WestGrid Autumn School 2020

Fahim Alam

Logging in to a remote system

You can connect to a remote HPC system via SSH (secure shell) using a terminal on your laptop. Linux and Mac laptops have built-in terminals, whereas on Windows we suggest using a free version of MobaXterm that comes with its own terminal emulator and a simple interface for remote SSH sessions.

```
fahim@fahim:~/Desktop$ ssh user005@cassiopeia.c3.ca
Password:
```

The password was rightly.really.proud.mutt

Navigating directories

To see the current directory, type “pwd”.

```
[user005@login1 ~]$ pwd
/home/user005
```

To see what we have in this directory, we can type “ls”

```
[user005@login1 ~]$ ls
projects  scratch
```

We see the files and the directory names but in order to distinguish them “ls -F”

```
[user005@login1 ~]$ ls -F
backup.txt  projects/  scratch@  thesis/
```

To see the hidden file,
we can type “ls -F -a” or we can combine them “ls -Fa”, we will see some file starts with (.) which are hidden files.

```
./  ../  .bash_logout  .bash_profile  .bashrc  projects/  scratch@
```

Absolute path ends with “/”. To change directory, we “cd directory name”. To go to home directory, type “cd” or “cd ~”. To go to previously visited directory “cd -”. To go top of the directory type “cd /”. To see the content in a particular directory, type “ls directory_name/”.

Getting help

To get help or to see manual page, type “man command” and type q to stop.

```
[user005@login1 ~]$ man ls
```

If manual page does not exist, we can use “help –command”.

Creating things

To create a directory, type ‘mkdir directory_name”.

```
[user005@login1 ~]$ mkdir thesis
[user005@login1 ~]$ cd thesis
```

To create a file type nano or type “nano filename.extension” then crtl + x, then press y.
To list the content of the file “cat filename.extension”. Or we can simply create a file in a specific directory type “nano directory_name/filename.extension” and to see it “cat directory_name/filename.extension”.

```
[user005@login1 thesis]$ nano
[user005@login1 thesis]$ cat draft.txt
```

To remove a directory type “rm -r directory_name/”. Use caution when using this. To remove a file in a particular directory, “rm directory_name/filename.extension”.

Moving and copying things

To copy a file, “cp file newfile”

```
[user005@login1 thesis]$ cp draft.txt backup.txt
[user005@login1 thesis]$ ls
backup.txt  draft.txt
[user005@login1 thesis]$ cat backup.txt
This is the draft of my thesis.
```

To see the attributes, type “ls -l”. To move a file to home folder use “..” and “.” to current directory.

```
[user005@login1 thesis]$ mv backup.txt ..
[user005@login1 thesis]$ mv ../backup.txt .
```

To move more than a file type “mv file file2 new_directory_name/”.
To move them back to current directory “ mv file/ new_directory_name file2/ new_directory_name .”
Moving is also same as renaming.
To rename a file, “rm oldfile newfile”.

Archiving and compressing

We can download a file from the internet, we can use “wget url ” but to give a different name type “ wget url -0 chosen_name.zip”.
The link: wget http://bit.ly/bashfile -O bfiles.zip

```
[user005@login1 tmp]$ wget http://bit.ly/bashfile -O bfiles.zip
```

To see the log,

```
[user005@login1 tmp]$ ls -l
total 244
-rw-r-----. 1 user005 user005 248892 Nov  2 18:45 bfiles.zip
```

To see the file size in kb type “ls -lh”.
To unzip, the file, “unzip filename.zip”

```
[user005@login1 tmp]$ unzip bfiles.zip
```

To see the content,

```
[user005@login1 tmp]$ ls -F data-shell/
cities.csv  data/       north-pacific-gyre/    writing/
creatures/  molecules/  wellsInvisibleMan.txt
```

To archive, we can type “tar cvf data.tar data-shell/”, where, c- create,v-visualize,f-follows. Now if check the size of the file, we see that the file is not archived it.

```
[user005@login1 tmp]$ ls -lh data.tar
-rw-r-----. 1 user005 user005 740K Nov  2 18:55 data.tar
```

To finalize, type “gzip data.tar”. Then see the size using ls -lh and you will see that file has been shrunk. To unzip the file, “gunzip data data.tar”. Then extract the file using “tar xvf data.tar” where x-extract.

Managing many files with Disk ARchiver (DAR)

File transfer

From your laptop, create a directory and create a file. Then to send this file to remote host, type “scp filename.ext host@name:home/”. Here- scp is secure copy.

