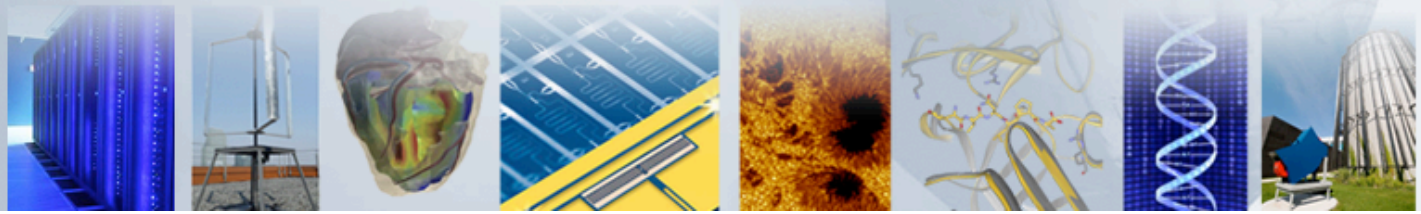
First Steps on Calcul Québec's Servers

Daniel Stubbs

March 16, 2017

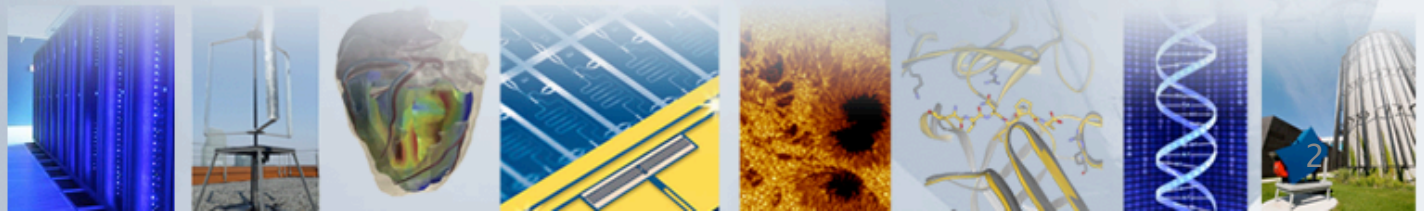
Calcul Québec



Linux

- A form of Unix, an operating system whose origins go back to the 1960s, i.e. long before the creation of Windows and MacOS.
- There is a sharp distinction between the the operating system as such and its graphical display (windows and the use of a pointing device).
- When you connect to a Calcul Québec server, your interaction with the operating system will normally be based on a command line interface, i.e. the keyboard.
- This may seem old-fashioned but it's very robust and powerful, once a few basic commands are learned.

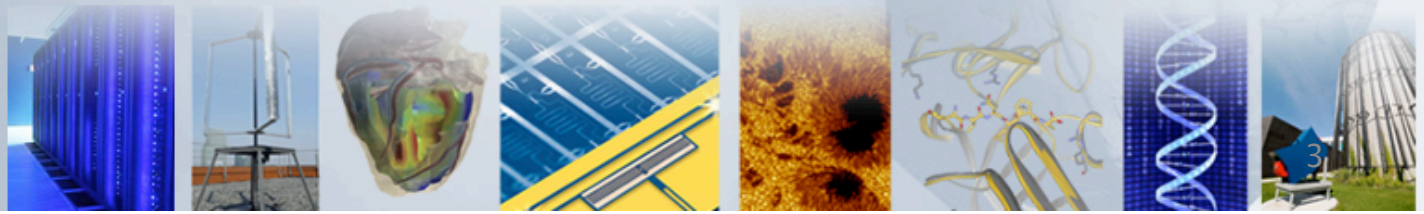
Calcul Québec



Summary

1. Connecting to Clusters
2. The Linux Filesystem
3. Text Editors
4. Transferring Files
5. Bash
6. Software Modules
7. Compiling Your Code
8. Using CPLEX
9. Job Submission

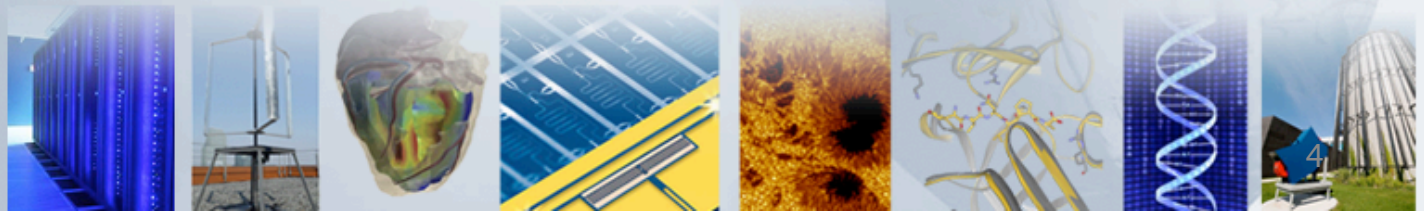
Calcul Québec



Connecting to Clusters

- The only way to connect to a Calcul Québec server is using `ssh` (Secure Shell).
- With ssh, your password is encrypted before being sent over the network.
- To connect, you obviously need to know the name of the machine (e.g. helios.calculquebec.ca), your username and finally your password.
- The first time that you connect to a machine, ssh will ask if you want to store the remote machine's key to which you normally answer "yes".

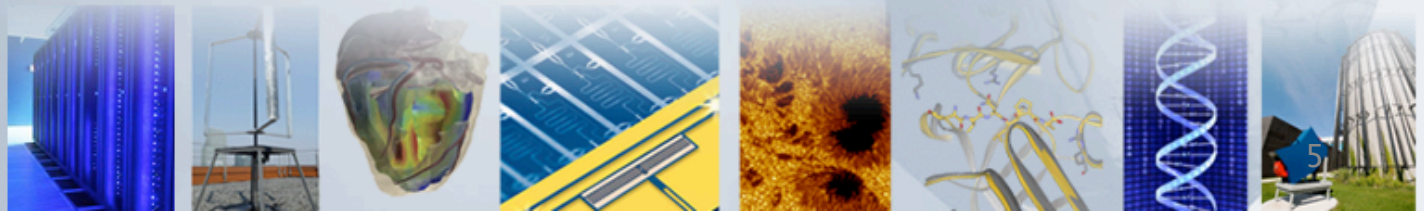
Calcul Québec



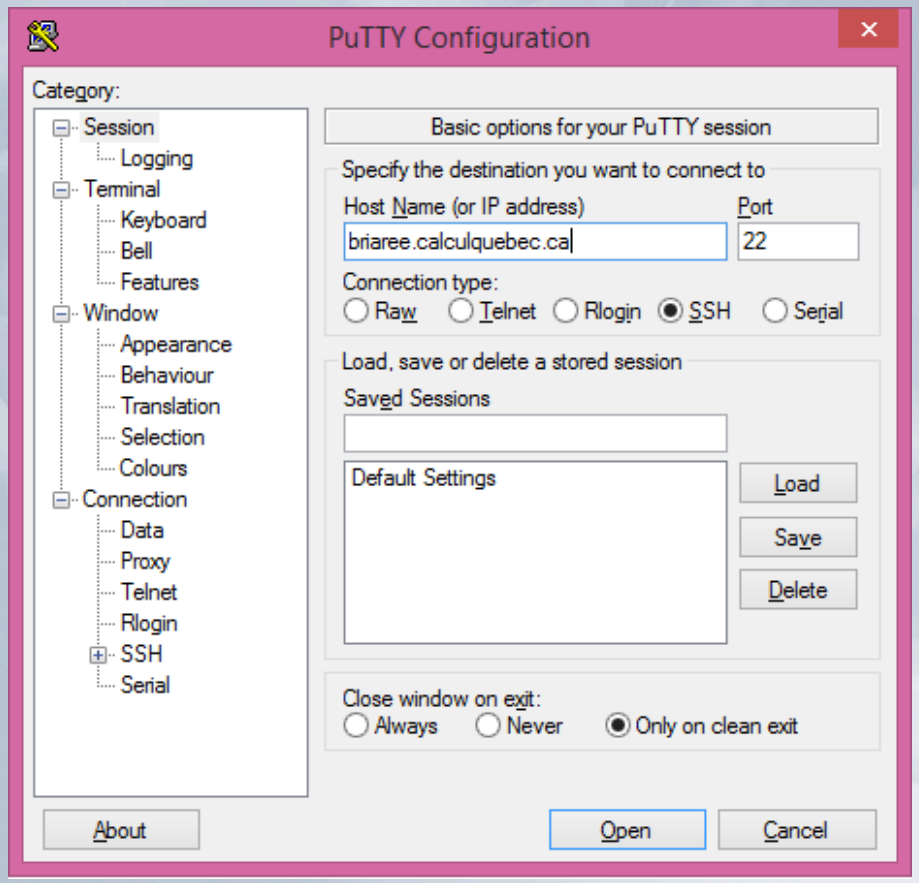
Connecting to Clusters

- If you use OS X, you already have ssh – you simply need to open Terminal.app (in /Applications/Utilities) and type
  `ssh username@machine_name`
- Windows doesn't come with a default ssh client but it's easy to download a free one.
- Common choices are "PuTTY" and "MobaXterm" and if you use the former, this is what the program looks like when it starts up:

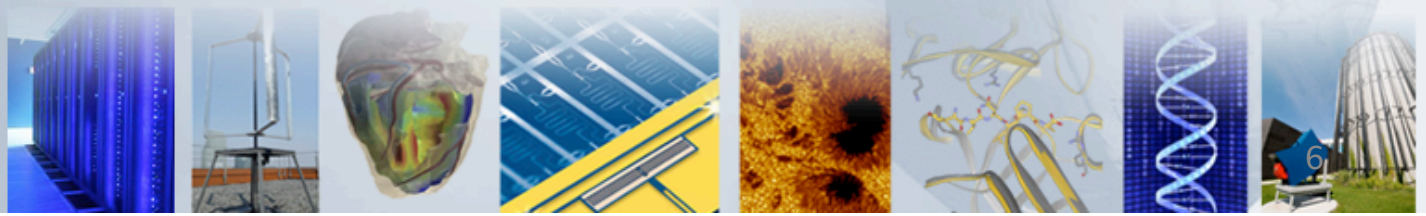
Calcul Québec



Connecting to Clusters



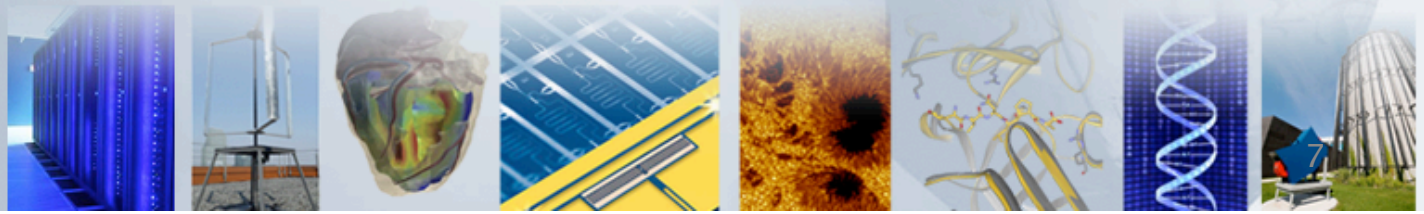
Calcul Québec



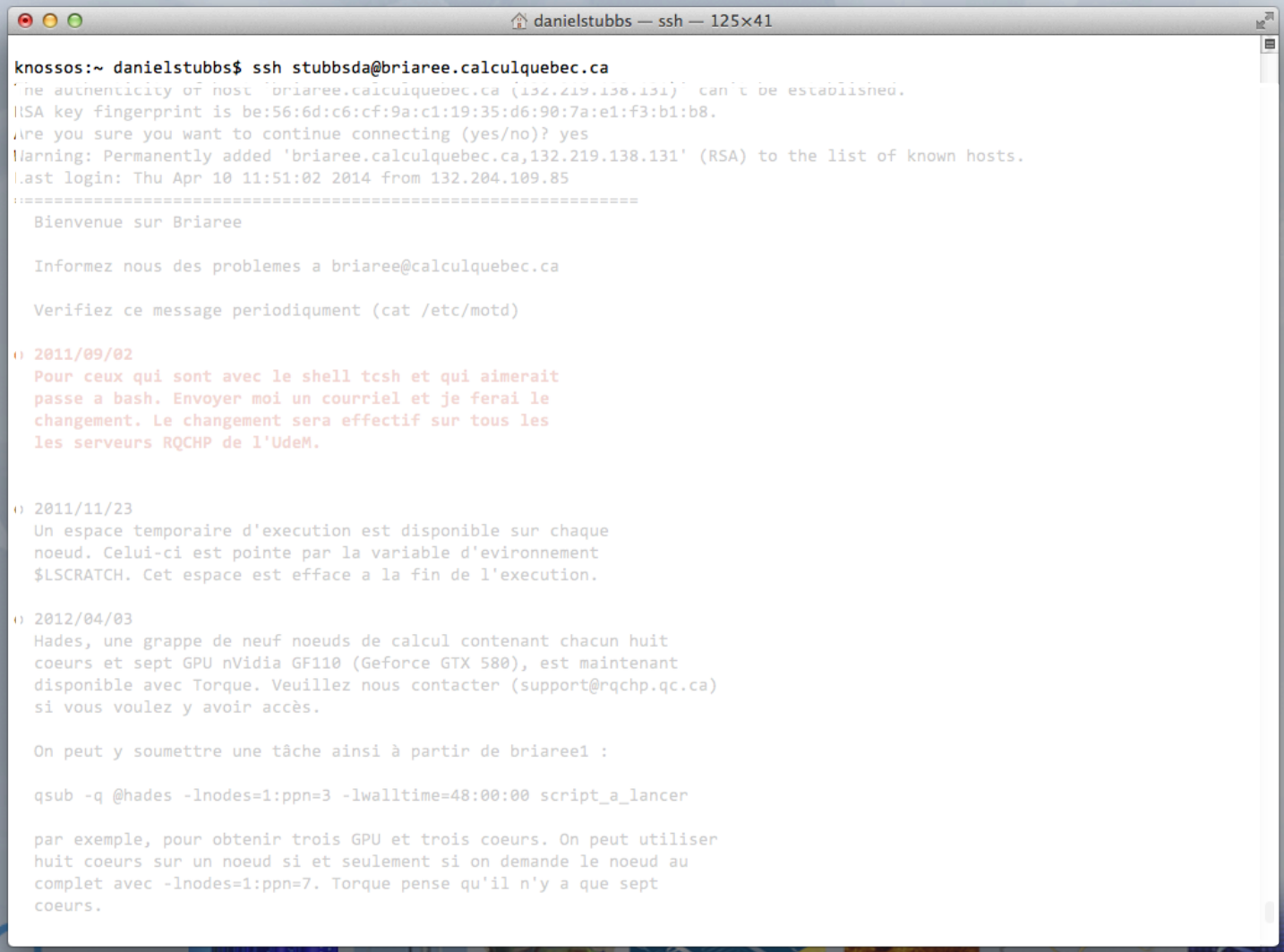
Connecting to Clusters

- You can also use a virtual machine by means of which you can run Linux (with its ssh client) inside of a machine running Windows or OS X.
- You can disconnect from a remote Linux machine by typing the command `exit`

Calcul Québec

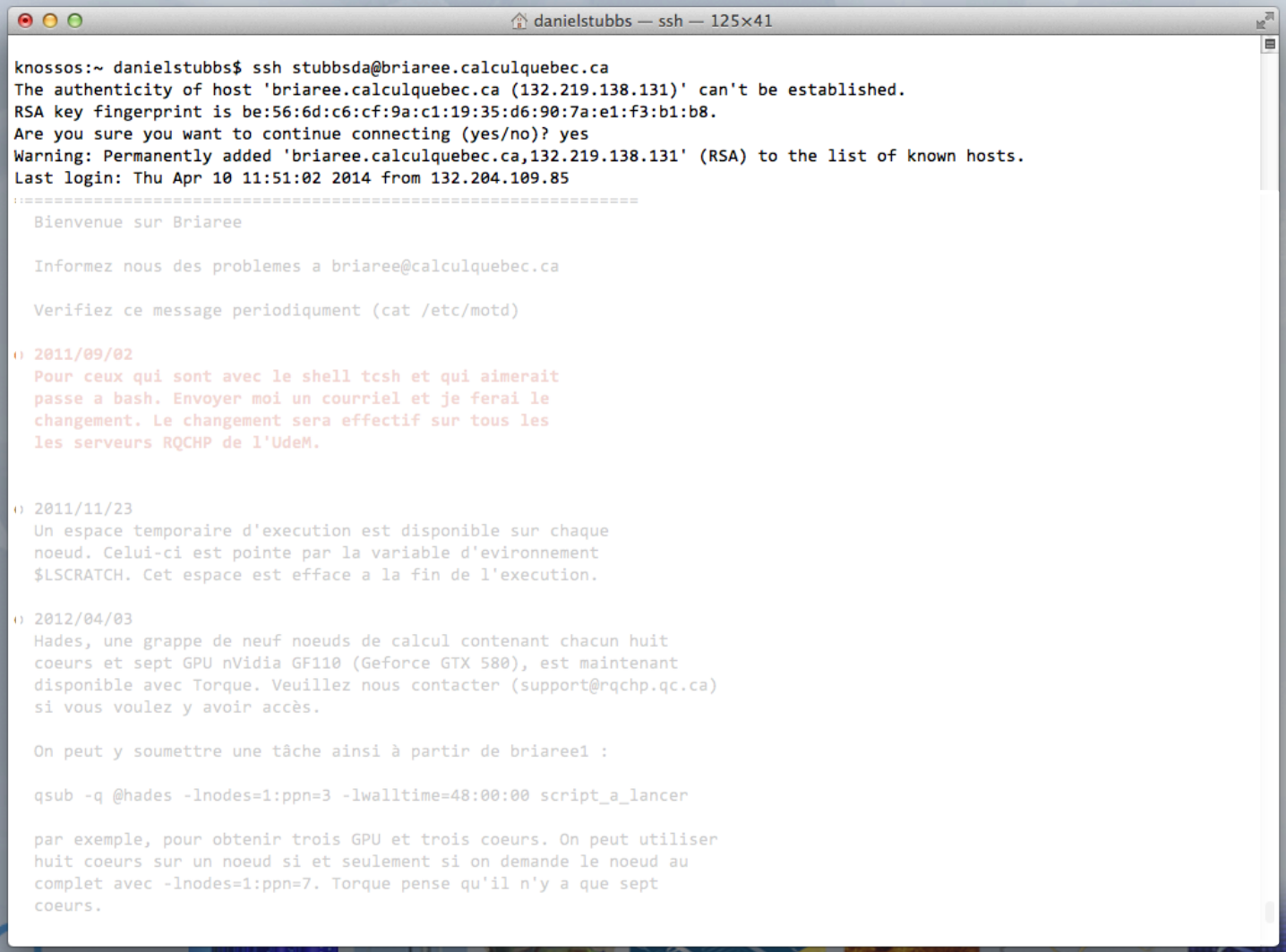


Connecting to Clusters



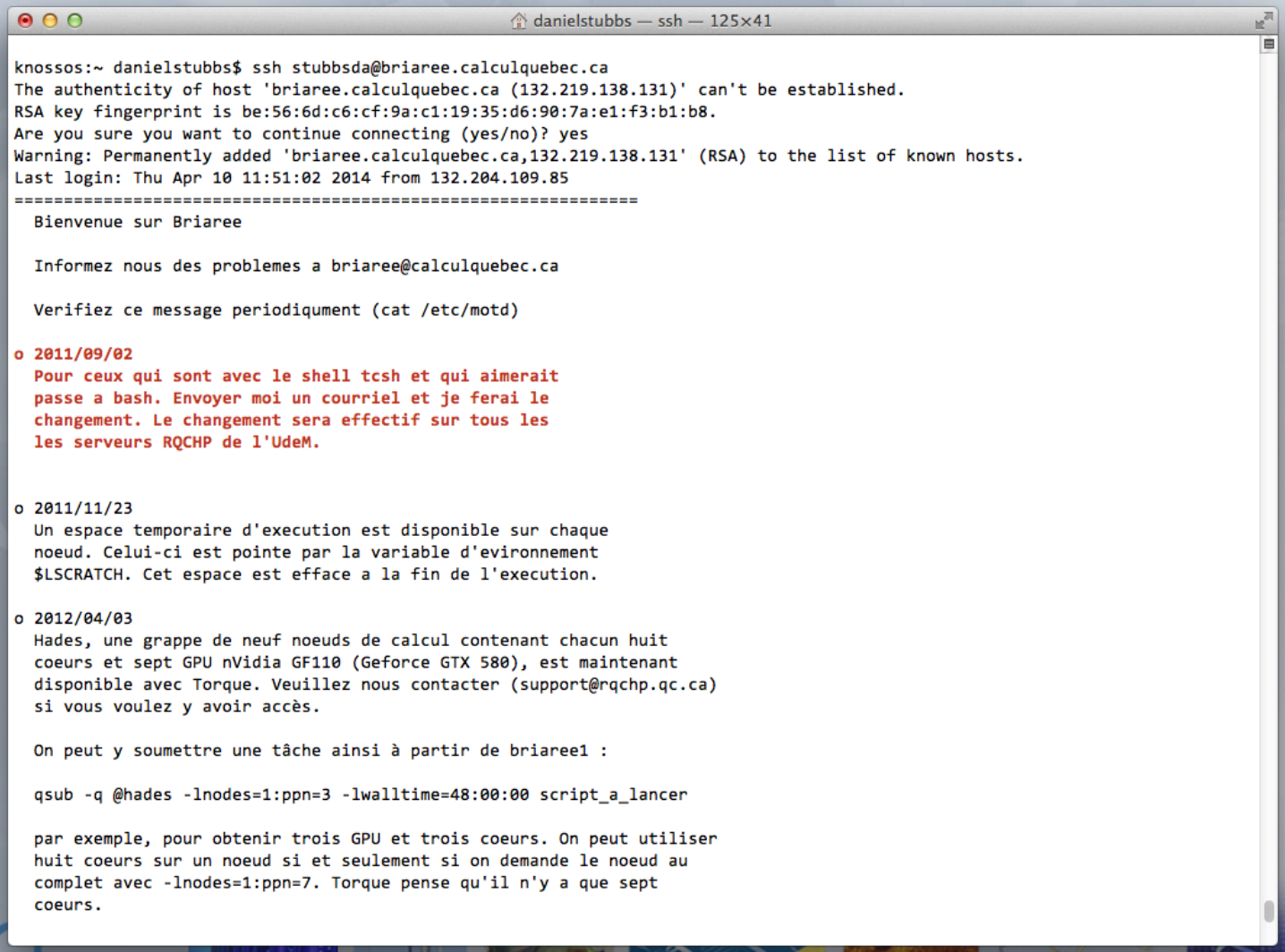
Calcul Québec

Connecting to Clusters



Calcul Québec

Connecting to Clusters

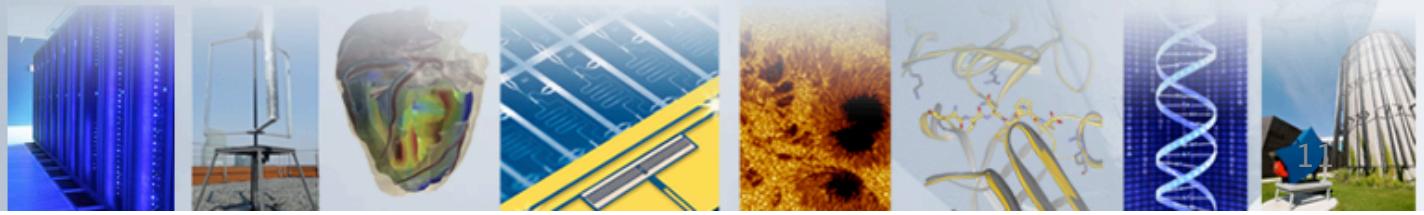


Calcul Québec

The Linux Filesystem

- With Linux (like OS X) the filesystem begins with the root directory " / ", the usual Windows nomenclature (C:\, D:\ etc.) doesn't exist.
- When you connect to a Calcul Québec server you always begin in your home directory, typically /home/username in Linux.
- During your connection you always see the "Message of the Day" from the remote server.
- The first time you connect your home directory is of course almost empty.

