Linux Command Line – A Guided Tour!

Lakshmipathi.G

Contents

Day-6

- 6.1 Remember this !!!!
- 6.2 How long the machine is up and running?
- 6.3 Who is it?
- 6.4 What's your hostname and version

Know your location:

Assume, your user name is "efg".
When you first login to a linux system, it will display "Last login: " Time and date and content of the file /etc/motd (this file is owned by the root user) and then it will greet you with a bash prompt, typically you will get something like -

```
[efg@fedori ~]$
```

"fedori" here refers the hostname. "$" denotes the bash shell is ready to accept your commands.

Where Am I ?

Let's understand how you landed in this location. By default, when you login, you will be redirected to a directory specified in the /etc/passwd file. This entry is created when the system-admin created an account for you and all /etc/ are configuration files. The /etc/passwd file has an entry for you, like -

```
efg:x:500:500:EFGname:/home/efg:/bin/bash
```

Let's split them to understand it-

efg - login name
x - Password (place holder in modern system, user password will be stored in another encrypted file /etc/sh
adow)
500 - unique user id
500 - groub id
EFGname - real name
/home/efg - home directory
/bin/bash - shell type

The summary of this entry would be - after verifying password for user "efg" in the /etc/shadow file,
allow user "efg" to login and place him under "/home/efg" with bash as the shell type.

You can get your current (working) directory using the command "pwd" (pRINT wORKING dIRECTORY).

What's a command

First of all, what is a command? It's a binary (also known as executable) file kept under a specific location. Guess where? Near the washing machine? No .... near the TV? ..no again.
Okay. Here is a hint, you can use another command called "whereis" to find the location of the command. Lets try it first:

```
[efg@fedori ~]$ whereis pwd
pwd: /bin/pwd
```

pwd binary can be found under the /bin directory. Wait a second, before running the pwd command.
There is more than one location where the binary files are kept.
You can view then using:

```
[efg@fedori ~]$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/efg/bin
```

When you type something on bash prompt,
```
[efg@fedori ~]$something
```

Bash assumes you entered a command called "something" and check it first on the current directory and then paths specified by the PATH environmental variable -so first it goes to "/usr/local/bin" and finds there is no file called "something",
then it checks /usr/bin and then /bin, etc.If it didn't find any file name it'll display the "command not found" error message as shown below-

```
[efg@fedori ~]$ something
bash: something: command not found

[efg@fedori ~]$ pwd
/home/efg
```

So you are in the right place, at your home directory as expected by your /etc/passwd entry.
How to "clear" above errors messages from the screen? ..there is command called ..
you figure this out yourself, the question itself has the "answer". :)

How to create a new directory?

You are at your home. It's nothing but a directory you can create your own directory using the command,

```
[efg@fedori ~]$touch dir0
```
If no error message is displayed after executing this command,then your directory is created.

Then in order to check the presence of the directory, use the "ls" command.
```
[efg@fedori ~]$ls
dir0
```
It should display your newly created directory.

Another way to create a directory is by using "mkdir" (mAkE dirECTORY)
```
[efg@fedori ~]$mkdir dir1
```

If no message is displayed after executing this mkdir command {it means it went without error}.
Then in order to list directory contents, use the "ls" command.
```
[efg@fedori ~]$ls
dir0 dir1
```
displays your newly created directories.

Analyzing ls command output

ls - command will display directory contents. If you want to know about it's permission,use "ls -l"

```
[efg@fedori ~]$ls -l
-rw-rw-r--  1 efg efg          0 2010-08-13 08:13 dir0
drwxrwxr-x  2 efg efg       4096 2010-08-13 08:15 dir1
```

Let's understand this output first, see the second line, can you see 4096, yes? yes? Okay,
then ignore it for now :D on its right side, you will see "date and time and directory name".
-date/time here refers creation time for this entry.And on its left side, you have four entries separated
by white fedori(blank fedori)."drwxrwxr-x 2 efg efg"
"efg efg" says user named "efg" who belongs to the group called "efg" is owner of this directory and group
member can also use this directory.

So what's is a group? We will talk about this while dealing with root user commands. For now just think of it
as your friends.

