Introduction to The Command Line II

Bertrand Brelier

SciNet HPC Consortium

9 June 2014

A good start

- These commands, and a few more we'll learn today, are enough to get started with using the command line.
- As you have seen, Unix commands are simple, and are designed to do one specific thing.
- By combining these commands together we will be able to do more interesting things.

Our commands

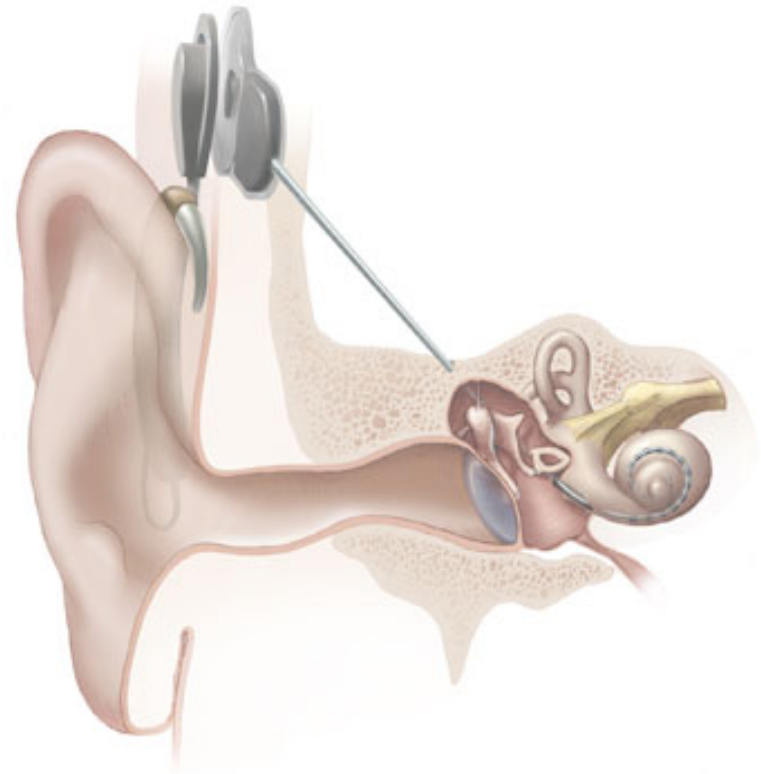
`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`mkdir dir`	create a directory
`rmdir dir`	delete a directory
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file

Example problem: Cochlear implant study

To participate, subjects visited the laboratory and one of the lab techs played an audio sample, and recorded their data. Data consists of:

- date of the test.
- information relating to the sounds played and heard during the test.
- subject ID.
- birth month/year.

Each set of test results were written out to a text file, one set per file.



source: wikipedia.org

Copying the data

```
[brelier.IBM-c5724425579] > pwd
/home/mobaxterm
```

```
[brelier.IBM-c5724425579] > df -h
Filesystem      Size      Used   Available    Use%   Mounted on
C:/Users/IBM_AD~1/AppData/Local/Temp/MOBAXT~1.1/bin
                29.9G    16.8G        13.1G     56%   /usr/bin
C:/Users/IBM_AD~1/AppData/Local/Temp/MOBAXT~1.1/lib
                29.9G    16.8G        13.1G     56%   /usr/lib
C:/Users/IBM_AD~1/AppData/Local/Temp/MobaXterm7.1
                29.9G    16.8G        13.1G     56%   /
C:/Users/IBM_AD~1/DOCUME~1/FakeHome
              29.9G    16.8G        13.1G     56%   /home/mobaxterm
C:              29.9G    16.8G        13.1G     56%   /drives/c
E:               7.2G   736.2M         6.5G     10%   /drives/e
```

```
[brelier.IBM-c5724425579] > cp -r /drives/e/SCMP101_shell .
```

```
[brelier.IBM-c5724425579] > cd SCMP101_shell
[brelier.IBM-c5724425579] >
```

