9TH DECEMBER 2013

INTRODUCTION TO UNIX

LOGIN INSTRUCTIONS

Start SSH Secure Shell

Click "Quick Connect"

Use the following details to LOGIN

Groups 1,2,3,4 are on the server "heron":

```
    hostname : 10.128.0.123
    port     : 22
    username : guest
    password : 12345
```

Home Folders:

```
    /home/guest/group1
    /home/guest/group2
    /home/guest/group3
    /home/guest/group4
```

Groups 5,6,7,8 are on the server "flamingo":

```
    hostname : 10.128.0.14
    port     : 22
    username : guest
    password : 12345
```

Home Folders:

```
    /home/guest/group5
    /home/guest/group6
    /home/guest/group7
    /home/guest/group8
```

EXERCISE 1: Basic navigation:

Once you are logged in, you will see the prompt

```
guest@heron:~$
or
guest@flamingo:~$
```

This is where you will type all your commands.

Type the following command "pwd" (means print working directory)

```
pwd
```

```
/home/guest
```

this is the home directory or home folder of the user "guest".
In unix folder=directory. They are the same thing.

Now type "ls" (show the list of files in the current folder/directory)

```
ls
```

```
group1 group2 group3 group4
```

If your group is group1, run the cd (change directory)command

```
cd group1
```

and type pwd (print working directory)

```
pwd
```

```
/home/guest/group1
```

This is the working folder of your group.

now you are going to make a folder/directory in which you are going to work.
This is done using the command "mkdir"

```
mkdir yourname
```

Type "ls" to check if you see the new folder , and then you can type

```
cd yourname
```

followed by

```
pwd
```

which should give you this

```
/home/guest/group1/yourname
```

This folder is the folder you are going to work in. All the files or folders you create/delete should be within this folder. Make sure you are inside this folder (and not in somebody else's folder when you run any commands).

```
ls
```

should not give any results. We haven't created any files/folders inside this folder yet.

Going back to a higher level folder:

```
pwd
```

```
cd ..
```

```
pwd
```

This should take you to one folder above
/home/guest/group1

```
cd yourname
```

will take you back to your folder /home/guest/group1/yourname

```
cd
```

```
pwd
```

should give
/home/guest/

You can type "cd" from anywhere and it will return you to the /home/guest folder.

```
cd /home/guest/group1/yourname
```

will take you directly to your folder.

Once you are back in your own folder, i.e. pwd gives you
/home/guest/group1/yourname

make a new folder

```
mkdir testfolder
ls -l
```

will give something like this..

```
total 4
drwxr-xr-x 2 guest guest 4096 2013-12-02 20:24 testfolder
```

removing a folder

```
rmdir testfolder
```

```
ls -l
```

will give you this

```
total 0
```

Brief summary:

```
cd        goto the home directory
cd ..     goto the directory above
cd ../..   go two directories above
pwd     print working directory
ls      list files and folders
mkdir   make a directory
rmdir   remove a directory
```

Variations for ls command (modifying behaviour using additional options)

```
ls -l long listing
ls -h print file sizes in reasonable units, not just bytes
ls -a list hidden files (beginning with .)
ls -R list directories recursively
ls -1 one item per line format
```

Try them all out

You can combine options e.g.

```
ls -lR
ls -ltrh
```

Getting help from the system, e.g. to find out more about the ls command and all its options, type this to access the complete manual

```
man ls
```

similarly, for the pwd command

```
man pwd
```

or

```
man mkdir
```

You can do this for any command.

even

```
man man
```

**** Try these commands :) ****

```
cal  (prints a calendar of the current month on the screen)
date (prints the current date and time)
clear (clears the screen)
```

Best practices:

* *Do not include spaces in names of folders or files*. If you want to make a folder with multiple words, use a hyphen or an underscore as a word separator

e.g

```
mkdir my_folder
```

or

```
mkdir my-folder
```

and ***not***

```
mkdir my folder
```

*** Unix commands are Case-Sensitive**,

`cd` is not the same as `CD` or `Cd`
similarly, if your foldername is "myfolder"
`cd myfolder` is correct
and
`cd MyFolder` is incorrect

Copying files:

Here is the basic structure of the command:

```
cp  source_file  destination_file
```

