Exercises, Part 1

1. Command line and using the archive server

This exercise familiarizes you with copying and moving files with the command line, the tar and gzip commands and also using the archive server.

Log into **hippu.csc.fi**

Move into the **$METAWRK** folder

```
cd $METAWRK
```

First copy some files to play with:

```
cp /p/appl/molbio/test_data/haplo.* .
```

Check what you copied.

```
ls
```

You can get more information on the files with

```
ls -l
```

Make a new folder called "haplo_files"

```
mkdir haplo_files
```

Move the files into the new folder

```
mv haplo.* haplo_files
```

Check that the copying was succesful

```
ls haplo_files
```

We will now package the haplo_files folder into a single file called haplo_files.tar and then compress it using the gzip program.

```
tar cvf haplo_files.tar haplo_files
gzip haplo_files.tar
```

Alternatively you can do this with single command

```
tar czvf haplo_files.tar.gz haplo_files
```

We then copy the resulting file into archive server

```
cp haplo_files.tar.gz $ARCHIVE
```

You can check that the copying was successful with

```
ls $ARCHIVE
```

You can then delete the files from your working directory

```
rm haplo_files.tar.gz
rm -r haplo_files
```

By default the computer asks you to confirm each deletion. This might not be desirable especially when deleting many files. You can override this behavior with the switch "-f". Just remember to be careful, especially when using wild-cards.

```
rm -rf haplo_files
```

You can retrieve the archived files in the following manner. First copy the file back to you working directory

```
cp $ARCHIVE/haplo_files.tar.gz .
```

Then uncompress

```
gunzip haplo_files.tar.gz
```

And finally open the tar package

```
tar xvf haplo_files.tar
```

You can also uncompress and open with a single command

```
tar xzvf haplo_files.tar.gz
```

2. Running a serial batch job

For this exercise we need the files we used in exercise 1.

Log into **vuori.csc.fi**

Vuori computing nodes can only see the **$WRKDIR**, so we need to copy the files there

```
cd $WRKDIR
mkdir merlin
cd merlin
cp $METAWRK/haplo_files/* .
```

Write a serial batch script called "merlin_job" that uses one core, requires 1 GB memory and 15 minutes wall clock time. Also capture the stderr and stdout (options -o and -e).

Use the following command line for merlin:
`merlin -d haplo.dat -p haplo.ped -m haplo.map --best > merlin.out`

Remember to load the merlin module!

Submit the job

```
sbatch merlin_job
```

Check the status of the job

```
squeue
```

After the job has finished you check what resources it used with

```
sacct
```

To see details of a single job you can use

```
sacct –l –j <jobid>
```

sacct lists a rather long list of values, many of which are of interest mainly to sysadmins. To get a more manageable output you can use the –o option to define which fields you want to see.

```
sacct -o jobid,jobname,maxrss,maxvmsize,state,elapsed -j <jobid>
```

Here we have elected to show jobid (JobID), jobname (JobName), maximum used memory (MaxRSS), maximum used virtual memory (MaxVMSize), state of the job (State) and elapsed time (Elapsed). These can be helpful when we decide on the resource allocation parameters for our next similar job.

For details on the different available fields and other options see

```
man sacct
```

When the job has finished review the results. What files were created? What do they contain?

3. Running an array job

Make directory called "arraytest".

Copy file
/p/appl/molbio/test_data/ex4.tar.gz

to the arraytest folder and open the package as before.

Write a batch job script that runs an array job that runs the EMBOSS program getorf for each of the sequences. Reserve 1 core, 1 GB memory and 15 min wall clock time.

Command line for getorf is of the format:
`getorf <inputfile> <outputfile> -minsize 100`

Remember to load the emboss module.

Hint: There are many ways to do this exercise. One option is to make a list file of the input file names and then go through that

Making a list:
```
ls *.seq > list
```

Command lines for the batch script
```
set seq =  `sed -n "$VUORI_JOBINDEX"p list`
getorf $seq "$seq".out -minsize 100
```

`sed –n <line number>p` returns a single line from the input file. Here we use the $VUORI_JOBINDEX variable to go through all the lines one at a time. We use the sed command to set the variable $seq and the use the variable to set the input and output file name arguments for the program.

