UNIX Course

Foundation Tutorial

© FDM Group Ltd 2014. All Rights Reserved.

Any unauthorised reproduction or distribution in part  
or in whole will constitute an infringement of copyright

# What is UNIX and why UNIX?

UNIX is a powerful operating system. There are many commands in UNIX and many of these commands can do many things. UNIX is secure, stable and used by many users, particularly developers.

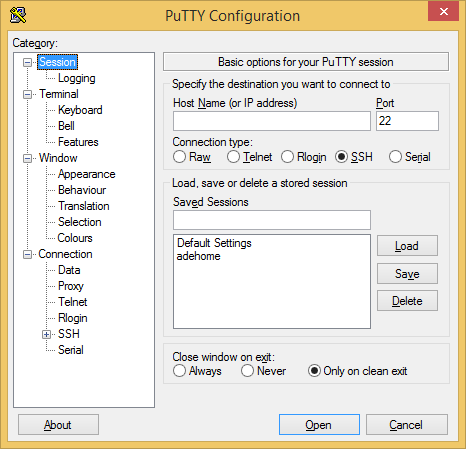
UNIX is an operating system written in a computer language called C. UNIX is used worldwide by programmers and developers. In addition to this, there are a range of compilers written for the UNIX operating system which allow programs written in C(or C++ or Java) to be compiled on UNIX. So there is a very close relationship between UNIX, C, C++ and Java.

It exists in many versions, some with full Graphic User Interfaces and others without. The version used in the classroom will not have a Graphical User Interface. The benefit of using such an operating system is that you learn how to use the command line tools much better. We will use PUTTY to access the UNIX system in this course.

# 1. Setting up Putty for use with UNIX

The following section shows how to configure Putty on your computer. You will need to repeat these steps if you use another computer.

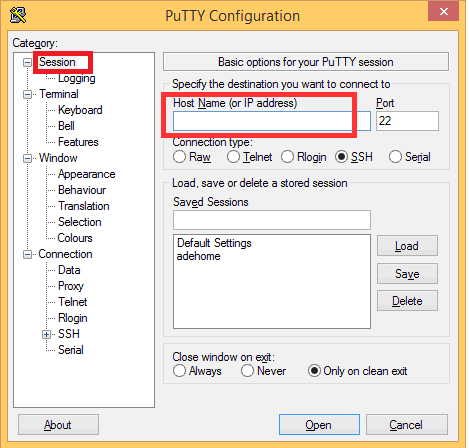
On the computers in the classroom (or if you have downloaded Putty at home), double click on the "Putty" icon. The following window appears:



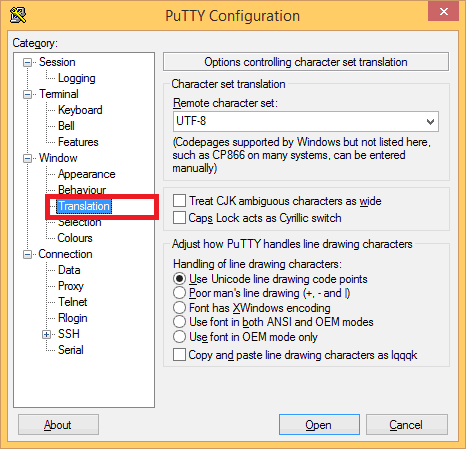
On the left of the screen, click "Session". On the right of the screen, enter in the "Hostname" textbox:

**unix.fdmgroup.com**

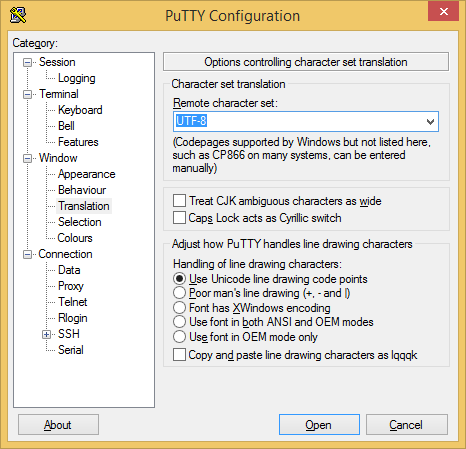
This is shown on the following screen:



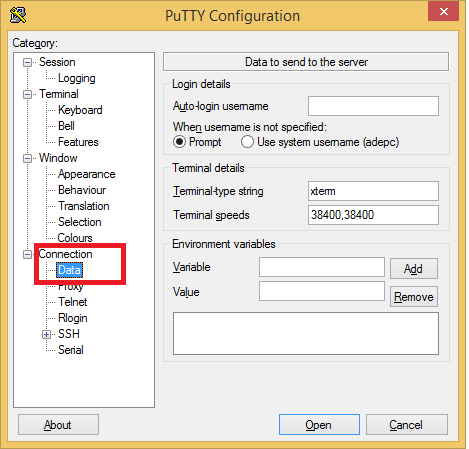
On the left of the screen, click on "Window"-> "Translation":



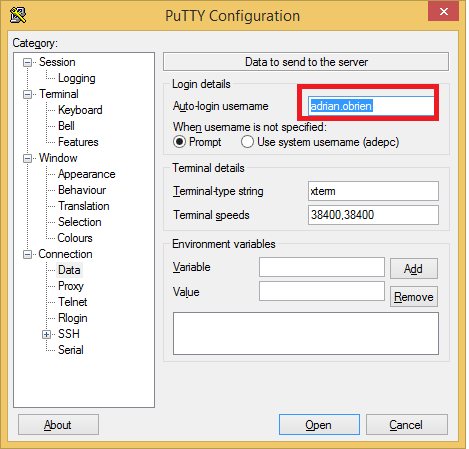
In the "Remote Character Set" entry, ensure that "UTF-8" is selected:



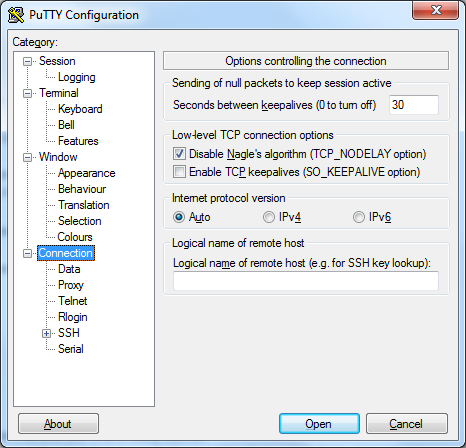
On the left of the screen, click on "Data".



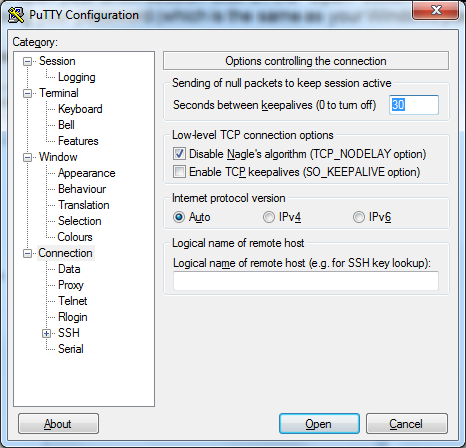
In the "Auto-login username" entry, type in your username:



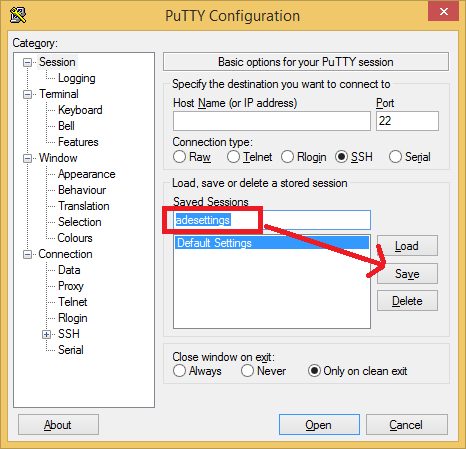
One additional setting is very useful. Click on “Connection”:



Then change the setting entitled “Seconds between keepalives” to the number 30. This will allow your session to remain active and this will ensure you are not disconnected from the UNIX session:



You can now save these settings on this computer by returning to the "Session" screen (click "Session" on the left of the screen):



Type a name for the settings under the message "Saved Sessions". Then click the "Save" button. This will save the settings you have provided.

You can load these settings later by selecting a Saved session and then clicking on the "Load" button. To open your UNIX session click on the "Open" button and you will be asked to login using your password (which is the same as your Windows password).

# 2. Using Putty at home

You can download Putty for use at home. When you click on www.putty.org it takes you to the following site:

http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe

where you can download the executable file.

You will to need change the settings as before.

Once you have set up your PUTTY session you can start typing in commands as shown in the following section.

