Linux Practical

Where am I and what's there?

At the command line, **p**rint the current **w**orking **d**irectory to find out where you are:

```
pwd
```

You should see a result of the form: **/home/*username*** – this directory is known as your home directory. Take a note of it; you will need it later. Also note that we'll use *username* (in italics) a lot in this practical. Whenever you see it, you should replace it with your actual username.

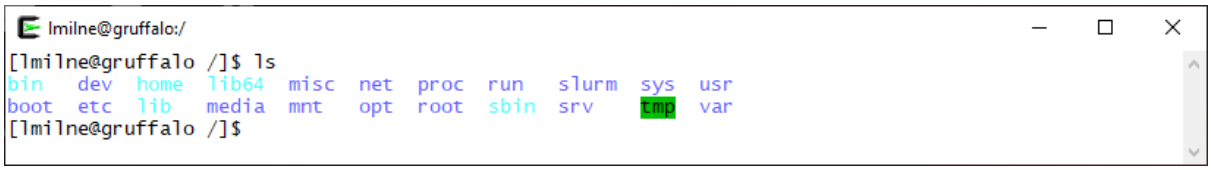
Now have a look at the root (slash) file system. The **cd** (change directory) command will get you there:

```
cd /
```

Enter the following command, which lists the contents of the current directory:

```
ls
```

You should see something resembling the following:



See if you can use **cd** to navigate from the root back to your home directory at **/home/*username***.

Once back home (just type **cd** on its own if you get really lost), list the contents of your home directory. It may not have much in it if you've not used the cluster before.

> **Hint: Where am I!?**
>
> If you get lost, typing **pwd** (for "print work directory") will always tell you where you are, but you can also take a glance at the SSH window's title bar – many common tools will mirror your current directory there.

Navigate to your "scratch" directory

Now move into your "scratch" directory. This is an area of your storage where you should do most of your working. Note that the scratch directory is not backed up.

To move there, first ensure you are in your home directory (type **cd** on its own to go there) and then type:

```
cd scratch
```

Make a directory for the training course

In your scratch directory, and at the command line, type:

```
mkdir training
```

This will make a directory called "training" - confirm it's been created by using the **ls** command.

Change to the training directory

Let's use the new directory. To do this, enter:

```
cd training
```

Now where are you? Remember you can always find out with **pwd**.

You should be at **/home/*username*/scratch/training**

Type at the command line:

```
cd ..
```

Now where are you?

Note that **..** is a special directory that moves you up one level in the directory tree. A single dot **.** means the current directory, and a double-dot **..** means the parent directory.

You can combine multiple "move up one level" shortcuts into a single **cd** command, eg:

```
cd ../../..
```

Chances are this put you right back to the root (**/**) directory again, so move back home with **cd**.

Pay attention to the title bar in your SSH client window too – it will track the current directory for you too. If it shows the **~** character, that's ok, as **~** is a special shortcut for your home directory.

Hint: Using the mouse in the command line

You can't click the mouse cursor around the Linux command prompt like you're used to, but it does still offer basic cut and paste. Drag with the mouse to highlight text, then simply right-click to paste it back into the window. You can also double-click a word (eg a filename) and instantly copy it into your current command with a right-click.

Create a directory

Return to the **training** directory. If you are a bit lost you can go there by typing **cd** to go into your home directory, then changing into the training directory like this:

```
cd

cd scratch/training
```

Or you can use the **~** shortcut as part of the path to the training directory in a single command:

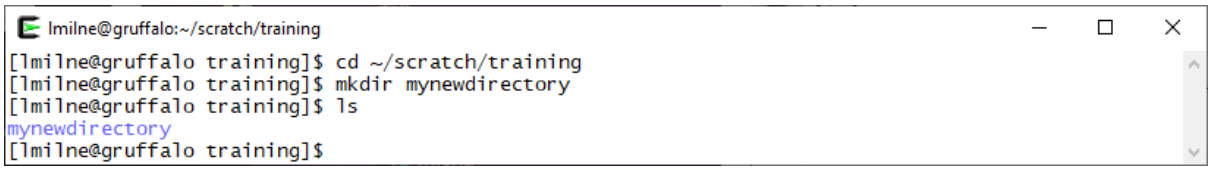
```
cd ~/scratch/training
```

> **Hint: Special keys**
>
> Typing long file or directory names is annoying, but luckily there's an easier way. You only need to type part of a name and then hit the TAB key – Linux will autocomplete the rest of it for you. You can also use HOME to jump the cursor to the start of a line, and END to jump to the end of a line.