`scp` is useful, but what if we don’t know the exact location of what we want to transfer? Or perhaps we’re simply not sure which files we want to transfer yet. `sftp` is an interactive way of downloading and uploading files. Let’s connect to a cluster with `sftp`:

```
[local]$ sftp userXXX@cassiopeia.c3.ca
```

```
sftp> pwd    # show our remote working directory
sftp> lpwd   # show our local working directory
sftp> ls     # show the contents of our remote directory
sftp> lls    # show the contents of our local directory
sftp> cd     # change the remote directory
sftp> lcd    # change the local directory
sftp> put localFile    # upload a file
sftp> get remoteFile   # download a file
```

Tapping the power of Unix

Wildcards, redirection to files, and pipes

```
[user005@login1 molecules]$ ls p*
pentane.pdb  propane.pdb
```

“ls *(character) will give us filename starts with that character. So see the size of the file and line type “wc filename” or “wc -l filename”. To get the same extension file,

```
[user005@login1 molecules]$ ls *.pdb
cubane.pdb  ethane.pdb  methane.pdb  octane.pdb  pentane.pdb  propane.pdb
[user005@login1 molecules]$
```

```
[user005@login1 molecules]$ ls *th*
ethane.pdb  methane.pdb
```

```
[user005@login1 molecules]$ wc -l *.pdb
  20 cubane.pdb
  12 ethane.pdb
   9 methane.pdb
  30 octane.pdb
  21 pentane.pdb
  15 propane.pdb
 107 total
```

To export the results in a file, we can use “>”. Try “wc -l *.(character) > file.txt”.

```
[user005@login1 molecules]$ cat list.txt
  20 cubane.pdb
  12 ethane.pdb
   9 methane.pdb
  30 octane.pdb
  21 pentane.pdb
  15 propane.pdb
 107 total
```

But the result we have is not sorted.

```
[user005@login1 molecules]$ cat sorted.txt
   9 methane.pdb
  12 ethane.pdb
  15 propane.pdb
  20 cubane.pdb
  21 pentane.pdb
  30 octane.pdb
 107 total
```

To get the first three line of a file type “head -3 filename.txt” and last line we get, “last -3 filename.txt”.

```
[user005@login1 molecules]$ head -3 sorted.txt
   9 methane.pdb
  12 ethane.pdb
  15 propane.pdb
[user005@login1 molecules]$
```

We can get the same thing by piping.

“wc -l *.(character) | sort -n | head -3”

```
[user005@login1 molecules]$ wc -l *.pdb | sort -n | head -3
   9 methane.pdb
  12 ethane.pdb
  15 propane.pdb
```

To create a file, type “touch filename”.

Aliases

Aliases are one-line shortcuts/abbreviation to avoid typing a longer command, e.g.

```
$ alias ls='ls -aFh'
$ alias pwd='pwd -P'
$ alias hi='history'
```

Loops

```
[user005@login1 creatures]$ for file in *.dat
> do
> echo $file
> done
basilisk.dat
unicorn.dat
```

To understand echo,

```
[user005@login1 creatures]$ echo hello > a1.txt
[user005@login1 creatures]$ cat a1.txt
hello
[user005@login1 creatures]$ cp a1.txt a2.txt
[user005@login1 creatures]$ echo secondline >> a2.txt
[user005@login1 creatures]$ diff a1.txt a2.txt
1a2
> secondline
```

```
user005@login1 creatures]$ for j in *.dat
 do
 echo $j
 ls -l $j
 echo
 done
asilisk.dat
rw-r--r--. 1 user005 user005 1838 Nov 17  2015 basilisk.dat

nicorn.dat
rw-r--r--. 1 user005 user005 1833 Nov 17  2015 unicorn.dat
```

Brace notations:

```
[user005@login1 creatures]$ touch a1.txt a2.txt a3.txt
[user005@login1 creatures]$
```

With brace notations,

```
[user005@login1 creatures]$ touch a{1..3}.txt
```

To remove the file rm a{1..3}.txt.
Back to the loops, we can put the result in a different file.

```
[user005@login1 creatures]$ for file in *.dat
> do
> mv $file original-$file
> done
[user005@login1 creatures]$ ls
original-basilisk.dat  original-unicorn.dat
```

Scripts and functions

```
[user005@login1 creatures]$ nano process.sh
```

We need a shebang to execute the file,

```
  nano 2.6.3

#!/bin/bash
echo Hello there !!!
```