# 3. Unix Navigation commands-used very frequently

### pwd

**pwd** shows your current location. As you navigate around the UNIX system, you will always have a location. The UNIX prompt will also show your current location. A typical directory tree is shown below:

/

/bin

/etc

/dev

/home

When you log in to unix the initial working directory will be your home directory:

/home/firstname.lastname/

For example:

/home/adrian.obrien/

In the example above we used the command pwd alone. We did not supply any other information to the command. In this situation, we have supplied no options or arguments to the command.

**cd**

**cd dirname** changes your location to the specified directory. For example, if we use the command:

cd games

we will move our location to the games directory (if it exists). In other words, the word games is an **argument** to our cd command.

You can change your location to any location in the UNIX operating system. If the directory does not exist then you will receive an error message. If you successfully change location, then when you use the **ls** command you will list the files in the new directory.

If you want to move “up” a directory (towards the / directory) in the directory tree then use the command:

**cd ..**

You can always return to your home directory by using the command:

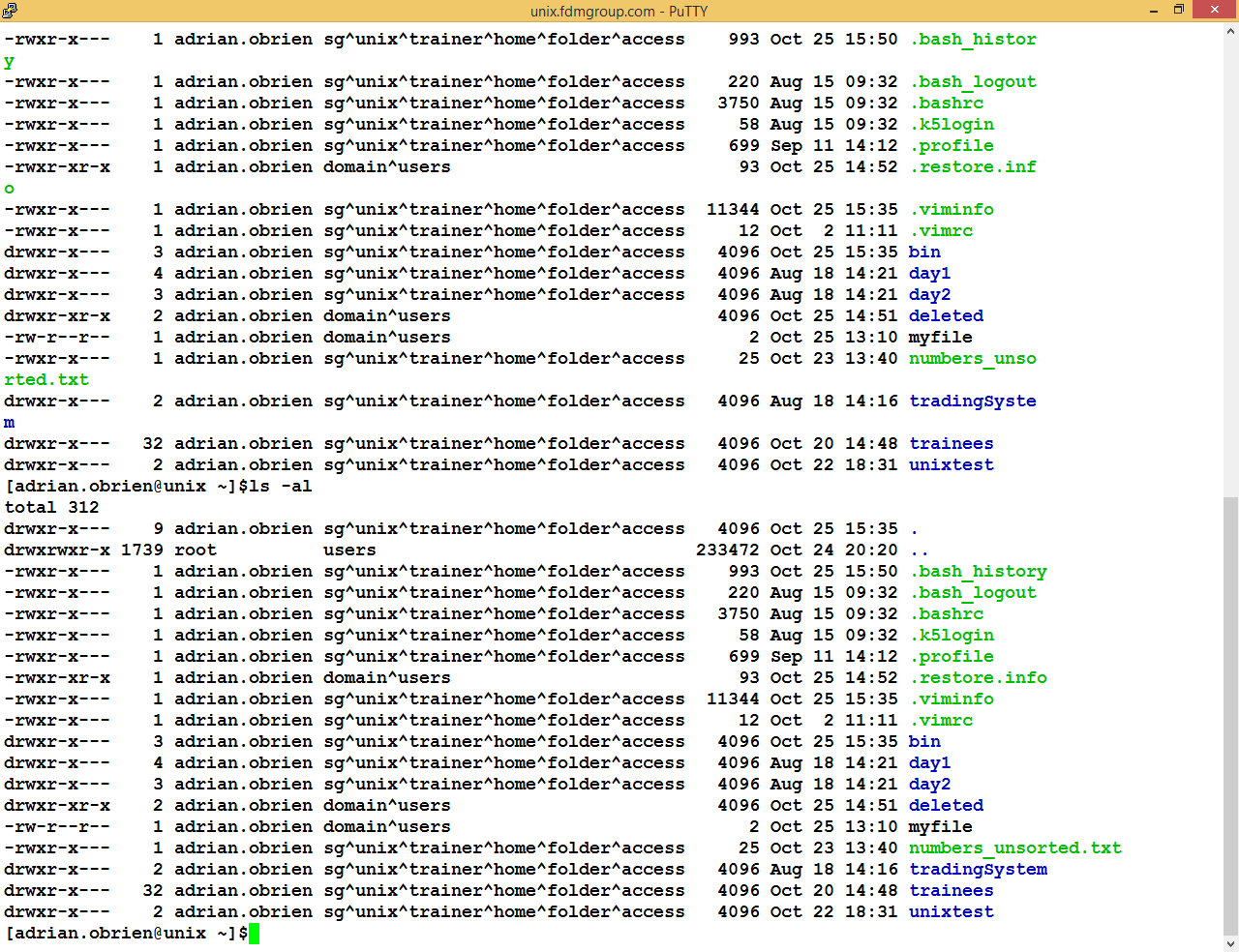
**cd (or cd ~)**

### ls

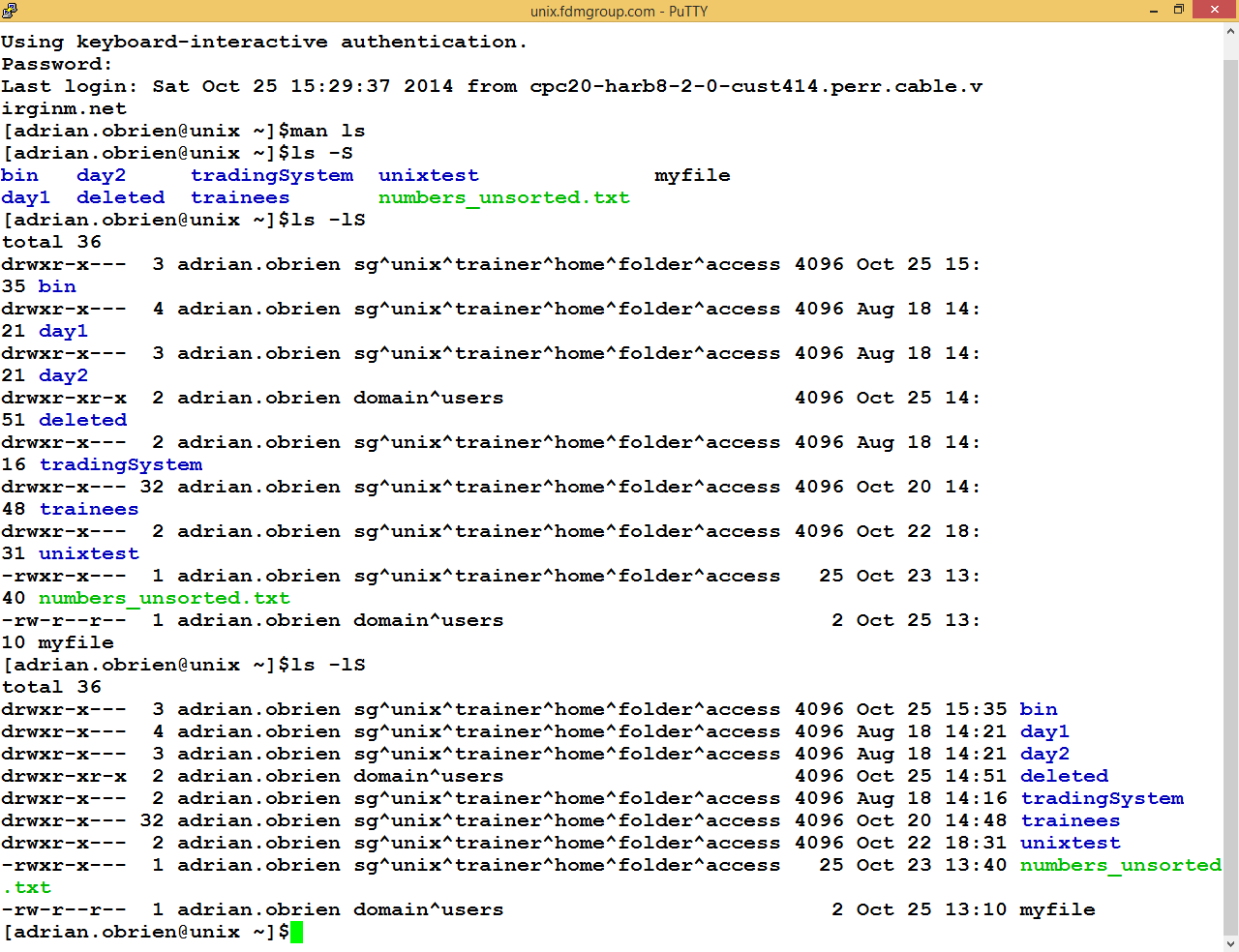
**ls** lists your files.   
**ls -l** long listing: contains the file size, file owner and access privileges and modified time/date.   
**ls -a** lists all files, including hidden files.