Calcul Québec



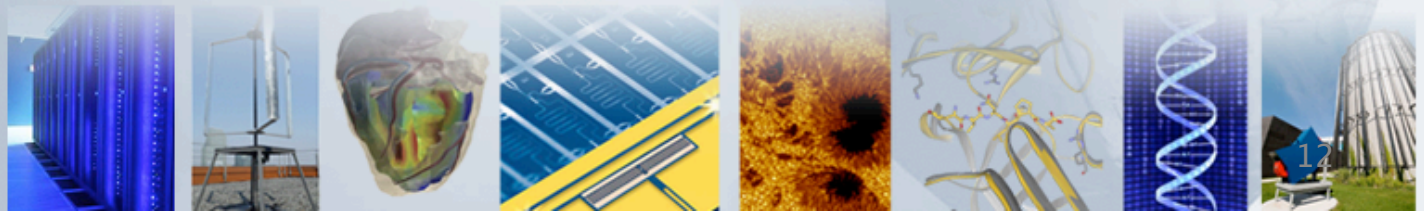
The Linux Filesystem

- Some basic commands for navigating around the filesystem are:

`mkdir`	Create a directory
`ls`	Display the contents of a directory
`cd`	Change directory
`cp [-r]`	Copy a file or directory
`mv`	Move a file or directory
`rm [-r]`	Delete a file or directory
`pwd`	Display the current directory

- You can learn more about these commands by using the command `man`

Calcul Québec



The Linux Filesystem

- There are also some useful shortcuts:

~ your home directory,

. the current directory,

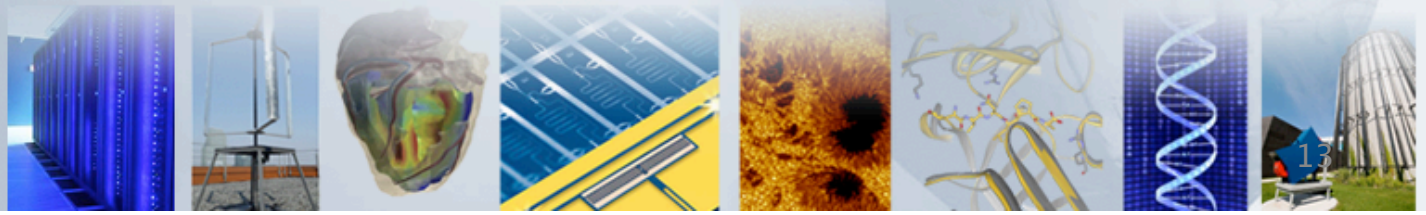
.. the directory above the current directory.

- Linux also has the wildcard * that can be used to carry out a command on several files, e.g.

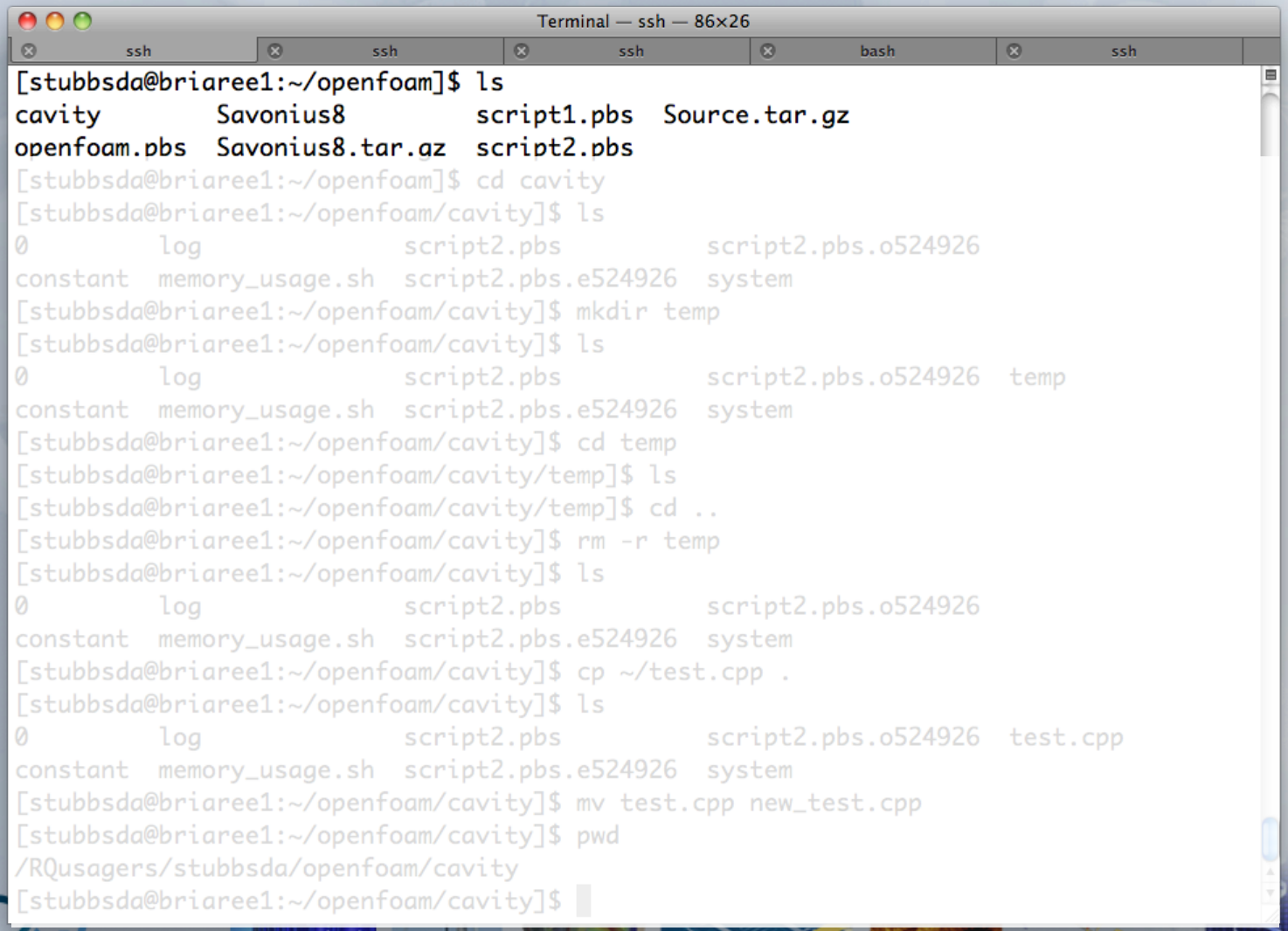
```
ls *.py
```

to see all the files whose name ends in `.py`

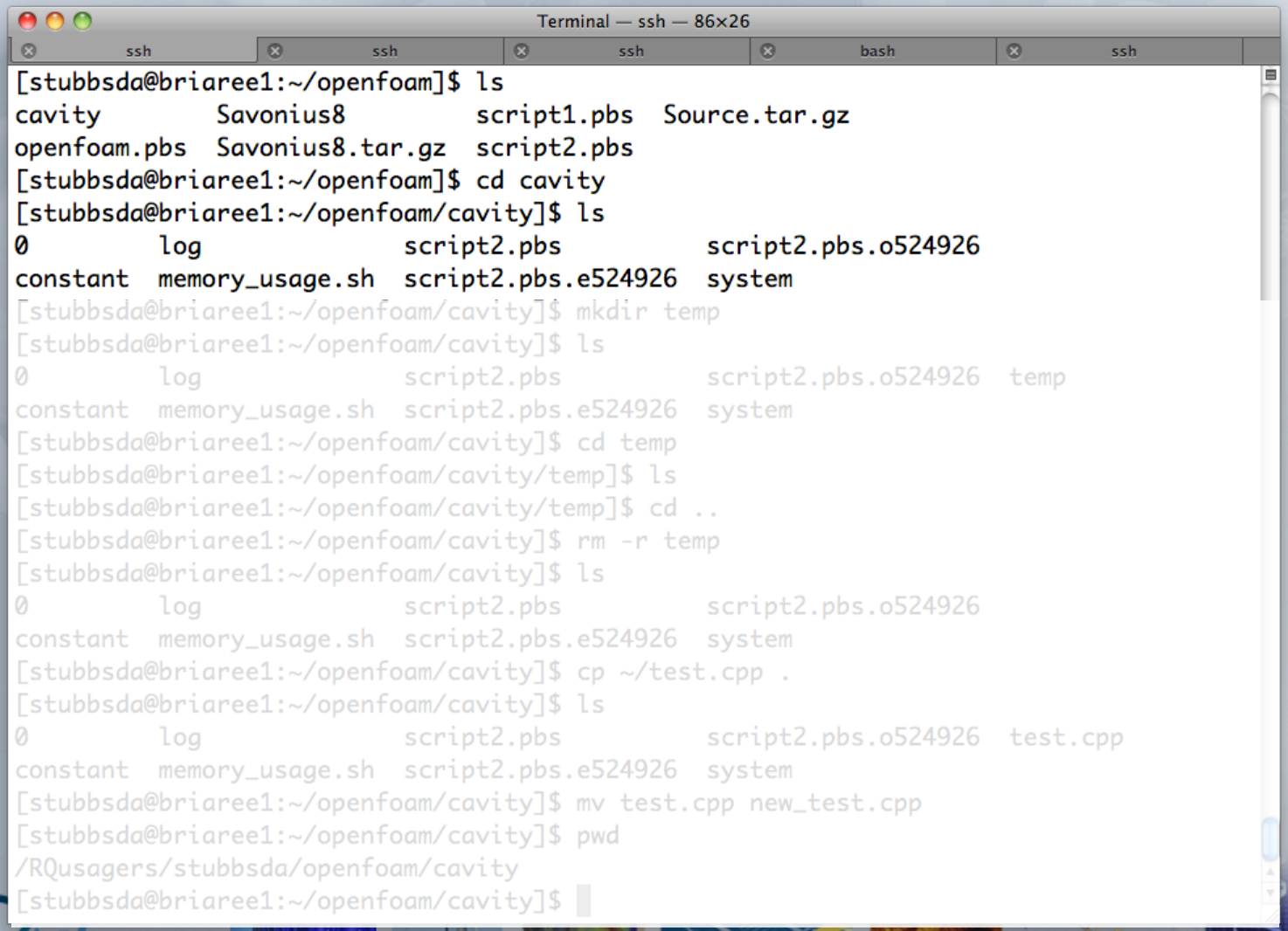
- In general the commands are all lower case, a practice common to Unix, C and C++, and the operating system assumes you know what you're doing.



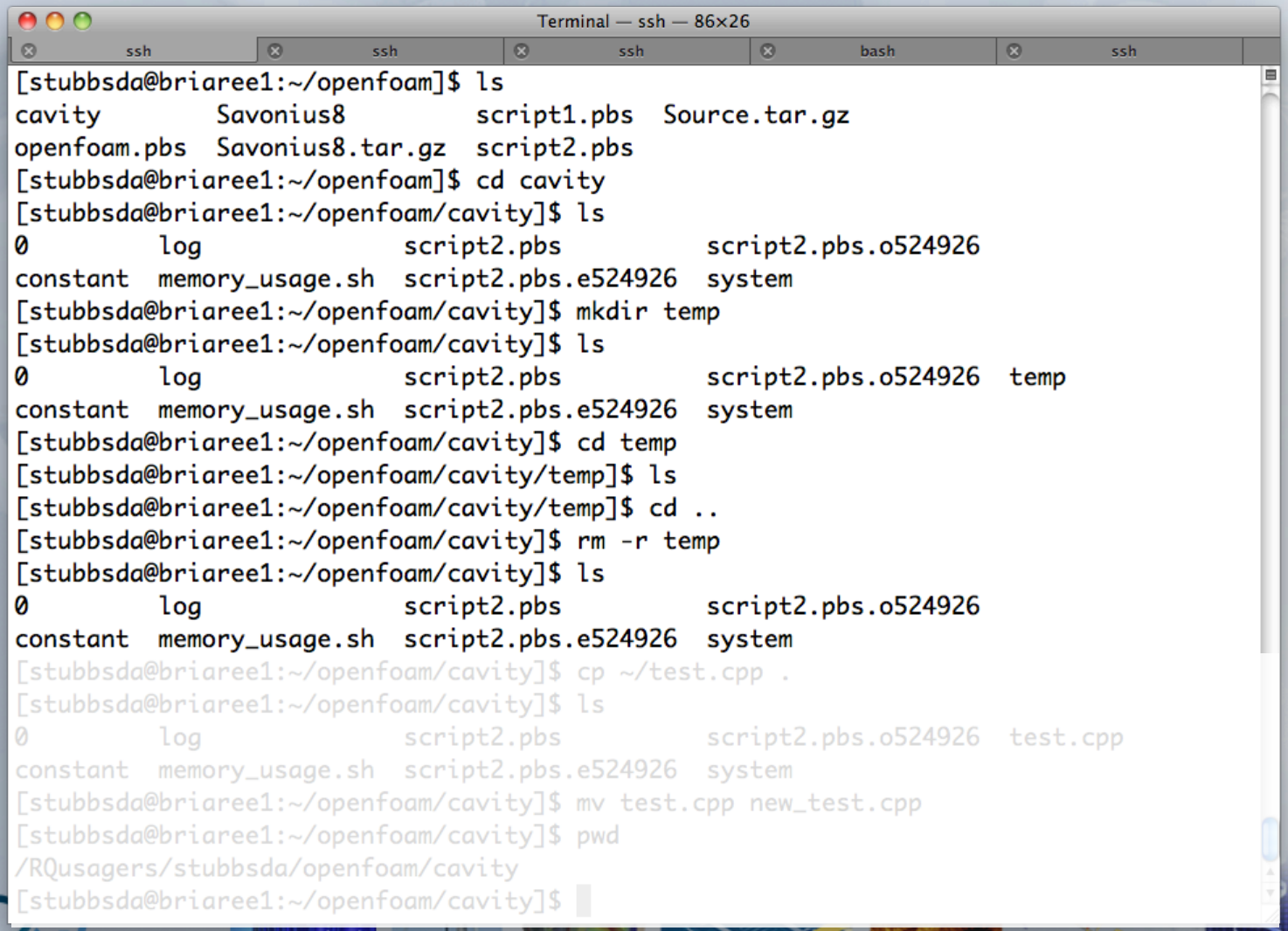
The Linux Filesystem



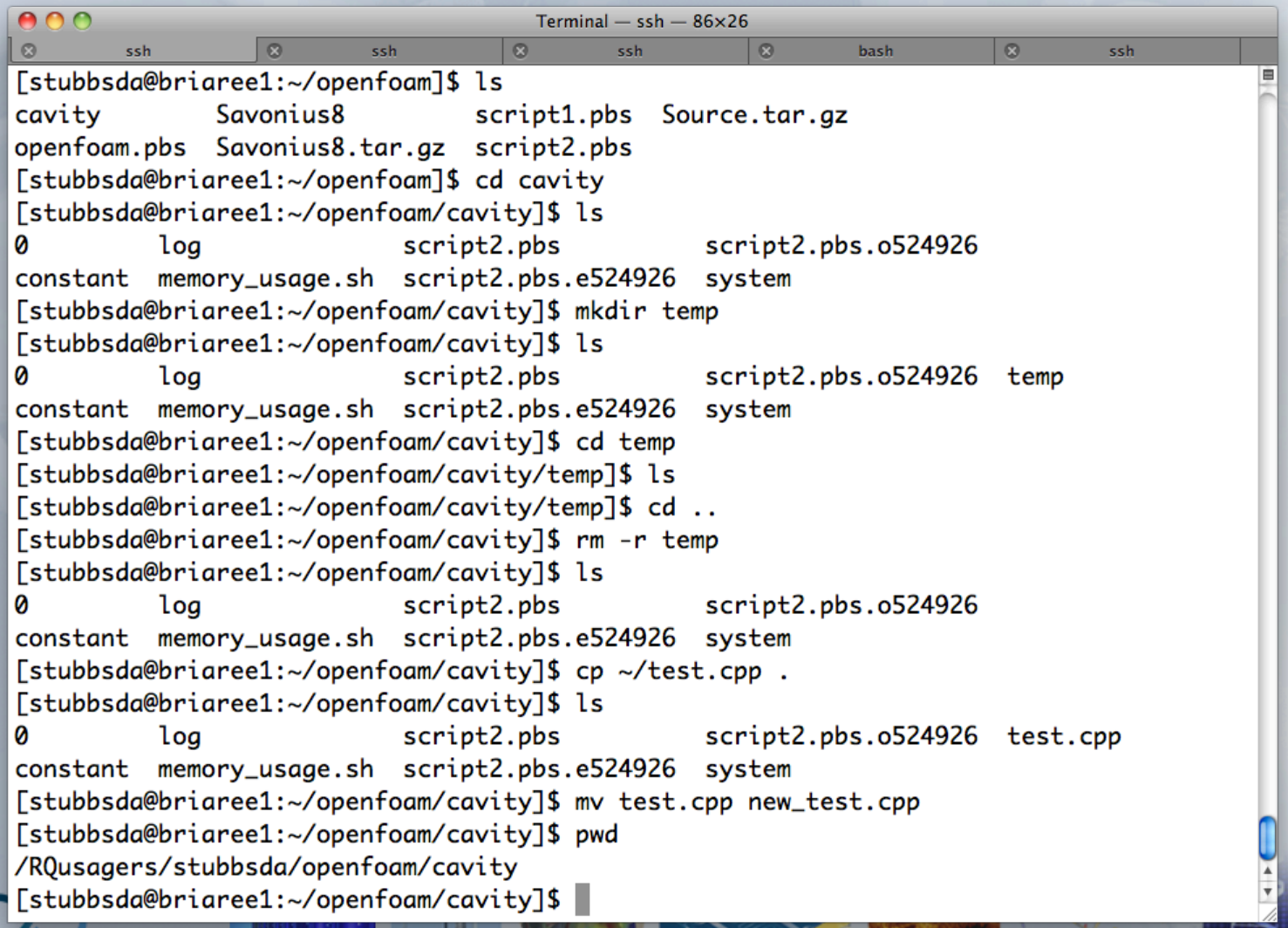
The Linux Filesystem



The Linux Filesystem



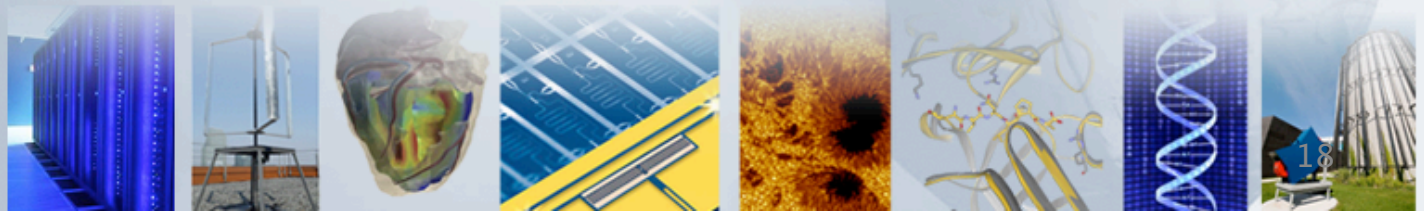
The Linux Filesystem



The Linux Filesystem

- When choosing names for your files and directories, avoid blank spaces and accented characters as these can lead to frequent problems
- It's wiser to use `mes_pensees` instead of "mes pensées".