To execute the created process.sh

```
[user005@login1 creatures]$ bash process.sh
Hello there !!!
```

For a bit complicated case,

```
[user005@login1 molecules]$ nano process.sh
[user005@login1 molecules]$ chmod +x process.sh
[user005@login1 molecules]$ ./process.sh

[user005@login1 molecules]$ ./process.sh one two three
one
```

where the process.sh contains just

```
#!/bin/bash
echo $1
```

Now we modify this file and add some more arguments.

```
  nano 2.6.3                          File: process.sh

#!/bin/bash
echo the first and third arguments are: $1 and $3
```

```
[user005@login1 molecules]$ nano process.sh
[user005@login1 molecules]$ chmod +x process.sh
[user005@login1 molecules]$ ./process.sh one two three
the first and third arguments are: one and three
```

Now lets try something different.

```
[user005@login1 molecules]$ chmod +x process.sh
[user005@login1 molecules]$ ./process.sh one two three
one
two
three
```

We will go to the molecules directory and do some more scripting.

```
#!/bin/bash
for molecule in $@
do
 echo $molecule
 head -3 $molecule
 wc -l $molecule
 echo -----
done
```

```
[user005@login1 molecules]$ ./process.sh *th*
ethane.pdb
COMPND      ETHANE
AUTHOR      DAVE WOODCOCK  95 12 18
ATOM      1  C           1      -0.752   0.001  -0.141  1.00  0.00
12 ethane.pdb
-----
methane.pdb
COMPND      METHANE
AUTHOR      DAVE WOODCOCK  95 12 18
ATOM      1  C           1       0.257  -0.363   0.000  1.00  0.00
9 methane.pdb
```

Variables

```
[user005@login1 molecules]$ myvar=3
[user005@login1 molecules]$ echo myvar
myvar
[user005@login1 molecules]$ echo $myvar
3
```

How about from a script? Let's rewrite our process.sh and get myvar executed.

```
#!/bin/bash
echo $myvar
```

```
[user005@login1 molecules]$ chmod +x process.sh
[user005@login1 molecules]$ ./process.sh

[user005@login1 molecules]$ export myvar=3
[user005@login1 molecules]$ ./process.sh
3
```

To reset the variable simply type “unset myvar”.

Some built-in variables,

```
[user005@login1 molecules]$ echo $HOME
/home/user005
[user005@login1 molecules]$ echo $HOME/tmp
/home/user005/tmp
[user005@login1 molecules]$ ls $HOME/tmp
a1  a2  bfiles.zip  data-shell  data.tar
[user005@login1 molecules]$ ls ~/tmp
a1  a2  bfiles.zip  data-shell  data.tar
[user005@login1 molecules]$
```

One more is, $PATH.

```
[user005@login1 molecules]$ echo $PATH
/cvmfs/soft.computecanada.ca/easybuild/software/2017/avx2/Compiler/gcc7.3/openmp
i/3.1.2/bin:/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/gcc-7.3.0/bin:/cvm
fs/soft.computecanada.ca/easybuild/software/2017/Core/imkl/2018.3.222/mkl/bin:/c
vmfs/soft.computecanada.ca/easybuild/software/2017/Core/imkl/2018.3.222/bin:/cvm
fs/soft.computecanada.ca/custom/bin/computecanada:/cvmfs/soft.computecanada.ca/e
asybuild/bin:/cvmfs/soft.computecanada.ca/custom/bin:/cvmfs/soft.computecanada.c
a/nix/var/nix/profiles/16.09/bin:/cvmfs/soft.computecanada.ca/nix/var/nix/profil
es/16.09/sbin:/opt/software/slurm/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/u
sr/sbin:/opt/puppetlabs/bin:/home/user005/.local/bin:/home/user005/bin
[user005@login1 molecules]$
```

Last but not least, $PS1,

```
[user005@login1 molecules]$ echo $PS1
[\u@\h \W]\$
```

Set up bash custom prompt (PS1)

```
[user005@login1 ~]$  echo $PS1
[\u@\h \W]\$
[user005@login1 ~]$ PS1="FahimAlam :"
FahimAlam :
```

```
FahimAlam :PS1="[\d \t \u@\h:\w ] $ "
[Mon Nov 02 22:08:10 user005@login1:~ ] $ PS1="[\d \t \u@\Fahim:\w ] $ "
[Mon Nov 02 22:09:11 user005@\Fahim:~ ] $ PS1="[\d \t Fahim:\w ] $ "
[Mon Nov 02 22:09:30 Fahim:~ ] $
```