Let us copy files from your group's data folder (which is located in /home/guest/group1/data/day1)

```
cp /home/guest/data/day1/pch.txt .
```

The “.” is a special way to say that copy these files to the current directory. You could also give a more complete path, e.g.

```
cp /home/guest/data/day1/pch.txt /home/group1/
```

Run some more of these commands to get more files.

```
cp /home/guest/data/day1/pch.faa .
cp /home/guest/data/day1/pch.gbk .
cp /home/guest/data/day1/pch.ffn .
```

Let’s see what we have now in our folder

```
ls -l
```

```
total 6284
-rw-r--r-- 1 guest guest  578007 2013-12-02 20:53 pch.faa
-rw-r--r-- 1 guest guest 1541770 2013-12-02 20:53 pch.ffn
-rw-r--r-- 1 guest guest 4206425 2013-12-02 20:53 pch.gbk
-rw-r--r-- 1 guest guest  101263 2013-12-02 20:53 pch.txt
```

Description of these files

pch.gbk	: genbank file of the complete genome of Prochlorococcus marinus
pch.faa	: fasta file of all proteins
pch.txt	: tab-delimited tabular text file of genes, locations, annotations
pch.ffn	: fasta file containing the gene sequences of all proteins

File extensions in unix e.g. .gbk, .faa , .txt in these examples are normally only meant to indicate what kind of data is contained within the files and are a useful tool to identify what might be inside a
file without opening it.

to make a new copy of the genbank file, do this

```
cp pch.gbk copy1.gbk
cp pch.gbk copy2.gbk
cp pch.gbk copy3.gbk
```

deleting a file: use the “rm” command.

```
rm copy1.gbk
```

to remove several files, you could give multiple file names, e.g.

```
rm copy2.gbk copy3.gbk
```

or you could use the "*" character again, e.g. like this…

```
rm copy*.gbk
```

Here the "*" character matches all files that start with the word "copy" followed by anything in between, and with a ".gbk" at the end.

You can test the use of the "*" in the ls command.

e.g. try the following commands and see what happens

```
ls *.gbk
```

```
ls p*
```

```
ls *.txt
```

```
ls -l *.txt
```

```
ls -lh *.gbk
```

```
ls -lh p*
```

Copying multiple files at the same time:

```
cp /home/guest/data/day1/* .
```

"*" is a special character in unix that is used for pattern matching. By default, "*" matches everything. The command above essentially asks that copy "all" files in the folder /home/guest/data/day1/ to the current directory.

This will copy all files that you need for your exercise on day1 to your folder.

Let's take a look at these files

```
ls
```

END OF EXERCISE 1

EXERCISE 2: Working with files

What's inside a file? Here is how to checking contents of a file:

The "more" command

```
more pch.gbk
```

this will show the first page of the file, use the spacebar to move forward page by page and use Enter to move forward line by line. Press "q" to stop viewing.

now you can see what's inside all the other files as well,

```
more pch.faa
more pch.txt
more pch.ffn
```

The "head" command

The head command lets you take a look at the lines at the beginning of a file, e.g.

```
head pch.gbk
```

will show you by default the top 10 lines in the file

you can also ask it to show any number of lines in your file, e.g.

```
head -20 pch.gbk
```

this will show you the top 20 lines
or

```
head -32 pch.gbk
```

this will show you the top 32 lines

```
head pch*
```

will show you the top 10 lines of all files whose names start with "pch"

The "tail" command

Similar to the head command, there is a tail command, which shows you the last lines of a file. Try the following commands

```
tail pch.gbk
tail -20 pch.gbk
tail -35 pch.gbk
tail pch*
```

The "wc" command

wc means "word count"

do

```
man wc
```

to read what the command does

try these

```
wc pch.gbk
   66394   259063  4206425 pch.gbk
```

The three numbers are the number of lines, number of words and the number of bytes in the file pch.gbk

You can ask wc to provide you only with the number of lines, like this

```
wc -l pch.gbk
```

try this

```
wc -l pch.txt
```

this gives you the number of rows in the tab-delimited file pch.txt

you can also do this

```
wc -l pch*
```

and it will provide the number of lines in all files that begin with a “pch” and also a sum of all these numbers.