Calcul Québec

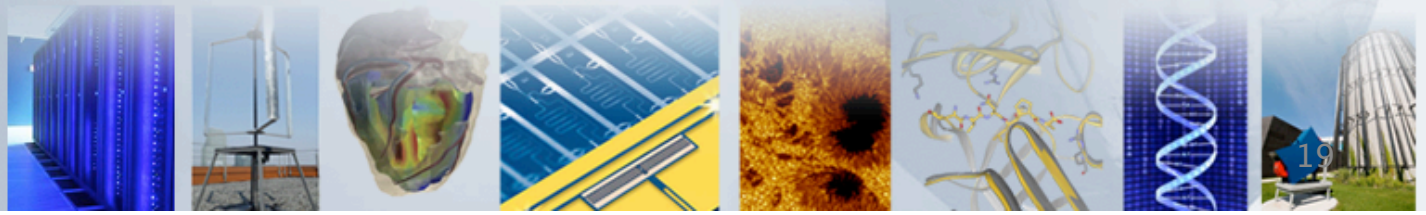


The Linux Filesystem

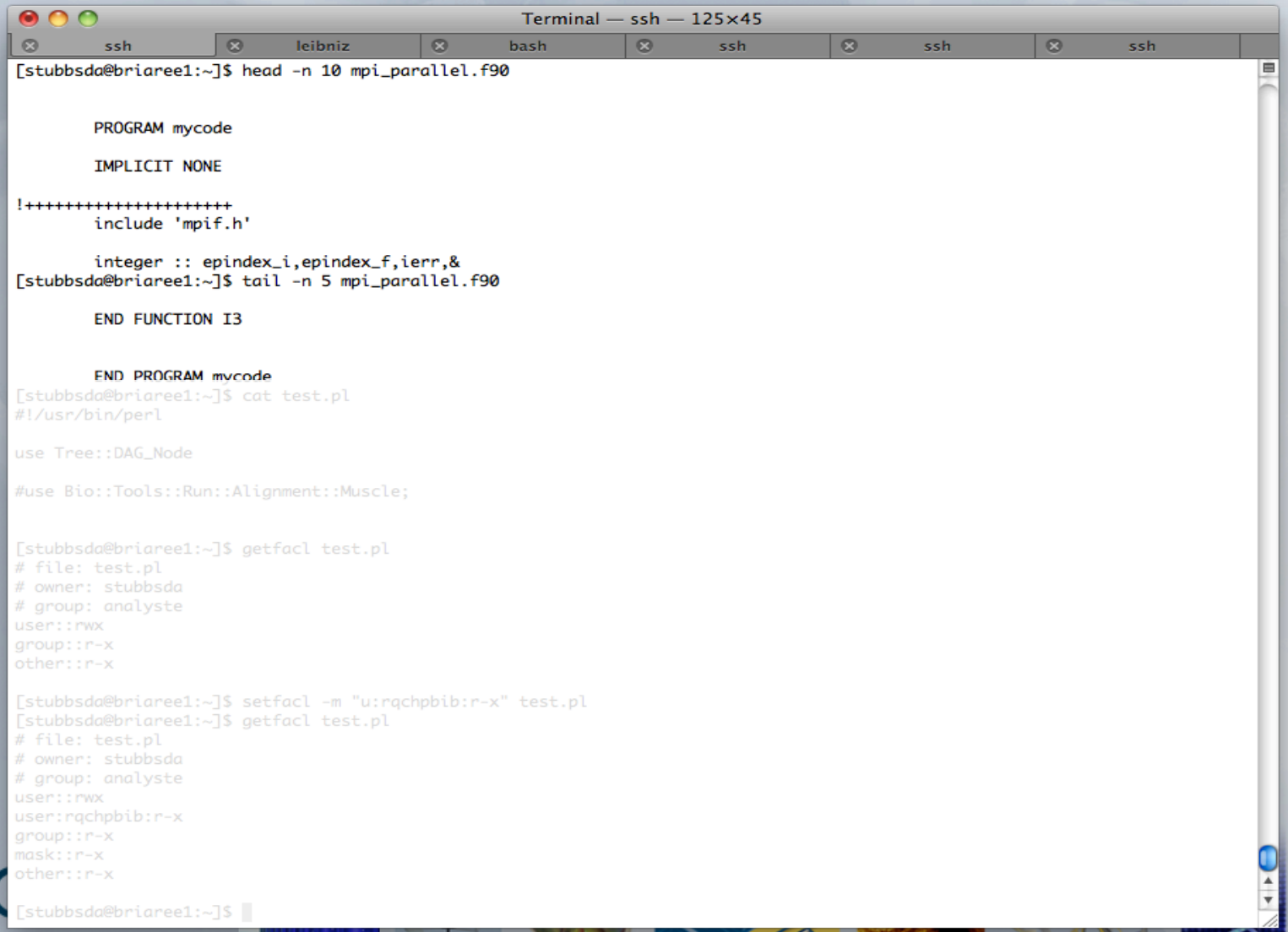
- There are a variety of tools for displaying the content of a text file.
- You can begin with the command `wc` (word count) which gives the size of a file in terms of the number of lines, words and bytes.
- To see the actual contents of the file, you can use `cat` (the entire file), `more` and `less` (interactive control of the display).
- If you're only interested in the beginning or end of a file, you can use `head` and `tail`.
- The following command shows the last 50 lines of this file:

```
tail -n 50 resultat.txt
```

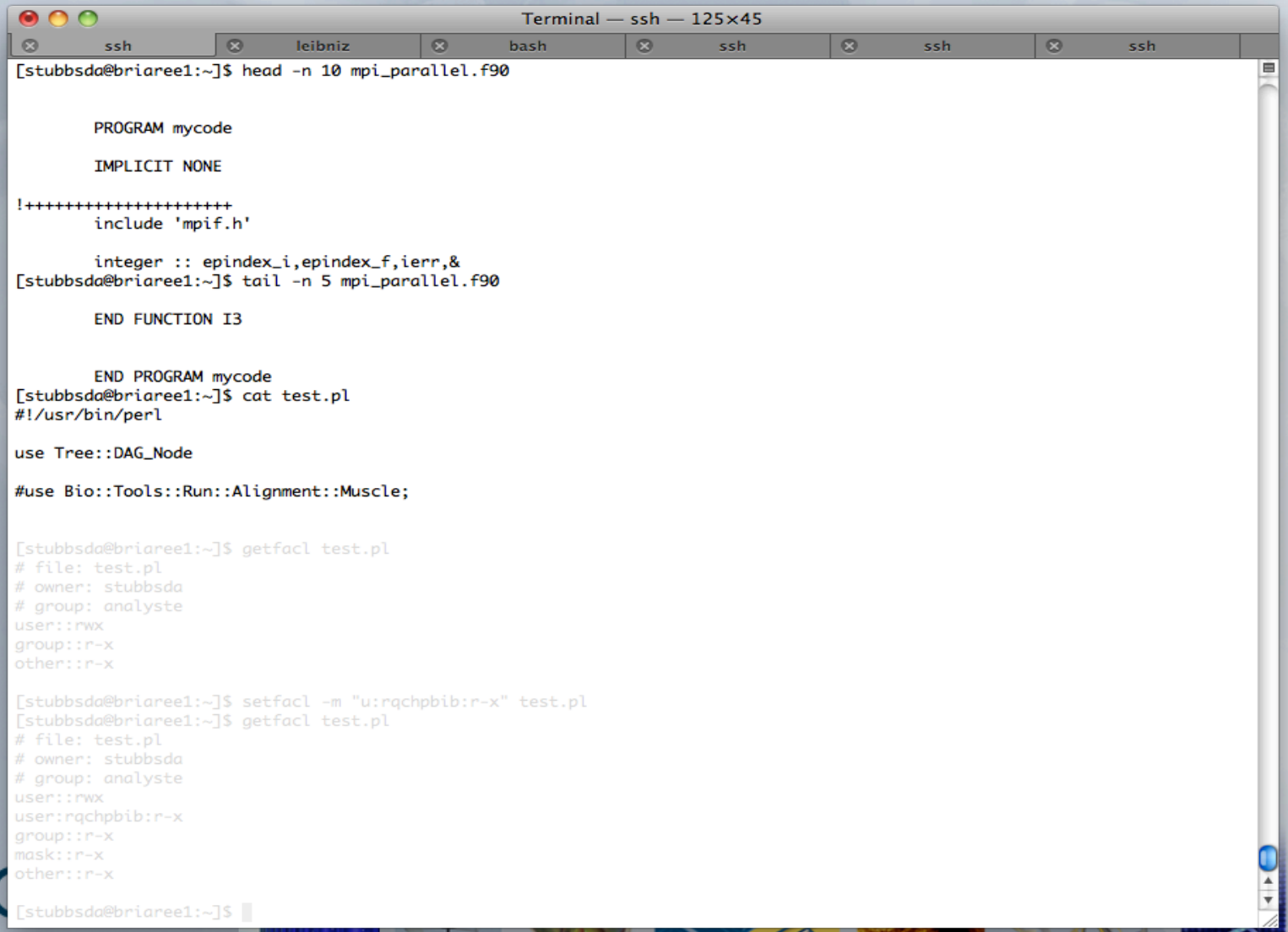
Calcul Québec



The Linux Filesystem



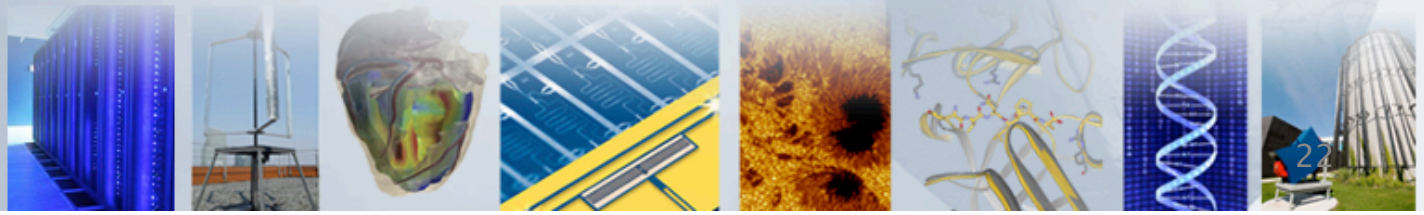
The Linux Filesystem



Text Editors

- With `mkdir` you can create directories but to create a text file you need to use a text editor.
- There are several different ones available in Linux and you are free to choose the one which you prefer.
- Among the most simple is `nano` and with more features and complexity there is `vi`.
- These two editors rely exclusively on the keyboard but there are text editors with a graphical interface like `emacs` and `nedit`.
- These last two are very similar to programs like Notepad (in Windows) and TextEdit.app (in OS X).

Calcul Québec



Text Editors

- You can begin by using nano, where you open a file with the command

```
nano file_name
```

- You can use the arrow buttons of the keyboard to move around, delete letters and add text. Most commands for the editor begin with the Ctrl button, for example:

`Ctrl-X` to exit the editor

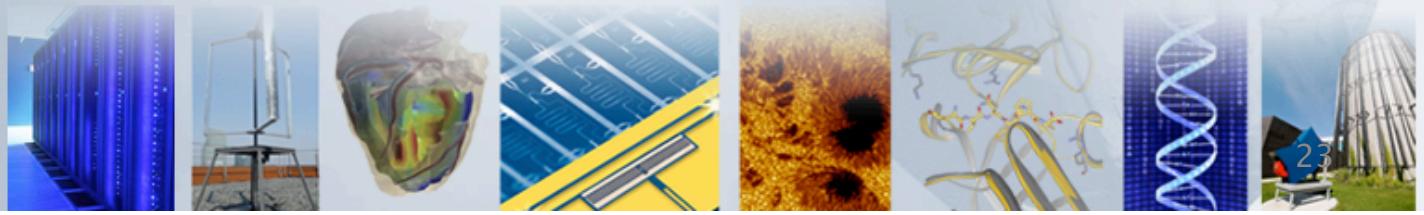
`Ctrl-W` to search for a word or phrase

`Ctrl-O` to write the current content to the file

`Ctrl-K` to delete a line

`Ctrl-G` for help

Calcul Québec

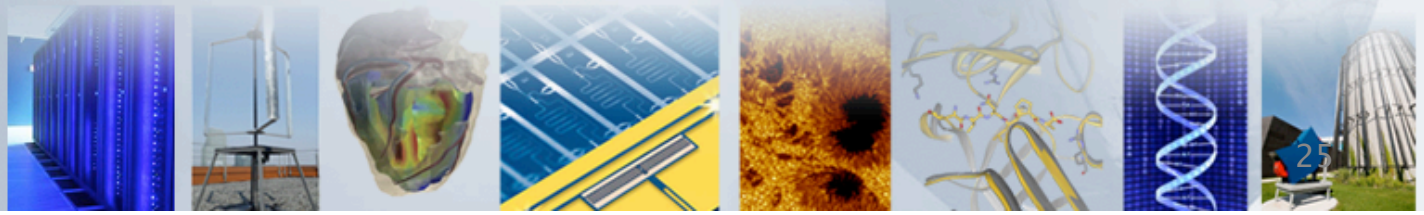


Text Editors

Text Editors

- One problem associated with text files is how to determine the end of a line?
- Windows doesn't answer this question the same way that Linux does, so we can have a pair of tools to handle the conversion of line endings from one standard to another: `dos2unix` and `unix2dos`.
- It's essential to avoid using word processing software like Word, Pages or OpenOffice: these programs do not create genuine plain text files.
- Under Windows the best idea is to use *Notepad*.
- Under OS X, open Terminal.app and then you can use nano, vi or emacs.

Calcul Québec



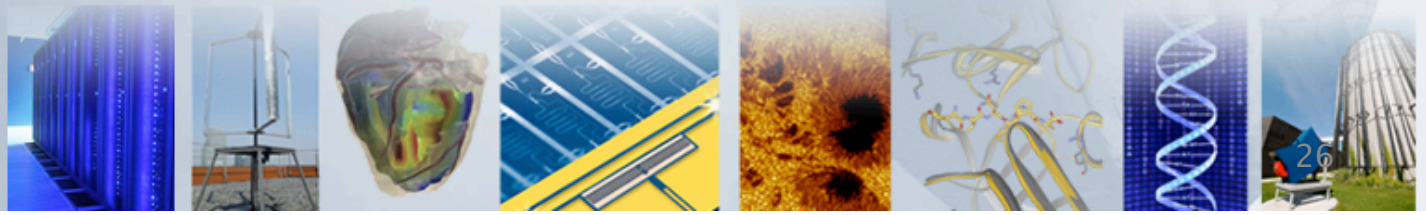
Transferring Files

- To transfer files between Calcul Québec servers and your workstation you should use `scp` and `sftp`.
- These two programs belong to the same family as `ssh` and also encrypt the connection.
- The program `scp` works like the Unix command `cp` (copy):

```
scp username@machine:research/out.dat result.dat
```

- As for `sftp`, you use it like `ftp`: you can use `cd` to move around and `put/get` to transfer files.
- Under Windows, you can use a program like *WinSCP* which functions in a manner similar to Windows Explorer (with a graphical interface etc.).

Calcul Québec

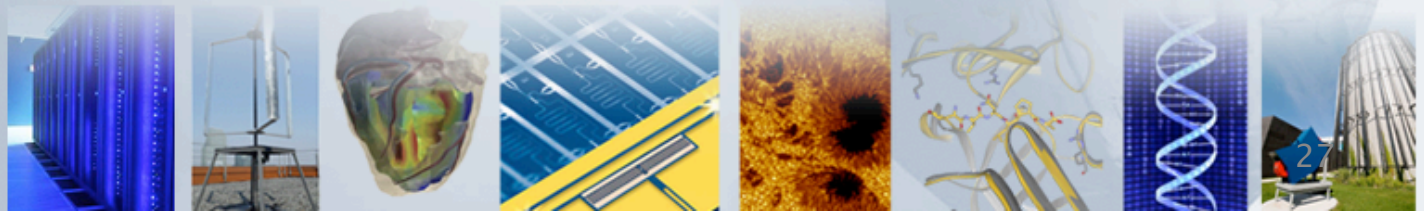


Transferring Files

- Since it's unlikely that your workstation has an ssh server, you should always start your file transfers from your workstation and not from the remote Calcul Québec machine.
- If you have a lot of data to transfer or need to transfer it over a great distance (e.g. from Vancouver to Montreal), you should consider using Globus.
- Globus is a national service national to simplify file transfer between the servers of Compute Canada.
- To learn more you can read the following page

https://docs.computecanada.ca/wiki/Globus

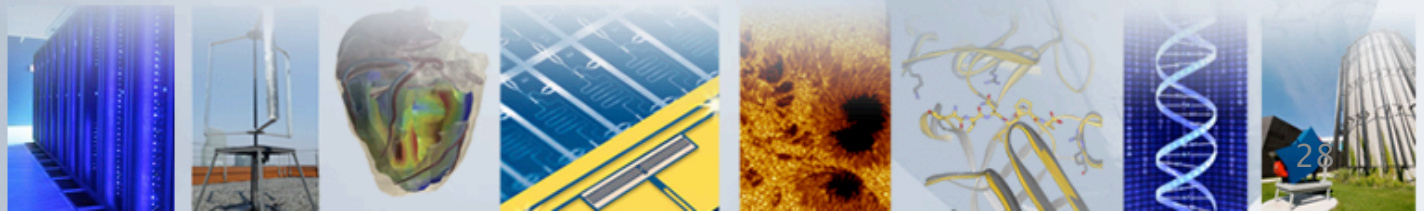
Calcul Québec



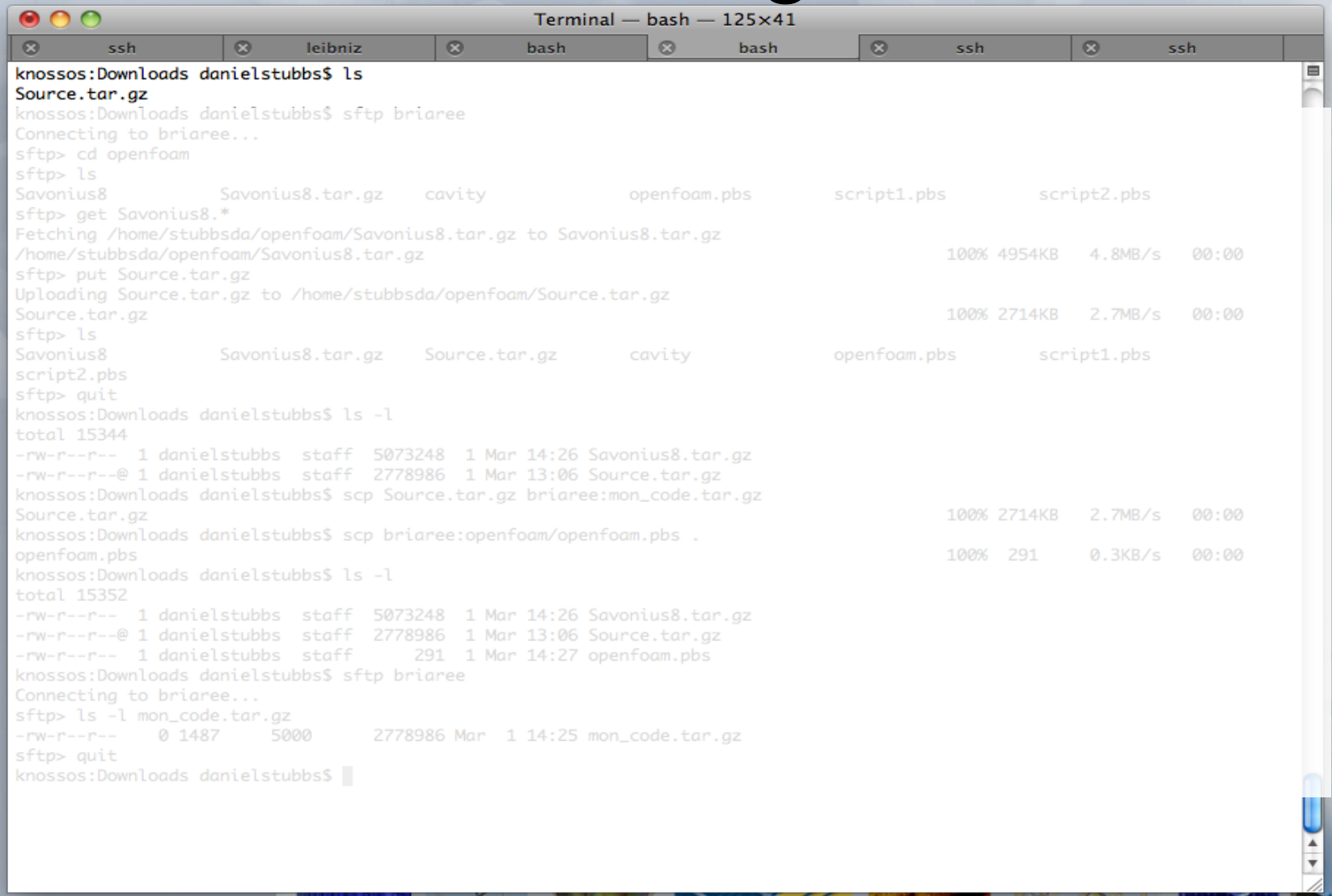
Transferring Files

- If you're planning on transferring a lot of data (tens of gigabytes or more), we would prefer if you discuss this with technical support staff before uploading the data.