Functions

```
[user005@login1 ~]$ greeting(){
> echo hi there!
> }
```

If we just type “greeting” we will hi there.
Rewrite the function as,

```
[user005@login1 ~]$ greeting(){
> echo the arguments are $@
> }
[user005@login1 ~]$ greeting a b c
the arguments are a b c
```

To get a specific output,

```
[user005@login1 ~]$ greeting(){ echo the arguments are $2; }
[user005@login1 ~]$ greeting a b c
the arguments are b
```

To get the number of output,

```
[user005@login1 ~]$ greeting(){ echo the arguments are $#; }
[user005@login1 ~]$ greeting a b c 4 5
the arguments are 5
```

All together as a script,

```
function greeting(){
 echo the 3rd argument is $3
 echo we have $# arguments and they are $@.
}
```

Since its a text file i.e does not have shebang. We do the following steps to run it.

```
[user005@login1 ~]$ nano greeting.sh
[user005@login1 ~]$ source greeting.sh
[user005@login1 ~]$ greeting a b c 4
the 3rd argument is c
we have 4 arguments and they are a b c 4.
```

Try something different,

```
function greeting(){
 echo the 3rd argument is $3
 echo we have $# arguments and they are $@.
}
function combine(){
 if [$# -eq 0 ]; then
   echo No arguments specified.
   return 1
 fi
 dir=$RANDOM$RANDOM
 mkdir $dir
 mv $@ $dir
 echo moved these files into $dir
}
```

```
[user005@login1 molecules]$ ls
2343431308  cubane.pdb  greeting.sh  octane.pdb   process.sh   sorted.txt
834320731   ethane.pdb  list.txt     pentane.pdb  propane.pdb
[user005@login1 molecules]$ combine propane.pdb
-bash: [1: command not found
moved these files into 8096515
[user005@login1 molecules]$
```

We can use this function in a different directory.

```
user005@login1 molecules]$ cd ../creatures/
user005@login1 creatures]$ ls
al-basili  nal-unicor  process.sh
user005@login1 creatures]$ combine nal-basili
bash: [1: command not found
oved these files into 180957679
```

Searching inside files with grep

We need to change our directory for this case,

```
[user005@login1 creatures]$ pwd
/home/user005/tmp/data-shell/creatures
[user005@login1 creatures]$ cd ..
[user005@login1 data-shell]$ cd writing
[user005@login1 writing]$
```

We will work with the file haiku.txt and find “day” using grep.

```
[user005@login1 writing]$ ls -F
data/  haiku.txt  old/  thesis/  tools/
[user005@login1 writing]$ cat haiku.txt
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
[user005@login1 writing]$ grep day haiku.txt
Yesterday it worked
Today it is not working
[user005@login1 writing]$
```

To get exactly day, we have to type " grep -w day haiku.txt" in this case, there will be no result since we dont have any word named day in the file.

```
[user005@login1 writing]$ grep -i -w today haiku.txt
Today it is not working
[user005@login1 writing]$
```

To get the line number as well type " grep -n -i -w today haiku.txt". To get the lines which does not contain the word "today", we use,

```
[user005@login1 writing]$ grep -i -w -v -n today haiku.txt
1:The Tao that is seen
2:Is not the true Tao, until
3:You bring fresh toner.
4:
5:With searching comes loss
6:and the presence of absence:
7:"My Thesis" not found.
8:
9:Yesterday it worked
11:Software is like that.
```

```
[user005@login1 writing]$ grep -w of haiku.txt
and the presence of absence:
```

Finding files with find

```
[user005@login1 writing]$ find . -name haiku.txt
./haiku.txt
[user005@login1 writing]$
```

Here, “.” stands for current directory.
To get all the text file,

```
[user120@login1 writing]$ find . -name "*.txt"
./haiku.txt
./data/two.txt
./data/one.txt
```

To search for files

```
[user120@login1 writing]$ find . -type f
./tools/old/oldtool
./tools/format
./tools/stats
./old/.gitkeep
./haiku.txt
./thesis/empty-draft.md
./data/two.txt
./data/one.txt
```

To search for directory

```
[user005@login1 writing]$ find . -type d
.
./tools
./tools/old
./old
./thesis
./data
```