The “cat” command

The cat command, given a file name, throws out the contents of the file on your screen. How this is useful will become apparent but try these commands

```
cat pch.faa
cat pch.ffn
cat pch.gbk
cat pch.txt
```

you can give multiple files to cat

```
cat pch*
cat *
```

and watch as the data from all the files pours onto the screen.

Saving the results from the output of a command

For example, you ran the command

```
wc -l pch.txt
1758 pch.txt
```

and you wou

```
wc -l pch.txt > result.txt
```

where the “>” character is used to dump what was supposed to be on the screen not onto the screen but into a file which we want to call result.txt.
ld like to save these results to an output file.

Here is how..

Try a few more examples…

```
ls -1 > list_of_files.txt
cat pch.txt > copy_of_pch.txt
head -9 pch.faa > oneseq.fasta
tail -20 pch.txt > last.txt
```

EXERCISE 3:

The “cut” command

We are going to use the Groups.txt file to show the use of the cut command. Take a look at this file

```
more Groups.txt
```

This file contains 4 columns, group number, first name, last name and country of all the participants in this course.

The cut command is used to cut columns from a file. You can cut any column from the file provided you know the column number of the column you want to cut.In the Groups.txt file, here are the column numbers

Column 1: group
Column 2: first_name
Column 3: last_name
Column 4: country

These columns are separated by a tab character, which is invisible. (remember that the tab character is not the same as the space character)

Try this, cutting the first column from the file

```
cut –f1 Groups.txt
```

this should print out on the screen the first column of the file, now try one by one all the columns.

```
cut –f2 Groups.txt
cut –f3 Groups.txt
cut –f4 Groups.txt
```

You can also cut two columns, like this

```
cut –f1,4 Groups.txt
```

this should print the first column (group) and the last column (country) on the screen

you can also specify a range of contiguous columns, e.g.

```
cut –f1-3 Groups.txt
```

this will print the first 3 columns

you can combine ranges with commas, like this

```
cut –f1,3-4 Groups.txt
```

this is the same as

```
cut –f1,3,4 Groups.txt
```

You can of course save the results using the ">" as described above, e.g.

```
cut –f1,4 > group_and_country.txt
```

and check what's in the file by using

```
more group_and_country.txt
```

or

```
cat group_and_country.txt
```

What if your data file is not tab-delimited by comma-delimited ? Take a look at the Groups.csv file (csv is a common extension used to identity files that contain the "," character as a separator between different columns.

```
more Groups.csv
```

You will see that the data in this file is identical to that in Groups.txt with the difference that you can see the different columns separated by a ",". With such a file, which has a column separator other than a tab, you have to tell the cut command what is character in the file, using the –d option (-d for delimiter)

```
cut –f1 –d',' Groups.csv
```

this should give the same result as

```
cut –f1 Groups.txt
```

<u>The "sort" command</u>

The "sort" command, as the name indicates is used to sort data. We will again use the Groups.txt file to explain the sort command.

Sorting a file based on a particular column

```
sort –k1 Groups.txt
sort –k2 Groups.txt
sort -k3 Groups.txt
sort -k4 Groups.txt
```

each of these commands will sort the Groups.txt by the first, second, third and fourth column respectively. Note that the column number is specified using -k option.

Similar to the cut command, sort by default considers that your file is tab-delimited. What about comma separated data (as in the file Groups.csv) ? Once again, you need to tell what is the character in the file that's used as a delimiter. For the sort command, this is done using the option -t. Try this for the Groups.csv file

```
sort -k1 -t',' Groups.csv
sort -k2 -t',' Groups.csv
sort -k3 -t',' Groups.csv
sort -k4 -t',' Groups.csv
```

However, all these sorts are alphabetical sorts, how do you sort numerically?

We will use the file Numbers.txt for this example (because the data file Groups.txt is already sorted numerically by group number)

Take a look at the file that contains some numbers.

```
more Numbers.txt
```

This will print 20 random numbers on your screen. You can see they are unsorted.

Try this first

```
sort Numbers.txt
```

(this is the same as `sort -k1 Numbers.txt`)

You will see that the result is sorted alphabetically. That's not what we want. We want to sort these numbers in ascending order. This is how to do it.

```
sort -n Numbers.txt
```
or
```
sort -n -k1 Numbers.txt
```

The -n option tells sort to treat the data as numbers. You see that the numbers are now sorted as we wanted.