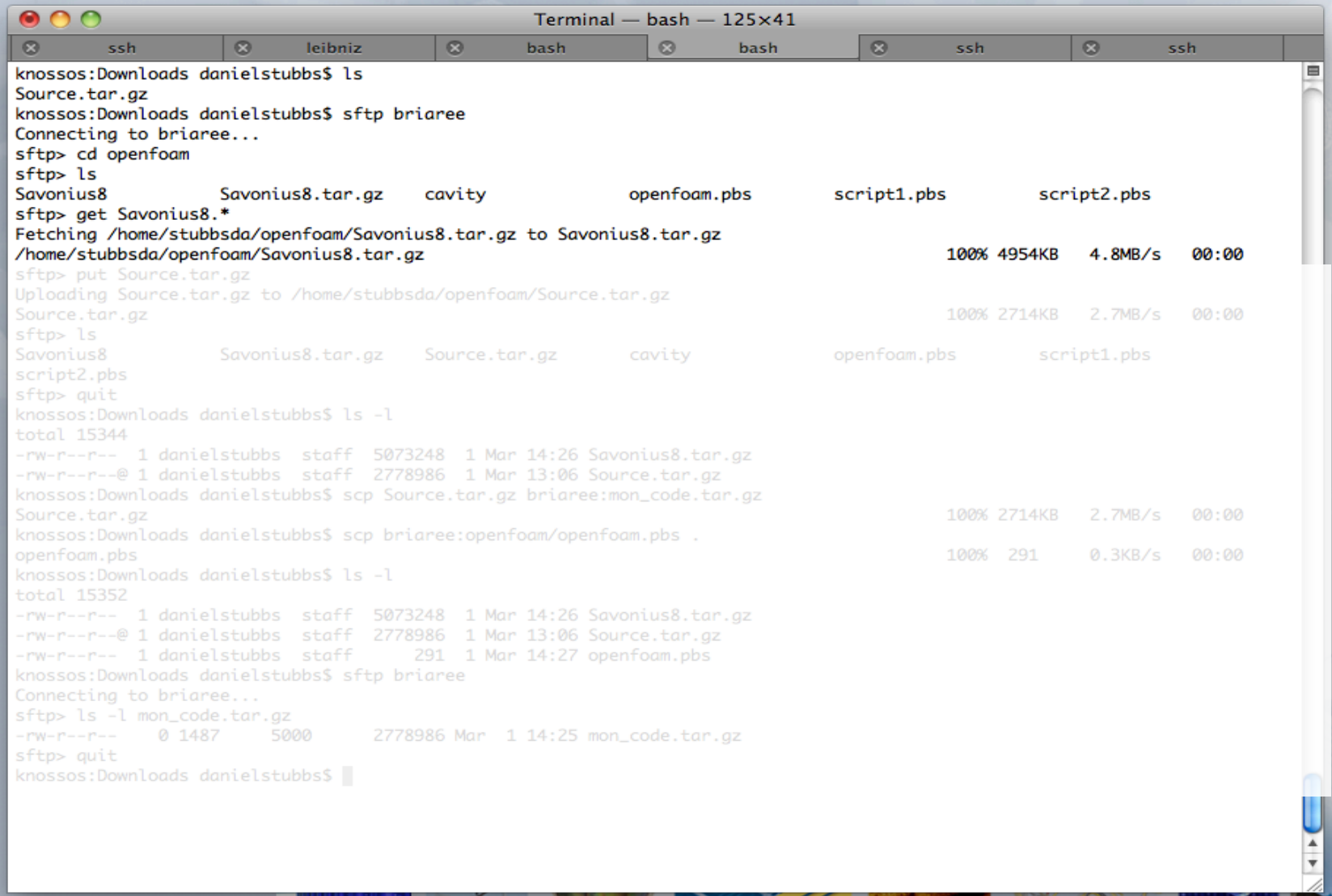
Calcul Québec



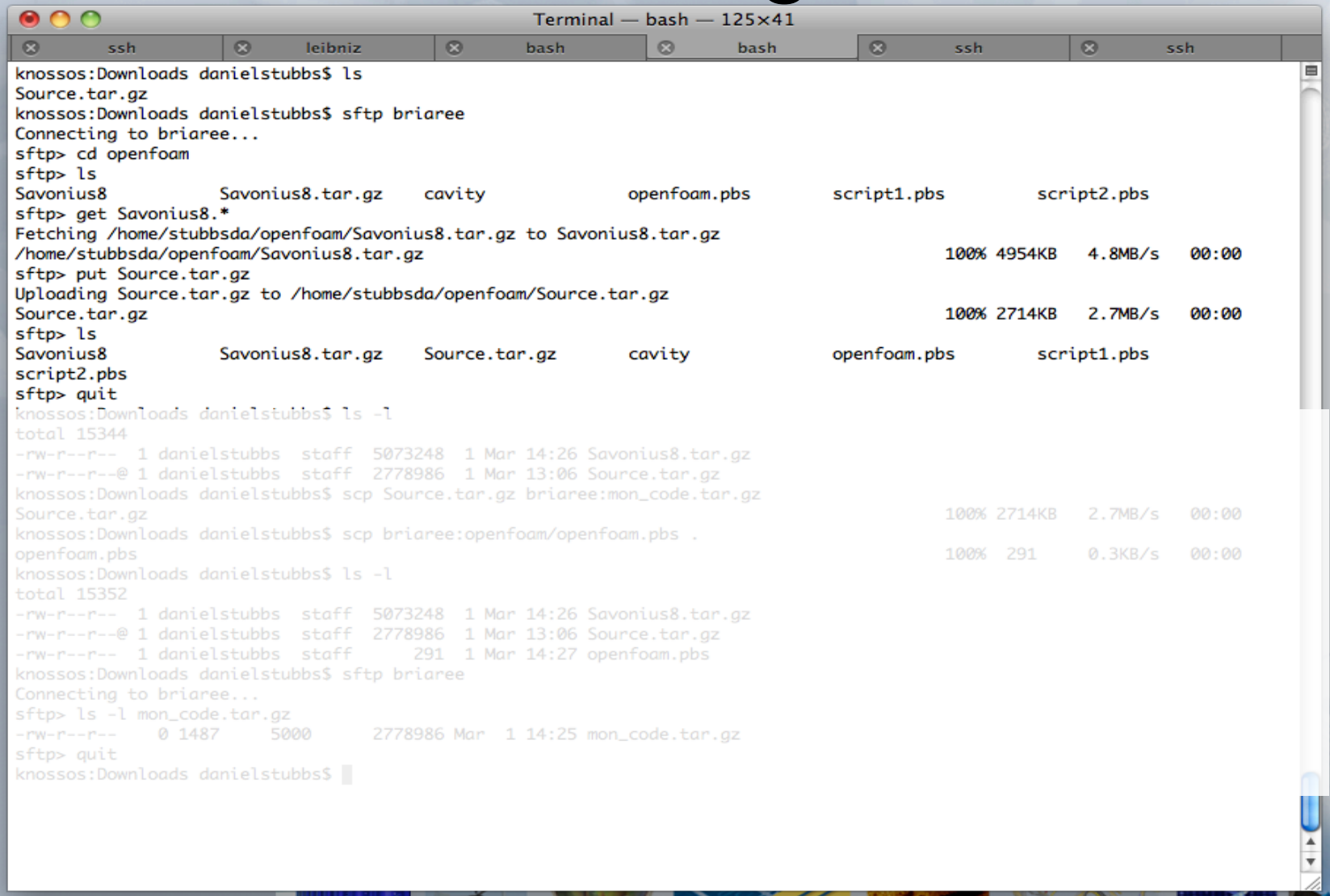
Transferring Files



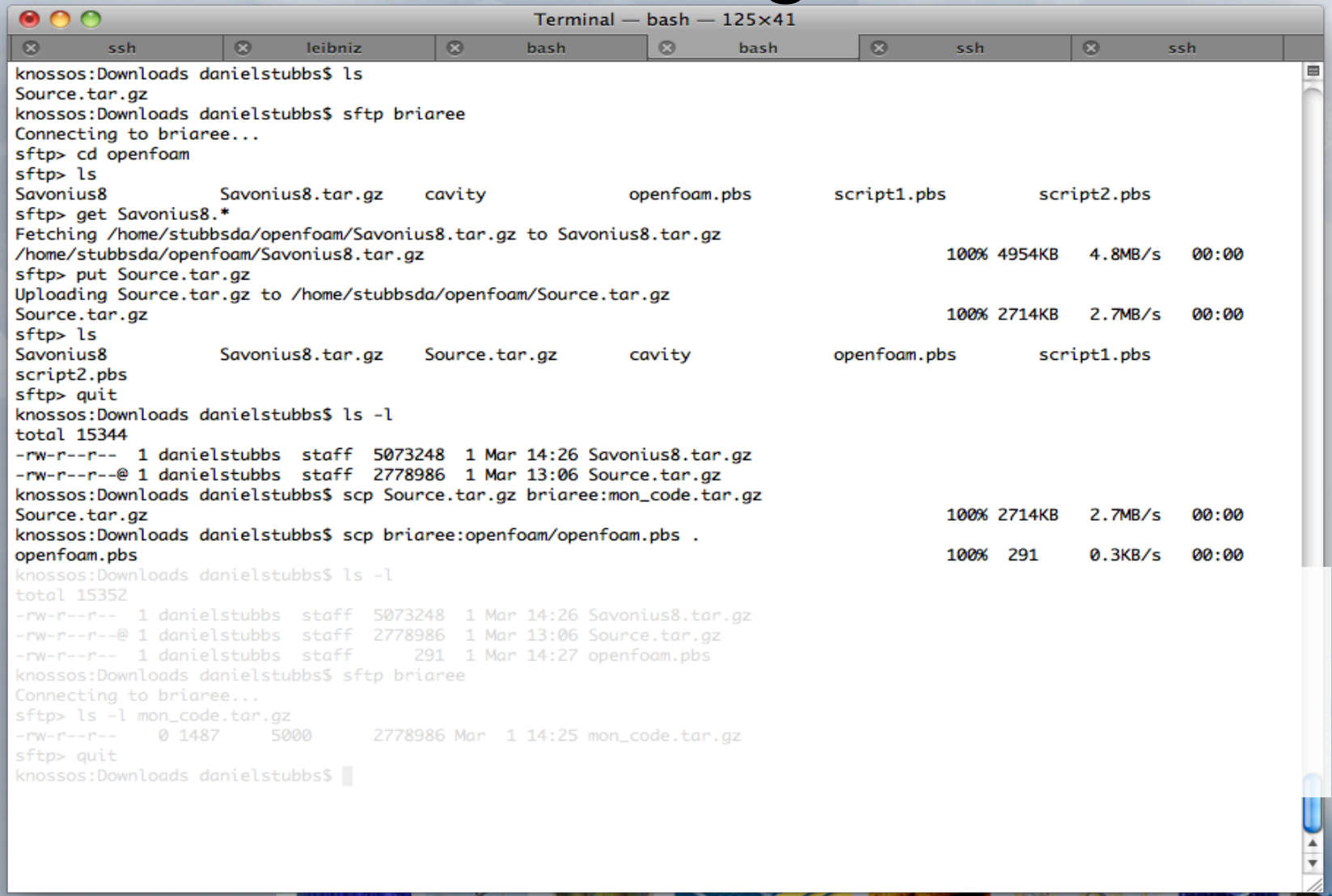
Transferring Files



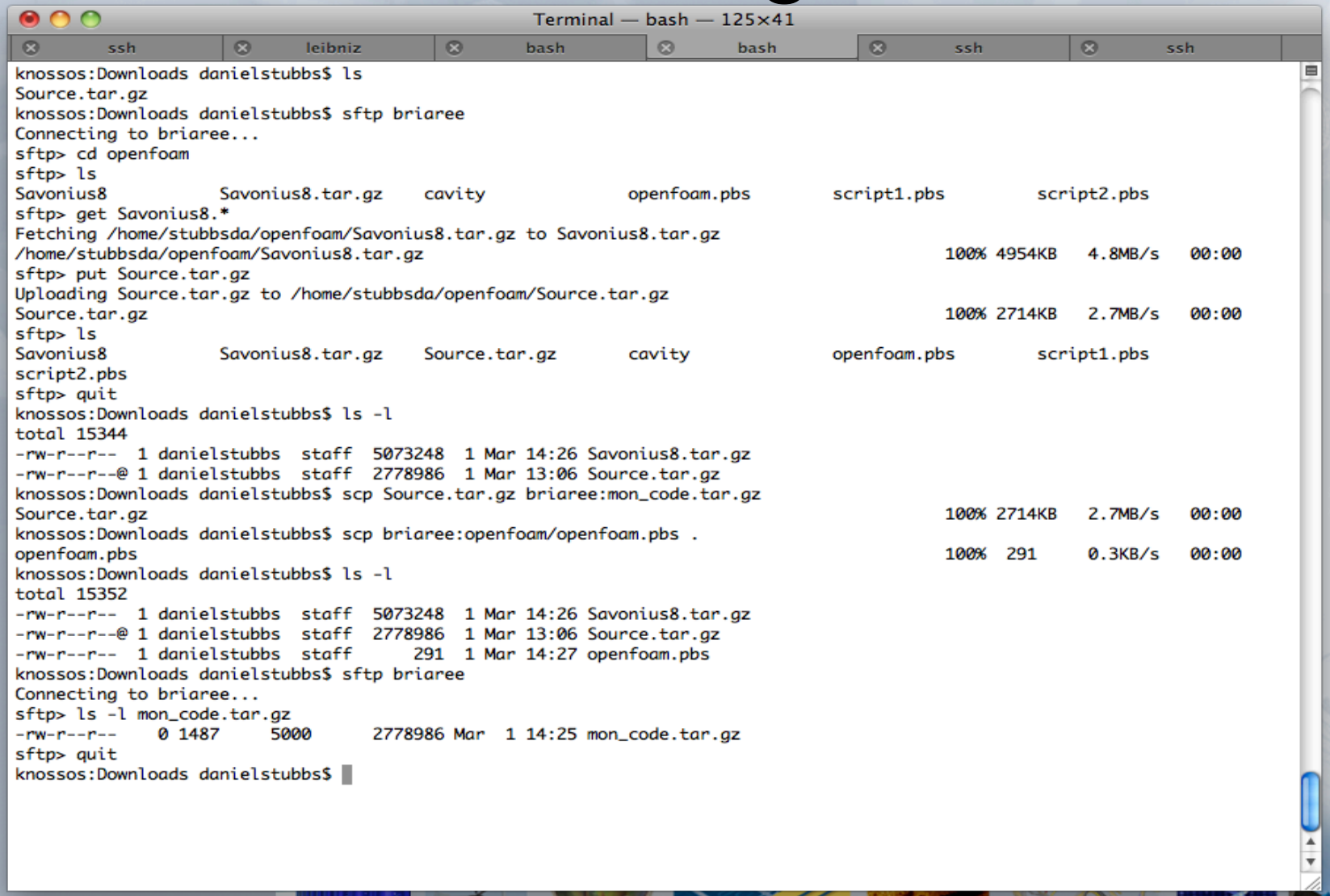
Transferring Files



Transferring Files



Transferring Files

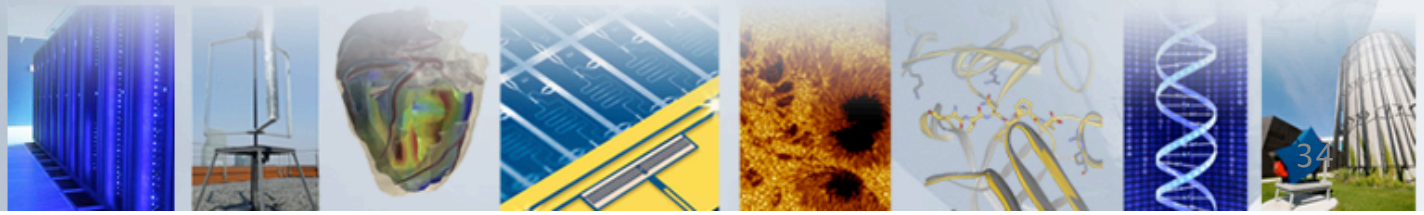


Bash

- In the Unix vocabulary, the "shell" (which interprets your commands) is a software layer which is a bridge between the kernel and the user.
- There are several different shells in existence for Linux but the default on Calcul Québec servers is the bash (Bourne Again Shell).
- You can customize this shell environment by modifying the file .bashrc in your $HOME.
- You can for instance create your own shortcuts for common commands, such as:

```
alias ll='ls -l'
```

Calcul Québec



Bash

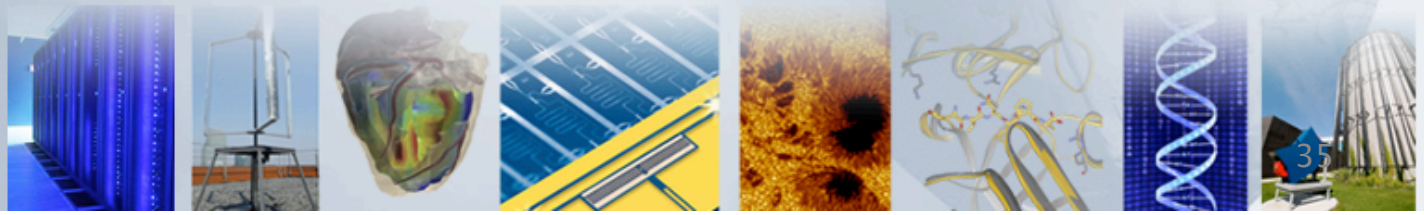
- In other cases, you can modify the value of an *environment variable*.
- Some environment variables are already defined for you, like $HOME, $PATH and, on Calcul Québec machines, $SCRATCH.
- To see the current value of an environment variable you can type

```
echo $variable_name
```

- If this variable isn't defined, the system returns nothing.
- To give a value to a variable you use the command

```
export variable_name=value
```

Calcul Québec



Bash

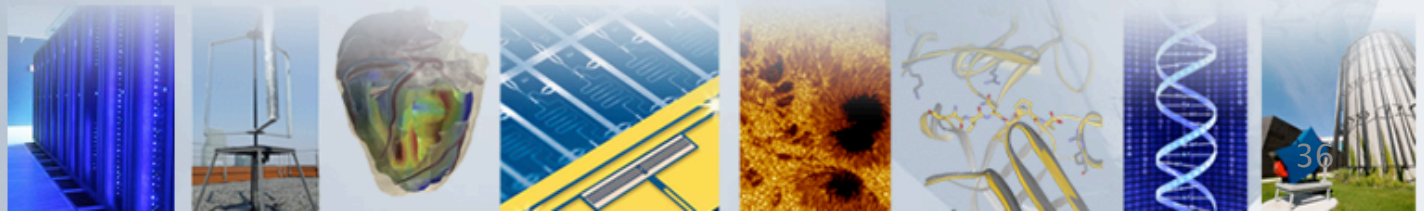
- One of the most important environment variables is the $PATH.
- It determines where the operating system will look for the applications that you call.
- The $PATH is a list of directories separated by a colon (:).
- When you type a command, Linux will search each one of these directories in a sequential order until it finds the command.
- You can see where Linux found the command by typing

```
which command_name
```

- To append a directory to your PATH,

```
export PATH=$PATH:/new/directory
```

Calcul Québec



Bash

Terminal — ssh — 120×41

ssh | leibniz | bash | ssh | ssh | ssh

```
[stubbsda@briaree1:~/openfoam]$ ls
cavity  openfoam.pbs  Savonius8  Savonius8.tar.gz  script1.pbs  script2.pbs
[stubbsda@briaree1:~/openfoam]$ ll
-bash: ll: command not found
[stubbsda@briaree1:~/openfoam]$ alias ll='ls -l'
[stubbsda@briaree1:~/openfoam]$ ll
total 5000
drwxr-xr-x  5 stubbsda analyste    8192 Feb 26 09:38 cavity
-rw-r--r--  1 stubbsda analyste     291 Oct 11 13:48 openfoam.pbs
drwxr-xr-x 13 stubbsda analyste    8192 Aug  6  2012 Savonius8
-rw-r--r--  1 stubbsda analyste 5073248 Aug  6  2012 Savonius8.tar.gz
-rwxr-x---  1 stubbsda analyste     221 Aug  6  2012 script1.pbs
-r--r--r--  1 stubbsda analyste     216 Sep 27 12:13 script2.pbs
[stubbsda@briaree1:~/openfoam]$ echo $CC

[stubbsda@briaree1:~/openfoam]$ export CC=gcc
[stubbsda@briaree1:~/openfoam]$ echo $CC
gcc
[stubbsda@briaree1:~/openfoam]$ echo $HOME
/RQusagers/stubbsda
[stubbsda@briaree1:~/openfoam]$ echo $PATH
/home/apps/Logiciels/VisualVM/visualvm_134/bin:/home/apps/Logiciels/Java/jdk1.7.0_05/bin:/home/apps/Logiciels/MPI/intel/
openmpi/1.4.4/bin:/home/apps/intel/composerxe-2011.4.191/bin/intel64:/opt/torque/x86_64/bin:/usr/local/bin:/bin:/usr/bin
:/usr/local/sbin:/usr/sbin:/sbin:/home/apps/bin:/usr/lpp/mmfs/bin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/sbin:/home/rqchpp
bs/maui/3.3.1/maui-3.3.1/bin:/usr/local/bin:/RQusagers/stubbsda/bin
[stubbsda@briaree1:~/openfoam]$ which gcc
/usr/bin/gcc
[stubbsda@briaree1:~/openfoam]$ which mpirun
/home/apps/Logiciels/MPI/intel/openmpi/1.4.4/bin/mpirun
[stubbsda@briaree1:~/openfoam]$ export PATH=$PATH:/RQusagers/stubbsda/openfoam
[stubbsda@briaree1:~/openfoam]$ echo $PATH
/home/apps/Logiciels/VisualVM/visualvm_134/bin:/home/apps/Logiciels/Java/jdk1.7.0_05/bin:/home/apps/Logiciels/MPI/intel/
openmpi/1.4.4/bin:/home/apps/intel/composerxe-2011.4.191/bin/intel64:/opt/torque/x86_64/bin:/usr/local/bin:/bin:/usr/bin
:/usr/local/sbin:/usr/sbin:/sbin:/home/apps/bin:/usr/lpp/mmfs/bin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/sbin:/home/rqchpp
bs/maui/3.3.1/maui-3.3.1/bin:/usr/local/bin:/RQusagers/stubbsda/bin:/RQusagers/stubbsda/openfoam
[stubbsda@briaree1:~/openfoam]$ which script1.pbs
~/openfoam/script1.pbs
[stubbsda@briaree1:~/openfoam]$
```