Within **training** create another new directory by typing at the command line:

```
mkdir mynewdirectory
```

Confirm the new directory is there by using **ls**

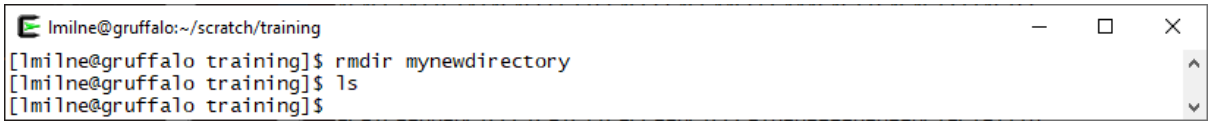


Remove a directory

Type at the command line:

```
rmdir mynewdirectory
```

Now get a listing of the contents to confirm it has been removed.



Make another directory

Make a directory call **tmp** in your training directory and change into it.

```
mkdir tmp

cd tmp
```

Create an empty file

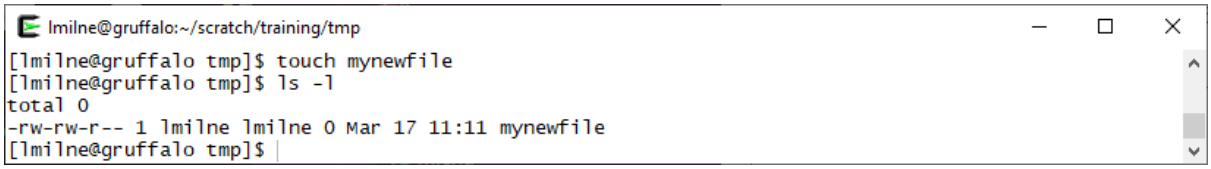
Within the **tmp** directory, type:

```
touch mynewfile
```

This special command just creates an empty file with the given name. Now type the following command, an enhancement to the previous **ls** command

```
ls -l
```
(note this is "minus ell" not "minus one")

Can you work out how big the file you have created is? Or when was it created? The **-l** option for **ls** gives additional information about files.

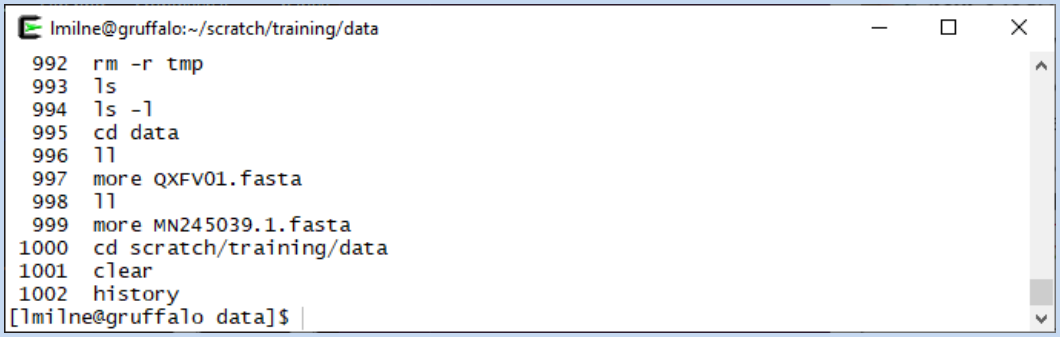


Note, you can also use **ll** ("ell ell", for long listing) which is a special shortcut (or "alias") for the same command.

Hint: What have I done?

The up and down arrow keys can be used to scroll backwards and forwards through recently entered commands. Resubmit a command by hitting ENTER, or even edit and then resubmit – this is great for when you've mistyped a long path and need to try again without having to retype everything.