To submit the job we need to know the number of lines in "list". You can find this out with

```
wc –l list
```

or

```
sed -n '$=' list
```

You can also include these commands directly into the command line. (Note that the inner and outer quotes are different. See the lecture notes on using different quotes):

```
array_sbatch -file arrayjob -from 1 -to `sed -n '$=' list`
```

Exercises, part 2

1. Moving query sequences from your local machine to Hippu with Scientist's User Interface

The *MyFiles* tool in scientist's interface offers a handy way to transport files between CSC and your local computer. However, Firefox and Explorer browsers have a general limit of 2 GB for uploading files. For larger files you can use Chrome or Safari browsers. If this is not feasible Scientist's User Interface provides you another, a bit more complicated tool that you can use for data transport. That is: using *GSI SSH console* and *SFTP session*.

In this exercise we use SFTP just to see, how the system works. The file we are going to upload is so small that it could be easily moved to CSC by using the MyFiles tool.

But first Download the file (R.fasta) to your computer with you browser. You can find it from:

https://extras.csc.fi/biosciences/kurssit/R.fasta

Then do the transport with GIS SSH console:

A. Login to Scientist's User Interface (https://sui.csc.fi)

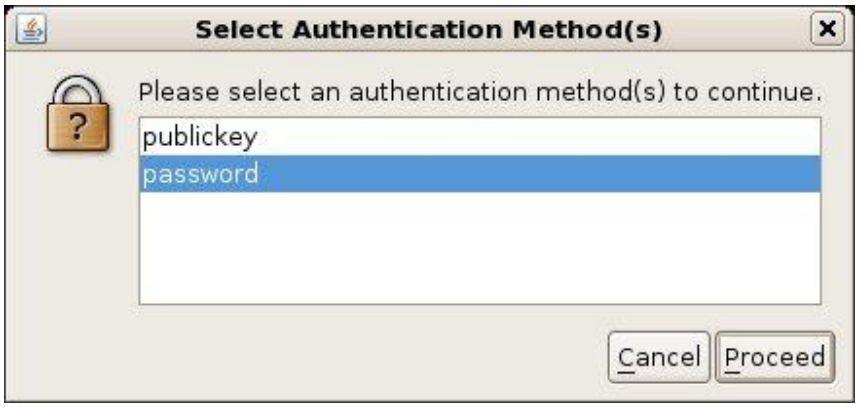
B. Launch the SSH Console by clicking the icon in the Services Desktop or by selecting the tool from the Services Menu.

C. When the console is launched, it asks for your user account and the name of the server you want to connect (the password will be asked only later on). As you wish to copy data for to Hippu, you should define:

Username: ***your_account***
Remote Host: **hippu.csc.fi**

And then launch the SSH console.

When the Java based console program starts you may need to accept launching the process in you local machine and allow the process connect CSC. When the console is running it will ask you to choose the authentication method to be used. Select: **password** and press **Proceed**.



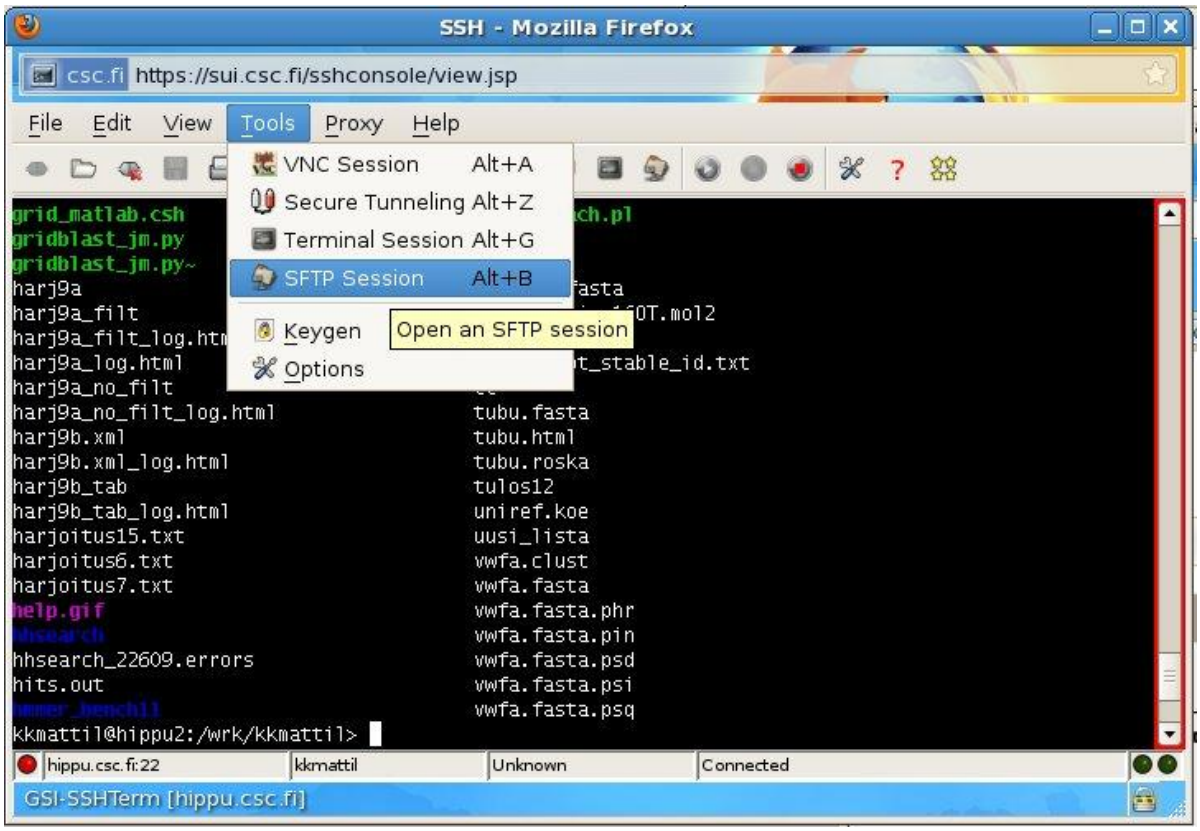
After this a Password Authentication window opens. Type in your password and press OK.