Lets check the first entry -
drwxrwxr-x

It denotes the permissions for this directory. In GNU-Linux,
the first "d" letter mention its a directory.
In general,rwx stands for read, write and execute.
then we have rwx repeated twice.
First three letters, (rwx) always denotes owner and second three (rwx) always refers
to group or friends and the final three,(r-x) always refers to all others or unknown
people who uses this machine. "-" here denotes,others don't have write permission,
they can only view/search this directory.

Now let's examine the first line,
-rw-rw-r-- 1 efg efg   0 2010-08-13 08:13 dir0
Yes, I can hear you screaming, "It doesn't have 'd' letter at it's begining? What happened?"
Remember the command used to create this one, "touch" - I lied about it :)- touch creates an empty file not a directory. So there is only one way to create directory,
that is to use "mkdir" command.

A touching note: -if the file doesn't exists, the touch command will create an empty file,
if file already exists, it updates its timestamp.

Okay,back to exploration-
Since "dir0" is regular file - it don't have a "d" at beginning, instead uses "-". Only special
files are denoted by first entry (a directory is also a special file).
rw-, says owner "efg" can read and write into this file and he can't execute it.
Similarly next three

fields "rw-",users belong to group "efg" can read and write.Final three entries "r--" says other can read/view

this file content ,but they can't modify or execute it.

since touch creates an empty file 0 - denotes its size. and date and time of creation along with file name is
displayed on the rest of the line.

Now we have two entries on our home current directory and lets visualize them.- Close your eyes and think
about it! ..wait don't sleep :)

```
/home/efg/  (our working directory.)
        |
        |-------------dir0
        |-------------dir1/
```

how to move "into" newly created directory? Can you guess the command for it? If you think "into" as the answe
r,
congrats!!!, you are dead wrong :)

Move into a directory

use "cd" command,which stands for "cHANGE dIRECTORY",to move into a specific directory.

Before you use cd command on directory, what will happen, if you use it with our file
dir0? Let's try that first -

```
[efg@fedori ~]$ cd dir0
bash: cd: dir0: Not a directory
```

"cd" is wise, it checks "dir0" and realizes it don't have "d" on it's beginning and says
this error message. Good.Now move on,

```
[efg@fedori ~]$cd dir1
```

We moved into a whole new directory. We can verify this by using "pwd" command and list the directories
content using ls and ls -l . Enjoy.

Oh oops ..there is nothing! Try it again but this time pass 'a' as an option for ls command
 "ls -la" . a surprise awaits you!

```
[efg@fedori dir1]$ ls -al
total 8
drwxrwxr-x 2 efg efg 4096 2010-08-13 08:15 .
drwx------ 5 efg efg 4096 2010-08-13 08:15 ..
```

What's . and .. ?

Whats this "." dot and ".." dot dot refers here? Let's wear the hat of Sherlock Holmes and
investigate this case as "the Curious case of . and .. " Two striking things are both entries
has "d" in front and same time stamp.

Check the timestamp of "dir1" when it's created using mkdir. Got it? both are same,so these
two entries "." and ".." are created when mkdir command was used. What they actually refer to?
Here comes the ls command to the rescue again, now pass "i" option to it.

```
[efg@fedori dir1]$ ls -lia
total 8
1966463 drwxrwxr-x 2 efg efg 4096 2010-08-13 08:15 .
1966452 drwx------ 5 efg efg 4096 2010-08-13 08:15 ..
```

Inode numbers

Numbers in front are called "Inode" numbers. These numbers are actually used by linux kernel.
Each file has a unique inode number.We use filename or directory names but kernel uses only these numbers.

Repeat the same procedure for the /home/efg directory, you can use ls command and pass
the directory name as argument:

```
[efg@fedori dir1]# ls -lia /home/efg
1966452 drwx------  5 efg efg 4096 2010-08-13 21:22 .
 172034 drwxr-xr-x 13 root   root   4096 2010-08-13 08:56 ..
1966461 -rw-------  1 efg efg  109 2010-08-13 08:10 .bash_history
1966457 -rw-r--r--  1 efg efg  18 2010-08-13 08:10 .bash_logout
1966455 -rw-r--r--  1 efg efg  176 2010-08-13 08:10 .bash_profile
1966453 -rw-r--r--  1 efg efg  124 2010-08-13 08:10 .bashrc
1966460 -rw-rw-r--  1 efg efg          0 2010-08-13 08:13 dir0
1966463 drwxrwxr-x  2 efg efg       4096 2010-08-13 08:15 dir1
```