SciNet compute • calcul CANADA

Copying the data, continued

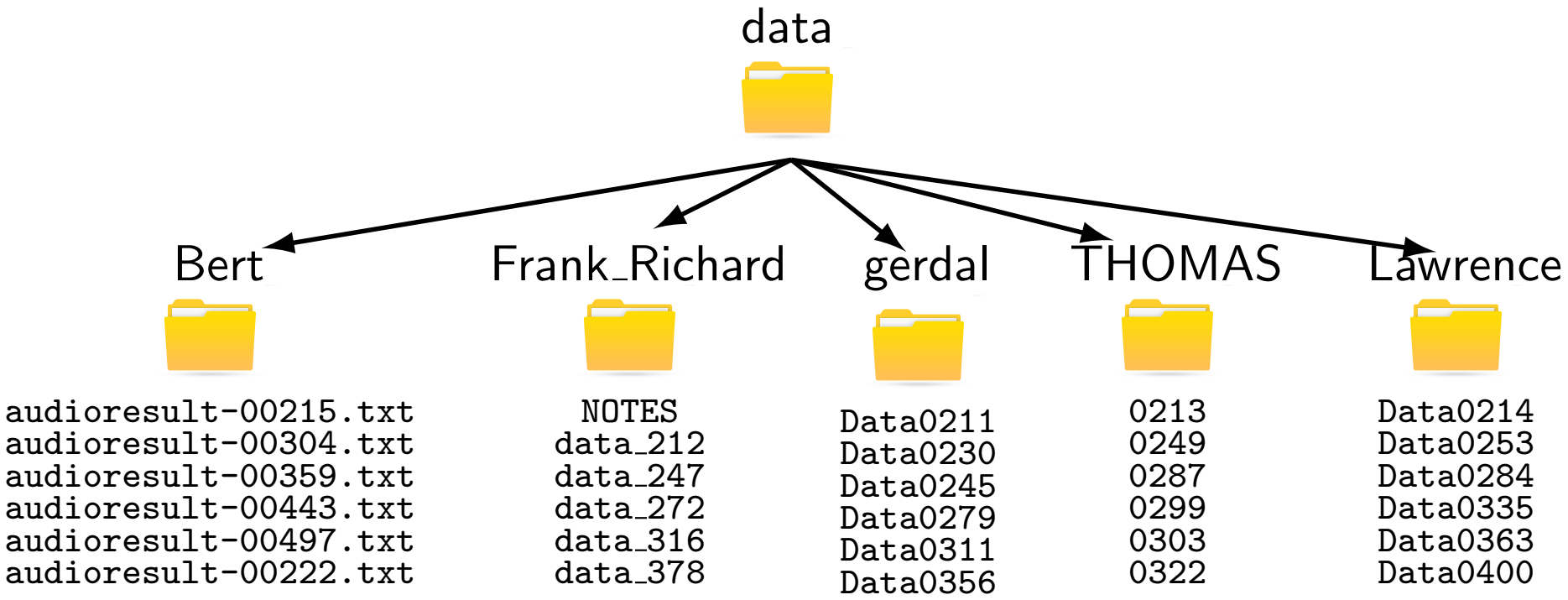
```
[brelier.IBM-c5724425579] > ls
BashSessionData.tar.gz
[brelier.IBM-c5724425579] > tar -z -x -f BashSessionData.tar.gz
[brelier.IBM-c5724425579] >
[brelier.IBM-c5724425579] > ls -F
data/ BashSessionData.tar.gz
```

What happened?

- A 'tar' file (also called a 'tarball') is a file in which has been bundled a number of other files, for easy of moving around.
- `tar` handles a tar file,
  - `-z` means gunzip it.
  - `-x` means extract the contents of the file.
  - `-f` specifies which file you are applying this command to.

The data is now in the 'data' directory.

Data is a complete mess



Multiple directories, inconsistent file names, some directories have a NOTES file...