What about sorting these numbers in descending order ? Use the -r (reverse) option

```
sort -r -n -k1 Numbers.txt
```

and you should get the numbers sorted with the largest number on top and smallest at the bottom.

Instead of typing -r,-n,-k1 separately, the sort command also allows you to say this more succinctly

```
sort -rnk1 Numbers.txt
```

Let's try the reverse sorting once again, but using the Groups.txt file

```
sort -r -n -k1 Groups.txt
```
(remember column on has numbers so we use -n)

Reverse alphabetical sorting by the second/third/fourths columns

```
sort -r -k2 Groups.txt
sort -r -k3 Groups.txt
sort -r -k4 Groups.txt
```

The "grep" command

The grep command is used to search text files matching a particular pattern. For example, we would like to know how many people from Netherlands are listed in the file Groups.txt

```
grep "Netherlands" Groups.txt
```

It should show you all the rows in the file Groups.txt that match the pattern "Netherlands".

What happens if you typed "netherlands" instead of "Netherlands"

Try it,

```
grep "netherlands" Groups.txt
```

You don't get any results at all. Why ? because unix is Case-Sensitive as explained before. "netherlands" is not the same as "Netherlands". However, if you would like to tell grep that I don't care whether its Netherlands or Netherlands, match either of them or even NeTHerLaNds, you can give the -i option, which means do a case-insensitive match. Now, try this

```
grep -i "netherlands" Groups.txt
```

This should give you the same result as

```
grep "Netherlands" Groups.txt
```

Try this

```
grep -i "NeTHerLaNds" Groups.txt
```

What if you wanted to only count how many lines match your pattern instead of looking at the results themselves ? Use the –c option

```
grep -i “netherlands” -c Groups.txt
```

This should print a number, 4 in this case which is the number of lines that match the pattern given by use.

What if you want to match 2 patterns ? e.g. Netherlands or France. Try the –E option which tells grep that we are going to provide whats called a regular expression. We specify this either or choice by using the “pipe” character “|”. Below “netherlands|france” essentially means “netherlands OR france”. The –E tells grep to interpret our pattern as a a “regular expression” and not just a simple text word.

```
grep -i -E “netherlands|france” Groups.txt
```

We can still use the –c option to count how many lines matched the pattern “Netherlands OR france”

```
grep -c -i -E “netherlands|france” Groups.txt
```

What if you want to know what are the lines right above and below the line that matched the pattern?

Try this first

```
grep “Morocco” Groups.txt
```

This should print a single line. Now try the following

```
grep -A 2 -B 2 “Morocco” Groups.txt
```

The –A 2 and –B 2 tell grep to also print 2 lines AFTER (-A) and 2 lines BEFORE (-B) the line to which the Morocco pattern matched.

How to print lines that “do not” match the pattern given?

```
grep “Netherlands” Groups.txt
```

this we know prints all lines matching the pattern Netherlands. But now if you use the –v option, this prints all lines that do not match the pattern “Netherlands”

```
grep -v “Netherlands” Groups.txt
```

you can of course use regular expressions like this

```
grep -v -E “Netherlands|Denmark” Groups.txt
```

or longer expressions

```
grep -v -E -I “Netherlands|Denmark|germany” Groups.txt
```

We can now use grep to count how many proteins are present in the fasta file pch.faa

Take a look at the file pch.faa first

```
more pch.faa
```

Note that in a fasta file, each record begins with a ">" character. This is well defined. So we can use this to count how many sequences are present in the file. Try this

```
grep –c ">" pch.faa
```

we can count how many hypothetical proteins are present in the file as well,

```
grep –c –i "hypothetical" pch.faa
```

we get the same answer by running this on the file pch.txt that contains all the annotations in a tabular format

```
grep –c –i "hypothetical" pch.faa
```

The "uniq" command

You have by now seen the kind of data that exists in the file Groups.txt. The fourth column is the country column. Let's cut the country column and save it to an output file called column4.txt.

```
cut –f4 Groups.txt > column4.txt
```

look at it..

```
more column4.txt
```

now we would like to count how many times does each country occur in this file.