Now a ssh console window opens to your screen.

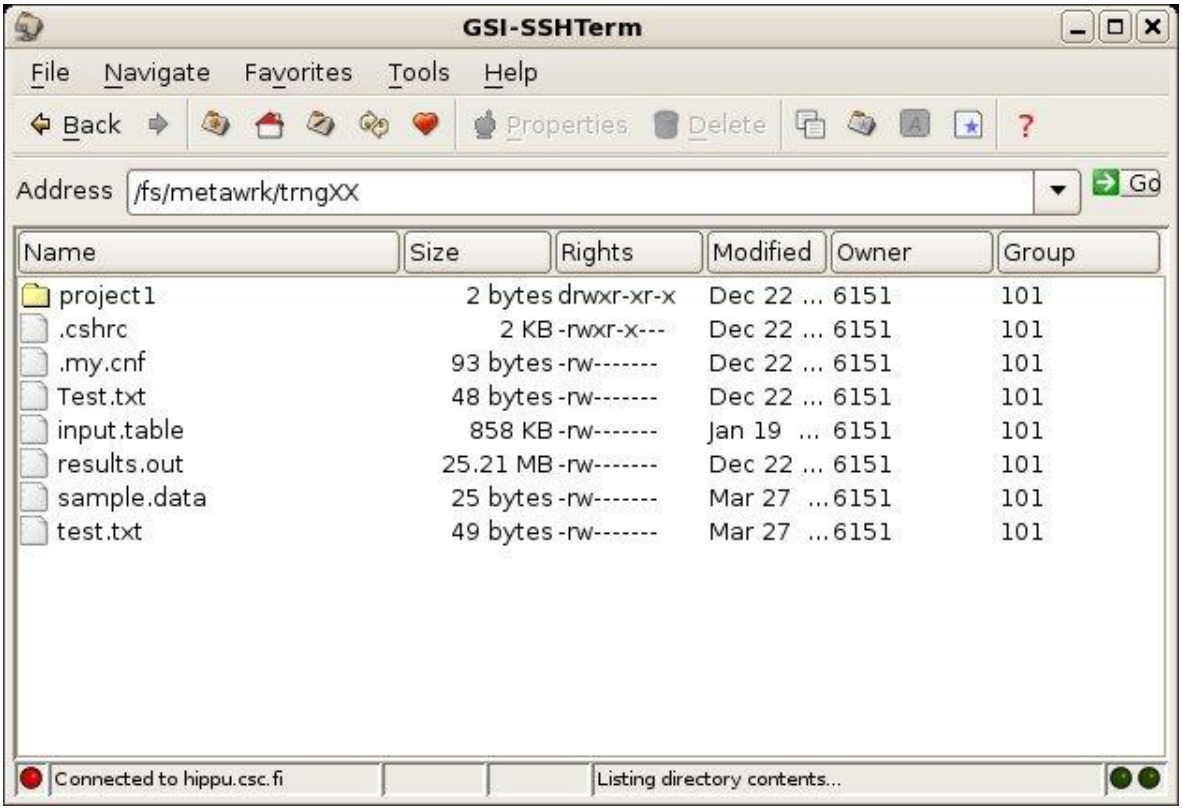
D. You can use the console to use the CSC servers, but it contains also other tools than just the command line client. You can use it for example for file transport or secure tunneling of server ports. In this case you can utilize the SFTP file transport tool.