More Surprises ! Who created all those files that begin with a dot?. Our investigation takes a new turn,
looks like this case will take sometime to solve. -Don't worry! Its not a judical enqury, so we will have verd
ict soon :)

Let's focus on how "." and ".." files created under dir1 first and we can check others file, a little later.

```
1966463 drwxrwxr-x  2 efg efg       4096 2010-08-13 08:15 dir1
```

this entries inode number is same as "." file under "dir1". So the file /home/efg/dir1/. denotes the directory
 itself.
And Just look above under "ls -lia /home/efg" here we have inode number same as /home/efg/dir1/.. - so initia
l verdict is,
all directories have two files named "." and ".." where "." refers to that directory itself and ".." refers to
 its parent

directory.

Absoulte and Relative path names

These ('.' and '..') are used for relative path names, so far we have used absolute path names.
Absolute path names begin with / (root) example : /home/efg/dir1
Relative path names begin with "." or ".." or without "." or ".." or "/"
Check your current path - using pwd command it will tell its "/home/efg/dir1"
you can access dir0 file using relative path like,

```
[efg@fedori dir1]$ls -l ../dir0
```

You can access files under current directory using "."

since we don't have any files here,first create a file using touch
```
[efg@fedori dir1]$touch giis.txt
```

and access it using
```
[efg@fedori dir1]$ls -l ./giis.txt
```

here is the final relative path method,which won't use . or .. or /

simply call,
```
[efg@fedori dir1]$ls -l giis.txt
```

Day-2

Understanding and manipulating file contents:

Okay, now lets understand the dot "." files under /home/efg namely ".bash_history" ,
".bash_logout" , ".bash_profile" , ".bashrc".

Go back to /home/efg using command
[efg@fedori dir1]$cd ..

Did you understand what above command really mean? ...what? NO? then re-read topic "Move into a directory".Then Came back ..I'll count till 10000 and wait for you. 10000,9999,9998...3,2,1,0. okay.Now lets first understand ,.bash_history ,file is used
to keep track of shell commands. You can view the content of file using "cat" command.

[efg@fedori ~]$cat .bash_history

will display commands and everything you have typed in shell prompt. "history" command used
to read this file and display it on screen.
[efg@fedori ~]$history
It can used by root user to keep track of what you are doing.. I can hear you "aaawk!! how to remove
this feature? Can I delete this file?" yes you can delete this file using the "rm" command.

[efg@fedori ~]$rm .bash_history
This will prompt you for confirmation. Since any deleted files can't be recovered, unless you have
fail-safe tools or hoping that FS didn't remove its final references from the journal log.

But the sad news is, this bash_history will be created automatically whenever you type something into
bash prompt. But this file is useful, we will see that in next section.

".bash_profile" , ".bashrc" files have environmental variable specific to this user. Say for example,
you want to change bash prompt

[efg@fedori ~]$

to it look like root# - so that you can fool your friend, saying you got root access. So edit your
.bash_profile and add following line -

export PS1="root#"

How to append this entry to file .bash_profile? Read on...

Understanding cat command:

we have used "cat" command view the content of .bash_history earlier.Lets understand some basic terms of GNU-L

inux.

your kepboard (or anyother input device) is referred as STDIN - standard input device,
your screen (or anyother output device) is referred as STDOUT - standard output device,
finally normally screen is also referred as STDERR- standard error device.

by default cat reads from stdin and writes into stdout device. (ie) it reads from your keyboard and write into
 screen.

To verify this simply type cat,
[efg@fedori ~]$cat
now type something as input (cat is reading from keyboard) and press "Enter" key,it will display whatever you
typed on the screen itself,which is your
default stdout.

Did you now realize what's stdio and stdout? so now type something like junk word m5do5(btw, to come out of "c
at" command press "ctrl" and "d" simulateously -which
tells the cat command - end of input,take me back to bash prompt) ...were where we? at some junk m5do5 :)

[efg@fedori ~]$ m5do5
bash: m5do5: command not found
Error message is displayed on default stderr ,which is your screen.

So when you said,"cat .bash_history" ,you are telling cat command ,"hey cat,this time ,treat given file as std
in".
and cat did the same,it read your file content and displayed on stdout.