Try this

```
sort column4.txt > sorted_column4.txt
```

```
more sorted_column4.txt
```

You see that the file still contains the header (look for the word "country"). We can remove it using grep –v, like this

```
grep -v country > only_countries.txt
```

Now we come to the uniq command which will do the magic for us

```
uniq only_countries.txt
```

this prints the name of each country only once (duplicates have been removed). But we still do not have the counts. Use the -c option now…

```
uniq -c only_countries.txt
```

this finally prints out on the screen the number of times each country occurs in the file. *Remember, whenever you have to run a uniq command, you will have to sort your data beforehand, so that duplicate lines occur together.*

Using Unix Pipes

Till now, we have only run a single unix command at a time. Like this…

```
cat Groups.txt
cut -f1 Groups.txt
sort -f1 Groups.txt
```

Now we will learn how we can combine unix commands together so that we can work more efficiently.

For example, we would like to obtain the first names of all the people in the Groups.txt that are from the country Spain.

How would we do it using the commands one at a time ? we might do something like this…

```
grep “Spain” Groups.txt > people_from_spain.txt
```

this should match all the lines in the file Groups.txt that contain the word Spain and write to an output file called people_from_spain.txt. Using the “>” character in this way is called “redirecting output”.

and then we could just cut the second column from the file, like this

```
cut -f2 people_from_spain.txt
```

this gives us the first names of all people in the file that are from Spain.

Using the “pipe” character “|”, we can skip over the creation of this intermediate file people_from_spain.txt altogether, like this…

```
grep “Spain” Groups.txt | cut -f2
```

Instead of directing the output of the grep command to a file, we send it to the cut command directly by inserting the pipe character in the middle. This achieves the same result, because the cut command is able to read the lines coming to it from the pipe one by one and process them as they are provided to it. Many unix commands are able to read data from pipes (and of course from files) . We will see more examples below.

We do not need to stop at combining two commands. Try this now…

```
grep “Spain” Groups.txt | cut -f2 | sort
```

Now we have managed to even sort the names alphabetically.

Let’s try the following now…

```
cut –f4 “Groups.txt” | sort | uniq –c
```

This should give us the counts of each country in the file Groups.txt

How many people are present in each group ? try this…

```
cut –f1 “Groups.txt” | sort | uniq –c
```

In which groups are people from Spain distributed?

```
grep “Spain” Groups.txt | cut –f1 | sort | uniq –c
```

Are there any people who have the same first name?

```
cut –f2 “Groups.txt” | sort | uniq –c
```

You can also use the cat and more commands in pipe, like this…

```
cat Groups.txt | grep Netherlands
```

```
more Groups.txt | grep France
```

```
cat Groups.txt | grep –E –i “netherlands|france|germany”
```

Let’s try some examples using the file pch.txt, which contains a table of the genes present in the genome of Prochlorococcus marinus. Look at the file carefully,

```
more pch.txt
```

There are 6 columns in the file. It is a tab-delimited file. The columns are as follows: gene identifier, feature type, start position in genome, end position in genome, strand (1 is + and -1 is -) and then the description of the gene.

How many types of features are present in the file?

```
cut –f2 pch.txt | sort | uniq –c
```

How many genes are present on each strand?

```
cut –f5 pch.txt | sort | uniq –c
```

How many genes are annotated with the word “transporter”?

```
grep –c –i “transporter” pch.txt
```

make a list of all genes that match the pattern “photo”

```
grep –i “photo” pch.txt
```

EXERCISE 4: Additional programs

We are now going to learn how to use a few more programs available to you. These programs are available in the folder /home/guest/bin.

The convertseq program

Biological sequence data can be in several different formats, fasta, genbank etc, and sometimes you need a simple method to convert from one to the other. The “convertseq” program provides the facility to do so.
For example, the file pch.gbk is a genbank file containing all the predicted genes and the complete genome sequence of the genome of Prochloroccocus marinus. Take a look…

```
more pch.gbk
```
or
```
cat pch.gbk
```

we are going to convert the genbank format file to a simple fasta file that contains only the genome sequences

```
cat pch.gbk | convertseq genbank fasta > pch.fna
```

here the word “genbank” indicates to the program that the file that is being supplied (through the pipe) is in genbank format, and the word “fasta” is what we would like to convert it to.