You can also use the **history** command to display a recent history of everything you've typed:



If you see an entry you want to re-run, use the **!** (always called "bang" in the Linux world), followed by the number of the command as shown by history. For example:

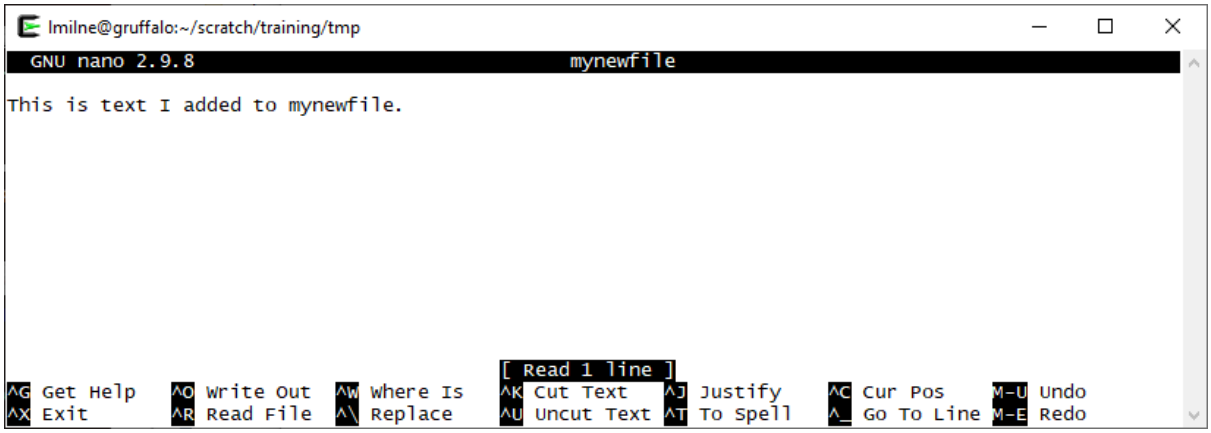
```
!997
```

Edit a file

You can now edit the empty file using the text-editor **nano**. This is useful for editing README files, configuration files, scripts and so on. To edit a file using **nano** simply type

```
nano mynewfile
```

Your ssh client window should now look like this, and you should be able to start typing to add text to the file.



The menu at the bottom of the **nano** window shows a variety of options, the larger your window the more options are shown at once. For the full set of options view the help section by typing **CTRL+G** while running **nano**. As you add or edit text in **nano**, note that the cursor cannot be moved by clicking with the mouse but is controlled by the cursor keys on your keyboard.

To save your work, exit **nano** by doing **CTRL+X,** then it will ask if you want to save (press **Y** or **N,** or **CTRL+C** to cancel) and ask what the filename of the saved file should be (just press **ENTER** if you want to save over the same file). If you then list the directory again you should be able to see that **`mynewfile`** is no longer an empty file.

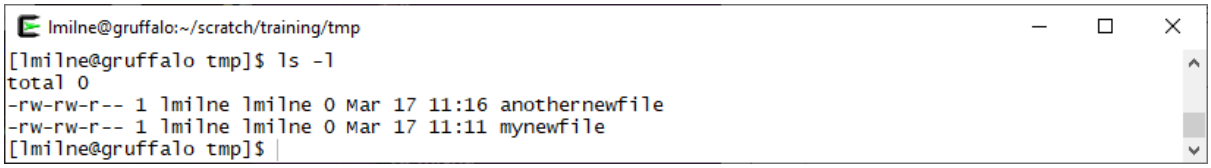
Copy a file

Make sure you're still in your **`/home/`*`username`*`/scratch/training/tmp`** directory, and then type:

```
cp mynewfile anothernewfile
```

The **`cp`** command will make a copy of a file.

Confirm that **`anothernewfile`** has been created and that it is the same size as the original **`mynewfile`**. Are the creation dates and times the same? (Note: If you did the last couple of steps very quickly they may be the same but **`anothernewfile`** should be newer than **`mynewfile`**.)



Move a file

In your **/home/*username*/scratch/training/tmp** directory, type:

```
mv anothernewfile ~
```

Confirm that the file is no longer in the current directory.

```
lmilne@gruffalo:~/scratch/training/tmp
[lmilne@gruffalo tmp]$ mv anothernewfile ~
[lmilne@gruffalo tmp]$ ls -l
total 0
-rw-rw-r-- 1 lmilne lmilne 0 Mar 17 11:11 mynewfile
[lmilne@gruffalo tmp]$
```