Input/Output Redirection

you can also override stdout,and ask "cat,please treat file1 as my input device and redirect the output to fil
e2 instead of default stdout,which is screen."

">" is called redirection operator for output stream.

[efg@fedori ~]$cat .bash_history > history_file.txt

"<" is input stream, using

 [efg@fedori ~]$cat .bash_history

is same as,telling cat to get input from file followed by "<" symbol.
[efg@fedori ~]$cat < .bash_history

">>" refers to append the content to existing file.

So to append the environment variable,
[efg@fedori ~]$echo "PS1=\"root#\"" >> .bash_profile

above command will set the enviromental files. echo command by default,displays output in stdout,we told echo
to redirect it to file.
\ is used here as an escape charater.we will see about escapte character,little later.

Check your file .bash_profile ,the last line would be PS1="root#" using cat command,but wait there is two othe
r common method to view content of file.

head & tail of file

[efg@fedori ~]$head .bash_history
head command will display ,first 10 lines of a file ,if you want to change number of line ,pass and option "-4
" .

[efg@fedori ~]$head -4 .bash_history
will display first four lines from the file.

tail,You guessed it correctly, is exact opposite of head.it will display last 10 lines and you can manipulate
it by passing the number,

[efg@fedori ~]$tail -1 .bash_history
PS1="root#"

Now logout and re-login again
[efg@fedori ~]$logout

Now provide your username & passwd. Eureca ! you will see this-
root#

".bash_logout" can used to have commands that ends to be executed when you log off. How about putting the comm
and to delete .bash_history in .bash_logout?

wait,.bash_history or history command is indeed useful one.

When you run history command ,it will display commands along with numbers on its right. If you want run one of
 those commands again,you can simply using

[efg@fedori ~]$!1
will the run command show by:
[efg@fedori ~]$head 1 .bash_history

Day-3

More File operations:

Now lets start manipulating files itself,rather than its data.

Taking file backup with cp

How to create copy of the file?,Say,we need to take a backup of .bash_history file from your home directory to
another directory dir1.
[efg@fedori ~]$cp .bash_history dir1

now verify the dir1 contents -

[efg@fedori ~]$ls -la dir1

You should find the file now it two places one under your home directory and another under "dir1".

Verifying file integrity

Seems to be there but how can we 100% sure ,that file integrity is not compromised?
you can use 'head' and 'tail' to verify few starting and ending lines,if they are fine - we are okay!.
what if something in middle got messed up? but wait they universally (yeah,even martians use this trick) acce
pted
method. Use md5sum.

[efg@fedori ~]$md5sum .bash_history
8599e55388ecef39d8905b1c67b044ef .bash_history

[efg@fedori ~]$md5sum dir1/.bash_history
8599e55388ecef39d8905b1c67b044ef dir1/.bash_history

As you can see both numbers/checksum are same.Thus we can be sure both files are same.

How to move file from one location to another?

you can simply move a file to another directory
[efg@fedori ~]$mv giis.txt dir1
now verify the dir1 contents -
[efg@fedori ~]$ls -la dir1

Since we moved (not copied) this file will be available only one location which is "dir1".

Here is a question for you,If you copying 10MB file into a directory and also moving (mv) 15GB file into a dir
ectory "dir1".
Which operation will complete first ? is it 10MB copy? or 15GB move?

Protecting files with chmod:

As we seen above,someone can use cp and mv command to copy/move your files to another location.How to prevent
,someone from accessing your files?
Remember 'ls -l' which we used long long ago? yeah,use it again.

```
[efg@fedori ~]$touch secretfile

[efg@fedori ~]$ls -l secretfile
-rw-rw-r--.  1 efg efg              0 2011-05-04 22:40 secretfile
```

As you can see owner and group is given read/write permission and 'others' given read-only permission.
Clearly other can't write something into your 'secretfile' since 'w' is missing for others.
but they can copy your file to some other location. In order to avoid that ,we can use 'chmod'.

```
[efg@fedori ~]$ chmod o-r secretfile
[efg@fedori ~]$ ls -l secretfile
-rw-rw----. 1 efg efg 0 2011-05-04 22:40 secretfile
```

o stands for others and '-r' means remove read permission.
If you want to remove rw from groups too,then use 'g' followed '-rw'

```
[efg@fedori ~]$ chmod g-rw secretfile
[efg@fedori ~]$ ls -l secretfile
-rw-------. 1 efg efg 0 2011-05-04 22:40 secretfile
```