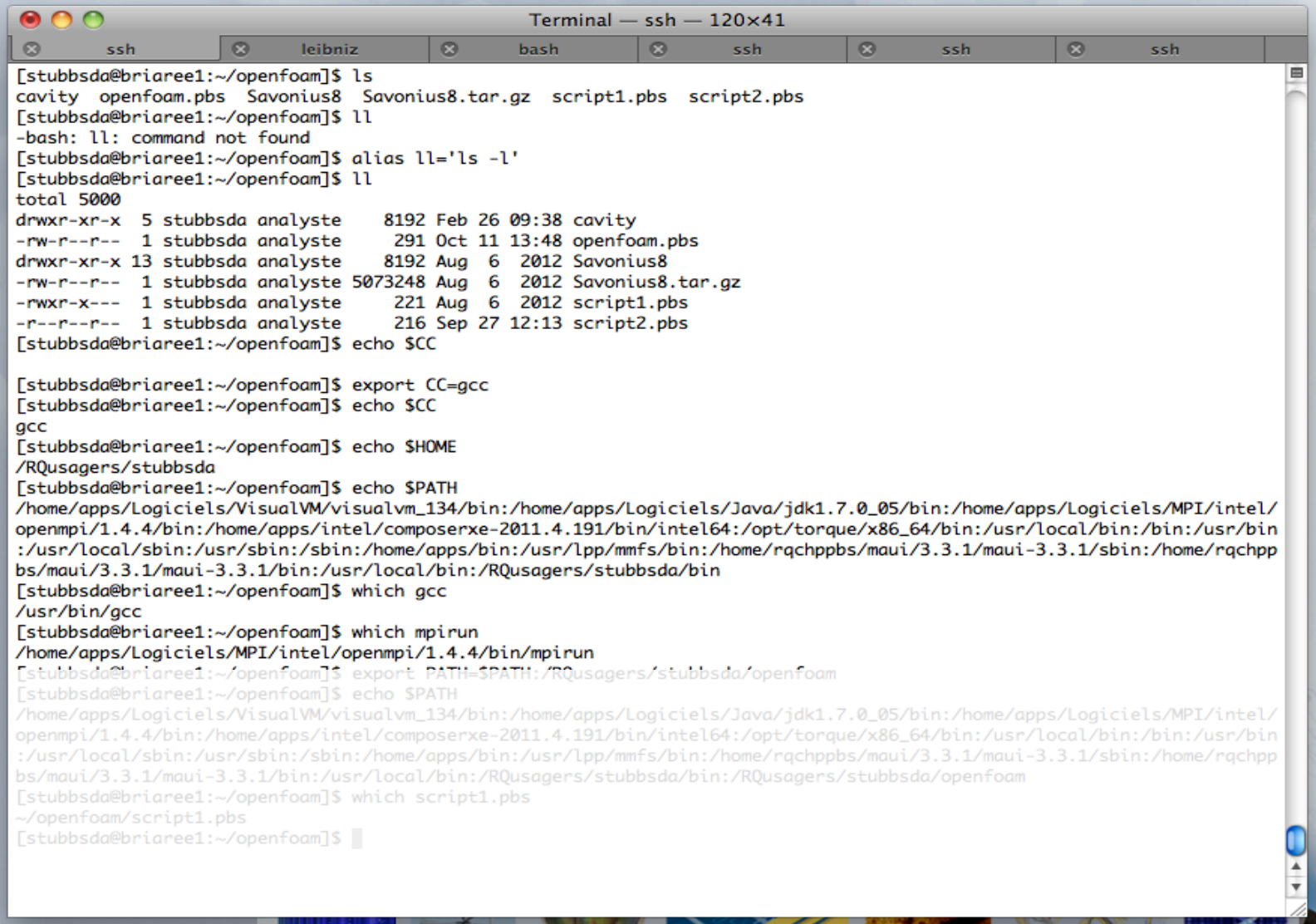
Bash

Terminal — ssh — 120×41

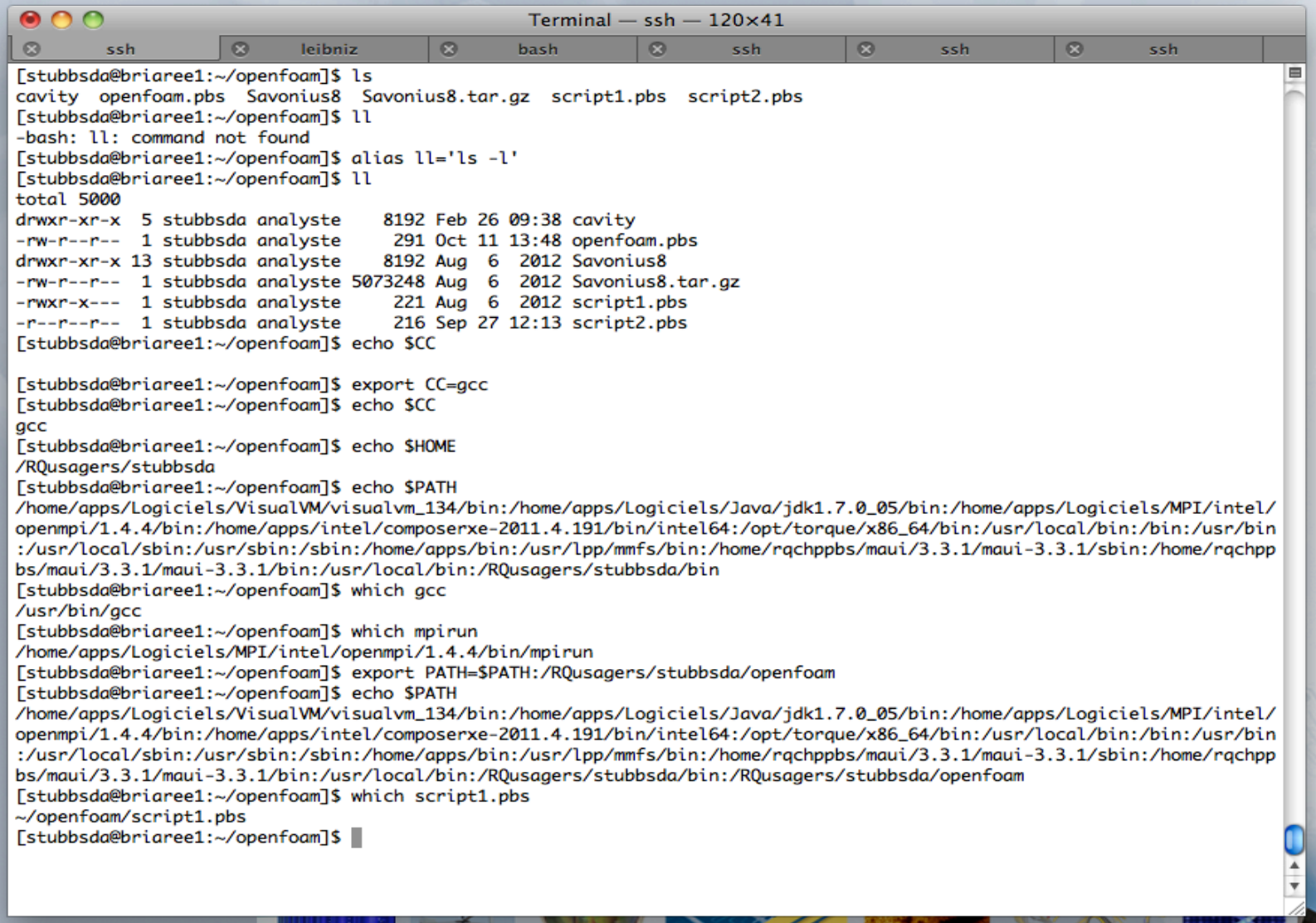
ssh | leibniz | bash | ssh | ssh | ssh

```
[stubbsda@briaree1:~/openfoam]$ ls
cavity  openfoam.pbs  Savonius8  Savonius8.tar.gz  script1.pbs  script2.pbs
[stubbsda@briaree1:~/openfoam]$ ll
-bash: ll: command not found
[stubbsda@briaree1:~/openfoam]$ alias ll='ls -l'
[stubbsda@briaree1:~/openfoam]$ ll
total 5000
drwxr-xr-x  5 stubbsda analyste    8192 Feb 26 09:38 cavity
-rw-r--r--  1 stubbsda analyste     291 Oct 11 13:48 openfoam.pbs
drwxr-xr-x 13 stubbsda analyste    8192 Aug  6  2012 Savonius8
-rw-r--r--  1 stubbsda analyste 5073248 Aug  6  2012 Savonius8.tar.gz
-rwxr-x---  1 stubbsda analyste     221 Aug  6  2012 script1.pbs
-r--r--r--  1 stubbsda analyste     216 Sep 27 12:13 script2.pbs
[stubbsda@briaree1:~/openfoam]$ echo $CC

[stubbsda@briaree1:~/openfoam]$ export CC=gcc
[stubbsda@briaree1:~/openfoam]$ echo $CC
gcc
[stubbsda@briaree1:~/openfoam]$ echo $HOME
/RQusagers/stubbsda
[stubbsda@briaree1:~/openfoam]$ echo $PATH
/home/apps/Logiciels/VisualVM/visualvm_134/bin:/home/apps/Logiciels/Java/jdk1.7.0_05/bin:/home/apps/Logiciels/MPI/intel/
openmpi/1.4.4/bin:/home/apps/intel/composerxe-2011.4.191/bin/intel64:/opt/torque/x86_64/bin:/usr/local/bin:/bin:/usr/bin
:/usr/local/sbin:/usr/sbin:/sbin:/home/apps/bin:/usr/lpp/mmfs/bin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/sbin:/home/rqchpp
bs/maui/3.3.1/maui-3.3.1/bin:/usr/local/bin:/RQusagers/stubbsda/bin
[stubbsda@briaree1:~/openfoam]$ which gcc
/usr/bin/gcc
[stubbsda@briaree1:~/openfoam]$ which mpirun
/home/apps/Logiciels/MPI/intel/openmpi/1.4.4/bin/mpirun
[stubbsda@briaree1:~/openfoam]$ export PATH=$PATH:/RQusagers/stubbsda/openfoam
[stubbsda@briaree1:~/openfoam]$ echo $PATH
/home/apps/Logiciels/VisualVM/visualvm_134/bin:/home/apps/Logiciels/Java/jdk1.7.0_05/bin:/home/apps/Logiciels/MPI/intel/
openmpi/1.4.4/bin:/home/apps/intel/composerxe-2011.4.191/bin/intel64:/opt/torque/x86_64/bin:/usr/local/bin:/bin:/usr/bin
:/usr/local/sbin:/usr/sbin:/sbin:/home/apps/bin:/usr/lpp/mmfs/bin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/sbin:/home/rqchpp
bs/maui/3.3.1/maui-3.3.1/bin:/usr/local/bin:/RQusagers/stubbsda/bin:/RQusagers/stubbsda/openfoam
[stubbsda@briaree1:~/openfoam]$ which script1.pbs
~/openfoam/script1.pbs
[stubbsda@briaree1:~/openfoam]$
```

Bash


Terminal — ssh — 120×41
ssh
leibniz
bash
ssh
ssh
ssh
[stubbsda@briaree1:~/openfoam]$ ls
cavity openfoam.pbs Savonius8 Savonius8.tar.gz script1.pbs script2.pbs
[stubbsda@briaree1:~/openfoam]$ ll
-bash: ll: command not found
[stubbsda@briaree1:~/openfoam]$ alias ll='ls -l'
[stubbsda@briaree1:~/openfoam]$ ll
total 5000
drwxr-xr-x 5 stubbsda analyste   8192 Feb 26 09:38 cavity
-rw-r--r-- 1 stubbsda analyste   291 Oct 11 13:48 openfoam.pbs
drwxr-xr-x 13 stubbsda analyste   8192 Aug 6 2012 Savonius8
-rw-r--r-- 1 stubbsda analyste 5073248 Aug 6 2012 Savonius8.tar.gz
-rwxr-x--- 1 stubbsda analyste   221 Aug 6 2012 script1.pbs
-r--r--r-- 1 stubbsda analyste   216 Sep 27 12:13 script2.pbs
[stubbsda@briaree1:~/openfoam]$ echo $CC
[stubbsda@briaree1:~/openfoam]$ export CC=gcc
[stubbsda@briaree1:~/openfoam]$ echo $CC
gcc
[stubbsda@briaree1:~/openfoam]$ echo $HOME
/RQusagers/stubbsda
[stubbsda@briaree1:~/openfoam]$ echo $PATH
/home/apps/Logiciels/VisualVM/visualvm_134/bin:/home/apps/Logiciels/Java/jdk1.7.0_05/bin:/home/apps/Logiciels/MPI/intel/openmpi/1.4.4/bin:/home/apps/intel/composerxe-2011.4.191/bin/intel64:/opt/torque/x86_64/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/apps/bin:/usr/lpp/mmfs/bin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/sbin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/bin:/usr/local/bin:/RQusagers/stubbsda/bin
[stubbsda@briaree1:~/openfoam]$ which gcc
/usr/bin/gcc
[stubbsda@briaree1:~/openfoam]$ which mpirun
/home/apps/Logiciels/MPI/intel/openmpi/1.4.4/bin/mpirun
[stubbsda@briaree1:~/openfoam]$ export PATH=$PATH:/RQusagers/stubbsda/openfoam
[stubbsda@briaree1:~/openfoam]$ echo $PATH
/home/apps/Logiciels/VisualVM/visualvm_134/bin:/home/apps/Logiciels/Java/jdk1.7.0_05/bin:/home/apps/Logiciels/MPI/intel/openmpi/1.4.4/bin:/home/apps/intel/composerxe-2011.4.191/bin/intel64:/opt/torque/x86_64/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/apps/bin:/usr/lpp/mmfs/bin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/sbin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/bin:/usr/local/bin:/RQusagers/stubbsda/bin:/RQusagers/stubbsda/openfoam
[stubbsda@briaree1:~/openfoam]$ which script1.pbs
~/openfoam/script1.pbs
[stubbsda@briaree1:~/openfoam]$

Bash

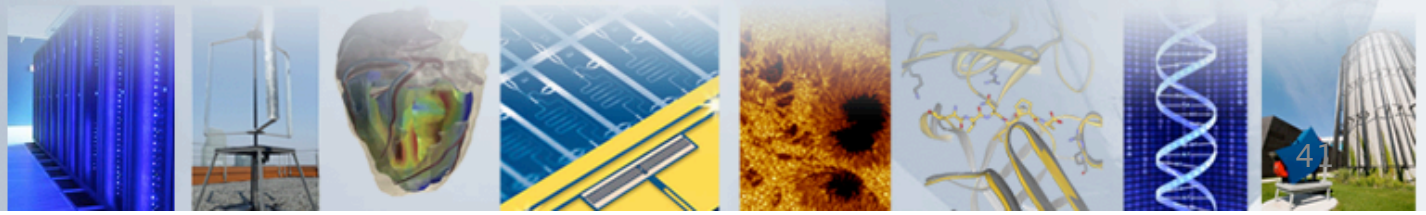

Terminal — ssh — 120×41
ssh
leibniz
bash
ssh
ssh
ssh
[stubbsda@briaree1:~/openfoam]$ ls
cavity openfoam.pbs Savonius8 Savonius8.tar.gz script1.pbs script2.pbs
[stubbsda@briaree1:~/openfoam]$ ll
-bash: ll: command not found
[stubbsda@briaree1:~/openfoam]$ alias ll='ls -l'
[stubbsda@briaree1:~/openfoam]$ ll
total 5000
drwxr-xr-x 5 stubbsda analyste   8192 Feb 26 09:38 cavity
-rw-r--r-- 1 stubbsda analyste   291 Oct 11 13:48 openfoam.pbs
drwxr-xr-x 13 stubbsda analyste   8192 Aug 6 2012 Savonius8
-rw-r--r-- 1 stubbsda analyste 5073248 Aug 6 2012 Savonius8.tar.gz
-rwxr-x--- 1 stubbsda analyste   221 Aug 6 2012 script1.pbs
-r--r--r-- 1 stubbsda analyste   216 Sep 27 12:13 script2.pbs
[stubbsda@briaree1:~/openfoam]$ echo $CC
[stubbsda@briaree1:~/openfoam]$ export CC=gcc
[stubbsda@briaree1:~/openfoam]$ echo $CC
gcc
[stubbsda@briaree1:~/openfoam]$ echo $HOME
/RQusagers/stubbsda
[stubbsda@briaree1:~/openfoam]$ echo $PATH
/home/apps/Logiciels/VisualVM/visualvm_134/bin:/home/apps/Logiciels/Java/jdk1.7.0_05/bin:/home/apps/Logiciels/MPI/intel/
openmpi/1.4.4/bin:/home/apps/intel/composerxe-2011.4.191/bin/intel64:/opt/torque/x86_64/bin:/usr/local/bin:/bin:/usr/bin
:/usr/local/sbin:/usr/sbin:/sbin:/home/apps/bin:/usr/lpp/mmfs/bin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/sbin:/home/rqchpp
bs/maui/3.3.1/maui-3.3.1/bin:/usr/local/bin:/RQusagers/stubbsda/bin
[stubbsda@briaree1:~/openfoam]$ which gcc
/usr/bin/gcc
[stubbsda@briaree1:~/openfoam]$ which mpirun
/home/apps/Logiciels/MPI/intel/openmpi/1.4.4/bin/mpirun
[stubbsda@briaree1:~/openfoam]$ export PATH=$PATH:/RQusagers/stubbsda/openfoam
[stubbsda@briaree1:~/openfoam]$ echo $PATH
/home/apps/Logiciels/VisualVM/visualvm_134/bin:/home/apps/Logiciels/Java/jdk1.7.0_05/bin:/home/apps/Logiciels/MPI/intel/
openmpi/1.4.4/bin:/home/apps/intel/composerxe-2011.4.191/bin/intel64:/opt/torque/x86_64/bin:/usr/local/bin:/bin:/usr/bin
:/usr/local/sbin:/usr/sbin:/sbin:/home/apps/bin:/usr/lpp/mmfs/bin:/home/rqchppbs/maui/3.3.1/maui-3.3.1/sbin:/home/rqchpp
bs/maui/3.3.1/maui-3.3.1/bin:/usr/local/bin:/RQusagers/stubbsda/bin:/RQusagers/stubbsda/openfoam
[stubbsda@briaree1:~/openfoam]$ which script1.pbs
~/openfoam/script1.pbs
[stubbsda@briaree1:~/openfoam]$

Software Modules

- In general, we prefer that Calcul Québec staff install the software and libraries that you need.
- We use use the command `module` to adapt all the environment variables necessary for the use of this software.
- The most common options are the following:

```
module list
module avail
module load module_name
module unload module_name
module purge
module swap old_module new_module
```