Combining find and grep

```
[user005@login1 writing]$ find . -name "*.txt"
./haiku.txt
./data/two.txt
./data/one.txt
```

To get the same result as a list,

```
[user005@login1 writing]$ echo $(find . -name "*.txt")
./haiku.txt ./data/two.txt ./data/one.txt
```

Using loop,

```
[user005@login1 writing]$ for file in $(find . -name "*.txt")
> do
>
>   echo $file
> done
./haiku.txt
./data/two.txt
./data/one.txt
```

To get pattern, we can combine grep with find,

```
[user005@login1 writing]$ for file in $(find . -name "*.txt");do grep day $file;
 done
Yesterday it worked
Today it is not working
"My dear Mr. Bennet," said his lady to him one day, "have you heard that
down on Monday in a chaise and four to see the place, and was so much
till the day before; so it will be impossible for her to introduce him,
pleasant, I can tell you, to be making new acquaintances every day; but
In a few days Mr. Bingley returned Mr. Bennet's visit, and sat about
was obliged to be in town the following day, and, consequently, unable
day before the ball by hearing, that instead of twelve he brought only
"My dear Mr. Bennet," said his lady to him one day, "have you heard that
down on Monday in a chaise and four to see the place, and was so much
```

```
[user005@login1 writing]$ find . -name "*.txt" | xargs grep day
./haiku.txt:Yesterday it worked
./haiku.txt:Today it is not working
./data/two.txt:"My dear Mr. Bennet," said his lady to him one day, "have you hea
rd that
./data/two.txt:down on Monday in a chaise and four to see the place, and was so
much
./data/two.txt:till the day before; so it will be impossible for her to introduc
e him,
./data/two.txt:pleasant, I can tell you, to be making new acquaintances every da
y; but
./data/two.txt:In a few days Mr. Bingley returned Mr. Bennet's visit, and sat ab
out
./data/two.txt:was obliged to be in town the following day, and, consequently, u
nable
./data/two.txt:day before the ball by hearing, that instead of twelve he brought
 only
./data/one.txt:"My dear Mr. Bennet," said his lady to him one day, "have you hea
rd that
./data/one.txt:down on Monday in a chaise and four to see the place, and was so
much
```

Text manipulation

We change our directory.

```
[user005@login1 writing]$ pwd
/home/user005/tmp/data-shell/writing
[user005@login1 writing]$ cd ..
[user005@login1 data-shell]$
```

We will work with the file “wellsInvisibleMan.txt” to see the existence of the file type “ls -l”
Then we try to find number of occurrence the word “invisible”.

```
[user005@login1 data-shell]$ grep invisible wellsInvisibleMan.txt | wc -l
60
[user005@login1 data-shell]$
```

Replace invisible to visible and put it into a different file,

```
[user005@login1 data-shell]$ sed 's/[Ii]nvisible/supervisible/g' wellsInvisibleM
an.txt > supervisible.txt
[user005@login1 data-shell]$ grep supervisible supervisible.txt|wc -l
173
[user005@login1 data-shell]$
```

Now we will remove all the quotation mark and put in to a new file called a1.txt.

```
[user005@login1 data-shell]$ cat wellsInvisibleMan.txt | tr -d "[:punct:]" > a1.
txt
```

To see whether we succeeded or not, we check the last few lines of the original file.

```
[user005@login1 data-shell]$ tail-10 a1.txt
-bash: tail-10: command not found
[user005@login1 data-shell]$ tail wellsInvisibleMan.txt
"Once I get the haul of them--_Lord_!"

"I wouldn't do what _he_ did; I'd just--well!" He pulls at his
pipe.

So he lapses into a dream, the undying wonderful dream of his life.
And though Kemp has fished unceasingly, no human being save the
landlord knows those books are there, with the subtle secret of
invisibility and a dozen other strange secrets written therein.
And none other will know of them until he dies.
```

And then lastly the newly created file,

```
[user005@login1 data-shell]$ tail a1.txt
Once I get the haul of themLord

I wouldnt do what he did Id justwell He pulls at his
pipe

So he lapses into a dream the undying wonderful dream of his life
And though Kemp has fished unceasingly no human being save the
landlord knows those books are there with the subtle secret of
invisibility and a dozen other strange secrets written therein
And none other will know of them until he dies
[user005@login1 data-shell]$
```

