Scripting for Life Science Researchers

Command line practical

October 21, 2013

Preparation

Install Cygwin (Windows users only)

Go to the course website[1] and follow the "Preparations" instructions to install Cygwin.

Software installation

Install the following editors:

- `nano`.
- `emacs` (`xemacs` for Linux, `emacs-w32` for Windows).

Exercises

Hints:

- Keep in mind that bash is case sensitive.
- Whenever "course" is used, replace it with your user name.
- Auto completion can be done by pressing the `TAB` key.

Files and directories

Find out where we are in the directory tree and list the content of directories. Try the following commands:

```
$ pwd
$ ls
$ ls .
$ ls ..
$ ls /home
$ ls /home/course
```

Now, create a directory and a text file using `nano`.

```
$ mkdir test_dir
$ cd test_dir
$ pwd
$ nano file1.txt
```

[1] https://humgenprojects.lumc.nl/trac/humgenprojects/wiki/scripting

We have now opened the `nano` text editor and are editint the content of the file `file1.txt`. Try to create a `FASTA` formatted file, something like:

```
>header1
ATGCGT
>header2
GTCAAA
```

Hints for using `nano`:

- When done press `CTRL-X`.
- When asked to save changes press `y`.
- Press enter to confirm the name of the file.

Examine the directory and its content in detail. Use the `-l` and `-h` flags of `ls` for long listing and human readable output formatting respectively.

```
$ ls -lh
```

You can see the file size, who created it, and when it was created.

Redirecting

Create an other text file named `file2.txt` in the same directory. Make sure to put some text in this file as well.

Check the contents of the directory. If everything went well you should see the two files you created.

You can see the contents of a file with `cat`. Be careful with large files, it can take a very long time to print them to the screen.

```
$ cat file1.txt
$ cat file2.txt
$ cat file1.txt file2.txt
```

You might want to concatenate both files and write the result to a new file. We can redirect the output from `cat` to a file with the "greater than" sign (`>`).

```
$ cat file1.txt file2.txt > concat.txt
$ cat concat.txt
```

A way to quickly figure out how many lines a file contains is to use `wc` (word count) with the `-l` option.

```
$ wc -l concat.txt
```

With the `head` command we can print any number of lines from the top of the file to the terminal. Use the `-n` flag to indicate the number of lines you want to print.

```
$ head -n 2 file1.txt
```

Try different values for `n`. The command `tail` works in a similar way but instead of printing the first lines, it will print the last ones. Try some `tail` commands.

Some commands accept wildcards as input. The star (`*`) for example. Examine the following commands.

```
$ cat *
$ wc -l *
```

Files and directories can be removed with the `rm` command.

```
$ rm file1.txt
$ ls
```

You can see that the file is gone. Now we want to get rid of the whole directory. Before we can do so we need to leave the current directory, then we can recursively remove the directory using `rm` with the `-r` flag.

```
$ cd ..
$ ls
$ rm -r test_dir
$ ls
```

The directory `test_dir` and all of its content is now removed.

Piping

To chain processes we can use the pipe (`|`) symbol. The file `/etc/services` contains a list of internet network services, the content is not important, but it is a large text file that serves our purposes for the following examples.

First, we use `cat` to show the content of this file, then we use a pipe to connect this output to the input of `less`. In this way, we will be able to conveniently scroll through the lines.

```
$ cat /etc/services
$ cat /etc/services | less
```

Hints for using `less`:

- Scroll down and up using the arrow keys.
- Press `q` when you are done.

Try to redirect output from `ls` to `less`.

The command `grep` searches for a given pattern, it will only print the lines matching this pattern. It requires input that we can provide by piping the output from `cat` to `grep`.

Suppose we want to have information about the `ssh` protocol.

```
$ cat /etc/services | grep ssh
```

We can save the lines of interest to a file in the same way we concatenated files earlier with the "`>`" symbol.

```
$ cat /etc/services | grep ssh > ssh.txt
$ cat ssh.txt
```

The `ps` command gives a list of current processes.

```
$ ps
```

To quickly count how many processes are running we pipe the output to `wc`.

```
$ ps | wc -l
```

Writing a script

Create 100 files files to work with.

```
$ mkdir exercise
$ cd exercise
$ for i in {1..100}; do ls -s > file_$RANDOM.txt; done
```

Now write a bash script that:

- Creates a list of the 20 files with the most lines in them.
  - Hint: `wc -l <file>`
  - Hint: To get only filenames from `sort`, pipe output to this: (explained later)
    `sed 's/ \+/ /g' | cut -d' ' -f3`
- Rsyncs all files except the 20 with the most lines in it to a folder inside this directory named "`backup`".
  - Hint: `--exclude-from`
- Move the 20 largest files to a folder named "`large_files`".
  - Hint: Read what the `-p` option can do for `mkdir` when the script is run again.
- For all the steps catch any errors into a file.