The sort command in Linux/Unix – explained

sort command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts a file assuming that the contents are in ascii. The use of options in sort command, can also allow to sort data numerically.

$ sort file_name↲ allows to sort a file in alphabetical order

$ sort -u file_name ↲ allows to sort a file in alphabetical order and remove duplicates.

$ sort -n file_name ↲ allows to sort a file numerically.

$ sort –r **file_name** ↲ allows to sort a file in reverse order.

$ sort -rn file_name ↲ allows to sort a file numerically, in reverse order.

$ sort file_name1 file_name2 ↲ will sort two files in ascending order

Files can be saved with multiple fields and delimiter.

If space or tab is used in a file to delimit the fields, then we do not need to specify the delimeter in the sort command. However if a special character like "," is used, then we need to specify in the sort command that "," represents a delimeter between fields.

Each field/column is represented using the option "k" . E.g. 3rd column will be represented as "k3".

Suppose we have a file named "data1.txt" containing:

5 3 2

4 1 3

6 2 1

To sort on third column, we use the sort command as follows:

$ sort –nk3 data1.txt ↲

Now, suppose we have a file “data2.txt” containing:

1,XXX,45.5

5,BBB,53.2

3,YYYY,30.3

To sort on third column, we use the sort command as follows:

$ sort -t"," -nk3 data2.txt ↲

Here **–t”,”** indicates that “,” is used to separate fields

The grep command in Linux/Unix

The `grep command` is used to search text or searches the given file for lines containing a match to the given strings or words. By default, grep displays the matching lines. Use grep to search for lines of text that match one or many regular expressions, and outputs only the matching lines. grep is considered as one of the most useful commands on Unix and other Linux operating systems.

To search for a keyword in a file, we simple type:

```
grep <keyword> <file_name>
```

e.g. to search for the word share in myFile

$ grep share myFile ↲

You can search recursively i.e. read all files under the /etc/ directory for a string share ↲

$ grep -r share /etc/ ↲

The above commands will match all words containing the word “share” e.g. “myshare”, “shareyours” etc…

If we want to do word match then we enclose the word to search in single quotes, e.g.

$ grep -r ‘share’ myFile ↲

To count the number of times a specific word occurs in a file, we enclose the word in single quotes as shown below

$ grep -c 'word' /path/to/file ↲

awk command - used for printing/displaying contents of a file

awk is a programming language which allows easy manipulation of structured data and the generation of formatted reports. awk stands for the names of its authors "**A**ho, **W**einberger, and **K**ernighan"

- awk allows to view a text file as records and fields.
- like common programming language, awk has variables, conditionals and loops
- awk has arithmetic and string operators.
- awk can generate formatted reports

Consider a file employee.txt containing the following info.

```
100  Thomas  Manager    Sales       $5,000
200  Jason   Developer  Technology  $5,500
300  Sanjay  Sysadmin   Technology  $7,000
400  Nisha   Manager    Marketing   $9,500
500  Randy   DBA        Technology  $6,000
```

To print the contents of employee.txt, we use awk as follows:

$ awk '{print;}' employee.txt ↲

Print the lines which matches with the pattern

$ awk '/Thomas/

> /Nisha/' employee.txt ↲

Note: Each pattern is represented on a new line

To print only the second and the fifth field

$awk '{print $2,$5;}' employee.txt ↲

To employees whose id >200

$ awk '$1 >200' employee.txt ↲

sed command in Linux/Unix

sed is a Stream Editor used for modifying the files in unix (or linux). Whenever you want to make changes to the file automatically, sed comes in handy to do this. Most people never learn its power; they just simply use sed to replace text. You can do many things apart from replacing text with sed. Below are described some features of sed with examples.

Consider a file “sample.txt” containing the following info:

```
Computing is the best.
But bioinformatics is better.
It takes long to shift from computing to bioinformatics
Oh computing! what a field is computing! I wish I did biology
```

To replace or substitute the first occurrence of word “computing” with “biology” on each line of sample.txt:

$sed 's/computing/biology/' sample.txt ↲

Replacing the second occurrence of word “computing” with “biology” on each line of sample.txt:

sed 's/computing/biology/2' sample.txt ↲

Replacing all occurrences of word “computing” with “biology” on each line of sample.txt:

sed 's/computing/biology/g' sample.txt ↲

Some important things about the practical sessions

1. Note that we are running a virtual machine **(vm)** with the Linux operating system. It is a mere coincidence that the machines are actually running Linux and the vm is also running Linux. It could be the case that a computer has Windows or Mac OS as its main operating system and it could run a vm which has Linux. The reason we are running this vm is that it is preloaded with the data and software that will be required for the practical sessions.