Let’s check the output

```
more pch.fna
```

In this example, the file pch.gbk only contains a single sequence. The convertseq program does not

We can of course convert fasta files to genbank format as well… like this..

```
cat pch.faa | convertseq fasta genbank > pch_proteins.gbk
```

To read the complete help of the convertseq program type this

```
convertseq -h
```

The “lenseq” program

The lenseq program provides a simple way to compute lengths of sequences. It can also deal with a number of different biological sequence formats.

Let us calculate the lengths of all the proteins in the fasta file pch.faa

```
cat pch.faa | lenseq fasta
```

this will print on the screen the results in 3 columns, where the first column is the identifier of the protein, the second is the length, and the third is the description.

Now we can use this to find the longest protein in the genome, like this

```
cat pch.faa | lenseq fasta | sort -n -k2
```

lenseq can also deal with genbank files, e.g.

```
cat pch.gbk | lenseq genbank
```

gives you the length of the genome.

To read the complete help of the lenseq program type this

```
lenseq -h
```

The “gcseq” program

The gcseq program provides a simple way to compute gc content and lengths of sequences. It can also deal with a number of different biological sequence formats.

Let’s find the gc content of the genome of Prochlorococcus marinus,

```
cat pch.gbk | gcseq genbank
```

output has 4 columns, genome identifier (accession number), gc content, length of the genome and the description of the genome

the file pch.ffn contains all the genes of the genome in fasta format. Take a look.

```
more pch.ffn
```

we can use gcseq on this entire fasta file, like this

```
cat pch.ffn | gcseq fasta
```

if we want we can of course save the results to a file

```
cat pch.ffn | gcseq fasta > pch.ffn.gcseq
```

To read the complete help of the gcseq program type this

```
gcseq -h
```

The “faslice” program

The faslice program provides a simple way to extract a few fasta sequences from a file. It works only with fasta files (not with genbank).

It we want only the first five sequences from a file, we can do this..

```
cat pch.faa | faslice 1 5 > first_5_seqs.faa
```

where “1 5” specifies that take all sequences beginning from the first one to the 5th one.

You can also get sequences from the middle of the file

```
cat pch.faa | faslice 30 40  > 30_40_seqs.fa
```

getting sequences from the end,

```
cat pch.faa | faslice 1000 -
```

will get sequences starting from 1000 and all those after it till the end of the file.

To read the complete help of the faslice program type this

```
faslice -h
```

Read the help of the following programs and learn to use them yourself:

lenfilter, gcfilter

use the following commands to get the help

```
lenfilter -h
gcfilter -h
```

Compressing and Decompressing files

Sequence files are usually provided to you as compressed files. Here are a few examples.

```
gzip pch.gbk
```

this will compress the file pch.gbk using “gzip”. The output file name will be pch.gbk.gz. Note that pch.gbk file now does not exist anymore, only its compressed version exists. Now we will uncompress this file using “gunzip”.

```
gunzip pch.gbk.gz
```

now the pch.gbk file is restored.

Files may also be compressed/uncompressed using the zip/unzip programs.
Try this.

```
zip pch.gbk
unzip pch.gbk.zip
```

Another option is bzip2/bunzip2

```
bzip2 pch.gbk
bunzip2 pch.gbk.bz2
```

Compressing complete folders/directories

The “tar” command can be used make a single file from a complete folder. This file usually has the extension .tar. This file in turn, may be compressed by using a compression program e.g gzip or bzip2. Try this

We will make a sample folder and copy some files to it.

```
mkdir testfolder
cp pch.* testfolder
```

now we run the tar command on the folder testfolder

```
tar -cvf testfolder.tar testfolder
```

here testfolder.tar is the outputfile name and after that we have provided the name of the folder “testfolder” that we want to “tar”.
You will see that a file called testfolder.tar appears in your current directory. (use “ls” to see). The original folder testfolder also exists.
Now we will compress this testfolder.tar file using gzip.

```
gzip testfolder.tar
```

now we have a compressed a complete folder into a single file (testfolder.tar.gz). This makes it easy to transfer lots of files from one place to another.

Now go inside the testfolder and delete all the files one by one.

```
cd testfolder
ls
rm pch*
cd ..
rmdir testfolder
```

Now we have deleted the original folder. Let’s restore the folder by using the testfolder.tar.gz file.Only a single command is required.

```
tar -xzvf testfolder.tar.gz
```

and now testfolder is back.
Go inside and check if the files are intact.