Hidden files are files whose filename starts with a dot symbol: The file **.profile** is a hidden file because the filename starts with a dot(.) symbol. These types of files are hidden because they are often system files.

The **ls** command has many other options and you can combine these. For instance if you use the command **ls** -al this will list all the hidden files and provide a long listing of those files:



If you use the command **ls -lS** this will display a long listing and sort them by file size:



The command **ls -al** is equivalent to:

**ls -a -l**

**ls -l -a**

**ls -la**

In other words, we can separate the command line options supplied to the ls command.

We can also place them in any order.

### mv

The **mv** command moves a file to another location:

You can use the following command to give a file a different name:

**mv filename1 filename2**

You can also use the command to move a file to a different directory:

**mv filename1 directory**

### When you run these commands there will be no output unless there is an error locating the files or directories. For this reason it is useful to list the files in the directory first so that you know what files are there.

### cp

The **mv** command moves a file to another location but the cp command copies a file to another location.

You can use the following command to make a copy of a file:

**cp filename1 filename2**

You can use the following command to copy a file to a different directory

**cp filename1 directory**

### mkdir

**mkdir dirname** creates a new directory.

When you list the contents of the current directory the new directory (which you have just created) will be shown.

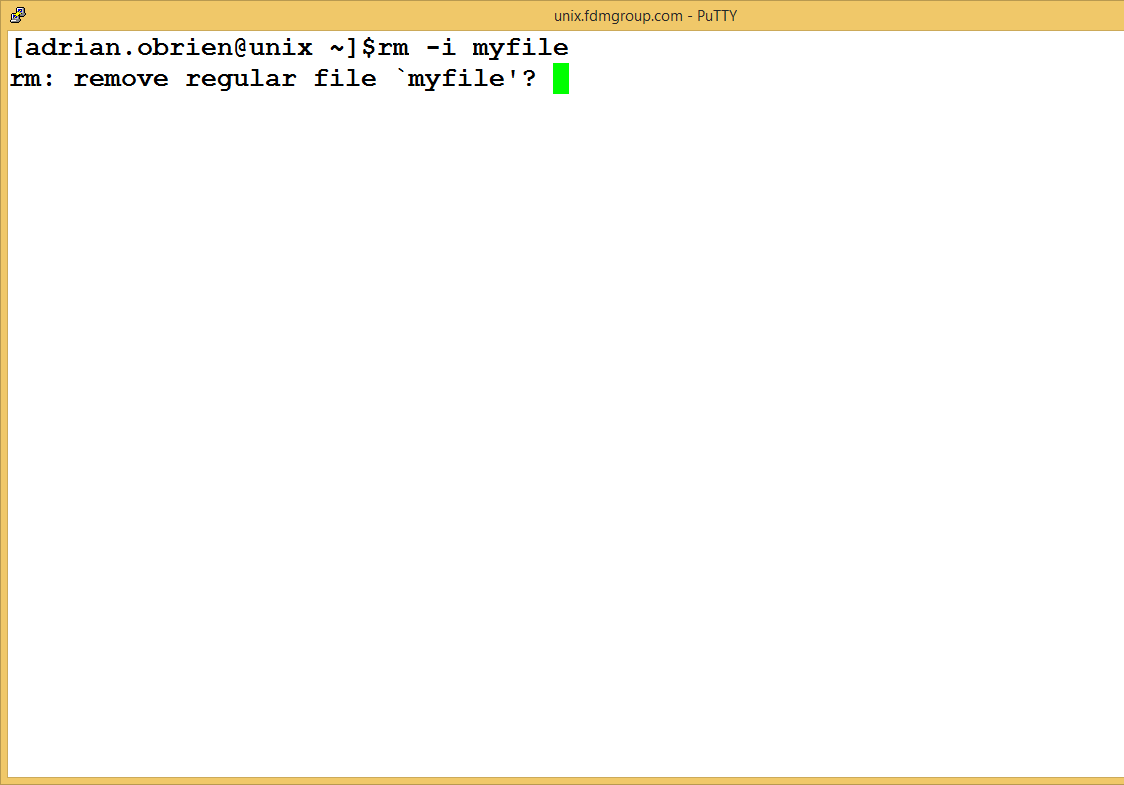
### rm

**rm filename** removes a file.

**WARNING: rm \***

This command will removes all your files and will provide no further information. For this reason this command has many options such as the option **-i**, which will ask you for confirmation before deleting the file:

**rm -i filename**



You will need to confirm that you wish to delete the file by pressing the y (yes) key. Otherwise the file will remain in place.

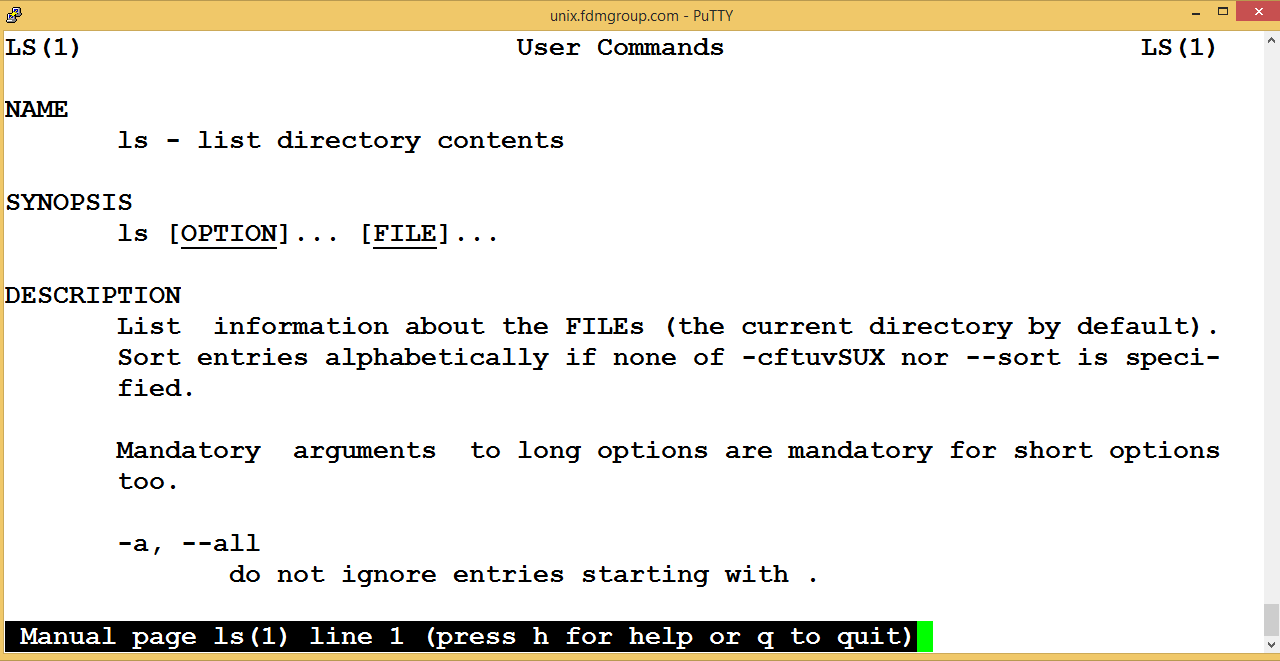
**man**

One very useful command is the **man** command. This provides more information on any command in UNIX and lists the options for a command and provides more detail on what the command actually does.

For example, when you type the command:

**man ls**

you are presented with the following screen:



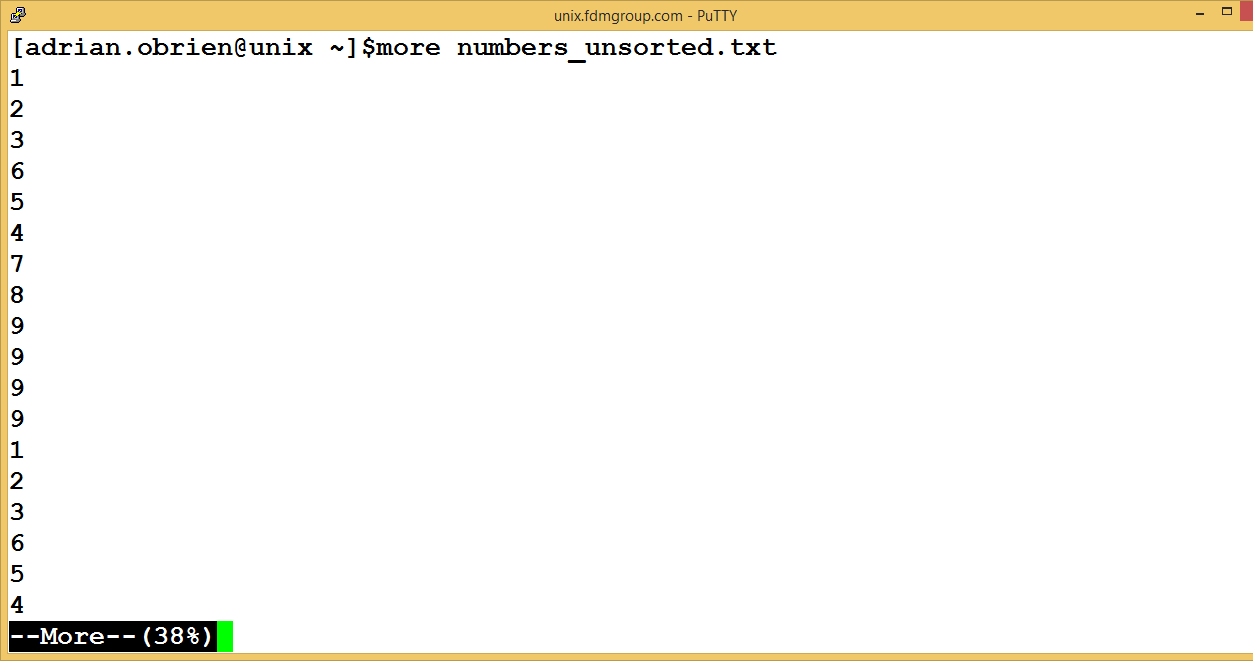
For example, if you use the following ‘man’ commands, this will give you more information on those commands:

|  |  |  |  |
| --- | --- | --- | --- |
| man ps | man ls | man pwd | man cp |
| man mkdir | man cat | man more | man man |

# 4. Viewing Files-used very frequently

### more

**more filename** displays the file one screen at a time. Pressing the SPACE key moves "down" the file one screen at a time and pressing the "q" key exits the display, returning the user to the UNIX prompt. For example:



If you do not wish to see the remaining pages of the file, press the q key.

### cat

You can use the cat command to perform various tasks:

**cat filename** displays the entire contents of the file (to the end of the file). If the contents file extends over several pages then the contents of the entire file will be displayed, ending with the final page of the file.

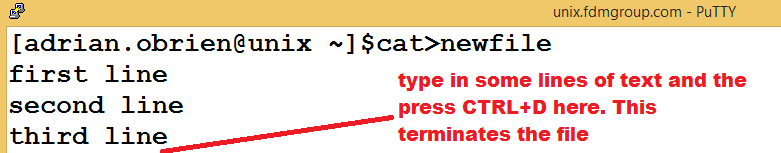
**cat>filename** allows you to type lines of text on screen and these lines will be added to the file. When you want to end the input, press the CTRL+D keys.

e.g. If you type in the following input at the command prompt:

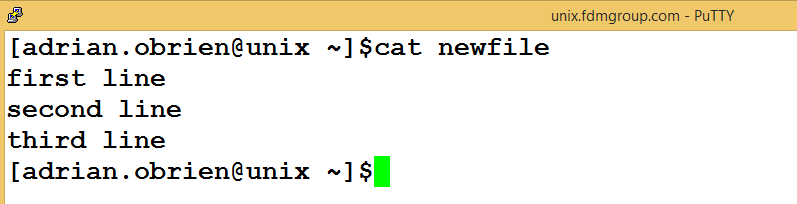
**cat>filename**

A  
B  
C  
Then press : <CTRL+D>

The file will contain three lines of text containing A, B and C on separate lines. Here is another example:



Typing in the command cat newfile shows the contents of the file:



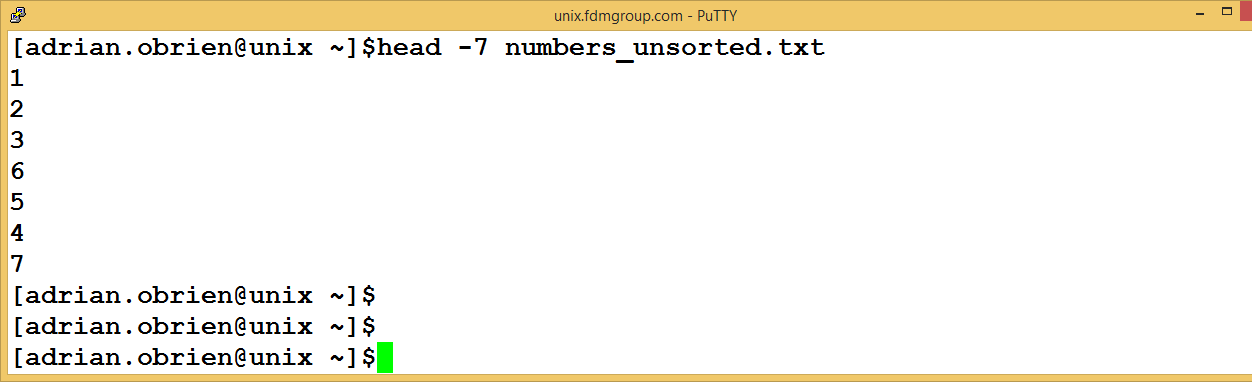
**cat>>filename**. This does the same as cat>filename but it appends the text to the end of the file. Press the CTRL+D keys to end the input.

### head and tail

**head -n** filename is used to display the first **n** lines of the file called filename

**tail -n** filename is used to display the last **n** lines of the file called filename

For example to display the first 7 lines of a file:



### wc

**wc *filename*** displays how many lines, words, and bytes there are in a file

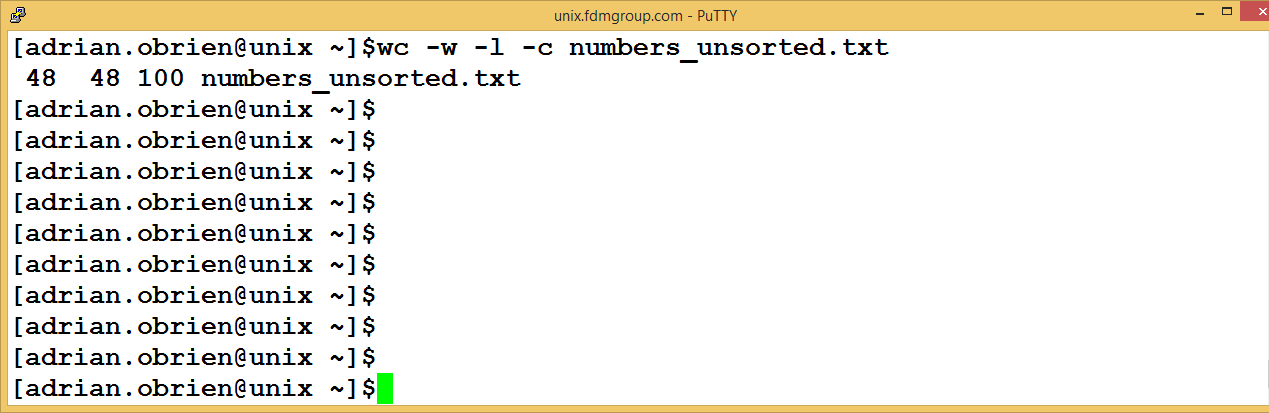
you can use the following variations:

**wc -l *filename***

**wc -w *filename***

**wc -c *filename***

### For example the following command will display the number of words, lines and bytes in a file:



### chmod

**chmod** lets you change permissions on your files. By default you can read, write and execute your files but you may sometimes want to change these permissions.

For example,

**chmod 777 filename** allows the file to be read from, written to and executed by everyone.

**chmod 700 filename** allows full access to the file for the owner but no access to anyone else.

**The three digits above are broken down:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **USER** | | | **GROUP** | | | **OTHERS** | | |
| **4** | **2** | **1** | **4** | **2** | **1** | **4** | **2** | **1** |
| **r** | **w** | **x** | **r** | **w** | **x** | **r** | **w** | **x** |
|  |  |  |  |  |  |  |  |  |
| **r** | **w** | **x** | **r** | **w** | **x** | **r** | **w** | **x** | **777** |
| **r** | **w** | **x** | **-** | **-** | **-** | **-** | **-** | **-** | **700** |
| **r** | **w** | **x** | **r** | **-** | **x** | **r** | **-** | **x** | **755** |
| **r** | **w** | **x** | **r** | **-** | **-** | **r** | **-** | **-** | **744** |

**The alternative is to use the following syntax:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **who** | **how** | **what** |  |
|  | **🡻** | **🡻** | **🡻** |  |
|  | **U** | **+** | **R** |  |
| **chmod** | **G** | **-** | **W** | **filename** |
|  | **O** | **=** | **x** |  |

**chmod u+r filename** : adds read access for the user(owner),

**chmod g-w filename:**  removes write access for people in the same group.

**chmod o=x filename:** makes the file executable for others.

You can also combine these.