Goal: create a single file format, using CLI.

Pipelines of commands

- So far we've used the following technique to combine commands:

```
[brelier.IBM-c5724425579] > pwd
/home/mobaxterm/Desktop
[brelier.IBM-c5724425579] > cd data/Lawrence
[brelier.IBM-c5724425579] > wc Data* > all-wcs
[brelier.IBM-c5724425579] > more all-wcs
```

  which creates a temporary file we don't care about; we just want to scroll through the wc results.

- This combination of actions, output of one command goes straight to another, is so common and useful that the shell has a special feature to do this:

```
[brelier.IBM-c5724425579] > wc Data* | more
```

- The 'pipe' allows you to chain together small commands.

The sort command

The sort command can take a number of important flags:

- `-n`: sort by number (not lexicographic; 101 $<$ 30 without `-n`).
- `-k [num]`: sort by the k'th column.
- `-r`: reverses order.

```
[brelier.IBM-c5724425579] > sort -n -k 3
all-wcs
...
9 24 150 Data0515
9 24 153 Data0214
468 1246 7644 total
```

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3
...
9 24 150 Data0515
9 24 153 Data0214
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`arg`	mandatory argument
`[arg]`	optional argument

SciNet compute • calcul CANADA

Pop quiz!

Modify this code to print only the smallest, and then only the largest, data file.

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3
...
9 24 150 Data0515
9 24 153 Data0214
468 1246 7644 total
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`arg`	mandatory argument
`[arg]`	optional argument

Pop quiz!

Modify this code to print only the smallest, and then only the largest, data file.

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3
...
9 24 150 Data0515
9 24 153 Data0214
468 1246 7644 total
```

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3 | head -n 1
9 24 144 Data0234
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`arg`	mandatory argument
`[arg]`	optional argument

Pop quiz!

Modify this code to print only the smallest, and then only the largest, data file.

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3
...
9 24 150 Data0515
9 24 153 Data0214
468 1246 7644 total
```

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3 | head -n 1
9 24 144 Data0234
```

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3 -r | head -n 2
468 1246 7644 total
9 24 153 Data0214
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`arg`	mandatory argument
`[arg]`	optional argument

SciNet compute • calcul CANADA

Pop quiz!

Modify this code to print only the smallest, and then only the largest, data file.

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3
...
9 24 150 Data0515
9 24 153 Data0214
468 1246 7644 total
```

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3 | head -n 1
9 24 144 Data0234
```

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3 -r | head -n 2
468 1246 7644 total
9 24 153 Data0214
```

```
[brelier.IBM-c5724425579] > wc Data* |
sort -n -k 3 -r | head -n 2 | tail -n 1
9 24 153 Data0214
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`arg`	mandatory argument
`[arg]`	optional argument

SciNet compute • calcul CANADA

Our first shell script

It's time to write our first shell script. To do this we need to use a text editor. Linux has many text editors, and they're all difficult to use.

In MobaXterm try Tools > MobaTextEditor, or you can use Notepad. Create a file called 'biggest':

```
#!/bin/bash
wc * | sort -n -k 3 | tail -n 2 | head -n 1
```

And run it by typing

```
[brelier.IBM-c5724425579] > rm all-wcs
[brelier.IBM-c5724425579] > source biggest
9 24 153 Data0214
```

MobaXterm makes the file executable by default. As such, we can just run it like a regular program:

```
[brelier.IBM-c5724425579] > ./biggest
9 24 153 Data0214
```

Largest range - grep

Number of characters is probably unimportant for our analysis. What about the data with the largest range?

```
[brelier.IBM-c5724425579] > grep Range
Data0352
Range:  2
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`arg`	mandatory argument
`[arg]`	optional argument

grep prints the lines from the input that contain the search argument.