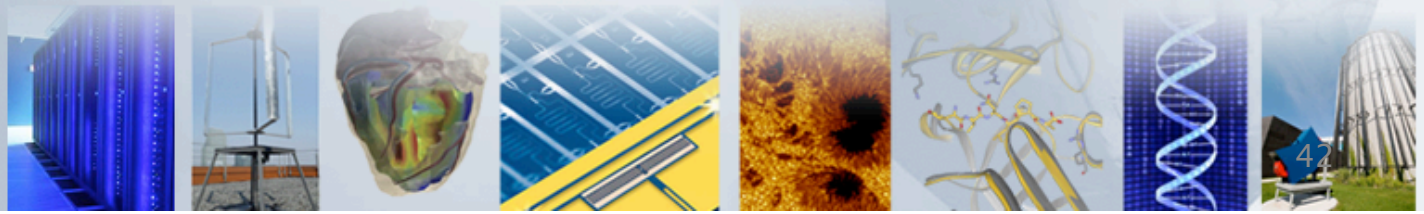
Calcul Québec



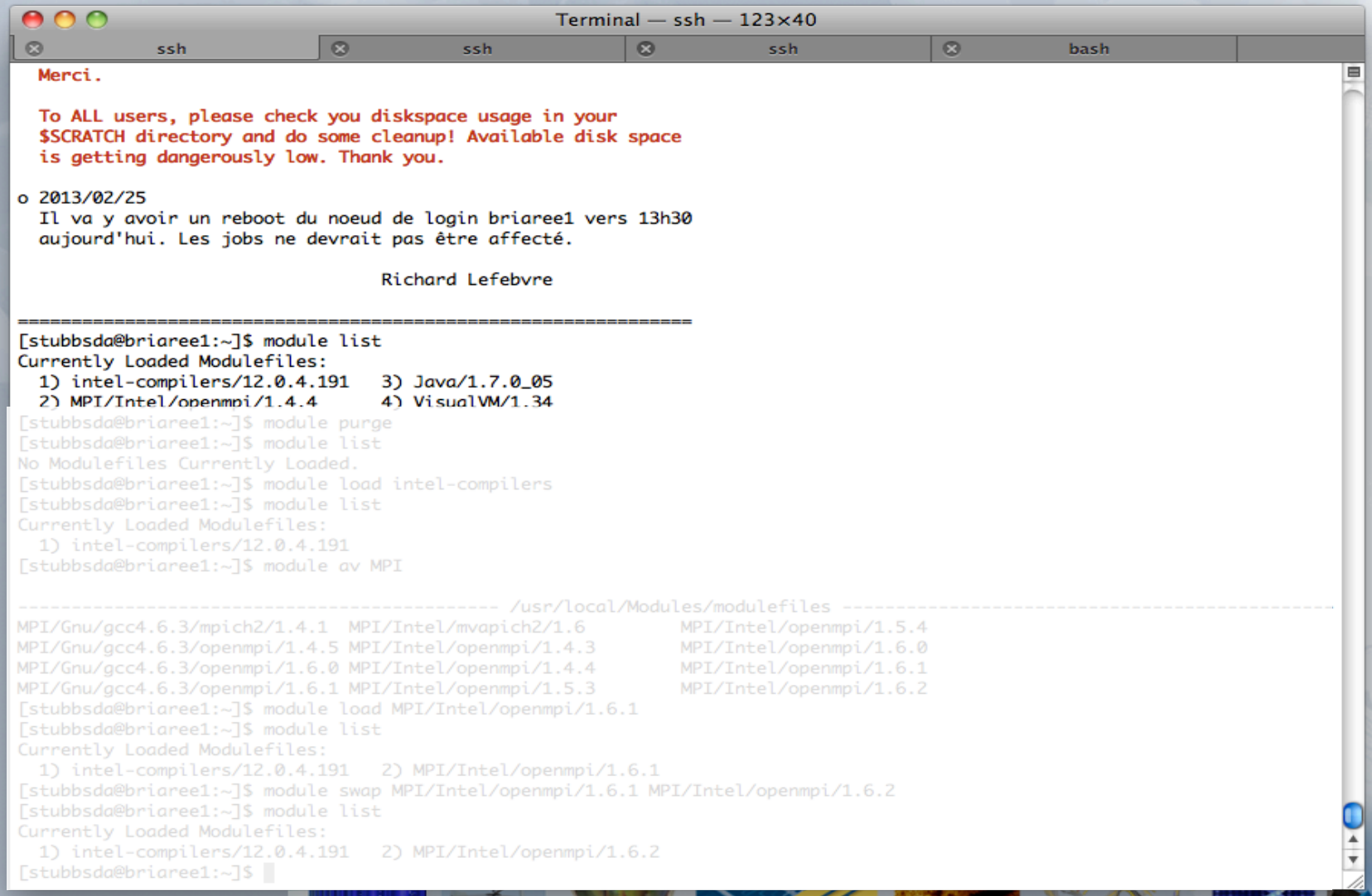
Software Modules

- With the module command, you can choose a particular version of a program while another user can use some other version.
- You can also automatically load modules by adding the line `module load` at the end of the .bashrc file in your $HOME.
- There are sometimes dependencies among modules that have to be satisfied, in which case you can first execute the command `module load A` and then `module load B`.

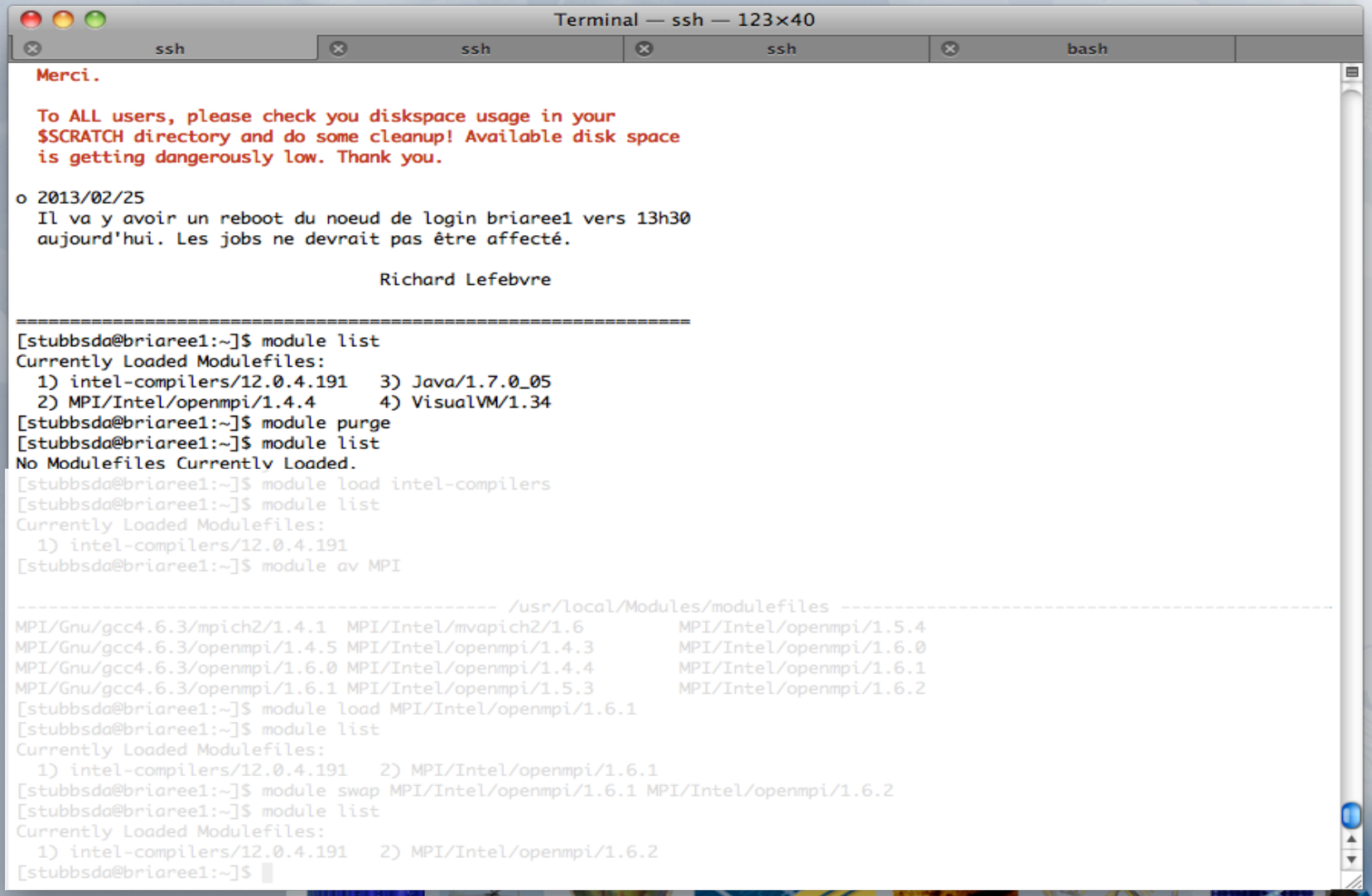
Calcul Québec



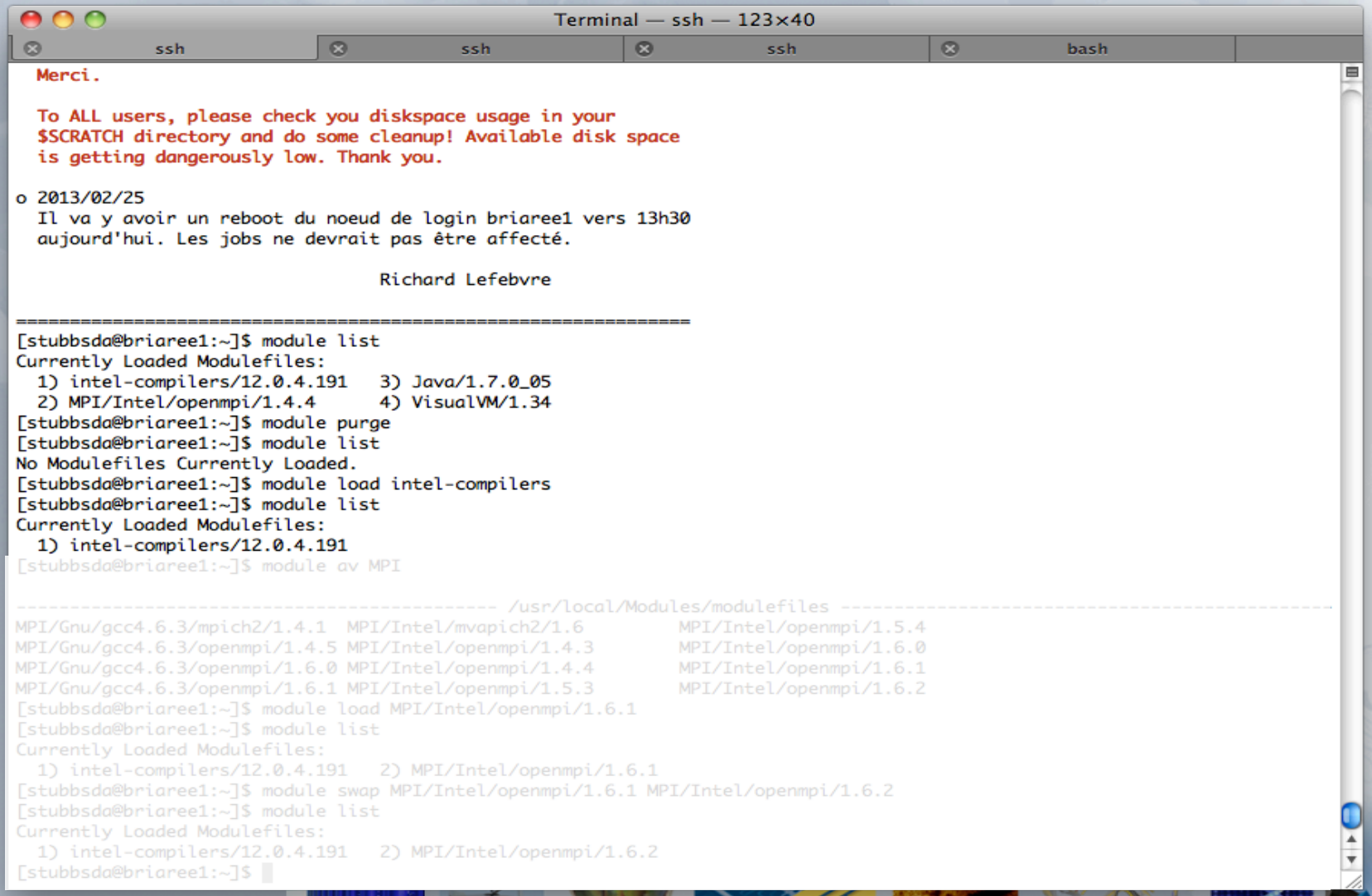
Software Modules



Software Modules



Software Modules



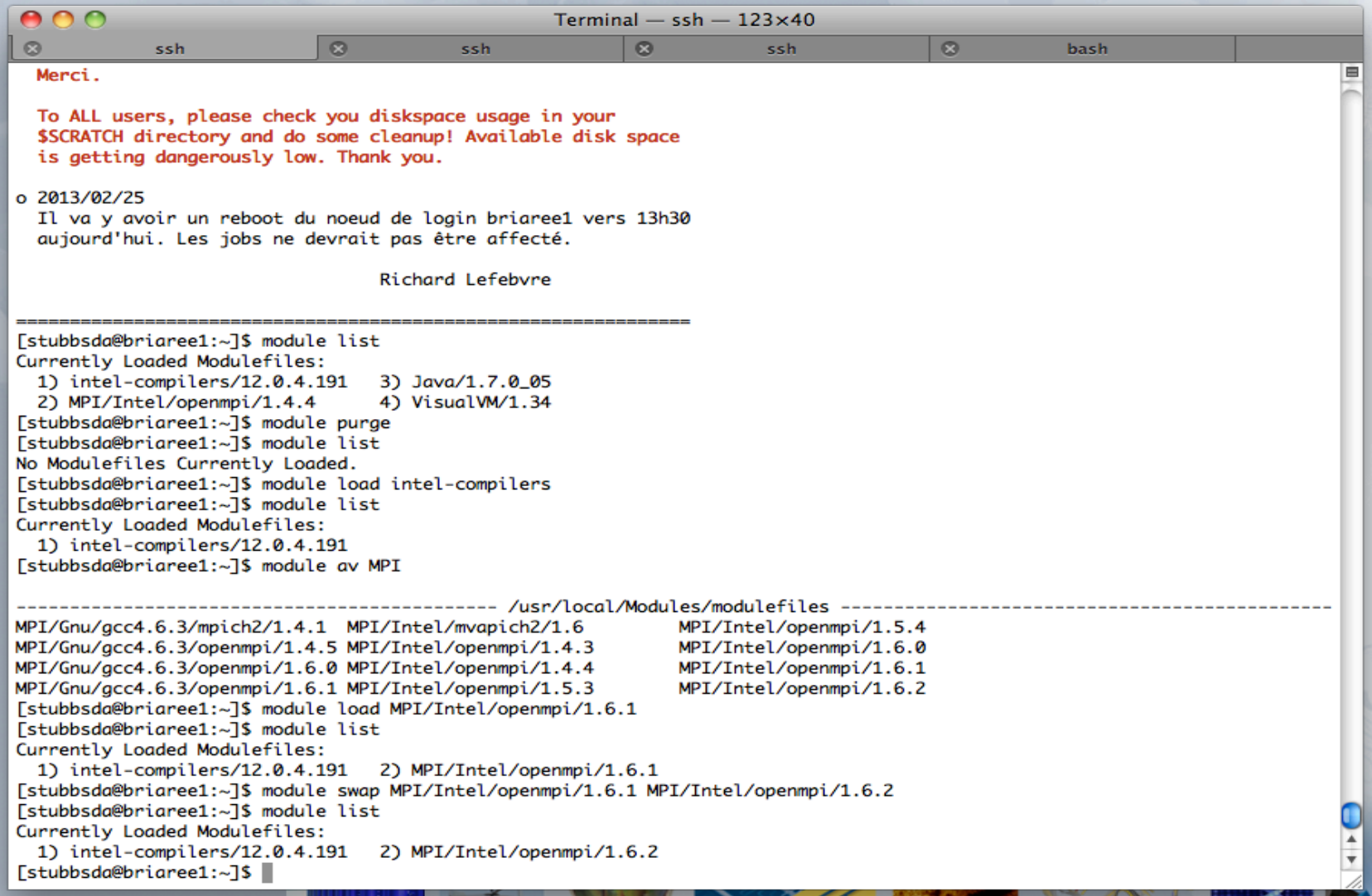
Software Modules

Terminal — ssh — 123×40

ssh | ssh | ssh | bash

```
  Merci.

  To ALL users, please check you diskspace usage in your
  $SCRATCH directory and do some cleanup! Available disk space
  is getting dangerously low. Thank you.

o 2013/02/25
  Il va y avoir un reboot du noeud de login briaree1 vers 13h30
  aujourd'hui. Les jobs ne devrait pas être affecté.

                                Richard Lefebvre

================================================================
[stubbsda@briaree1:~]$ module list
Currently Loaded Modulefiles:
  1) intel-compilers/12.0.4.191   3) Java/1.7.0_05
  2) MPI/Intel/openmpi/1.4.4      4) VisualVM/1.34
[stubbsda@briaree1:~]$ module purge
[stubbsda@briaree1:~]$ module list
No Modulefiles Currently Loaded.
[stubbsda@briaree1:~]$ module load intel-compilers
[stubbsda@briaree1:~]$ module list
Currently Loaded Modulefiles:
  1) intel-compilers/12.0.4.191
[stubbsda@briaree1:~]$ module av MPI

------------------------------------------- /usr/local/Modules/modulefiles -------------------------------------------
MPI/Gnu/gcc4.6.3/mpich2/1.4.1  MPI/Intel/mvapich2/1.6         MPI/Intel/openmpi/1.5.4
MPI/Gnu/gcc4.6.3/openmpi/1.4.5 MPI/Intel/openmpi/1.4.3        MPI/Intel/openmpi/1.6.0
MPI/Gnu/gcc4.6.3/openmpi/1.6.0 MPI/Intel/openmpi/1.4.4        MPI/Intel/openmpi/1.6.1
MPI/Gnu/gcc4.6.3/openmpi/1.6.1 MPI/Intel/openmpi/1.5.3        MPI/Intel/openmpi/1.6.2
[stubbsda@briaree1:~]$ module load MPI/Intel/openmpi/1.6.1
[stubbsda@briaree1:~]$ module list
Currently Loaded Modulefiles:
  1) intel-compilers/12.0.4.191   2) MPI/Intel/openmpi/1.6.1
[stubbsda@briaree1:~]$ module swap MPI/Intel/openmpi/1.6.1 MPI/Intel/openmpi/1.6.2
[stubbsda@briaree1:~]$ module list
Currently Loaded Modulefiles:
  1) intel-compilers/12.0.4.191   2) MPI/Intel/openmpi/1.6.2
[stubbsda@briaree1:~]$
```

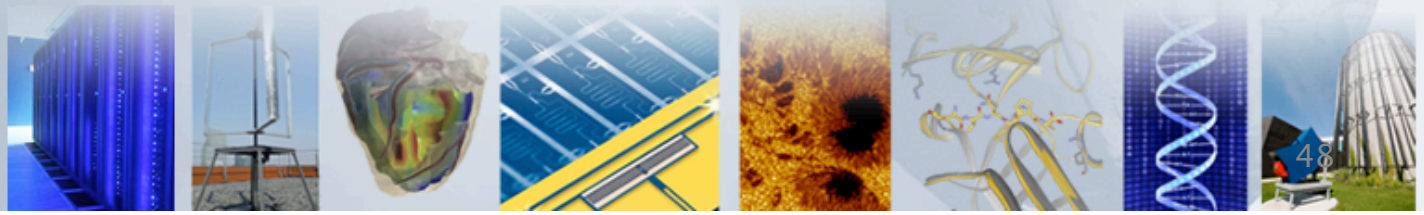
Software Modules



Compiling your Code

- Several of the modules discussed earlier concern compilers and development environments for languages like C, C++ and Java.
- The most widely used compilers for Linux are Gnu (gcc and g++) and Intel (icc and icpc).
- You're free to choose the compiler (or Java environment) and version that best suit your needs.
- You can also load modules corresponding to various external libraries that your code makes use of, such as CPLEX.
- Compiling your code can be relatively simple if it's just a single source file.

Calcul Québec

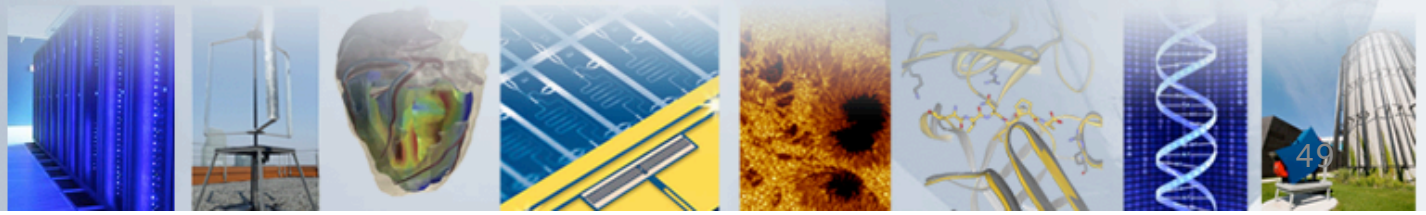


Compiling your Code

- A command like

```
g++ -O3 -march=native -o my_prog main.cpp -lcplex -lm
```

  may well suffice in this case.
- If you have multiple source and header files with various dependencies among them, the best way to handle the build process is using a makefile.
- This is a plain text file named `makefile` that is in the build directory for your code.
- This file contains a set of rules specifying how to build the object files (*.o) and then link these together along with any external libraries to create the binary file.