Be careful not to change permissions on a file/directory in such a way that you can no longer access the file/directory. This change will be irreversible. If in doubt, try out command first with an empty file.

### diff

This command allows you to compare two files to check if they contain the same contents.

**diff filename1filename2** compares two files, and shows where they differ

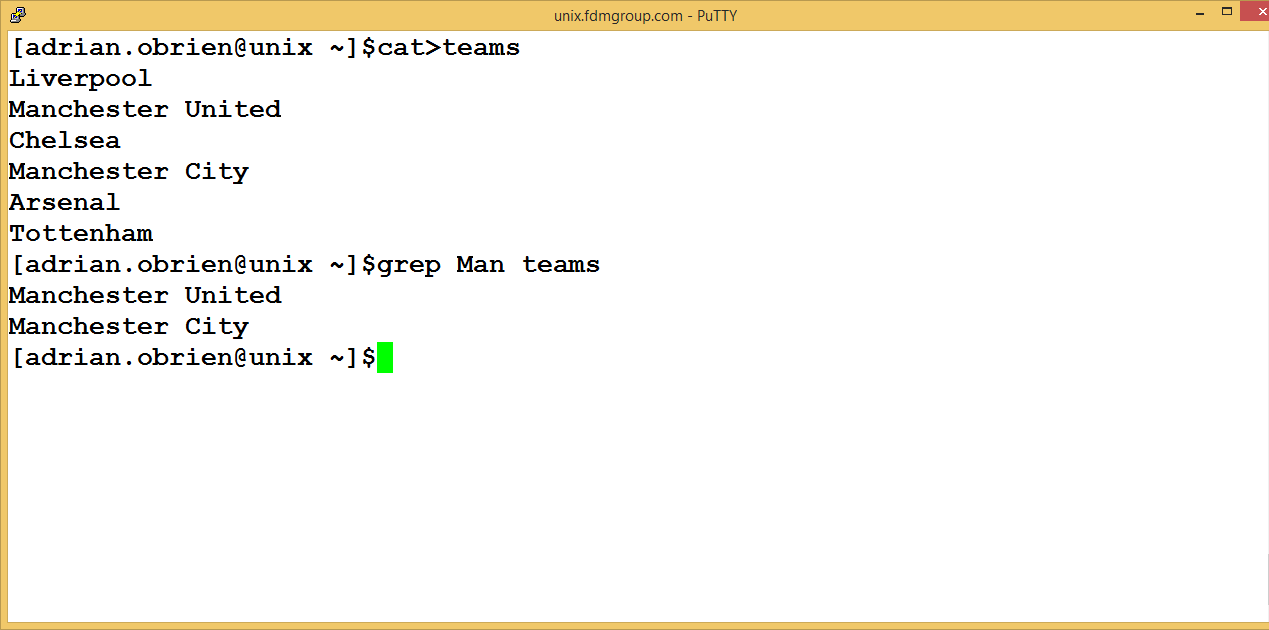
### grep

this command allows you to check if a file (or files) contain a string of text

**grep "searchstring"filename** looks for the string in a file.

**more filename | grep "searchstring"**can also be used.

In the example below we have created a file which contains some teams. Then we search for the word "Man" in this file. This produces two lines in the resulting output:



Some useful options with the grep command include:

|  |  |
| --- | --- |
| Option | Description |
| -i | Case insensitive search |
| -c | Prints a count of lines that match |
| -n | Prints the line numbers |
| -l | Prints the filenames in which the lines match |

### cut

this command is used for selecting a number of columns from a file (or the output from another command). You can specify column numbers or choose delimiters (column separators) and choose column numbers based on the delimiters.

For instance, the following code will display the first 4 columns of the file called filename. This is because we have used the –c option and requested column numbers 1-4.

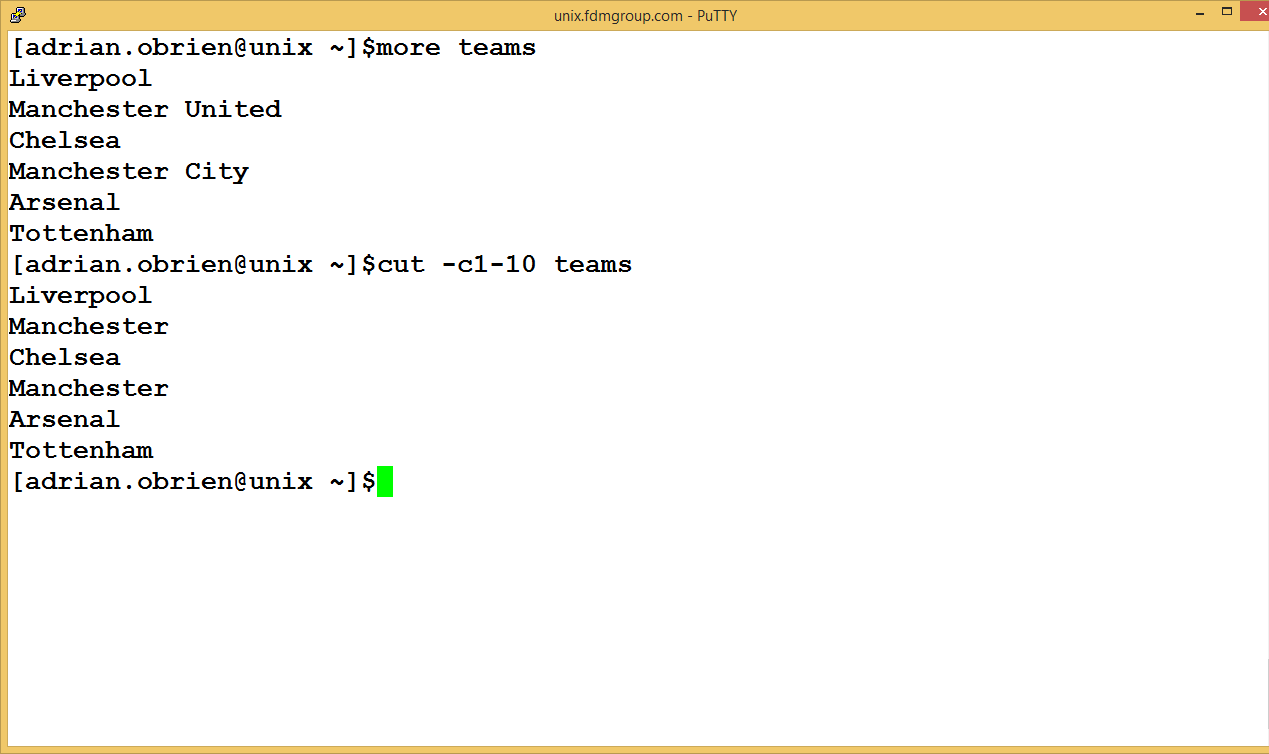
The next example shows the second column of the file (because we have used "-f2") when the: character is used as the separator between the columns:

**cut –c1-4 filename**

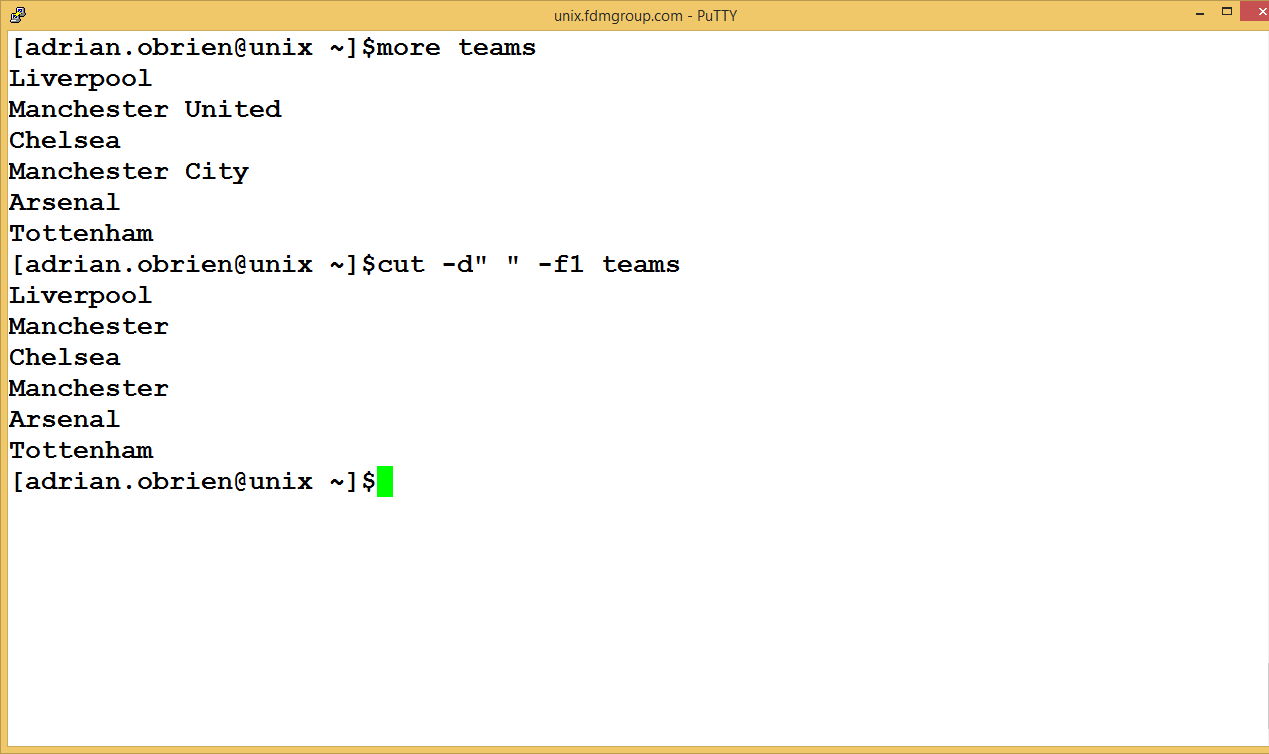
**cut -d":" -f2 filename**

In the following example, we have "cut" the first 10 columns of the file by using

"-c1-10" as our option in the cut command.



We could also have selected the first name of each team by "cutting" the first field (using the **space** character as the field separator and selecting field **1**) using the command cut -d**" "** -f**1:**



### tr

this allows you to search a file for a particular character and replace all the occurrences of this character with another character. For instance the following code will translate any ":" characters into "," characters.

**tr ":" "," filename**

date – this function displays the date:



echo – this function displays a message and/or a variable on screen

You can also create your own variables using the following code:

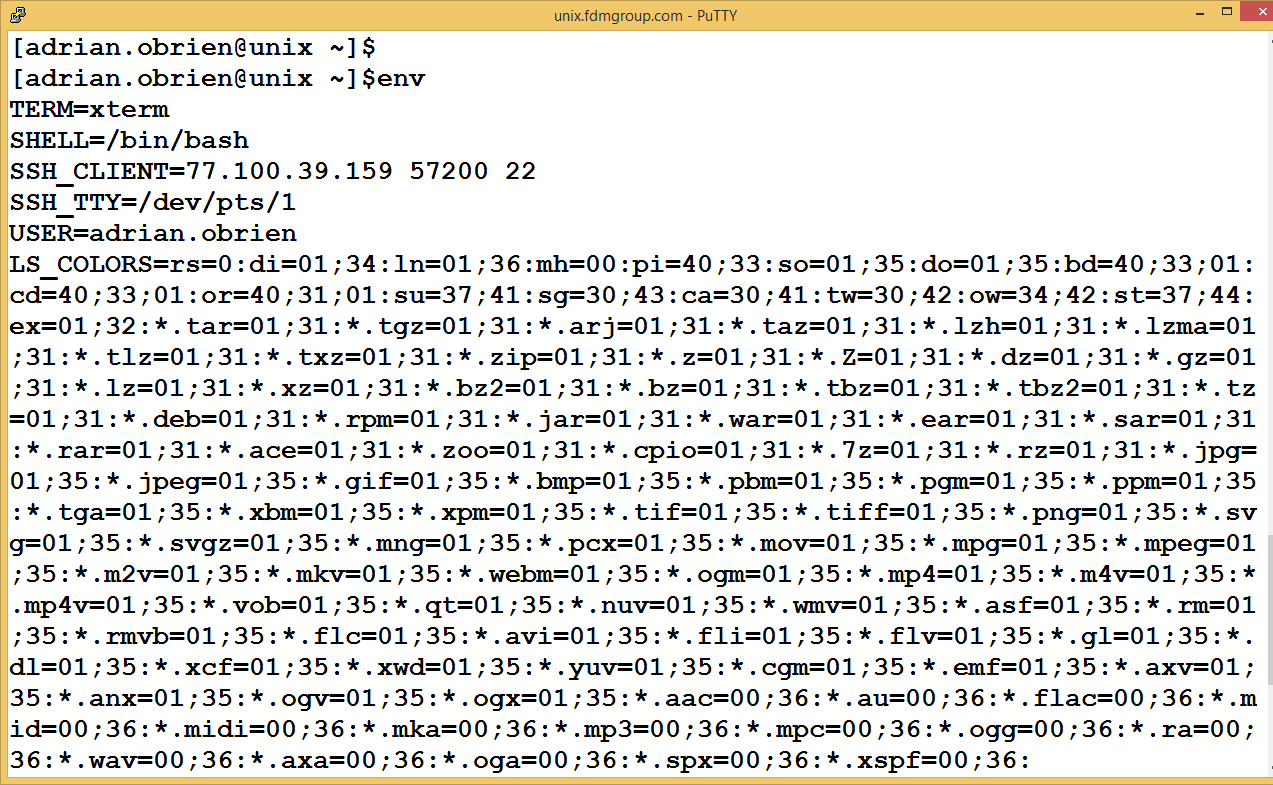
**myname="Adrian"**

You can write code which uses this variable:

**echo $myname** or

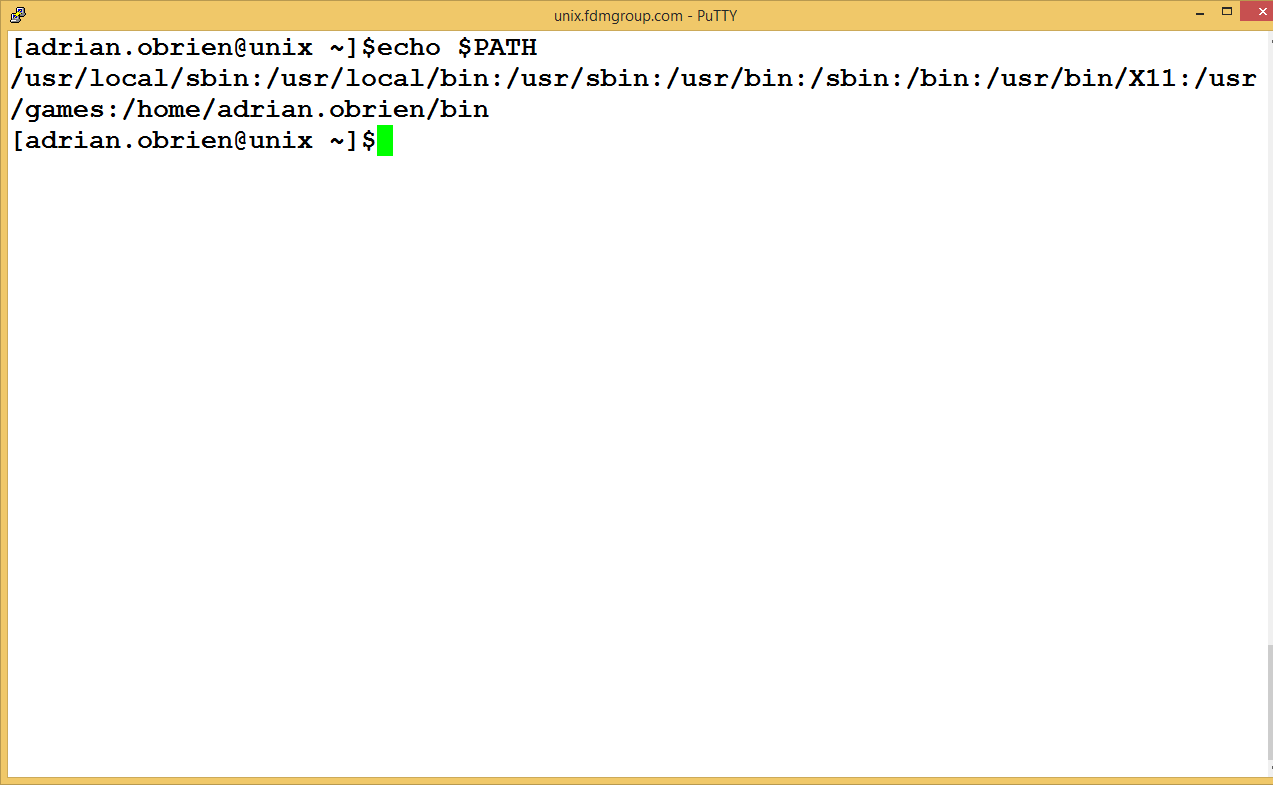
**echo ${myname}**

env - To list environmental variables in UNIX you can use the env command. Some of the output from this command is shown below:



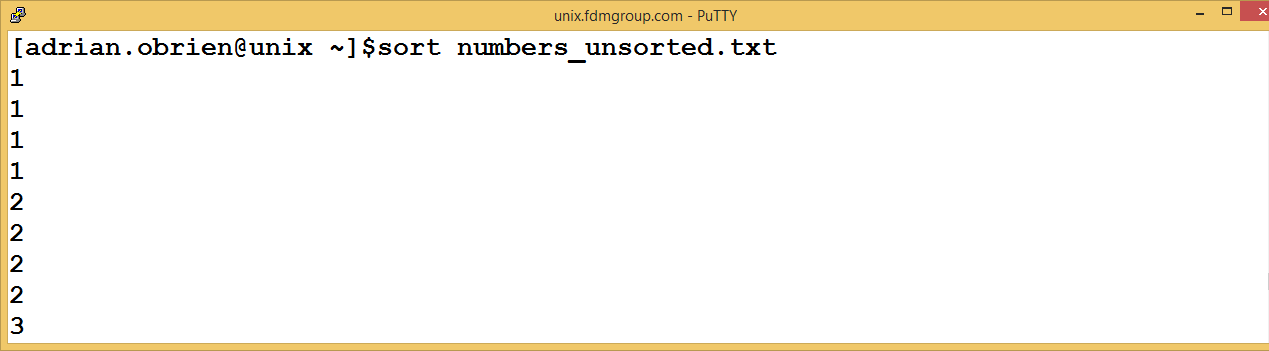
One frequently used environment variable is the PATH variable. If you want to display this variable you can use the command:

**echo $PATH**



rmdir – this command can be used to remove a directory

sort – this command sorts a file into alphabetical order. For example:



uniq - this command can be used to ensure there are no duplicates in a set of results

find - this is used to find a file(or files) which satisfy a condition

# 5. Streams: stdin, stdout, stderr - used very frequently

UNIX has a number of default settings for operating. These include accepting instructions from the keyboard or mouse and outputting results are error messages to the screen.

These are called stream and the 3 main streams in UNIX are:

1. The standard input stream
2. The standard output stream
3. The standard error stream

These streams can be redirected. For example results can be saved in a file by redirecting the standard output stream

**ls –l > dirlist**

further details of this later in section 10.

# 6. Information about other UNIX users -sometimes used

Sometimes you will need to find out about other UNIX users who have an account on the UNIX system. This tends to be a system administrator task.

w - this command shows who has logged in and what they're currently doing.

who - this command tells you who's logged on, and where they have logged in. This is a useful command if you need to know if the user is logging in remotely.

finger - there are two variations of this command

* finger username - provides information about that user specified, for instance when they last read their mail and whether they're logged in.
* finger returns information about all users

users - displays a list of the users currently logged in

whoami - returns your username.

# 7. Information about processes-often used

ps -u username - displays a list of processes being run by the user specified. Without a username, it will display your processes. The information includes the process ID, which you need if you have to kill a process.

kill PROCESSID - kills a process

jobs – this command is used to display the list of jobs

sleep - makes the terminal pause for a number of seconds

bg %JOBID - puts a job in the background

fg %JOBID - puts a job in the foreground

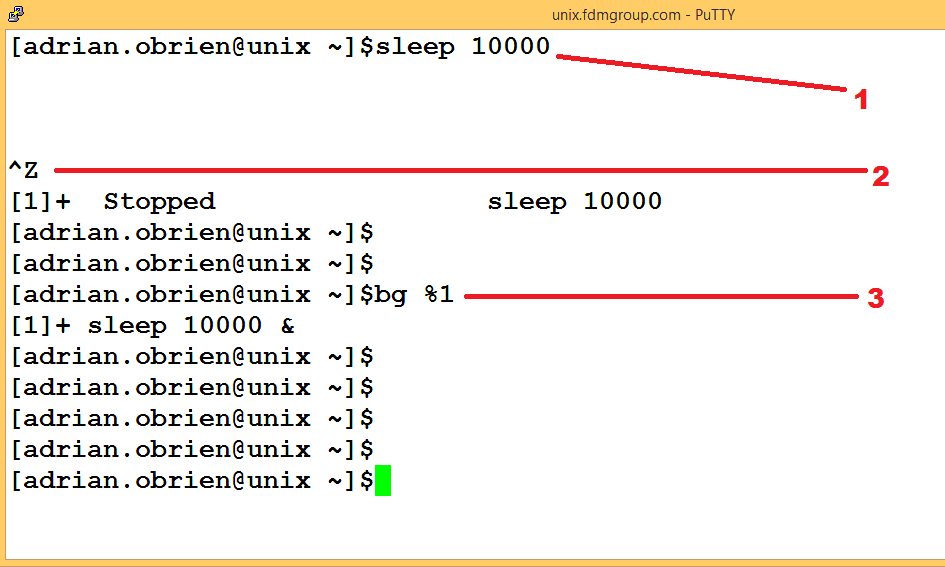
Here is an example of these commands being used together. In the following example we have:

1. started a process by typing the command: sleep **10000** (and pressing ENTER)

2. interrupt the process by pressing **CTRL+Z**

3. place the process in the background by using the command: **bg %1**

These are labelled in the diagram below:



These three steps have the effect of placing the sleep command in the background and this means that other commands can operate in the foreground if we wish.

If we wish to place the sleep process in the foreground again, we can do this by using the command

**fg %1**

and then place it in the background again using

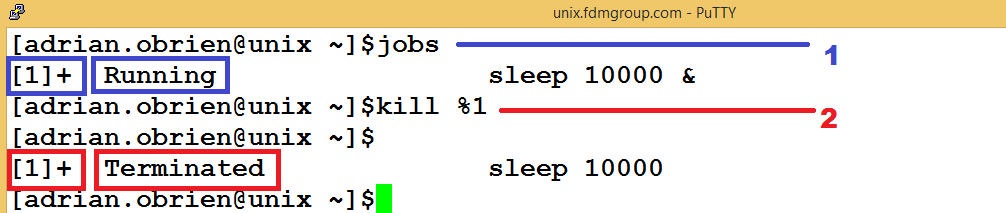
**bg %1**

If the job is in the background we can show this by listing the jobs and this is done using the jobs command.

In the following example, we list the jobs and then terminate the process using the kill command:

**1. list the jobs**

**2. kill the background job using the kill command and the job id.**



basename - displays the filename of a file

dirname -displays the directory in which a file is stored

quota -v shows the following information:

* your disk quota (how much space you have left in your account to store files)
* how much you're actually using

du filename --- shows the disk usage of the files and directories in *filename* (without argument the current directory is used). **du -s** gives only a total.

tput - this command can be used to change the appearance of the screen

# 8. Common errors

When you use first use UNIX, try to execute some commands or write some shell scripts, you will encounter errors. The following section discusses some of the more common errors. This is normal in a subject where every character matters and a space character in the wrong place can make a drastic difference.

**Error**: You try to use a file on the command prompt but it does not work.

**Solution**: You may be in a different directory to the one which contains the file you wish to use. Use the command **ls -al** to list the files to see if the file is present. Use the command cd to move to the correct directory. You can also use the command **pwd** to find your current location and move to the correct location. The UNIX prompt also shows your current location.

**Error**: You have tried to run the mv command but it does not work.

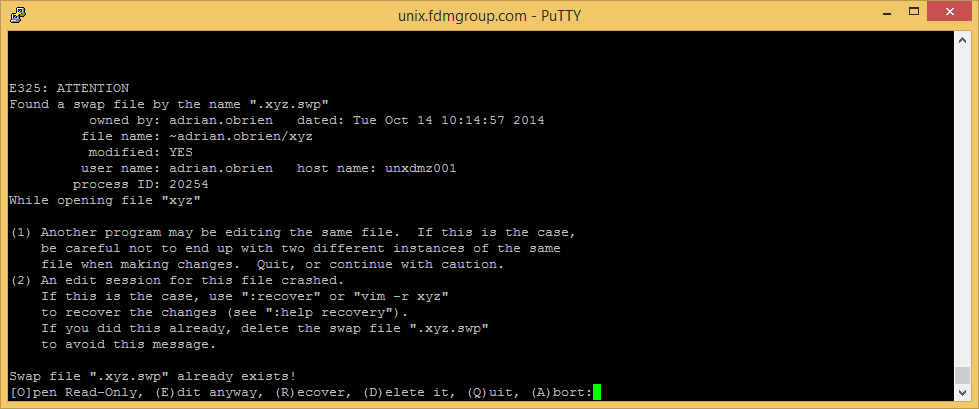
**Solutions**:

* You may have tried to move the file to a non-existent location
* The filename you supplied may not be valid (or it may be the name of an existing file, in which case you have just overwritten its original contents)
* You may have tried to use the **mv** command without supplying the destination

**Error**: You have tried to write a UNIX command but the terminal has "hung".

**Solution**: Press CTRL+C to stop the command

**Error**: You will often use vi to edit a file. If you lose your connection to UNIX after a period of inactivity then you sometimes encounter the following error: Abort/Retry/Ignore



**Solution**: Press A to abort. Then list the contents of the directory using the command:

**ls -al**

Locate the file which has a .SWP file extension. In the example above we were editing a file called myfile, so the file we need to remove is: **.myfile.swp.** The filename will start with a .(dot) as it is a hidden file.