Note: If you try to access the web on the main operating system then this data **will not be available** in the vm because the vm is a separate machine. So, you have to be careful if you run Google Chrome or Firefox for your practical sessions, **pls run the one which is in the vm**.

Similarly if are reading your slides on pdf in the main machine, you cannot copy text from there and paste in the vm.

Running the vm is like running two operating systems on the same computer; one which is the actual one and another which is the vm.

2. Linux/Unix provide a rich set of commands to facilitate the life of end-users. However it is not possible to know all of them. We normally learn about these commands when we have to use them. Using the command-line interface provides a lot of advantage in the sense that one can easily develop workflows. However, one must use the command-line interface often to be familiar with all the commands.

   Some interesting sources of reference for learning/using Linux/Unix are:

   a) http://www.ualberta.ca/~stothard/downloads/linux_for_bioinformatics.pdf

b) http://www.ee.surrey.ac.uk/Teaching/Unix/

The best way to use/learn the Linux/Unix command-line interface:

$ man man ↲ provides help on how to use man

$ man command ↲ gives information about specific command

Solutions to practical questions on sort, grep, awk and sed

Create a file named OsTypes.txt

```
vi OsTypes.txt, press i and then insert the text
```

Unix
Linux
Solaris
AIX
Linux
Ubuntu
Windows
HPUX
When you are finished, press “esc”,”:”,”w”,”q”.
This will save your file and exit vi.

Next sort the file OsTypes.txt

```
sort OsTypes
```

Sort and remove duplicates

```
sort -u OsTypes.txt
```

Create a file named numbers1.txt

Write the following in numbers1.txt

20

19

5

49

200

Next sort the file numbers1.txt numerically

```
sort numbers1.txt
```

Sort numbers1.txt numerically in

reverse order

```
sort –nr numbers1.txt
```

Create a file named numbers2.txt

Write the following in numbers2.txt

25

18

5

48

200

```
sort -n numbers1.txt fnumbers2.txt
```

Create another file brca1pcr.txt Write the following brca1pcr.txt

3,AUGGA,65536

2,GAYYU,262144

1,MGNGU,2048

5,AUHAA,12288

4,GARWS,65536

Sort the file brca1pcr.txt

```
sort brca1pcr.txt
```

Now sort brca1pcr.txt based on third field

```
sort -n -t"," -k3 brca1pcr.txt
```

Finally sort brca1pcr.txt based on third field in reverse order

```
sort -nr -t"," -k3 brca1pcr.txt
```

Search for for the palindromic sequence TAAATAAAT in

mito.hg19.simulated.r1.fastq

```
grep TAAATAAAT mito.hg19.simulated.r1.fastq
```

Search for TAAAT in mito.hg19.simulated.r1.fastq

```
grep TAAAT mito.hg19.simulated.r1.fastq
```

Search how many times TAAAT occurs in mito.hg19.simulated.r1.fastq

```
grep -c TAAAT mito.hg19.simulated.r1.fastq
```

Create the file data1.txt (which contains Sequence id, Sequence name, organism, Genus and Sequence size) as follows:

....

Display the contents of the file data1.txt using awk

```
awk '{print;}' data1.txt
```

Print the lines which match **Myco**

```
awk '/Myco/' data1.txt
```

Display the sequence id and the sequence size on each line

```
awk '{print $2,$5;}' data1.txt
```

Display the lines where sequence size>3000

```
awk '$5 >3000' data1.txt
```

Create the file data2.txt as follows:

....

Display the contents of data2.txt where word "unix" is replaced with "linux"

```
sed 's/unix/linux/g' data2.txt
```

Display the contents of data2.txt where

word "unix" is replaced with "linux" for the second occurrence on each line

```
sed 's/unix/linux/2' data2.txt
```