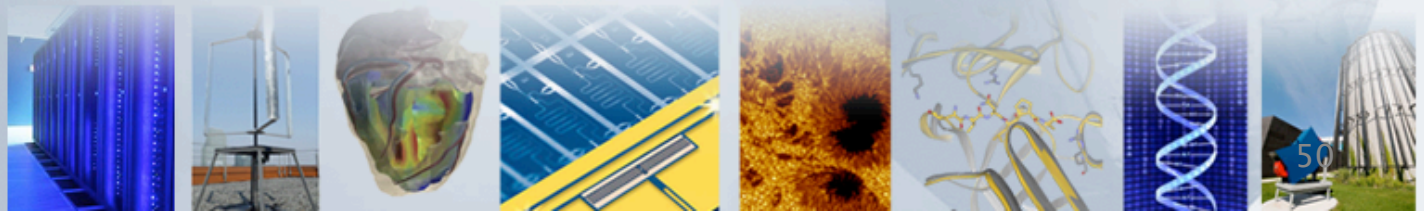


Calcul Québec

Compiling your Code

```
OBJS = code1.o code2.o main.o
CXX = g++
LIBS = -lcplex -lboost_system -lm
CXX_FLAGS = -O3 -march=native -Wall

prog: $(OBJS)
        $(CXX) $(CXX_FLAGS) -o prog $(OBJS) $(LIBS)

code1.o: code1.cpp global.h
        $(CXX) $(CXX_FLAGS) -c code1.cpp

code2.o: code2.cpp code2.h global.h
        $(CXX) $(CXX_FLAGS) -c code2.cpp

main.o: main.cpp global.h
        $(CXX) $(CXX_FLAGS) -c main.cpp
clean:
        rm -f $(OBJS)
        rm -f prog
```

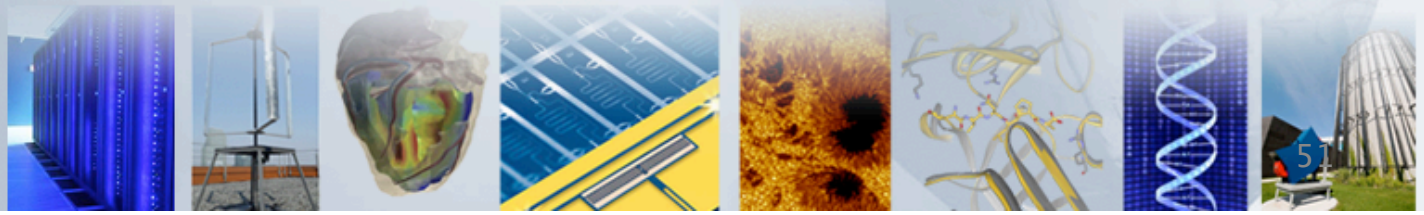
Calcul Québec



Compiling your Code

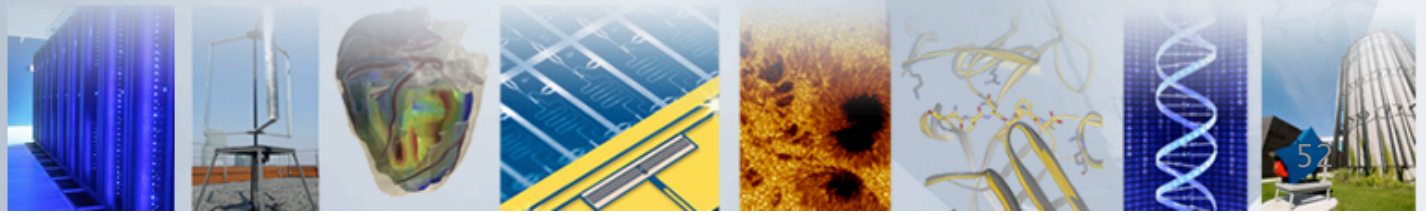
- This was a very simple makefile for a project involving just three source files and two header files but a makefile can be as complicated as needed.
- Note that any modules which you needed for building your program, such as external libraries, will also need to be loaded in order to run the resulting binary.
- If you're doing significant amounts of code development, it would be wise to consider learning a version control tool like Git, Mercurial or Subversion.

Calcul Québec



Using CPLEX

- The cluster Briarée at the Université de Montréal has several different CPLEX modules available for use.
- The latest version 12.7.0 is however unavailable due to the age of Briarée.
- You can define and solve an optimization model using `oplrun` or the C++ and Java interfaces that CPLEX provides.
- We will give a couple of examples here of using Java or C++ to solve a toy model.



Using CPLEX

Maximize

$x_1 + 2x_2 + 3x_3$

subject to

$-x_1 + x_2 + x_3 \leq 20$

$x_1 - 3x_2 + x_3 \leq 30$

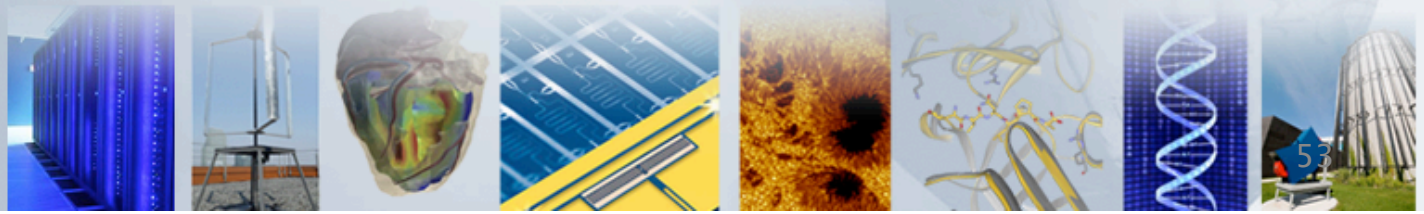
with these bounds

$0 \leq x_1 \leq 40$

$0 \leq x_2 < \infty$

$0 \leq x_3 < \infty$

Calcul Québec

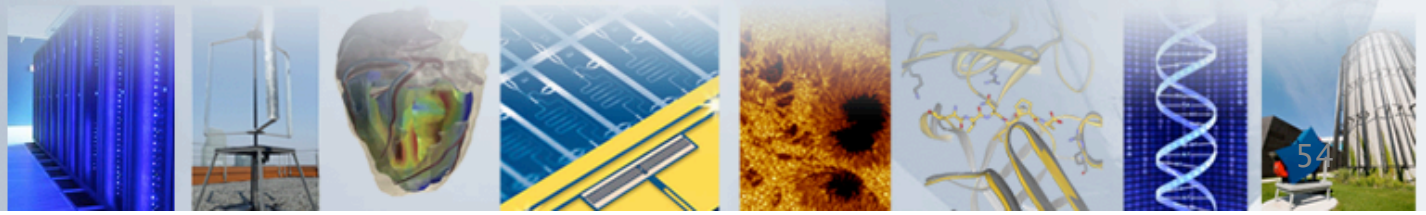


Using CPLEX

- We can solve this model in Java using the following code:

```
static void populateByRow(IloMPModeler model,IloNumVar[][]
   var,IloRange[][] rng) throws IloException
{
  double[] lb   = {0.0, 0.0, 0.0};
  double[] ub   = {40.0, Double.MAX_VALUE, Double.MAX_VALUE};
  String[] varname = {"x1", "x2", "x3"};
  IloNumVar[] x    = model.numVarArray(3, lb, ub, varname);
  var[0] = x;
  double[] objvals = {1.0, 2.0, 3.0};
  model.addMaximize(model.scalProd(x, objvals));
  rng[0] = new IloRange[2];
  rng[0][0] = model.addLe(model.sum(model.prod(-1.0,
   x[0]),model.prod( 1.0, x[1]),model.prod( 1.0, x[2])), 20.0,
   "c1");
  rng[0][1] = model.addLe(model.sum(model.prod( 1.0,
   x[0]),model.prod(-3.0, x[1]),model.prod( 1.0, x[2])), 30.0,
   "c2");
}
```

Calcul Québec

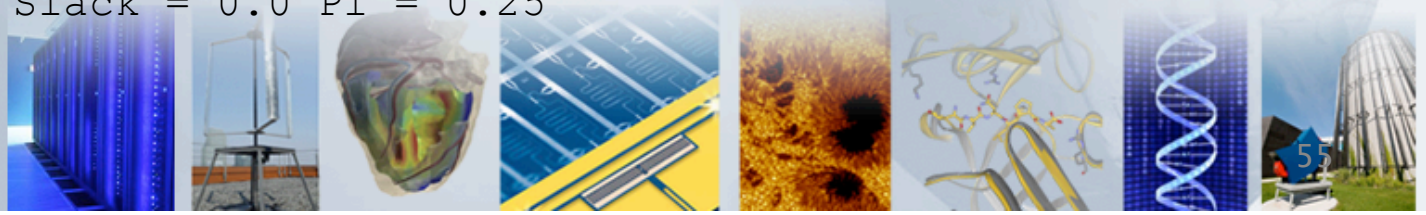


Using CPLEX

- We can compile and run it as follows:

```
$ javac  -classpath $CLASSPATH -O -d . LPex1.java
$ java  -d64 -Djava.library.path=$LD_LIBRARY_PATH -classpath
   $CLASSPATH: LPex1 -r
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time = 0.01 sec. (0.00 ticks)

   Iteration log . . .
Iteration: 1   Dual infeasibility =            0.000000
Iteration: 2   Dual objective =          202.500000
Solution status = Optimal
Solution value  = 202.5
Variable 0: Value = 40.0 Reduced cost = 3.5
Variable 1: Value = 17.5 Reduced cost = -0.0
Variable 2: Value = 42.5 Reduced cost = -0.0
Constraint 0: Slack = 0.0 Pi = 2.75
Constraint 1: Slack = 0.0 Pi = 0.25
```

Calcul Québec

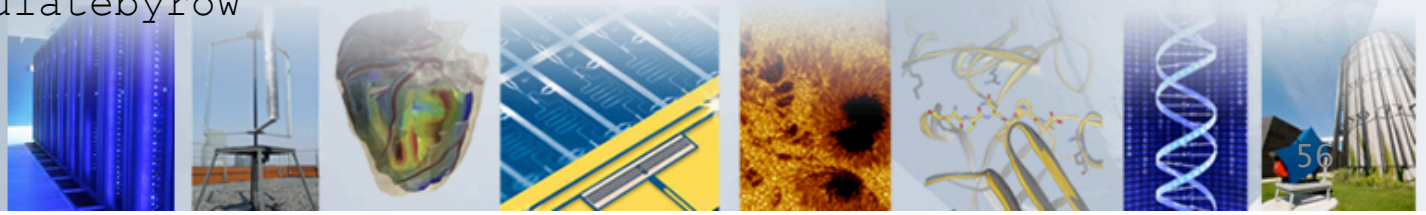


Using CPLEX

- To solve this model using C++ we can use this program:

```
static void populatebyrow (IloModel model, IloNumVarArray x,
   IloRangeArray c)
{
  IloEnv env = model.getEnv();
  x.add(IloNumVar(env, 0.0, 40.0));
  x.add(IloNumVar(env));
  x.add(IloNumVar(env));
  model.add(IloMaximize(env, x[0] + 2 * x[1] + 3 * x[2]));
  c.add( - x[0] +     x[1] + x[2] <= 20);
  c.add(   x[0] - 3 * x[1] + x[2] <= 30);
  x[0].setName("x1");
  x[1].setName("x2");
  x[2].setName("x3");
  c[0].setName("c1");
  c[1].setName("c2");
  model.add(c);
}  // END populatebyrow
```

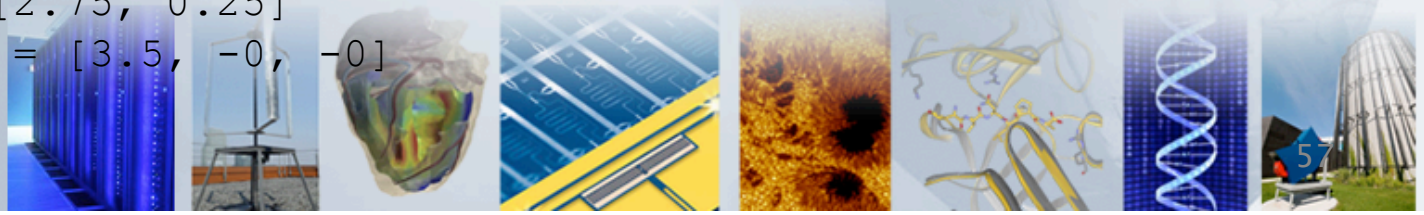
Calcul Québec



Using CPLEX

- To compile and run it we use the commands:

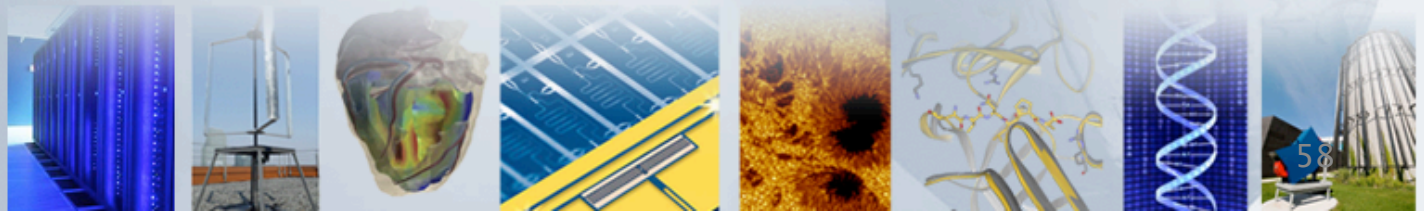
```
$ g++ -c -m64 -O -fPIC -fno-strict-aliasing -fexceptions -
   DIL_STD  ilolpex1.cpp -o ilolpex1.o
$ g++ -m64 -O -fPIC -fno-strict-aliasing -fexceptions -DIL_STD -o
   ilolpex1 ilolpex1.o -lconcert -lilocplex -lcplex -lm -lpthread
$ ./ilolpex1 -r
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time = 0.00 sec. (0.00 ticks)

   Iteration log . . .
Iteration: 1   Dual infeasibility =            0.000000
Iteration: 2   Dual objective =          202.500000
Solution status = Optimal
Solution value  = 202.5
Values        = [40, 17.5, 42.5]
Slacks        = [0, 0]
Duals         = [2.75, 0.25]
Reduced Costs = [3.5, -0, -0]
```



Using CPLEX

- The method in both cases is to prepare a small source code file containing the model, to compile it with the appropriate flags and options and then to execute it from the command line.
- On a Calcul Québec server like Briarée, preparing the source file and compiling can be done interactively on the login node but the final step should be done inside a job.
- You can also use the login node for very brief (a minute or two) tests to check that your compiled binary starts cleanly.
- In the next section we'll see how to submit a job.

Calcul Québec



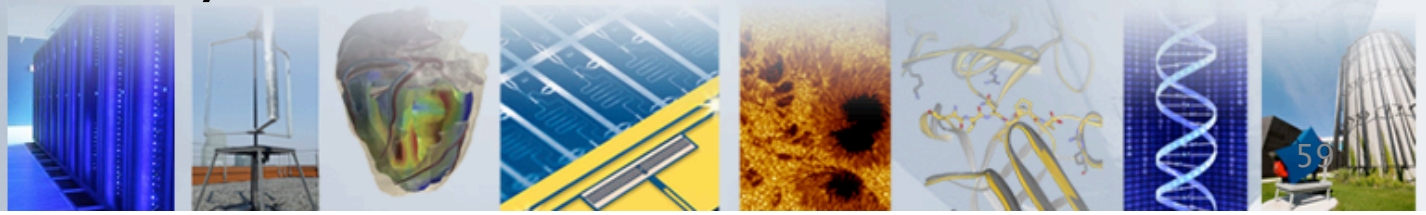
Exercises

- You should begin by connecting to Briarée using either your own account (if you have one) or the following guest account:

Username = `user06`
Password = `Red17path#3`
Machine = `briaree1.calculquebec.ca`

- Once you're connected, follow these steps:

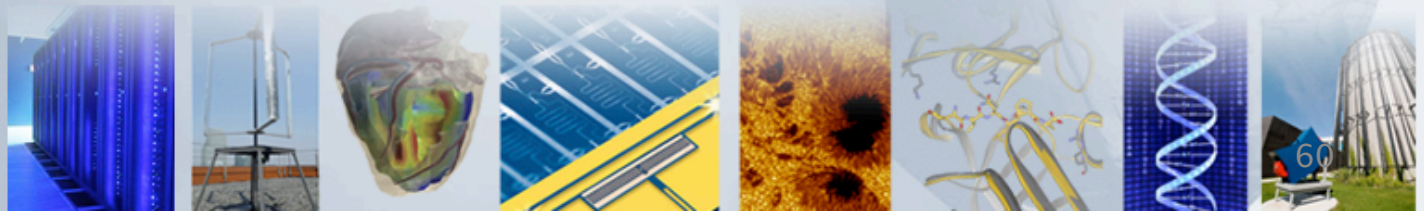
1. Create a directory using your name and enter it.
2. Copy the contents of the directory `/tmp/cplex-exercises` to this directory.
3. Enter the directory `cpp` or `java` and use the Makefile there to compile and run the sample problem: remember to load the necessary modules.



Job Submission

- The machine that you login to with ssh is what's called the login or head node of the cluster.
- This node is the gate to the cluster for everyone and is thus not at all appropriate for your computations which should take place on the cluster's compute nodes.
- You use a text editor to create a small file called a job script and which specifies the resources needed for the job as well as the actions to be performed, line by line.
- Once this file has been created you can submit the job to the scheduler by the command `qsub script.pbs`

Calcul Québec



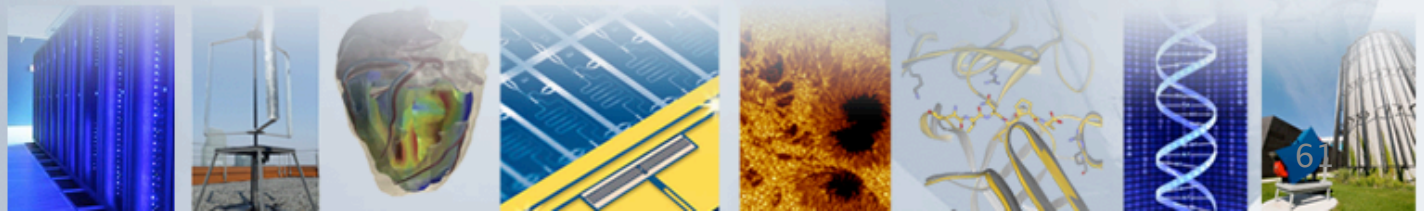
Job Submission

- To see the current state of the cluster you can use the command `qstat`.
- If you want to delete one of your jobs you can use the command

`qdel job_id`

- On Briarée you can use the command `pbs_free` to see how many processors are free.
- Some limits for the Calcul Québec machines at the UdeM: a job cannot run for more than 168 hours and if your job requires more than 48 processors you need to demonstrate that the software can use them efficiently.
- You can submit as many jobs as you want.

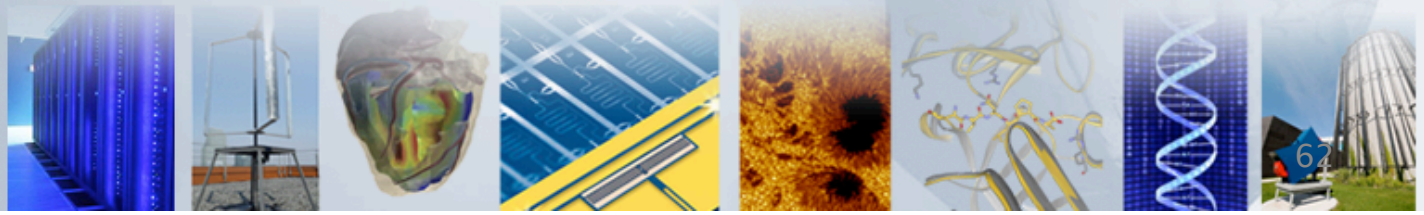
Calcul Québec



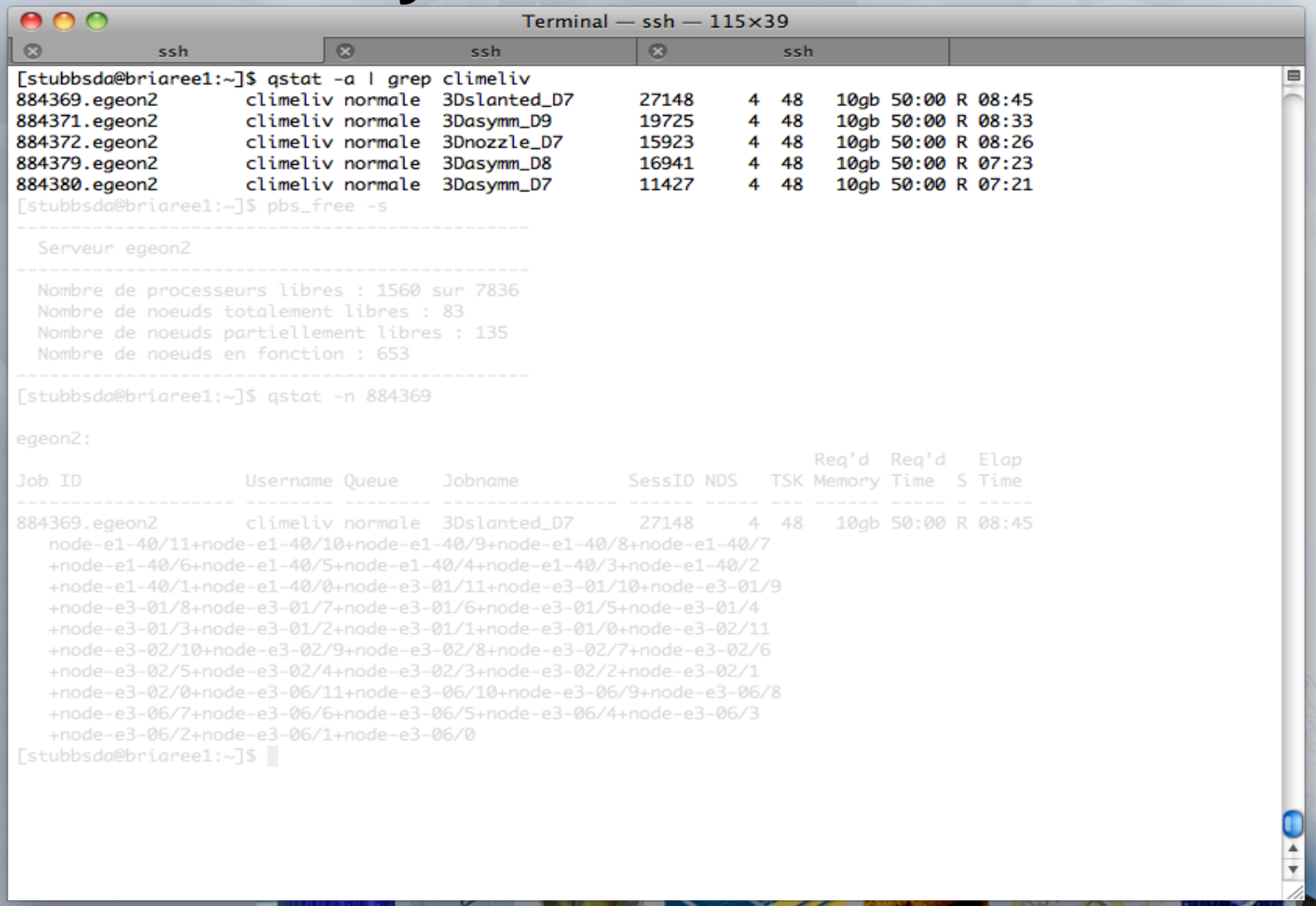
Job Submission

```
#!/bin/bash
#PBS -l walltime=52:00:00
#PBS -l nodes=1:ppn=12
#PBS -l mem=12gb
#PBS -j oe
#PBS -r n
#PBS -o output.txt

module load cplex_studio/12.6.0

cd $SCRATCH/research

./my_code p1 p2 > output.dat
```

