Linux command line exercises

Metadata: commands in this exercise: **grep**, tar, cat, cut, more, less, awk, sed, wc, sort, gnuplot
Metadata: example files from computational chemistry

In these instructions the *first* character "**$**" in the command examples should not be typed, but it denotes the command prompt.

Some command lines are too long to fit a line in printed form. These are indicated by a backslash "****" at the end of line. It should not be included when typing in the command. For example

```
$ example command \
continues \
and continues
```

Should be typed in as:

```
example command continues and continues
```

0) Download and unpack the exercise files:

Go to the workshop home page (www.csc.fi -> click at the workshop name link at the right column, scroll down and download the tar archive from the link, save as …)

Open a terminal, cd to the folder where you downloaded the archive, unzip and untar the file:

```
$ tar xzvf linux-exercise.tar.gz
```

The **v** (verbose) flag in the tar command shows the files with the path that are untarred. You see, that you'll get a number of files in a subdirectory named linux-exercises. Go to that directory with the **cd** command.

In that directory you should now have these files:

dimer.log : Gaussian quantum chemistry geometry optimization log file for a water dimer
dimer_scan.log : Gaussian log file for relaxed potential energy scan for stretching water dimer distance
freq.log : Gaussian log file for a frequency calculation
cp2k.out : cp2k calculation ascii output file
cp2k.xyz : xyz format molecular structure file of liquid water

1) have a look at the contents of some files

```
$ more cp2k.out
```

```
$ nano cp2k.out
```

…

2) How did the total energy change in the water dimer optimization run? (dimer.log)

The part in the log file where the energy has converged is shown below. The final energy is printed on the shadowed line.

```
Cycle  12  Pass 1  IDiag  1:
E= -152.637052486060     Delta-E=       -0.000000000001 Rises=F Damp=F
DIIS: error= 7.20D-08 at cycle   5 NSaved=   5.
NSaved= 5 IEnMin= 5 EnMin= -152.637052486060     IErMin= 5 ErrMin= 7.20D-08
ErrMax= 7.20D-08  0.00D+00 EMaxC= 1.00D-01 BMatC= 2.02D-13 BMatP= 1.38D-12
IDIUse=1 WtCom= 1.00D+00 WtEn= 0.00D+00
Coeff-Com:  0.795D-03 0.448D-01 0.134D+00 0.411D+00 0.410D+00
Coeff:      0.795D-03 0.448D-01 0.134D+00 0.411D+00 0.410D+00
Gap=     0.182 Goal=   None    Shift=    0.000
RMSDP=8.86D-09 MaxDP=9.24D-08 DE=-1.08D-12 OVMax= 9.92D-08

SCF Done:  E(RPBE-PBE) =  -152.637052486     A.U. after   12 cycles
           NFock= 12  Conv=0.89D-08     -V/T= 2.0059
KE= 1.517485854820D+02 PE=-4.434279894947D+02 EE= 9.792090969701D+01
Leave Link  502 at Thu Jan 29 09:51:27 2015, MaxMem=    33554432 cpu:         2.3
(Enter /appl/chem/G09RevD.01/g09/l601.exe)
Copying SCF densities to generalized density rwf, IOpCl= 0 IROHF=0.
```

How to get all those lines out?

```
$ grep "SCF Done:" dimer.log
```

What is the shortest string to grep that gives only these lines?

3) Was there something in the output that needs attention? (cp2k.out)

Programs often print out messages in case something goes wrong or the user has chosen questionable options. Is there anything in cp2k.out that we need to worry about?

Are there any warnings?

```
$ grep warning cp2k.out
```

What if the warning was capitalized? (or uppercase)

```
$ grep -i warning cp2k.out
```

4) look for the development of the convergence criteria, which of these is satisfied last?

```
        Item               Value     Threshold  Converged?
 Maximum Force            0.057661     0.000450     NO
 RMS     Force            0.022508     0.000300     NO
 Maximum Displacement     0.218107     0.001800     NO
 RMS     Displacement     0.108003     0.001200     NO
```

Try some of these, what do they do?

```
$ grep "Maximum Force" dimer.log
$ grep "m Forc" dimer.log
$ grep " Force" dimer.log
$ grep " Displa" dimer.log
$ grep -E "RMS     |Maximum " dimer.log
$ grep -A 4 "Threshold" dimer.log
```

Why bother? Sometimes the geometry optimizations with "floppy" modes (flat potential energy surface) fail to converge and just wiggle around. Looking at the RMS force and Maximum displacement relative to the total energy can reveal that this is actually happening.