Go to the “Tools” menu of the client window and select:

Tools | SFTP session



This launches a file transport terminal to your screen.



E. By default the file transport terminal shows the content of your CSC home directory. To move data to your $METAWRK directory at CSC change the directory path in the “Address” filed to:

/fs/metawrk/your_account

Where your_account is replace by the user account you are using. To upload your file, go to the “File” menu and select

File | Upload Files

Select the file to be uploaded (R.fasta) using the file selection window that opens to your screen and start the upload process.

2. Retrieving the target sequences from web

Later on in this exercise we want to compare our query sequences against the genome of bacteria *Pseudomonas aeruginosa*. This genome is not available at CSC but you can download it from the ftp site of NCBI. First you need to locate the file to be downloaded.

Point you web browser to address: **ftp://ftp.ncbi.nih.gov/genomes/Bacteria/**

In the listing you can find four *Pseudomonas_aeruginosa* genomes. Open the folder:
Pseudomonas_aeruginosa_PA7_uid58627.

There the nucleotide sequence is stored into file:
NC_009656.fna .

Right click this file name and copy the link location so you can use just paste the address to the command line after the *wget* command.

In the command shell, move to you **$METAWRK** directory:

```
cd $METAWRK
```

Then retrieve the data with command (all in single line)

```
wget ftp://ftp.ncbi.nih.gov/genomes/Bacteria/Pseudomonas_aeruginosa_PA7_
uid58627/NC_009656.fna
```

Check what you got with commands:

```
ls -ltrh
head NC_009656.fna
tail -20 NC_009656.fna
module load emboss
infoseq_summary NC_009656.fna
```

3. Using IDA

In this exercise you will use IDA storage service to store and share data. In this case we will use IDA through the iRODS commands in Hippu.

Setting up your IDA account

Before you can start using IDA you must first define your iRODS environment. This needs to be done only once.

Log in to Hippu and give commands:

```
module load iRODS
iinit
```

The *iinit* command asks the information of your iRODS (i.e. IDA) account. Define the values as below. Replace the trngXX with your training account (trng01, trng02 , etc), not with your personal account.

One or more fields in your iRODS environment file (.irodsEnv) are

missing; please enter them.

Enter the host name (DNS) of the server to connect to*:**ida.csc.fi*

Enter the port number:1247

Enter your irods user name:*trngXX*

Enter your irods zone:csc

Those values will be added to your environment file (for use by

other i-commands) if the login succeeds.

Enter your current iRODS password:
yuh28rex

The settings are stored to file .irods/.irodsEnv in your home directory. For smooth usage, let's add one more line to the .irodsEnv file.

```
echo "irodsHome /csc/internal/bifld/trngXX" >> .irods/.irodsEnv
```

Finally, copy your .irods folder also to your $WRKDIR to be able to use iRODS commands also in batch jobs in the Vuori cluster.

```
cp -r .irods $WRKDIR/
```

All the steps above need to be done only once. Later on it is enough to give just the setup command:

```
module load iRODS
```

and start using the iRODS commands. Try first ils command to see the content of your IDA home directory:

```
ils
```

Create a new IDA directory and move the *Pseudomonas aeruginosa* genome there in compressed format

```
imkdir genomes
icd genomes
gzip NC_009656.fna
iput NC_009656.fna.gz
ils -l
```

All the IDA training accounts belong to the same project an so they have a common shared directory. Change this shared directory as the current iRODS directory and check the content of the directory:

```
icd /csc/internal/bifld/shared
ils
```

Download a file called *query.fastq*. You will use the file later on in exercise 5.

```
iget query.fastq
```

4. Using screen

Screen is a virtual window manager. When running programs in screen they will continue to run even if your connection is cut of. It is mostly useful when running long interactive jobs on Hippu.

Screen is machine specific, so you need to take note which machine (hippu1-4) you are logged in. You can see it *e.g.* from the command prompt

hippu2 ~>

Begin a new screen session

```
screen
```

We can now try *de-attaching* and *re-attaching* to a screen

De-attach from screen press:

ctrl-a-d

or give command

```
screen -d
```

You can list the running screen sessions with

```
screen -ls
```

You should see something like this:

There is a screen on:
11870.pts-17.c553 (Detached)
1 Socket in /tmp/uscreens/S-trngXX.