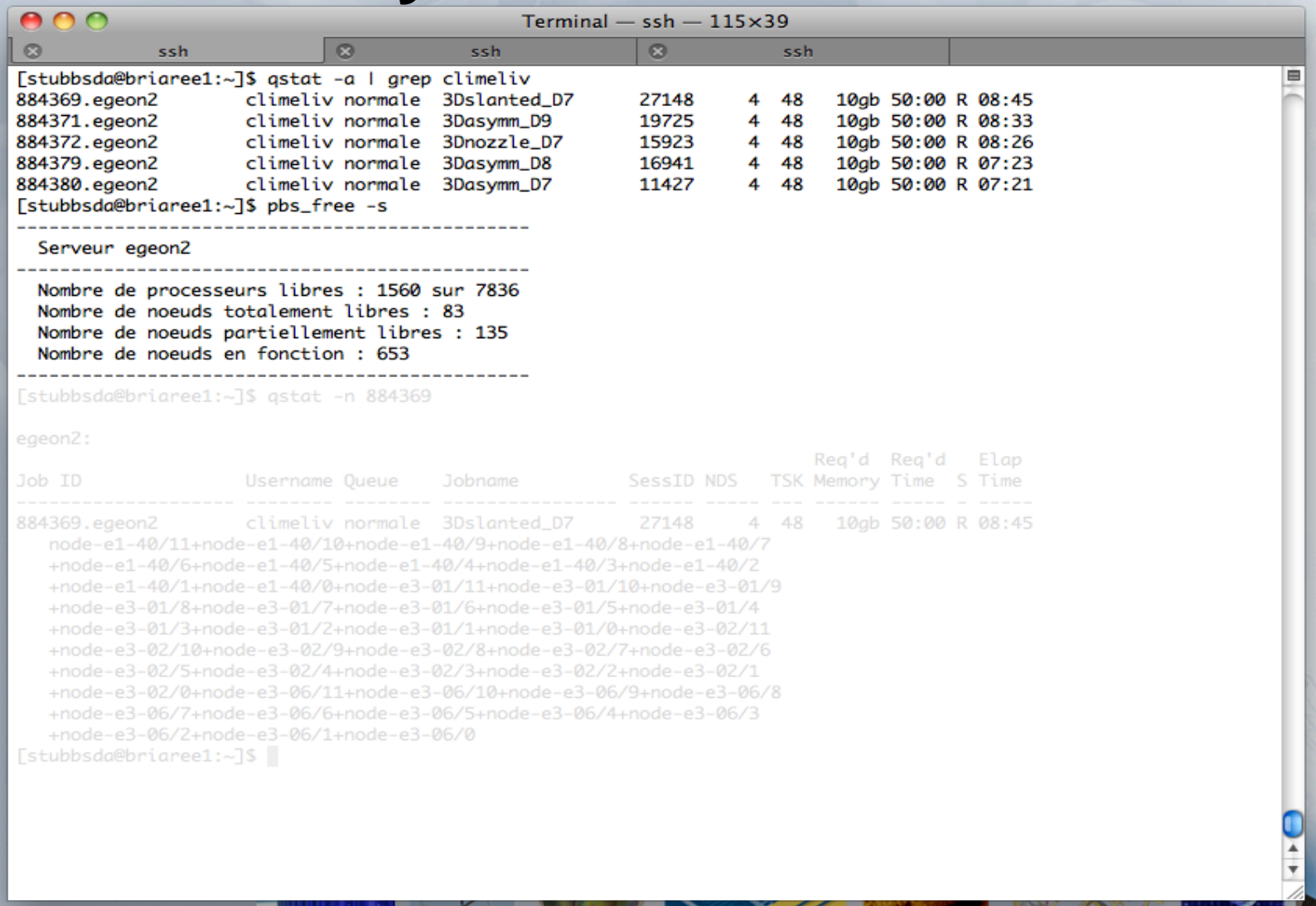
Calcul Québec



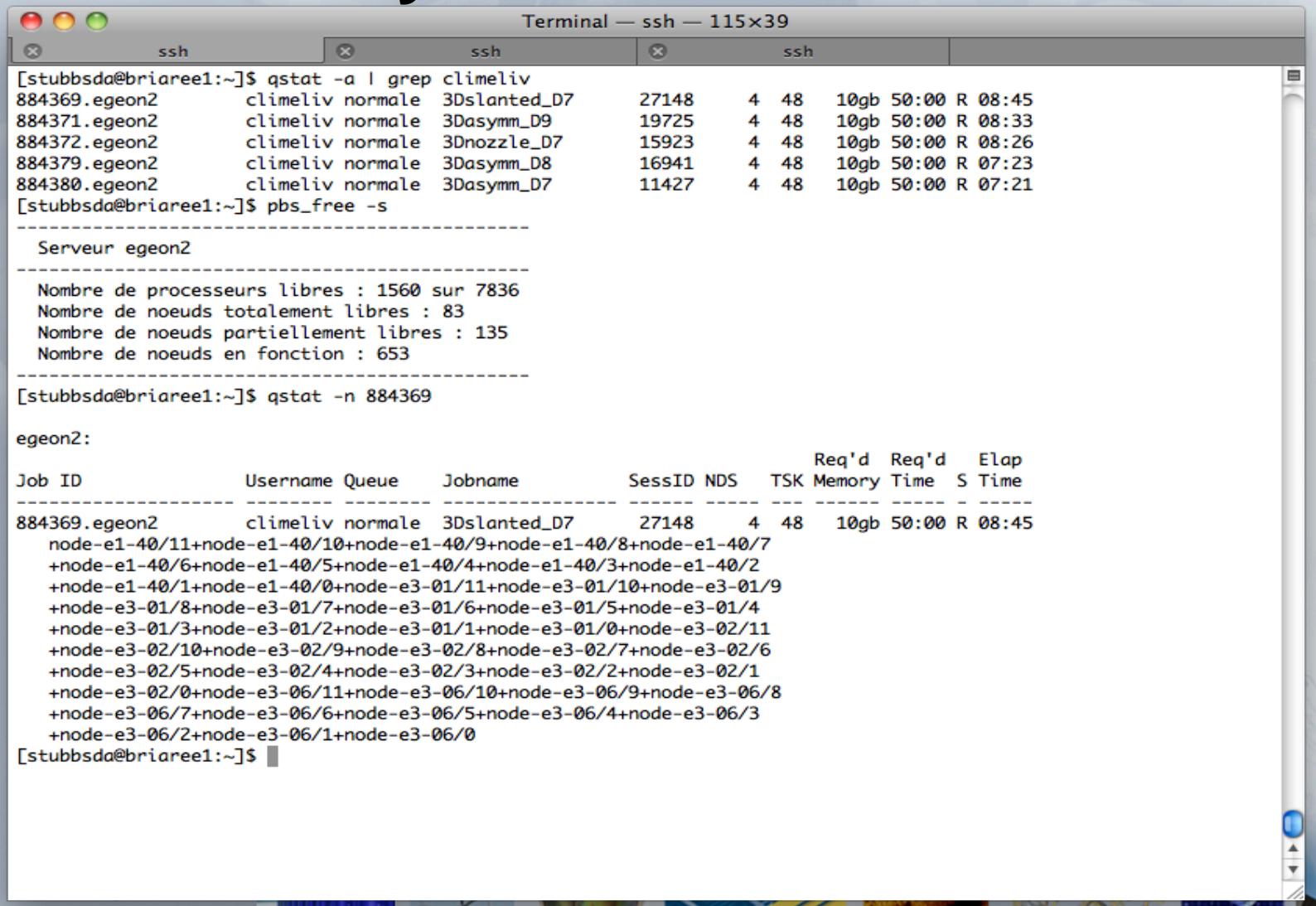
Job Submission



Job Submission



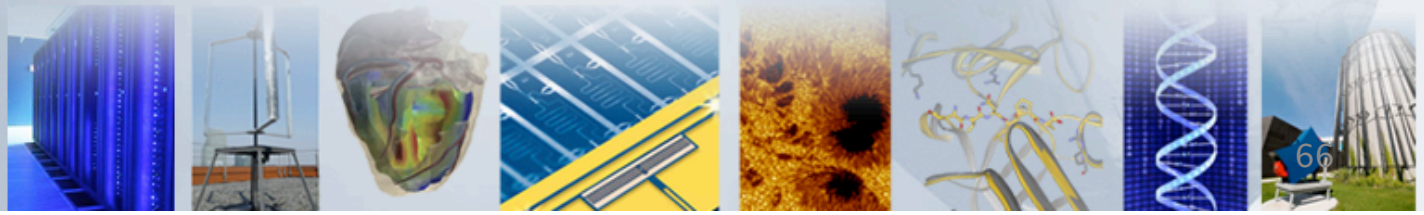
Job Submission



Job Submission

- A common situation is to run the same (serial) program with many different parameters, each of these calculations being independent of the others.
- While you can submit hundreds of individual jobs, one for each set of parameter values, there are alternative ways of organizing such calculations.
- One approach is to pack 12 calculations into a single job, since there are 12 CPU cores on a Briarée node.
- To do this we will have 12 lines in the job script and add a command at the end to wait for them to complete.

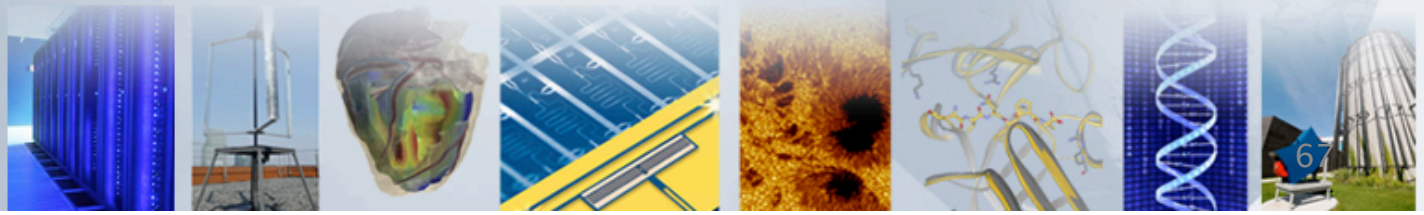
Calcul Québec



Job Submission

```
#!/bin/bash
#PBS -l walltime=10:00:00
#PBS -l nodes=1:ppn=12
#PBS -j oe
#PBS -r n
#PBS -o output.txt

module load software/2.3

cd research

./my_prog p1 &
./my_prog p2 &
./my_prog p3 &
.
.
.
./my_prog p12 &
wait
```

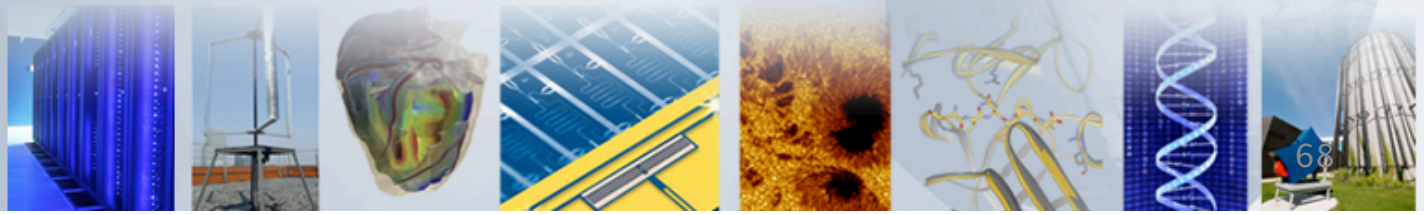
Calcul Québec



Job Submission

- This is certainly an improvement but still suffers from some limitations.
- What if some of these parameters lead to much longer run times than others?
- If we have hundreds or thousands of such computations to do, creating and managing all of these job submission scripts can quickly become tiresome.
- There is a better alternative using a program called Gnu Parallel, available as a module on all Calcul Québec machines.

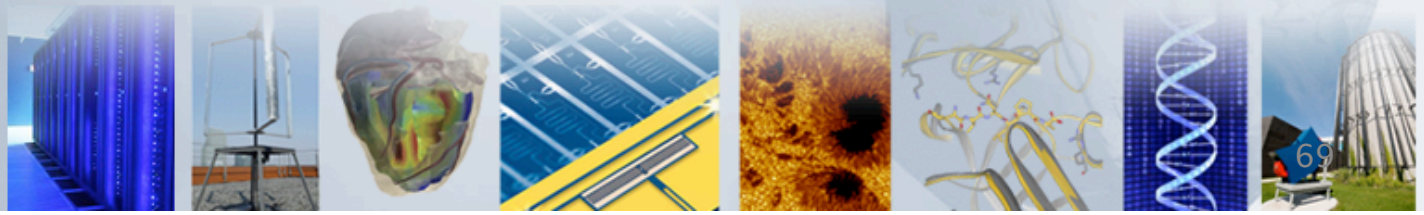
Calcul Québec



Job Submission

- With Gnu Parallel you specify the command to be executed as well as different sets of parameters for this command and the number of processors available.
- Gnu Parallel will then ensure that as soon as one sub-job is completed a new one is started.
- Gnu Parallel can also be used across more than one node and comes with a mechanism to write its progress to a log file.
- The different sets of parameters to be run can be specified using a rule or if there no such simple rule, a text file with one case per line can be given to Gnu Parallel.

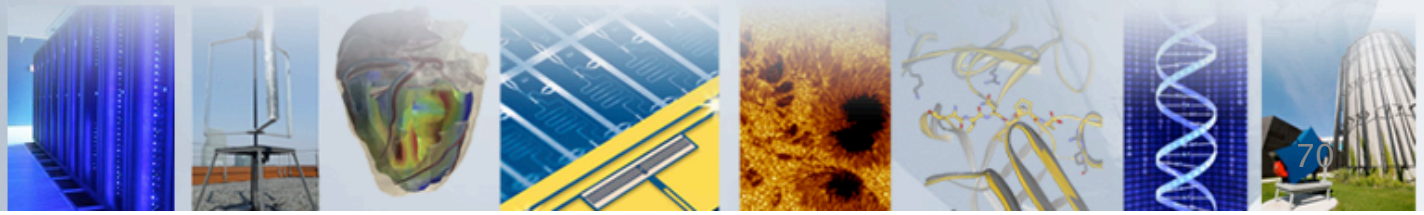
Calcul Québec



Job Submission

```
#!/bin/bash
#PBS -l walltime=10:00:00
#PBS -l nodes=1:ppn=12
#PBS -j oe
#PBS -r n
#PBS -o output.txt

module load GNUParallel/20141022

cd research

parallel --jobs 12 --workdir $PWD ./my_prog foo {1} ::: p1 p2 p3

parallel --jobs 12 --workdir $PWD ./my_prog {1} :::: parms.txt
```

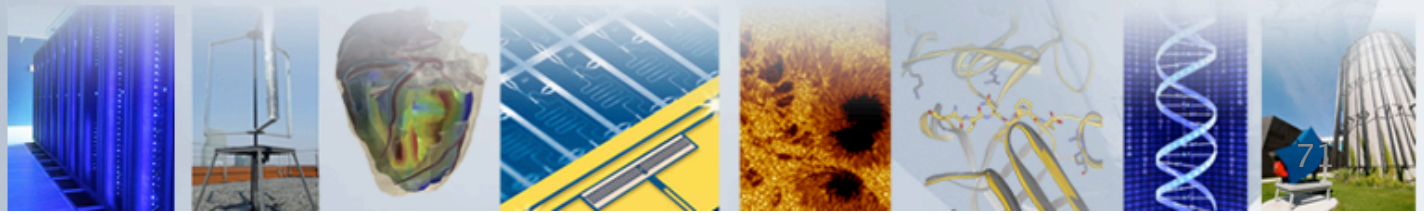
Calcul Québec



Job Submission

- A final option is the use of job arrays.
- With this method we use the PBS job scheduler to create sub-jobs that will be run as resources become available.
- We can access the identity of a sub-job in the submission script using the environment variable `$PBS_ARRAY_INDEX`
- If we need to provide different command line parameters for each sub-job, we can read these from a text file with one set of parameters per line.

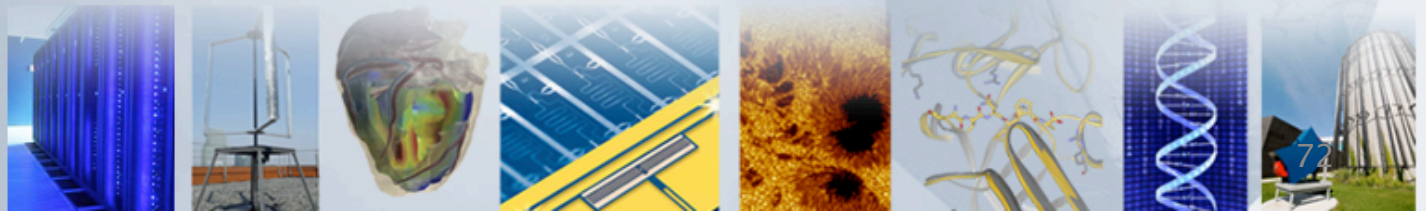
Calcul Québec



Job Submission

```
#!/bin/bash
#PBS -l walltime=1:00:00
#PBS -l nodes=1:ppn=1
#PBS -j oe
#PBS -r n
#PBS -J 1-200

cd research

parameters=`sed -n "${PBS_ARRAY_INDEX} p" input.txt`

parameterArray=($parameters)

x=${parameterArray[0]}
y=${parameterArray[1]}
z=${parameterArray[2]}

./my_prog $x $y $z
```

Calcul Québec



Additional Resources

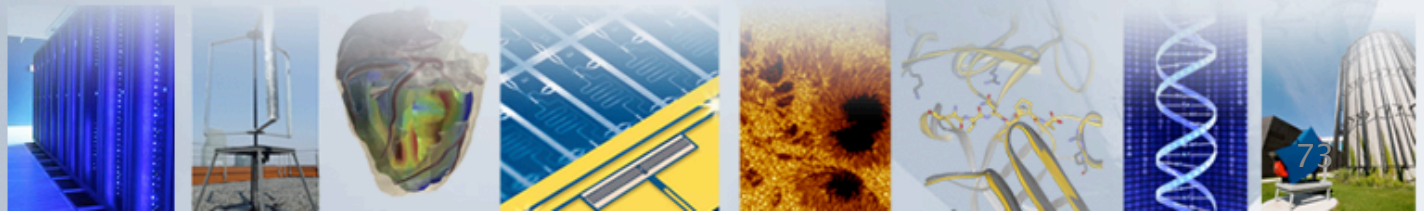
- Read the man page `man` for commands that you use.
- There are many online tutorials as well as other Internet documentation on command line Linux.
- Books like

The Linux Command Line: A Complete Introduction by W. Shotts

Beginning the Linux Command Line by S. van Vugt

- Given that the basic commands haven't changed since the beginnings of Unix in the 1970s, you can also make use of an older book on the subject.

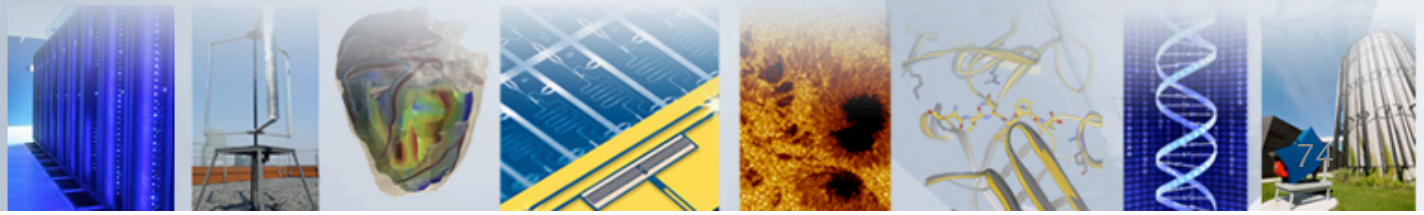
Calcul Québec



Additional Resources

- You can send an e-mail in French or English to the address support@calculquebec.ca
  to ask for help from the Calcul Québec staff.
- If you have a question about the use of a particular machine like Briarée you can send your e-mail to briaree@calculquebec.ca

Calcul Québec



Exercices

- Begin by connecting again to Briarée using either your own account (if you have one) or the following guest account:

Username = `user06`
Password = `Red17path#3`
Machine = `briaree1.calculquebec.ca`

- Once you're connected, return to the directory that you created for the first exercise and create a job submission script to run a CPLEX job for either Java or C++.
- This will be a serial job and should run in just five minutes (less in fact); remember to load the appropriate modules in your job script.