Largest range - grep

Number of characters is probably unimportant for our analysis. What about the data with the largest range?

```
[brelier.IBM-c5724425579] > grep Range
Data0352
Range:  2
```

```
[brelier.IBM-c5724425579] > grep Range *
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`arg`	mandatory argument
`[arg]`	optional argument

grep prints the lines from the input that contain the search argument.

Largest range - grep

Number of characters is probably unimportant for our analysis. What about the data with the largest range?

```
[brelier.IBM-c5724425579] > grep Range
Data0352
Range:  2
```

```
[brelier.IBM-c5724425579] > grep Range *
...
Data0544:Range:  6
Data0554:Range:  2
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`arg`	mandatory argument
`[arg]`	optional argument

grep prints the lines from the input that contain the search argument.

Largest range - grep

Number of characters is probably unimportant for our analysis. What about the data with the largest range?

```
[brelier.IBM-c5724425579] > grep Range
Data0352
Range:  2
```

```
[brelier.IBM-c5724425579] > grep Range *
...
Data0544:Range:  6
Data0554:Range:  2
```

```
[brelier.IBM-c5724425579] > grep -v Range
Data0352
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`arg`	mandatory argument
`[arg]`	optional argument

grep prints the lines from the input that contain the search argument.

Largest range - grep

Number of characters is probably unimportant for our analysis. What about the data with the largest range?

```
[brelier.IBM-c5724425579] > grep Range
Data0352
Range:  2
```

```
[brelier.IBM-c5724425579] > grep Range *
...
Data0544:Range:  6
Data0554:Range:  2
```

```
[brelier.IBM-c5724425579] > grep -v Range
Data0352
#
Reported:  Mon Jul 25 14:01:36 2011
Subject:  babyMcCartney281
Year/month of birth:  1991/07
Sex:  M
CI type:  14
Volume:  1
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`arg`	mandatory argument
`[arg]`	optional argument

grep prints the lines from the input that contain the search argument.

grep -v prints the lines from the input that *don't* contain the search argument.

Pop quiz!

Create a new script, called 'biggestRange', which prints out the data file which has the biggest Range.

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`arg`	mandatory argument
`[arg]`	optional argument

Pop quiz!

Create a new script, called 'biggestRange', which prints out the data file which has the biggest Range.

```
[brelier.IBM-c5724425579] > cat bigges-
tRange
#!/bin/bash

# a comment!  Use the # sign.

grep Ra * | sort -n -k 2 | tail -n 1
```

```
[brelier.IBM-c5724425579] > ./biggestRange
Data0531:Range:  9
```

```
[brelier.IBM-c5724425579] >
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`arg`	mandatory argument
`[arg]`	optional argument

SciNet compute • calcul CANADA

Cutting up the output

How do keep just part of the output?

Suppose we want just the file name?

```
[brelier.IBM-c5724425579] > grep Ra * |
sort -n -k 2 | tail -1
Data0531:Range:  9
```

```
[brelier.IBM-c5724425579] > grep Ra * |
sort -n -k 2 | tail -1 | cut -c -8
Data0531
```

Suppose we just want the range?

```
[brelier.IBM-c5724425579] > grep Ra * |
sort -n -k 2 | tail -1 | cut -c 10-
Range:  9
```

Suppose we just want something else?

```
[brelier.IBM-c5724425579] > grep Ra * |
sort -n -k 2 | tail -1 | cut -c 2-5
ata0
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`cut flags output`	cut part of output
`arg`	mandatory argument
`[arg]`	optional argument

- "-c" tells cut to cut characters.
- "-8" means keep up-to-and-including character eight.
- "10-" means keep 10 and higher.

SciNet compute • calcul CANADA

Suppose I need to save information

If I need to save something, I use variables.

```
[brelier.IBM-c5724425579] > grep Ra * |
sort -n -k 2 | tail -1 | cut -c -8
Data0531
```

```
[brelier.IBM-c5724425579] > i=‘grep Ra * |
sort -n -k 2 | tail -1 | cut -c -8‘
```

```
[brelier.IBM-c5724425579] >
```

```
[brelier.IBM-c5724425579] > echo i
i
```

```
[brelier.IBM-c5724425579] > echo $i
Data0531
```

NOTE: The ‘ symbol used here is the single quote in the upper-left corner of your keyboard!! Using this single quote will execute the command.