Now we convert all upper case letter to lower case,

```
[user005@login1 data-shell]$ cat a1.txt | tr '[:upper:]' '[:lower:]'> a2.txt
[user005@login1 data-shell]$ tail a2.txt
once i get the haul of themlord

i wouldnt do what he did id justwell he pulls at his
pipe

so he lapses into a dream the undying wonderful dream of his life
and though kemp has fished unceasingly no human being save the
landlord knows those books are there with the subtle secret of
invisibility and a dozen other strange secrets written therein
and none other will know of them until he dies
[user005@login1 data-shell]$
```

We will use sed command to replace space into a new line.

```
[user005@login1 data-shell]$ cat a2.txt | sed 's/ /\'$'\n/g'>a3.txt
```

Using “more a3.txt”, we see the following,

```
the
invisible
man

a
grotesque
romance

by
n
g
wells



contents
```

We can see there are lots of whitespaces. We can remove it.

```
user005@login1 data-shell]$ sed '/^$/d' a3.txt >a4.txt
user005@login1 data-shell]$
```

We see that there is no more spaces.

```
the
invisible
man
a
grotesque
romance
by
h
g
wells
contents
i
the
strange
mans
arrival
ii
mr
teddy
henfreys
first
impressions
iii
```

Now we will sort them alphabetically.

```
user005@login1 data-shell]$ cat a4.txt | sort | uniq -c > a5.txt
user005@login1 data-shell]$ more
```

```
   1 2d
1339 a
   1 aback
   2 abandoned
   1 abandoning
   1 abandonment
   1 abject
   5 able
   1 ableeding
   1 abnormal
   2 aboard
 114 about
  10 above
   3 abroad
   1 abroadin
   1 abrupt
  21 abruptly
   2 absolutely
   3 absorb
   1 absorbed
   3 absorbs
   1 abstraction
   1 absurd
```

To get the frequent words,

```
[user005@login1 data-shell]$ cat a5.txt | sort -gr > a6.txt
[user005@login1 data-shell]$
```

```
3303 the
2032 and
1339 a
1312 of
 985 to
 960 he
```

Column-based text processing with awk scripting language

Let’s change our directory back to writing “cd writing”.
Now lets open the haiku.txt by cat haiku.txt.
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.

We want to get the first word of every line.

```
[user005@login1 writing]$ awk '{print $1}' haiku.txt
The
Is
You

With
and
"My

Yesterday
Today
Software
```

To get the 2nd line we type “awk ‘{print $2}’ haiku.txt and instead of $2 ,if we tried $0 we will get the whole line.
Now we want to print everything up-to a character say ‘a’.

```
[user005@login1 writing]$ awk -Fa '{print $1}' haiku.txt
The T
Is not the true T
You bring fresh toner.

With se

"My Thesis" not found.

Yesterd
Tod
Softw
```

We create a new text and append a text to the new text as well by “>>”.

```
[user005@login1 writing]$ echo hello tom > hello.txt
[user005@login1 writing]$ echo hello john >> hello.txt
[user005@login1 writing]$ cat hello.txt
hello tom
hello john
```

We can change the names to a different name, say Adam.

```
[user005@login1 writing]$ awk ' {$2 = "Adam"; print $0}' hello.txt
hello Adam
hello Adam
```

When we type “ls-l” we lots of output and some of them are redundant. We can fix this,

```
[user005@login1 writing]$ ls -l
total 8
drwxr-xr-x. 2 user005 user005  36 Nov 17  2015 data
-rw-r--r--. 1 user005 user005 218 Nov 17  2015 haiku.txt
-rw-r-----. 1 user005 user005  21 Nov  3 00:45 hello.txt
drwxr-xr-x. 2 user005 user005  22 Nov 17  2015 old
drwxr-xr-x. 2 user005 user005  28 Nov 17  2015 thesis
drwxr-xr-x. 3 user005 user005  44 Nov 17  2015 tools
[user005@login1 writing]$ ls -l | awk '{print $5 " " $9}'

36 data
218 haiku.txt
21 hello.txt
22 old
28 thesis
44 tools
[user005@login1 writing]$
```

Awk can handle pattern too.

```
[user005@login1 writing]$ awk '/Yesterday|Today/' haiku.txt
Yesterday it worked
Today it is not working
[user005@login1 writing]$ awk '/Yesterday|Today/{print $3}' haiku.txt
worked
is
```

To print the lines from 3,

```
[user005@login1 writing]$ awk 'NR>3' haiku.txt

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
```

To get the lines from 2 and 3 rd only,
“awk ‘NR>1 && NR<4’ haiku.txt