**IMPORTANT: Be careful not to remove the original file which contains your code!!!**

**You may wish to type the following command to make sure you are removing the correct file(the –i option denotes an “interactive removal”):**

**rm –i .myfile.swp**

# 9. Mathematical operators

There are two formats for arithmetic expressions:

$[ expression ] and $(( expression ))

In the code example below, we use the following mathematical operators:

|  |  |
| --- | --- |
| Symbol | Meaning |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division(Integer division) |
| % | Modulus(calculates the remainder when dividing the numbers) |
| \*\* | Exponentiation(Power of)  Raise the first number to the power of the second number |

**echo 1 + 2 = $((1 + 2))**

**echo 6 - 6 = $[ 6 - 6 ]**

**echo 2 x 3 = $((2 \* 3))**

**echo 5 / 3 = $((5 / 3))**

**echo 7 % 6 = $((7 % 6))**

**echo 3 ^ 4 = $[ 3 \*\* 4 ]**

We can also specify numbers in a base numbering system other than base 10. The bases we will look at are:

* base 2(binary)
* base 8(octal)
* base 16(hexadecimal)

# convert binary number 1001

**let binResult=2#1001**

**echo $binResult**

# convert octal number 14

**let octResult=8#14**

**echo $octResult**

# convert hex number 0xFF

**let hexResult=16#FF**

**echo $hexResult**

# 10. redirection and piping

There are a number of other operators in UNIX. They are known as piping/redirection:

a. Redirection

We can send the results of a command to a file.

In the example below, the results of the ls command are redirected to a file. Note that nothing is shown on screen. All the output is sent to the file.

**ls >filename**

b. Redirection(append)

In this example, we are sending the results of the pwd command to a file. This is similar to the previous example, with the difference being that we are appending the output to the file.

**pwd >>filename**

c. Redirecting the error stream to a file

In this example we try to execute the (non-existent) command abc. As the command doesn’t exist, it will produce an error. Normally the errors will be sent to the screen but in this example, the errors are sent to a file. You can open the file and see the error message.

**abc 2>errors**

d. Redirecting the error stream to the standard output

In this example we try to execute the (non-existent) command abc. The difference in this example is that we are redirecting the error stream to the standard output. This is done by using the syntax: 2>&1

**abc 2>&1**

e. Piping:

In this example, we are grouping together commands and using the pipe symbol, | , to use the output from one command as the input for another.

**ls | grep ade | grep xyz | tr “:” “,” > OUTPUT.txt**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** |

Lets look at each step in turn:

1. The ls command is executed. This lists the directory contents. The output of the ls command is used for the grep command.
2. The grep command searches for the string and will produce a set of results.
3. The grep command searches for the string and will produce a set of results.
4. From the results of the previous command, the tr command converts any occurrences of ":" to ","
5. Finally the results of the tr command are redirected to a file called OUTPUT.txt

f. piping and tee

This command is the same as the previous example but we have used the tee command to create a file.

ls| grep ade| tee results

Note that when you use the tee command the result from the previous command (when using piping) is redirected to the file as well as the screen.

g. Input from a file

In this example we use a grep statement to search for a string in a file. The file we are using in this example is the standard input stream, a file called inputfile.txt

**grep ade < inputfile.txt**

# 11. DAYS 1-2: Table of commands used

During Days 1 and 2 we will cover the following commands:

|  |  |
| --- | --- |
| Command/topic | definition |
| **DAY 1** |  |
| **touch** | Create a new file |
| **ls** | Directory listing |
| **rm –r** | Remove directories |
| **mkdir** | Create a directory |
| **rmdir** | Remove a directory (empty) |
| **man** | Online manual |
| **cp** | Copy a file |
| **cd** | Change directory |
| **pwd** | display the current directory |
| **cat** | Display a file |
| **mv** | Move a file |
| **chmod** | Changes permissions  Remember the rules for chmod:  **u:**  **g: o:**  rwx rwx rwx  r=4, r=2, x=1  chmod 700: rwx for user only rwx --- ---  chmod 070: rwx for group only --- rwx ---  chmod 007: rwx for other only --- --- rwx |
| **wc –l**  **wc -w**  **wc -c** | Word count  -l: lines  -w: words  -c: characters |
| **more** | Displays a file |
| **less** | Displays a file |
| **head -n** | Display the first n lines of a file |
| **tail -n** | Display the first n lines of a file |
| **clear** | Clears the screen |
| **grep** | Looks for text in a file |
| **find** | Look for a particular file which is ……. |
| Wildcards  \*,?, [] | Used to enable a command to be performed on many files at the same time |
| Redirection/piping  >, >>, 2>  |, tee |  |
| Regular Expressions  ^, $, ., [], \*, \* | Used to define patterns, used by the grep command |
| **DAY 2** |  |
| **ps, ps –e** | List the processes |
| **jobs** | List the jobs |
| **sleep** | Sleep for a number of seconds |
| **bg %JOBID** | Run a job in the background |
| **fg %JOBID** | Run a job in the foreground |
| **kill PROCESSID** | Kill a process |
| **who**  **which**  **finger**  **users**  **sort**  **uniq**  **basename**  **dirname**  **id**  **du**  **hostname**  **tput**  **uname**  **echo**  **comm**  **cmp**  **tr**  **cut**  **find** | During the second morning, there is an exercise where students will choose a command from this list and use the UNIX man command to find out as much about the command as possible.  Then they will present their findings to the other students in the class. |

# 12. Other useful commands

|  |  |
| --- | --- |
| Other commands/tips |  |
| history | If you use the history command it will list the commands you have used in your UNIX session previously. If you wish to use a previously used command from your UNIX session you can use the up arrow to locate previous commands. Simply use the up and down arrows until you find the command you need. |
| echo | Prints to the screen |
| {1..100} expansion | Expands a range  (in this example 1 to 100) |
| for  while  case | Loop  Loop  Case statement |
| read | Read from the keyboard |
| alias | An alias can be used to create alternative, shortened names for commands. If you use a command (or a series of commands on one line) frequently then you can use an alias to represent this command(s). This can be used to save you time. For example:  alias ll='ls -al'  alias myalias='grep 'abc' filename | cut -c1-4' |
| vi common shortcuts:  ESC  :wq  :q  i | Command mode  Save and quit  Quit  Insert mode |

# Using vi

When you write shell scripts using vi remember the following:

1. shortcut keys in vi:

ESC command mode

i insert mode

:w save

:q quit

: wq save and quit

u (command mode): undo last action

1. To display line numbers in vi create a file in your home directory called **.vimrc** with the following content:

**set nu**

**nohl**

Save this file. You will notice that files opened in vi will now display line numbers.

1. When using vi you may find it easier to open two UNIX windows, one to edit the script in vi and another to run your script from the command prompt.

# Exam and project

Key points for the project exam later in the week

**1. Project:**

The project is open book.

You can start planning your project from Tuesday onwards and start writing the code for your project on Wednesday evening. You will be working on your project all day on Thursday in the classroom. The project must be YOUR OWN WORK.

The project must be submitted by Friday 12 noon.

You do this by putting your code in the ~/project directory. You can do this by using the command: **cd ~/project** and making sure that the code is in that directory

**2. Exam:**

The exam is a 1 hour paper-based exam and is closed book. The exam will ask a series of questions related to UNIX commands. You will type your answers in a UNIX window (to get the results of the UNIX command) and then write down the UNIX command on the answer paper.

The exam contains 30 questions.

**3. Backups**

Keep regular backups using this command:

**cp OriginalFile BackUpFile**

If you delete your project accidentally then the backup can be used and your project can be restored.

**4. General Coding Tips**

Use the command "ls -al" to show a complete file listing with hidden file sizes

Assume nothing-check and test everything. Write "defensive" code-it is safer.

With vi, make sure you know what these do:

ESC :wq Save and exit

ESC :q! Quit without saving

ESC :w Save

ESC u Undo last action

There are many guides on UNIX. In this course we will be using the BASH shell and one particular guide which covers the BASH shell is available on the following site:

<http://www.tldp.org/LDP/abs/abs-guide.pdf>