Now check to see if it is in your home directory with:

```
ls -l ~
```

Remember, **~** (tilde) is an alias for your home directory. It can save you having to type **/home/*username*** all the time.

```
lmilne@gruffalo:~/scratch/training/tmp
[lmilne@gruffalo tmp]$ ls -l ~
total 160476
-rw-rw-r-- 1 lmilne lmilne         0 Mar 18 09:40  anothernewfile
-rw-r--r-- 1 lmilne lmilne 178619914 Feb 17  2020 'HPC Blog.docx'
lrwxrwxrwx 1 lmilne lmilne        24 Mar 19  2020  projects -> /mnt/shared/projects/jhi
lrwxrwxrwx 1 lmilne lmilne        26 Mar 19  2020  scratch -> /mnt/shared/scratch/lmilne
[lmilne@gruffalo tmp]$
```

Now move the file back with the following command

```
mv ~/anothernewfile .
```

Remember the **.** means the current directory, so this command in full has really asked for **/home/*username*/anothernewfile** to be moved to **/home/*username*/scratch/training/tmp/anothernewfile**

```
lmilne@gruffalo:~/scratch/training/tmp
[lmilne@gruffalo tmp]$ mv ~/anothernewfile .
[lmilne@gruffalo tmp]$ ls -l
total 0
-rw-rw-r-- 1 lmilne lmilne 0 Mar 17 11:16 anothernewfile
-rw-rw-r-- 1 lmilne lmilne 0 Mar 17 11:11 mynewfile
[lmilne@gruffalo tmp]$
```

Now move up one level to the **~/scratch/training** directory

```
cd ..
```

Copy files

You can copy directories of files by supplying **cp** with the **-r** argument. The general form of this is **cp -r *source_directory destination_directory***.

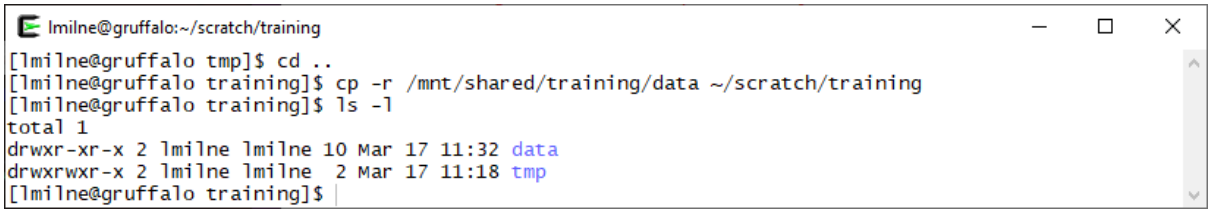
You can use this to take a copy of all the tutorial files and store them locally in your training directory:

```
cp -r /mnt/shared/training/data ~/scratch/training
```

This will take a few moments to run.

The **-r** option means recursively copy the directory, that is, copy the directory, its files, and any sub-directories (and their files) within it. Remember that **~** means your home directory, so what we've asked for is for the **data** directory located in **/mnt/shared/training** to be copied to your **scratch/training** sub-directory within your home directory.

Check that the data directory has copied to the training folder using **ls -l**



Move into the **data** directory.

Revisiting ls

ls has lots of useful options (that work alongside **-l**), such as:

-h **list file sizes in human-friendly KB, MB, etc**

-S **list (and sort) files by size**

-a **show hidden files as well**

Note the capital **-S** in the second row. That isn't the same option as **-s** (small s). Linux is case sensitive and it's *really* important to pay attention to this.

The **-a** option is another interesting one as it lists hidden files. In Linux, that's any file whose name begins with a dot, for example **.testHiddenFile** – Enter:

```
ls
```

