Introduction to Unix shell - Orientation

- 2016/17 Part III Systems Biology SEB module
- 8 Feb 2017, 10:00-13:00
- Bioinformatics Training Room, Craik-Marshall Building, Downing Site
- Alexey Morgunov

Introduction

This extended exercise is intended as a way to familiarise yourself with (or refresh your memory of) the most basic commands in shell. Work through the commands sequentially, trying to understand exactly what's happening at each step and answering the questions. Refer to Notes for help.

Orientation

Try the following commands (one command per line) to learn to navigate around your file system. Explore! N.B. The commands are not listed in any particular order in this section.

```
pwd
ls
cd [folder name]
ls -l     # what useful information is displayed here?
man ls    # hit <q> to exit, <space> to move down one page
ls -lht
cd ..
cd ~
cd -
cd .
```

Come back to some cosy home directory where you can play around without wreaking havoc on your computer. N.B. The commands in this section are listed in the order of suggested execution. Make sure you understand what is happening in each line and can answer the questions!

```
mkdir sandbox
cd sandbox
echo 'Hello, world!' > hello.txt
cp hello.txt ../hello.txt.copy    # where is the copied file? where is the original?
cp -r ../sandbox ../sandbox2      # you don't have to be in the same directory as the
file/folder
mv hello.txt ../hello.txt.copy2   # and now?
# how would you rename a file?
cd ..
rm hello.txt.copy2
rmdir sandbox
rmdir sandbox2    # why does this not work unlike the one above?
rm -r sandbox2
find . -name '*txt*'
find . -name '*txt*' -delete
```

Let's look at some redirection and pipes (see here) by exploring where these commands put our text. Use any text you like but make it different between inputs.

```
echo 'Hello, world!' > hello.txt
```

```
cat hello.txt
cat > file.txt      # <enter> to start a new line, <CTRL+D> when done entering text
cat file.txt
cp file.txt file2.txt
cat hello.txt >> file.txt
cat file.txt
cat hello.txt > file2.txt    # how is '>' different from '>>' ?
cat file2.txt
cat hello.txt file.txt > long.txt
cat long.txt
```

How best to look at our text files?

```
cat long.txt
less long.txt       # <q> to quit
gedit long.txt
head -3 long.txt
tail -3 long.txt
tail -n +3 long.txt
wc long.txt         # what options does 'wc' have? what can you do with it?
```

Now let’s do some sorting. Figure out how these functions and parameters work.

```
sort < long.txt > sorted.txt
cat long.txt | sort > sorted2.txt   # is there any difference between the two methods?
cat > animals.txt  # e.g. cat, dog, cat, hedgehog, horse, Dog, human, 1 more human
cat > numbers.txt  # e.g. 3, 4, 2, 1, 10, 102, 101, 10, 10, one too many
sort animals.txt   # what is the problem? how to overcome it?
sort -f animals.txt
sort -ur animals.txt
sort numbers.txt   # what is the problem? how to overcome it?
sort -n numbers.txt
uniq -c numbers.txt # what is the problem here?
sort -n numbers.txt | uniq -c
```

Let’s take a look at some `grep` and `sed`, but only some very basic examples. They can do much more than this!

```
grep "o" animals.txt
grep -v "o" animals.txt
grep "dog" animals.txt
grep -i "dog" animals.txt
sed "s/o/O/g" animals.txt
```

And just a couple more tricks. Also, worth checking out `cut`, `paste`, `join`, `diff` and `comm` in more detail.

```
rev animals.txt
cut -c3-5 animals.txt
paste - - < animals.txt
paste -d',' -s animals.txt
echo "hello" | tr l r
```

Finally, some important syntax issues. Compare the outputs of similar lines and figure out what is different. Use Notes as help if needed.

```
echo ls
echo "ls"
echo 'ls'
echo `ls`

echo $HOME
echo ${HOME}
echo \$HOME
echo \\$HOME
echo \\\$HOME

echo '$HOME | ls'
echo "$HOME | ls"
echo `$HOME | ls`

echo f{oo,ee,e}d
echo $((42+42))
```

License

Many of the shell scripting exercises are taken from Linux Shell Scripting Tutorial (LSST) v2.0 under a CC-BY-NC-SA license.

This material is released under a CC-BY-NC-SA license.