [ ]:
```
cat myfile.txt
```

In [ ]:
```
less myfile.txt
```

In [ ]:
```
nano myfile.txt
```

`Ctrl+x` to exit, `y` / `n` to save or not save the file.

Files and directories can be copied, moved, and removed. *There is no recycle bin here; if you delete something, it's gone forever. Remove with care.*

In [ ]:
```
cp myfile.txt newfile.txt
```

In [ ]:
```
mv newfile.txt my_new_directory
```

In [ ]:
```
cd my_new_directory
```

In [ ]:
```
mv newfile.txt delete_this.txt
```

In [ ]:
```
rm delete_this.txt
```

In [ ]:
```
cd ~
```

To delete directories and their contents **recursively**, use the `-r` flag.

In [ ]:
```
rm -r my_new_directory
```

Searching

We can use `grep` to search within text files. Move back to your home directory and try searching `myfile.txt`.

In [ ]:
```
grep "fun" myfile.txt
```

To find a file, use the `find` command.

In [ ]:
```
find ~ -name "myfile.txt"
```

Other useful tips

You can use the up arrow to scroll through your command history - useful for editing previous commands.

You can use tab to autocomplete commands and filenames - type the beginning of the file name and then hit tab. Hit it twice to see a list of options if there's more than one possibility.

Asterisks (*) are wildcard characters. For example, this would list all files inthe current directory enting in .txt: `ls *.txt`

If a text file is behaving strangely, the problem may be linebreaks. PCs, Macs, and Unix use different characters to tell a file the line has ended: `\r\n` , `\r` , `\n` , `^M` . Use a decent text editor than can produce Unix linebreaks (not notepad, never MS Word!). Notepad++ is good for Windows users. You can fix incorrect linebreaks using `dos2unix` and `mac2unix` .

You can see which directories will be searched when executing programs by examining your PATH variable: `echo "$PATH"`

In [ ]:
```

```