Then enter:

```
ls -a
```

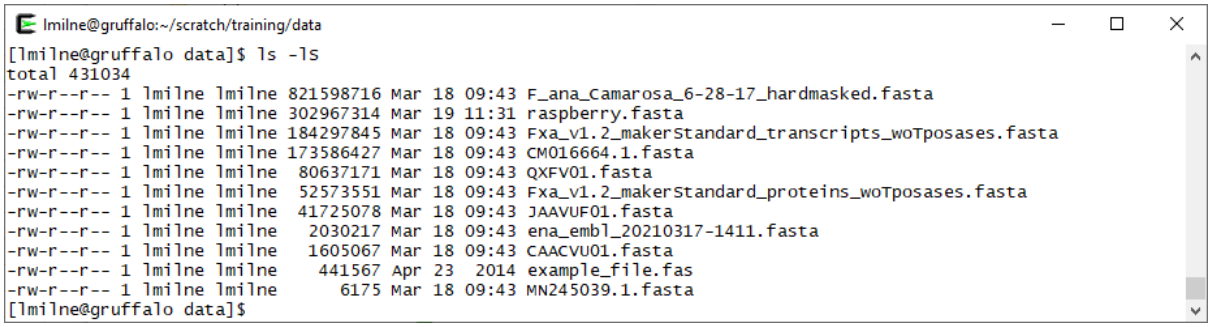
It's worth noting that some of the command line options only work in conjunction with others. For instance the output of **ls -h** will look no different than the output of **ls**. Combine it with **ls -l** though e.g. **ls -l -h** and the sizes specified in the **ls -l** output will be shown in a human

readable format, rather than just the raw number of bytes (which for large files will be in the millions).

Some tools allow you to combine command line options, eg **`ls -lh`** although if in doubt, just specify options separately as in the first form above.

At the command line enter:

```
ls -lS
```



(Remember case sensitivity, that's a capital **S**). Note in the image above that the output of **`ls`** differs from that of **`ls -S`** (but only in the ordering of the directories, the default ordering of **`ls`** is alphabetical). In the **`ls -lS`** example we can confirm that the files have been listed in size order (largest to smallest).

Many common tools and programs on Linux come with built-in help manuals – meaning we can check on what other options **`ls`** has by running:

```
man ls
```

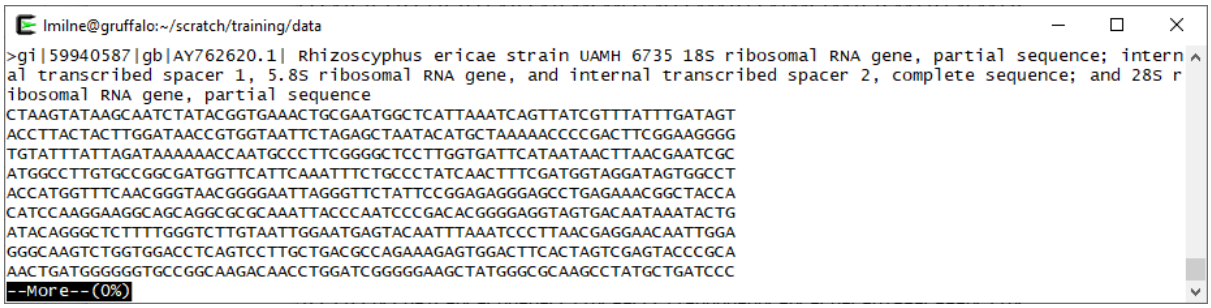
Viewing a file

You can view the contents of files on linux using the **`more`** command. Check you're still in the **`data`** directory (run **`cd ~/scratch/training/data`** if not). If you list this directory, there should be a file called **`example_file.fas`**

Let's view it now:

```
more example_file.fas
```

You should see a lot of text on the screen, as below.



You can scroll through the file by holding down the return key on your keyboard, or move through the file a page at a time by pressing the spacebar. You can return to the command prompt by pressing **q** or **CTRL+C**.

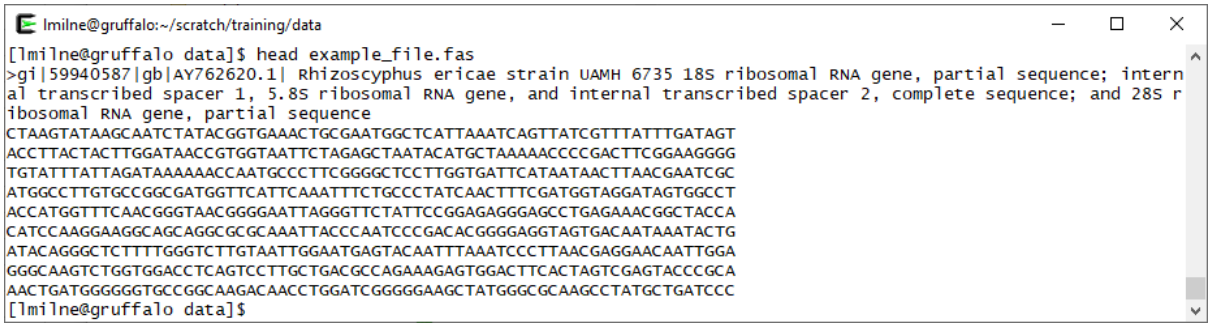
Peeking at a file

There are two other really useful tools for viewing files – these are **head** and **tail**.