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`cut flags output`	cut part of output
`arg`	mandatory argument
`[arg]`	optional argument

What happens if you use the other single quote?

Arguments for bash scripts

- We'd like to use our script on each directory, but not have a copy in each one.
- Move biggestRange down one directory, and change it so that it works on any directory's files:

```
[brelier.IBM-c5724425579] > pwd
/home/mobaxterm/Desktop/data/Lawrence
```

```
[brelier.IBM-c5724425579] > mv bigges-
tRange ..
```

```
[brelier.IBM-c5724425579] > cd ..
```

```
[brelier.IBM-c5724425579] > cat bigges-
tRange #!/bin/bash
grep Range ${1}/* | sort -n -k 2 | tail -1
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`cut flags output`	cut part of output
`arg`	mandatory argument
`[arg]`	optional argument

- When a command is run in the shell, its name is put in ${0}.
- All other arguments are put in ${1}, ${2}...

Arguments for bash scripts

```
[brelier.IBM-c5724425579] > pwd
/home/mobaxterm/Desktop/data
```

```
[brelier.IBM-c5724425579] > ls
Bert Frank_Richard Lawrence THOMAS
alexander biggestRange gerdal jamesm
```

```
[brelier.IBM-c5724425579] > ./biggestRange
Bert
Bert/audioresult-00384.txt:Range:  10
```

```
[brelier.IBM-c5724425579] > ./biggestRange
THOMAS
THOMAS/0336:Range:  10
```

```
[brelier.IBM-c5724425579] > ./biggestRange
james
grep:  james/*:  No such file or directory
```

```
[brelier.IBM-c5724425579] > ./biggestRange
jamesm
jamesm/data_517.txt:Range:  10
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`cut flags output`	cut part of output
`arg`	mandatory argument
`[arg]`	optional argument

Loops for bash scripts

Bash has loops, just like any language:

```
[brelier.IBM-c5724425579] >
```

```
[brelier.IBM-c5724425579] > cat
loop_biggestRange #!/bin/bash
for dir in alexander Bert Frank_Richard
do
echo "The biggest range in " ${dir} " is:"
./biggestRange ${dir}
done
```

```
[brelier.IBM-c5724425579] >
```

```
[brelier.IBM-c5724425579] >
./loop_biggestRange
The biggest range in alexander is:
alexander/data_462.DATA:Range:  10
The biggest range in Bert is:
Bert/audioresult-00384.txt:Range 10
The biggest range in Frank_Richard is:
Frank_Richard/data_538:Range:  10
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`cut flags output`	cut part of output
`for..do..done`	for loop in bash
`arg`	mandatory argument
`[arg]`	optional argument

If statements for bash scripts

Bash has if statements, just like any other language:

```
[brelier.IBM-c5724425579] > pwd /home/-
mobaxterm/Desktop/data
```

```
[brelier.IBM-c5724425579] > cat if_test.sh
#!/bin/bash
cd ${1}
for filename in *
do

if [ $filename == "NOTES" ]
then
echo "I'm a NOTES file."
fi

done
```

```
[brelier.IBM-c5724425579] > ./if_test.sh
jamesm
I'm a NOTES file.
```

```
[brelier.IBM-c5724425579] > ./if_test.sh
alexander
```

More commands

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`cut flags output`	cut part of output
`for..do..done`	for loop in bash
`if..then..fi`	if statement in bash
`arg`	mandatory argument
`[arg]`	optional argument

An easier way to avoid the NOTES file would be

