Introduction to unix/linux: Answers

1. Running

   ```
   ls ~/
   ```

   doesn't show any files beginning with a dot. Why not? How can you see them?

   Answer: Files beginning with a dot are treated as hidden by default. They're often used as configuration files. To view all files:

   ```
   ls -a
   ```

   Note that * also doesn't match files beginning with a dot. Hence (for example) *rm -r ** wouldn't remove any files beginning with a dot in the current directory, but would remove all other files and subdirectories.

2. What does running *cd -* do?

   Answer: *cd -* changes to the previous directory and prints out the name of the that directory.

3. Run the command

   ```
   mkdir ~/dir1/dir2
   ```

   How can the error be fixed?

   Answer: The problem is that *mkdir*, by default, only creates one directory level and assumes that all preceding subdirectories exist. One can create the required subdirectories first:

   ```
   mkdir ~/dir1
   mkdir ~/dir1/dir2
   ```

   but this is somewhat laborious. Instead, the *--parents* option will result in *mkdir* creating any subdirectories that are needed:

   ```
   mkdir -p ~/dir1/dir2
   ```

4. Create a directory and some file(s) in the directory. Try deleting the directory using *rmdir*. What happens? How can you delete it? (Hint: look at the *rm* man page.)

   Answer: *rmdir* only removes a directory if it's completely empty. *rm* has an option to delete a directory recursively (i.e. including its contents):

   ```
   rm -r non-empty_directory_to_be_deleted
   ```

5. The operators <, > and >> were described above. There's also (unsurprisingly) a << operator. Find out how to use it.

   Answer: See http://tldp.org/LDP/abs/html/here-docs.html.

7. Work out what each piece of this command does:

   ```
   cut -f1 -d" " ~/.bash_history | sort | uniq -c | sort -nr | head -10
   ```

Answer: *cut -f1 -d" " ~/.bash_history* prints the first word on each line in your ~/.bash_history file (which stores the commands you use): so it lists all the commands (without any arguments) that you've used. | *sort* sorts the list of commands alphabetically. | *uniq -c* takes the alphabetical list of commands and removes all lines which are identical to a previous line and prints the unique lines preceeded by a count of how many times each line occurred. | *sort -nr* sorts the list of commands and their frequency numerically and in reverse order (i.e. descending). *sort* sorts by the first field of input by default. | *head -10* prints the first 10 lines of the output from the previous command.

The entire command prints the 10 commands you've used most and how often you've used them.

8. Add an alias to your configuration file so that running ls (with no additional options) gives a coloured output.

   Answer: On Macs, add to you ~/.bashrc file:

   ```
   alias ls="ls -G"
   ```

   (This assumes that your ~/.bash_profile is set up as described in your notes.) On linux add something like

   ```
   alias ls="ls --color=auto"
   ```

   to your ~/.bashrc. (The exact argument might vary depending upon how recent your version of ls is, so check the manpage...)

9. Create a directory containing several files. Produce a tarball of that directory. Find out how to compress the tarball both at the same time the tarball is created and after an uncompressed tarball has been created.

   Answer:

   ```
   mkdir test_dir && cd test_dir && touch test_file1 test_file2 test_file3
   # produce a tar ball
   tar cvf test.tar *
   # produce a compressed tarball using gzip
   tar cvzf test.tar.gz *
   # produce a compressed tarball using bzip22
   tar cvjf test.tar.bz2 *
   # compress an existing tarball using gzip
   gzip test.tar
   # or compress an existing tarball using bzip2
   bzip2 test.tar
   ```

   Common compression algorithms on Linux are gzip and bzip2. (zip also exists like it does on Windows, but is less commonly used.) Uncompressing a file can be performed using gunzip or bunzip2, as appropriate.

10. Download this file: http://www.cmth.ph.ic.ac.uk/people/j.spencer/cdt/names.txt.

    a. Use grep to print your name and the names of the people in the lines directly above and below your name.

       Answer:

       For example:

       ```
       grep -B1 -A1 Peter names.txt
       ```

b. Use awk and sed to print the file in the format of FORNAME SURNAME rather than SURNAME, FORNAME.

Answer:

It's possible to do this in two steps, saving the intermediate output to a temporary file, but is easiest done using a pipe:

```
awk '{print $2, $1}' names.txt | sed -e 's/,//'
```

In fact, this can be accomplished in a single awk command:

```
awk '{sub(/,/,""); print $2, $1}' names.txt
```

Note that this assumes the all forenames and surnames are single words. Dealing with the case where this is not true is left as an exercise...

11. Modify the welcome script so that it will welcome an arbitrary number of users and print a help message if the first argument is "--help".

Answer:

```
#!/bin/bash
# A script to welcome a user.
# Usage: welcome.sh [user]
# If the user is supplied, welcome that user, otherwise welcome the current
# user.
if [[ $# -eq 0 ]]; then
    echo "Welcome `whoami`"
elif [[ "$1" == "--help" ]]; then
    echo usage: welcome.sh users
    echo "welcome a (space-separated) list of users"
else
    for user in $*; do
        echo "Welcome $user"
    done
fi
```

12. Download a tarball of data files from http://www.cmth.ph.ic.ac.uk/people/j.spencer/cdt/data_files.tar and extract the files in it. Each directory contains a single data file which contains output of a calculation of the dispersion energy between two 1D wires modelled using Hückel theory at various separations. Write a script which:

    a. Extracts the separation and output from each data file.

    b. Sorts the by the amount of separation.

    c. Prints out a data table.

    d. Save the data table to a file and plot it.

(Hint: start on the command line and figure out how to extract and print the required information from one data file first.)

Answer: All output files are called 'out'. The required information is on the lines beginnning "Separation" and "Energy". *grep Separation out* and *grep Energy out* selects the required information from an output file and *awk* can be used with a pipe to select the actual number. Hence:

```
#!/bin/bash
for d in run*; do
```

```
    cd $d
    z=`grep Separation out | awk '{print $3}'`
    energy=`grep Energy out | awk '{print $3}'`
    echo $z $energy
    cd ..
done | sort -g
```

The script loops over each directory. For each directory it goes into the directory, obtains the required information and prints it out. The |*sort -g* at the end takes the entire output from the *for* loop and sorts in numerically.

13. Find out about the chmod command and the unix concept of file-permissions. Use

```
ls -l
```

to show file permissions.

Answer: See http://en.wikipedia.org/wiki/File_system_permissions and http://en.wikipedia.org/wiki/Chmod (or google them...).

14. What does the command:

```
chmod 750 args.sh
```

mean?

Answer: It changes the file `args.sh` to be readable, writeable and executable by the user who owns the file, readable and executable by the group who owns the file (which may contain just the owner of the file or also other users) and prevents all other users from reading, writing or executing the file.

15. Why must directories be executable?

Answer: See the chmod man page.