You can use **head** to quickly view the contents of the start of a file:

```
head example_file.fas
```

You should see that the first ten lines of the file have been displayed (although there may be some line wrapping depending on your SSH client's window size).



Take a look at the man page for **head** (using **man head**) to see which options are available. Try to view the first twenty lines instead of the default ten.

tail is very much like **head**, but instead it enables you to view the end of a file. Enter:

```
tail example_file.fas
```

If you read the man page for tail (**man tail**) you'll note that it has a few more options than head. An interesting option for tail is **-f** which follows the end of the file as the file is growing. This can be really useful for following along with the output of a long running program.

Navigation reminder

Change back into the **tmp** directory you were previously in. There are a few ways you can do this, so pick one from the following:

You can use the **..** shortcut to move up one directory level:

```
cd ../tmp
```

Or, use **cd** to return to your home directory, then supply it with the *relative* path to the **tmp** directory:

```
cd
```

```
cd scratch/training/tmp
```

Or, use the *absolute* path to the **tmp** directory, which will work from any directory you're currently in:

```
cd /home/username/scratch/training/tmp
```

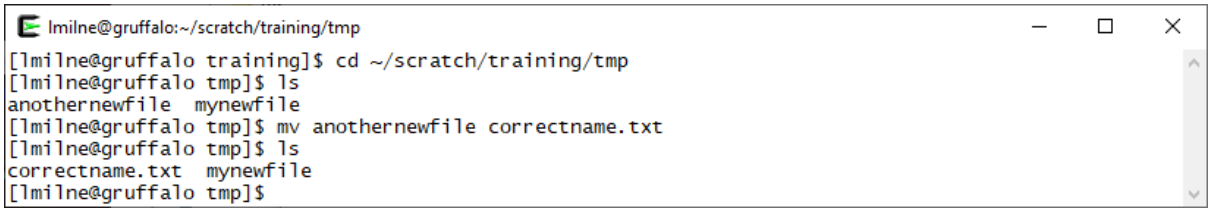
Rename a file

Earlier you created two files **anothernewfile** and **mynewfile** in the **tmp** directory. List it using **ls** to confirm they're both still there.

The **mv** (move) command can also be used to rename a file. For example:

```
mv anothernewfile correctname.txt
```

and use **ls** as always to check that things are as you expect.



Delete a file

The first thing to remember on almost all Linux file systems, is once you have a deleted a file it is gone! There is – usually – no recycle bin at the command line, so be careful.

Type the command:

```
rm correctname.txt
```

Check by listing the files that it really has gone.

Revisiting deleting directories

Return to your **training** directory again.

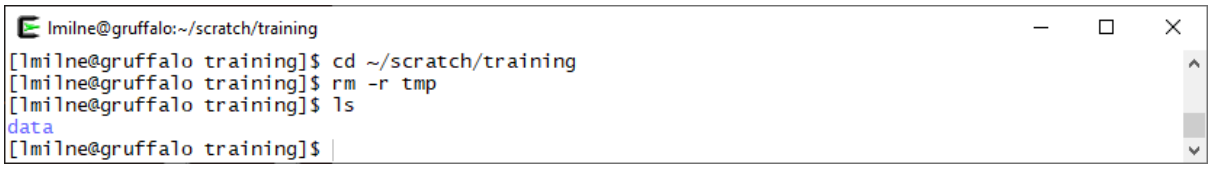
```
cd ~/scratch/training
```

Earlier you saw that **rmdir** will delete a directory, however it can only do so if the directory is empty. You're finished with the **tmp** directory and don't want that data clogging up the system anymore, so you want to delete the directory, but it's full of files and **rmdir** refuses to do the job…

Fortunately, **rm** accepts an additional (recursive) option which will let you remove a directory *and* all of its contents in one go:

```
rm -r tmp
```

To prove it's gone, list the current directory. You should no longer see tmp in the directory.



If you have time, why not try creating and deleting a few more files and directories for practice?