```
cd ${1}
ls * | grep -v NOTES
```

SciNet compute • calcul CANADA

Assignment:

Write a script, called cleanedData.sh, which does the following, when run from the data directory:

- Copies all of the data files from data/alexander, data/Bert, etc. to a new directory, 'data/cleaneddata'.
- Renames the data files in cleaneddata; the names must have the format "Data.NNN.txt", where NNN is the three-digit number that goes with the original file.

The final result should be 351 files, ranging from Data.211.txt to Data.561.txt, in the cleaneddata directory. No NOTES files should be present.

Be sure to ask questions. Use man pages if you need to. If there is some functionality that you would like, and you hope that a command exists that does that, ask me. You should be able to do this with everything you have been taught thus far.

Assignment:

```
#!/bin/bash
mkdir cleaneddata
for i in alexander  Bert  Frank_Richard  gerdal  jamesm  Lawrence THOMAS
do
for filename in $i/*
do
if [ $filename != "$i/NOTES" ]
then
    if [ $i == "alexander" ]
    then
        tmp=`echo $filename| cut -c 16-18`
        echo "cp $filename cleaneddata/Data.$tmp.txt"
        cp $filename cleaneddata/Data.$tmp.txt
    fi
    if [ $i == "Bert" ]
    then
        tmp=`echo $filename| cut -c 20-22`
        echo "cp $filename cleaneddata/Data.$tmp.txt"
        cp $filename cleaneddata/Data.$tmp.txt
    fi
    if [ $i == "Frank_Richard" ]
    then
        tmp=`echo $filename| cut -c 20-22`
        echo "cp $filename cleaneddata/Data.$tmp.txt"
        cp $filename cleaneddata/Data.$tmp.txt
    fi
    if [ $i == "gerdal" ]
    then
        tmp=`echo $filename| cut -c 13-15`
        echo "cp $filename cleaneddata/Data.$tmp.txt"
        cp $filename cleaneddata/Data.$tmp.txt
    fi
```

Assignment:

```
    if [ $i == "jamesm" ]
    then
        tmp=‘echo $filename| cut -c 13-15‘
        echo "cp $filename cleaneddata/Data.$tmp.txt"
        cp $filename cleaneddata/Data.$tmp.txt
    fi
    if [ $i == "Lawrence" ]
    then
        tmp=‘echo $filename| cut -c 15-17‘
        echo "cp $filename cleaneddata/Data.$tmp.txt"
        cp $filename cleaneddata/Data.$tmp.txt
    fi
    if [ $i == "THOMAS" ]
    then
        tmp=‘echo $filename| cut -c 9-11‘
        echo "cp $filename cleaneddata/Data.$tmp.txt"
        cp $filename cleaneddata/Data.$tmp.txt
    fi
fi
done
done
```

Assignment: other solution using function

```
#!/bin/bash
function MyCopy {
    num=$(($2 + 2))
    tmp=‘echo $1| cut -c $2-$num‘
    echo "cp $1 cleaneddata/Data.$tmp.txt"
    cp $1 cleaneddata/Data.$tmp.txt
}
mkdir cleaneddata
for i in alexander  Bert  Frank_Richard  gerdal  jamesm  Lawrence THOMAS
do
for filename in $i/*
do
if [ $filename != "$i/NOTES" ]
then
    if [ $i == "alexander" ]
    then
        MyCopy $filename 16
    fi
    if [ $i == "Bert" ]
    then
        MyCopy $filename 20
    fi
    if [ $i == "Frank_Richard" ]
    then
        MyCopy $filename 20
    fi
    if [ $i == "gerdal" ]
    then
        MyCopy $filename 13
    fi
```

Assignment: other solution using function

```
    if [ $i == "jamesm" ]
    then
        MyCopy $filename 13
    fi
    if [ $i == "Lawrence" ]
    then
        MyCopy $filename 15
    fi
    if [ $i == "THOMAS" ]
    then
        MyCopy $filename 9
    fi
fi
done
done
```