Since there is only one screen running you can reconnect it with command:

```
screen -r
```

If there would be several screens running you could specify one by the process id

```
screen -r 11870
```

Screen also works if the connection is cut off unexpectedly. We will test this later on.

5. Running BWA read mapping job in the grid

Aligning large sets of short sequences, reads, to a reference sequence set is an essential step in many next-generation sequencing (NGS) based analysis work-flows. For reads sets, containing hundreds of millions of short sequences, the mapping task can take weeks of computing.

One way to speed up the mapping tasks is to use grid computing. The read mapping tasks can be divided into numerous sub-tasks that can be executed in grid in parallel fashion and later on merged back together. In this exercise we use grid interface of Burrows-Wheeler Aligner, BWA. The interface is called *grid_bwa.* It executes automatically following five basic steps. 1. Checking input, 2. Indexing the reference. 3. Splitting the mapping task into sub-jobs, 4. Executing sub-jobs in the grid. 5. Collecting the results.

In the beginning, make sure you are in your screen session. You can check this by giving command:

```
screen -r
```

This command should now show you something like:

There is a screen on:

28744.pts-14.hippu1 (Attached)

There is no screen to be resumed.

Here the word *Attached* tells that you are within this screen.

Then move to your **$METAWRK** directory.

```
cd $METAWRK
```

setup BWA environment:

```
module load bwa
```

The update you gird proxy:

```
grid-proxy-init -rfc -valid 24:00
```

Now you are ready to launch the BWA job with command:

```
grid_bwa aln -query query.fastq -ref NC_009656.fna -out aln.bam -nsplit 20
```

Once the command is running you can step out form the screen session with by pressing:

ctrl-a-d

And return to the screen with command:

```
screen –r
```

6. Connection break simulation with screen

Close the ssh window, while you are in your screen session. This simulates the connection being cut off

Log in again. Make sure you log in to the correct machine.

You can list the running screens as before

```
screen -ls
```

The re-attach as before

```
screen -r
```

You should find everything as you left it. The grid BWA process is still running.
When the BWA process is finished you should close the screen session with command:

```
exit
```

after that try listing screens

```
screen -ls
```

The command should return something like:

No Sockets found in /tmp/uscreens/S-trngXX.

Screen can be useful especially when running programs that use stdout, but you should be careful and not leave unnecessary screen sessions hanging.

NOW, PLEASE DE-ATTACH FORM YOUR SCREEN AND LET THE GRID BWA RUN IN THE BACKGROUND WHILE YOU DO THE NEXT EXERCISE.

7. Working with data columns

Submit the BLAST search again using the *pb blast* that runs in Hippu. In this case our query set is rather small. For larger query sets you could use *pb blast* in Vuori and for really huge BLAST task you could run the job using Grid BLAST.

You should be able to submit the BLAST job with commands:

```
cd $METAWRK
module load blast+
pb blastn -query R.fasta -dbnuc NC_009656.fna -out pb_blast_results -outfmt 7
```

By running command:

```
head -50 pb_blast_results
```

You can see that the result file contains columns, where the query sequences are in the first column and the hit sequences in the second column. You can now check, how many hits each query sequence got with command:

```
awk '{print $1}' pb_blast_results | grep -v "#" | sort | uniq -c | sort -k1n
```

Study how this command works by executing it step by step:

```
awk '{print $1}' pb_blast_results
awk '{print $1}' pb_blast_results | grep -v "#"
awk '{print $1}' pb_blast_results | grep -v "#" | sort
awk '{print $1}' pb_blast_results | grep -v "#" | sort | uniq -c
awk '{print $1}' pb_blast_results | grep -v "#" | sort | uniq -c | sort -k1n
```

You can check the number of query sequences in the input file with command:

```
grep -c ">" R.fasta
```

Alternative ways to count number of sequences with EMBOSS:

```
module load emboss
seqcount R.fasta
infoseq_summary R.fasta
```

If you compare that to the number query sequences in the result file:

```
awk '{print $1}' pb_blast_results | grep -v "#" | sort | uniq -c | wc -l
```

You can notice that the command used above shows no information for those query sequences that did not get any matches.

Extra task:

Use linux commands to sort out those query sequences that did not produce any hits. There is no singe right way to do this task. You probably need several commands and temporary files to get the result. You can also try linux scripting.