5) Did the frequency calculation succeed? (freq.log)

Are there errors or warnings? Was the preceding geometry optimization successful *i.e.* the structure is a minimum on a potential energy surface? (*hint. were forces and displacements converged?*)

6) How many oxygen atoms there are in the cp2k.xyz file?

A line that specifies the coordinates for an oxygen atom looks like this:

```
  O         7.1808680000        1.7902530000        3.7253460000
```

Get all lines that have a capital O in them

```
$ grep O cp2k.xyz
```

How many lines was that?

```
$ grep O cp2k.xyz | wc
```

What does **wc** (short for *word count*) print out? Can we give it some flags? (*try* **man wc**, *or google for it*)

How many atoms in total? (*we know it's only hydrogen and oxygen atoms, i.e. O and H*)

```
$ grep -E "O|H" cp2k.xyz | wc -l
```

You could also count all the lines in the file and subtract the first two, which don't represent atoms.

```
$ wc -l cp2k.xy
```

What if your structure had also osmium atoms (Os). How would you change your commands?

7) How long did one iteration in cp2k.out take?

- The part in the output file that shows timing is like this:

```
*******************************************************************************
 ENSEMBLE TYPE                =                                          NVE
 STEP NUMBER                  =                                              1
 TIME [fs]                    =                                       0.500000
 CONSERVED QUANTITY [hartree] =                             -0.880615921354E+04

                                                 INSTANTANEOUS             AVERAGES
 CPU TIME [s]                 =                         290.20               290.20
 ENERGY DRIFT PER ATOM [K]    =          -0.810224161417E+02    0.000000000000E+00
 POTENTIAL ENERGY[hartree]    =          -0.880838552803E+04   -0.880838552803E+04
 KINETIC ENERGY [hartree]     =           0.222631448393E+01    0.222631448393E+01
 TEMPERATURE [K]              =                        305.326              305.326
 *******************************************************************************
```

You could try this to get the timing:

```
$ grep TIME cp2k.out
```

But that gives too many hits. To get only the line that has the wall clock time spent at each time step give:

```
$ grep "CPU TIME" cp2k.out
```

Is energy drift per atom speeding up? How to get access to those lines?

8) How to plot those times?

First direct them to a file

```
$ grep "CPU TIME" cp2k.out > times
```

Confirm they are there:

```
$ more times
```

Start gnuplot with **`gnuplot`** and give

```
plot 'times' using 0:5 with points
```

or with lines

```
plot 'times' using 0:5 with lines
```

Exit with **`quit`**

Explanation: in gnuplot the command is "plot", followed with the filename that has the data to be plotted (as it is a string, it needs to be quoted), "using" tells gnuplot to use the following columns in that file, "0" means the line number (first line=1, second line=2,… this will be the x-axis), ":" means to plot the second column as the function of the first column, "5":th column will be the y-axis, "with" is followed by what to plot at the coordinates, now it's "points" i.e. some symbols.

How to plot the energy drift per atom?

9) What is the average temperature based on cp2k.out?

The part in the output file that has this information looks like this:

```
 ******************************************************************************
 ENSEMBLE TYPE                =                                            NVE
 STEP NUMBER                  =                                              1
 TIME [fs]                    =                                       0.500000
 CONSERVED QUANTITY [hartree] =                              -0.880615921354E+04

                                              INSTANTANEOUS             AVERAGES
 CPU TIME [s]                 =                      290.20               290.20
 ENERGY DRIFT PER ATOM [K]    =          -0.810224161417E+02   0.000000000000E+00
 POTENTIAL ENERGY[hartree]    =          -0.880838552803E+04  -0.880838552803E+04
 KINETIC ENERGY [hartree]     =           0.222631448393E+01   0.222631448393E+01
 TEMPERATURE [K]              =                     305.326              305.326
 ******************************************************************************
```

First we want to **grep** all lines with the temperature:

```
$ grep TEMPERATURE cp2k.out
```

To get the average we can process the output of the **grep** command with **awk** and create a cumulative sum of the column with the temperature (**+=** adds the value in column 4 to the current value in variable **a**), then print that instantaneous value (in column 4), the number of records processed **NR** and the average up to that point:

```
$ grep TEMPERATURE cp2k.out | awk '{a+=$4;print $4,NR,a/NR}'
```