If you are someone like "Leonard" (hero from memento) ,and you want to extra careful ,even you do not to mess
up with it ,
then you yourself can remove its write permission like -

```
[efg@fedori ~]$ chmod u-w secretfile
[efg@fedori ~]$ ls -l secretfile
-r--------. 1 efg efg 0 2011-05-04 22:40 secretfile
```

After some time (15 minutes later :D) , suddenly realize you got to modify the this file,so need to add 'w' to
 yourself.
When removing permission we used '-' ,now we want to add them so whatelse the symbol would be other '+'?

so do it like -

```
[efg@fedori ~]$ chmod u+w secretfile
[efg@fedori ~]$ ls -l secretfile
-rw-------. 1 efg efg 0 2011-05-04 22:40 secretfile
```

Links

For example ,you created a new directory named "dir2"

```
[efg@fedori ~]$ mkdir dir2
```

and want to make sure that 'secretfile' is accessible from both dir1 and dir2.
(though secretfile has 0 size ,just assume it has around 5GB data thus size=5GB)

If you cp the 5GB secretfile to dir2 and dir1.On the whole it consume 15GB (5 each for secretfile at home,dir1
 and dir2)
Since file contents are same, its not a good idea to let them occupy 15GB.
Here is where links will be useful.How about creating links from dir1 and dir2 instead of cp them?

```
[efg@fedori ~]$ ls -lt secretfile
-rw-------. 1 efg efg 0 2011-05-04 22:40 secretfile
```

links are created by ln command like -

```
[efg@fedori ~]$ ln secretfile dir1/secretfile
```

check the dir1 contents -

```
[efg@fedori ~]$ ls -lt dir1/secretfile
-rw-------. 2 efg efg 0 2011-05-04 22:40 dir1/secretfile
```

yeah,we have it.

```
[efg@fedori ~]$ ls -lt secretfile
-rw-------. 2 efg efg 0 2011-05-04 22:40 secretfile
```

now wait a second , rewind 10 lines - check the first ls
it shows "1" before owner "efg" - right? But now check the last ls output
it shows "2". Why?
That's because secretfile is being linked from two location now after ln command.
One from original home and another from dir1.

So now if we create another link to dir2 - like

```
[efg@fedori ~]$ ln secretfile dir2/secretfile
```

Now the link count increased by 1,its new count 2+1 which is ..hmmmm...where the heck is my calculator ?!
yeah..got it..its 3.So the ls should shows as 3.

```
[efg@fedori ~]$ ls -lt dir2/secretfile
-rw-------. 3 efg efg 0 2011-05-04 22:40 dir2/secretfile
[efg@fedori ~]$ ls -lt dir1/secretfile
-rw-------. 3 efg efg 0 2011-05-04 22:40 dir1/secretfile
[efg@fedori ~]$ ls -lt secretfile
-rw-------. 3 efg efg 0 2011-05-04 22:40 secretfile
```

and it does :D

Now If you change the secretfile at home ,this will be reflected at dir1 and dir2.
For example , I'll append some data

```
[efg@fedori ~]$ echo "This is some random data for the file"  >> secretfile
[efg@fedori ~]$ ls -lt secretfile
-rw-------. 3 efg efg 38 2011-05-04 22:42 secretfile
```

Okay,now go and check dir1 and dir2 ,

```
[efg@fedori ~]$ cat dir2/secretfile
This is some random data for the file

[efg@fedori ~]$ cat dir1/secretfile
This is some random data for the file
```

Since all the links has single copy of the file,whatever you do via one link will be reflect via remaining lin

ks too.
Similarly , if you change from dir2 it gets reflected at dir1 and home and so on.

Say if you delete the file from dir1 ,the count will decrease by 1 and count will be 2.

```
[efg@fedori ~]$ rm -rf dir1/secretfile
[efg@fedori ~]$ ls -lt secretfile
-rw-------. 2 efg efg 38 2011-05-04 22:42 secretfile
```

true.
and If you delete it from dir2 too,then count will become 1.

```
[efg@fedori ~]$ rm -rf dir2/secretfile
[efg@fedori ~]$ ls -lt secretfile
-rw-------. 1 efg efg 38 2011-05-04 22:42 secretfile
```

true again
If you delete it from home ,then file will be gone forever.

