Command line activity

This is intended to be a very quick guided tour to get acquainted with the command line. We'll spend a few minutes in class and see how far people get. The goal is to experience interacting with a command line interface. If you find some aspect particularly interesting, feel free to branch off and investigate it as long as you can give a quick summary during the class discussion after the activity. Please use `http://bellard.org/jslinux/` to complete the activities.

1 The basics

1.1 Go to your home directory

To go to your home directory type:

```
cd ~
```

Type:

```
pwd
```

to display the path to your home directory.
Let's see what is in our home directory, type:

```
ls
```

There should be two files. Let's create a new directory called `thessi` by typing:

```
mkdir thessi
```

To see if it was created, type:

```
ls
```

Oops we actually wanted to create a file names `thesis`. Let's press the `Up` key a couple times to pull up our old command from our history, correct it and create the correct `thesis` file.

2 Getting help

Now if we run `ls` we can see our `thesis` directory and the incorrect `thessi`. We would like to get rid of `thessi` but we don't know the command for deleting a directory. Can you find the correct command on google?

Maybe a search query like "delete a directory linux"? Try out the solution and see if it deletes.

Did you find something about `rmdir` or maybe `rm -r` (careful with `rm -r` in the future, it's easy to permanently delete a lot of things at once)? There are often multiple ways to accomplish things in command line. Probably the easiest way to delete an empty directory is:

```
rmdir thessi
```

3 Canceling things

To cancel a command, you can use `Ctrl+c` (hold the `control` key and press `c` key). Let's test this out. Change your working directory to `/dev` and look around. Can you do this by yourself?

The commands are:

```
cd /dev
ls
```

We're going to look at the `urandom` file. To look at a file you can use the `cat` command (short for concatenate). First, lets try out tab completion. Type:

```
cat ur
```

and hit the Tab key. It should autocomplete to `cat urandom`. When you hit Enter your screen will start filling up with random gibberish without stopping a la the green text in the movie Matrix. Once that gets old, you can hit Ctrl+c to cancel. `urandom` is actually a special file that continuously returns random information although here we were just using it to practice cancelling.

4 Bioinformatics: Finding things in DNA

As an example of why command line could be useful lets look at a biological example. There is a handy `grep` command that can be used to look for strings like DNA motifs or genes. We'll need a fake DNA string so let's create one. First let's make sure we're in our home directory by typing `cd`. Then to create a fake DNA string, type something like (enter any random DNA sequence instead of directly copying):

```
echo ACACATATGATAGATATATGTGGAST > gene
```

We'll get into the details of how this works in another class but it should have created a file named `gene`. Type `ls` to make sure it was created. We might want to find how many As are in this DNA. To do this we can use the `grep` command with `-o` argument like `grep -o PATTERN FILE`. So that'll be:

```
grep -o A gene
```

Try looking for a few other DNA motifs. Remember you can press `Up` to edit your previous command.

5 Advanced: For loops

This is getting a bit more advanced but gives an example of how command line can make life easier. Often we want the computer to do something many times, e.g. look for DNA matches in 20 gene files or rename 100 analysis files. In programming, one way to deal with this is a for loop. We tell the computer for each of these items, do something. In bash you can create a for loop with something like:

```
for temp in MULTIPLE THINGS TO OPERATE ON;do
  ACTION ${temp}
```

```
done
```

Here I used all CAPS for places custom information will go. For example, to run blast on 3 files you could do something like:

```
for temp in file1 file2 file3;do
  blast ${temp}
done
```

Let's try a for loop. As a simple example, let's try saying hello to a bunch of people. Linux has a command `echo` which makes the computer print whatever comes after `echo`. For example:

```
echo Hello XXX
```

will print "Hello XXX".
Let's say we have to say hello to Jim, John, Bob, Mary and Pat. Can you follow the pattern above to greet each person by name?

You could do something like:

```
for temp in Jim John Bob Mary Pat;do
  echo Hello, ${temp}!
done
```

You should get an output like:

```
Hello, Jim!
Hello, John!
Hello, Bob!
Hello, Mary!
Hello, Pat!
```

Try saying "Goodbye" instead of "Hello". Remember you can use the `Up` arrow to edit your previous commands. Now we want to create a directory for each person. Try replacing the `echo` command to make 5 directories with each directory named for one of the people.
This same for loop pattern with a little editing could be used to run bowtie, blast or any other program.

6 Just for fun: Compile your first program

If you reached this far during class, then you must have flew through things. Let's get a little more advanced just for fun. `cd` back to your home directory and `ls` again to see the contents. There's a file called `hello.c`. Let's see what's in it using `cat` (remember you can hit Tab to autocomplete after typing `cat h`:

```
cat hello.c
```

This is a simple program written in the C programming language. We need to compile it into a program that we can run. To do this we can use the `tcc` command (short for tiny C compiler):

```
tcc hello.c -o hello
```

List the directory contents to see a new file called "hello" (the `-o hello` in the previous command tells the compiler what file to create). We can run this file by typing:

```
./hello
```

You should see a greeting from the program.
If you still have time, you can customize the C code by starting a text editor (like microsoft word but on the command line) called `vi`. `vi` is not the best text editor to introduce beginners to but we are limited on this Linux instance (again this is just for fun so no problem at all if things don't work). So to start editing the file type:

```
vi hello.c
```

You should see the contents of the `hello.c` file. Before pressing anything else, press the `i` key to get into editing mode. Now use the arrow keys to navigate to "Hello World". Delete `World` and replace it with your name (be careful not to delete the quotes). Hit the `Esc` key to exit editing mode. Type `:wq` and hit Return to save and quit. You can `cat hello.c` to see your edits. Compile `hello.c` as before and run it. You should see your customized greeting.

If you got that done already, try to make a for loop to repeat the greeting 20 times. Check google for a tutorial on for loops in C and try to implement it. When you start `vi`, hit `i` immediately and when done editing again hit `Esc` and type `:wq` to save.