This could also be done without separate grep command with

```
$ awk '/TEMPERATURE/{a+=$4;n++;print $4,n,a/n}' cp2k.out
```

> *Explanation: here we use a counter n, which is incremented each time the condition is met (the line has the string TEMPERATURE. n can then be used to calculate the cumulative average. In this case NR counts all lines in the file cp2k.out so we can't use that to calculate the average.*

How to plot both the instantaneous temperature and the cumulative average? (*hint: in gnuplot if you use replot instead of plot, the previous plot is retained*)

10) What is the smallest x-coordinate in cp2k.xyz?

X-coordinate is the first number, *i.e.* in the second column on the file. You can sort the file numerically (**-n**) according to the column (**-k**) you want.

```
$ cat cp2k.xyz | sort -n -k 2 | head
```

What are the coordinates of the O atom that has the smallest z-coordinate? (in the 4th column)

11) Working with data columns and fixing data format

Often it is necessary to change files slightly to use them in different analysis programs. This exercise simulates some typical changes you need to do. For example, NGS data from different sources may come in different syntax and to use them together needs fixing one or the other. This exercise shows an example on how to accomplish that.

Get file **`hsa.gff3`** from mirbase.org:

```
$ wget ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff3
```

1. Remove comment lines (lines starting with **#**)

```
$ grep -v "#" hsa.gff3 > tmp_nohash
```

2. Remove all lines that include tag '**`miRNA_primary_transcript`**'

```
$ grep -v "miRNA_primary_transcript" tmp_nohash > tmp_nomir
```

3. Change chromosome names (1st column) from format **`chr1, chr2`**, .. to format **`1, 2,...`**

```
$ cut -c 4- tmp_nomir > tmp_noscr
```

> *This command "cuts" i.e. prints everything from the 4th character on each line, i.e. cuts* ***away*** *the first three characters. This could also be done with sed. The following command would replace the first occurrence of "chr" on each line with nothing (what is between the second and third slash), i.e. remove those.* **`$ sed s/chr// tmp_nomir > tmp_noscr`**

4. The 9. column is now format

```
'ID=MIMAT0027618;Alias=MIMAT0027618;\
      Name=hsa-miR-6859-5p;Derives_from=MI0022705'.
```

Change the first item to format **`'gene_id "MI0006363_1"'`**

```
$ sed s/ID=/'gene id "'/ tmp_noscr > tmp_gene_id
```

> *Note that here we already print out the first " around the gene_id. We'll print the second " at the next stage.*

5. Leave out the last three columns i.e. **`Alias, Name`** and **`Derives`** entries and add the " after the **`gene_id`**.

```
$ awk -F ";" '{print $1 "\""}' tmp_gene_id > tmp_trimmed
```

> *First we tell* **`awk`** *to use ; as the field separator.* **`$1`** *now matches everything up to the first ; (i.e. until the* **`gene_id`** *code). Getting the* `"` *in place is a bit tricky. As* `"` *has a special meaning (it is not just a character) we need to escape it with* **`\`** *to mean just-the-character-*`"` *and not the meaning of* `"` *and finally quote that with* `"`*:s. An alternative way to do this in two steps is to use*

cut *to leave out everything after the first occurrence of* **;** *and then print the trailing " with* **awk** *(as above).*

```
$ cut -d ";" -f 1 tmp_gene_id | awk '{print $0 "\""}' > tmp_trimmed
```

6. Sort the file by chromosome and by miRNA start position (4. column). Make sure to sort the chromosomes in numerical order (**-n**), not in alphabetical (*i.e.* 1,2,3... not 1,10,11..)

```
$ sort -k1n,1 -k4n,4 tmp_trimmed > hsa.edited
```

Extra task. Make another file that only has entries from chromosome 2

It's possible to do the above in single command line, i.e. passing the output from the previous command as input to the next, but usually it is safer to use temporary files (until you know each of the steps work). Here is a one-liner that does steps 1-6:

```
$ grep -v "#" hsa.gff3 | grep -v "miRNA_primary_transcript" | cut -c 4- | \
      sed s/ID=/'gene id "'/ | awk -F ";" '{print $1"\""}' | \
      sort -k1n,1 -k4n,4 > hsa.edited
```