Above linking is known as "hard link". There another linking method known as "soft link".
The major different between the two is , hard link is twice as heavy as soft link :P ..hehe just kidding.

Lets begin with an example for soft link,

```
[efg@fedori ~]$ ln -s /home/efg/secretfile dir1/sym_file
```

we used the same ln command but added an option "-s" to create symbolic link or soft link.

Here is the dir1 contents , notice the first letter 'l' before permissions ,which refers to the file named "sy
m_file" is soft link.

```
[efg@fedori ~]$ ls -l dir1/
lrwxrwxrwx. 1 efg efg 21 2011-05-07 22:52 sym_file -> /home/efg/secretfile
```

And also note that secretfile's link count didn't increase to "2".

```
[efg@fedori ~]$ ls -l secretfile
-rw-------. 1 efg efg 38 2011-05-06 09:27 secretfile
```

Though, you can access the file ,modify it via "sym_file", removing "sym_file" won't remove the original file.
What i mean,is

```
[efg@fedori ~]$rm -rf dir1/sym_file1
```

won't affect secretfile by any means.

But if you delete the original file first ,then "sym_file" is known as "broken link" since it points to a file
which does not exist.

Day-4

Locate or find a file :

I have created a file named "hideme" , but I don't where resides now.How to locate it?
fastest way is to use "locate" command,

```
[efg@fedori ~]$ locate hideme
/home/efg/dir1/hideme
```

another way is to use a 'find' command,

```
[efg@fedori ~]$ find /home/efg -name hideme
/home/efg/dir1/hideme
```

which scans the directory /home/efg for file 'hideme'.

the difference between locate and find is that,locate command maintains database(which was created by updatedb
),it actually check that database for file.
updatedb is command which runs from time to time and update filename and its path on the database which was la
ter used by locate command.

But find command doesn't use any database,It simply the scans the directories everytime.
This is very basic usage of find command,but its very powerful command,next section(advanced find) deals with
it.

which one is that?

If you have multiple binary installed on different path.I mean,for example,assume you have php5 installed on /
usr/bin and php4 installed on /usr/local/bin.

```
[efg@fedori ~]$  whereis php
php: /usr/bin/php /usr/local/bin/php
```

below command uses php to print "Hello world".

```
[efg@fedori ~]$ php -r 'echo "Hello World\n";'
Hello World
```

How to find out which get executed when your simply typed "php".

to figure out that we can use,

```
[efg@fedori ~]$ which php
/usr/local/bin/php
```

thus when you simply use "php" it invokes binary from /usr/local/bin.

How which command works?.Remember in our first section, when we talked about PATH environmental,

```
[efg@fedori ~]$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/efg/bin
```

which command uses above search path.Though we have php on /usr/bin and /usr/local/bin ,since /usr/local/bin c
omes before /usr/bin,
that binary got executed.

Search for a text inside a file or files:

So now we know, how to find a file. But how to find a whether specific text available on a file or not?

For example,secretfile contents is -

```
[efg@fedori ~]$ cat secretfile
This is some random data for the file
```

In order to find whether the string "data" available on that file. we can use grep command.
grep command is used to search for given pattern on a file,If the pattern is found,it will print that whole li
ne.

```
[efg@fedori ~]$ grep 'data' secretfile
This is some random data for the file
```

To understand more about grep , we need to add more data to secretfile.

```
[efg@fedori ~]$  cat >> secretfile
This will go as second line.
Note that we used >> to append data to the file.
If you use > ,it will overwrite any existing contents.
I'm going to end this by pressing ctrl-D now..
^D
```

Lets check the contents again -

```
[efg@fedori ~]$ cat secretfile
This is some random data for the file
This will go as second line.
Note that we used >> to append data to the file.
If you use > ,it will overwrite any existing contents.
I'm going to end this by pressing ctrl-D now..
```

Lets try the same grep command , with option -n which display the line number too.

```
[efg@fedori ~]$ grep -n 'data' secretfile
1:This is some random data for the file
3:Note that we used >> to append data to the file.
```

Suppose we want to print all lines except the line containing the word 'data' -
Use v , invert option like-

```
[efg@fedori ~]$ grep -v 'data' secretfile
This will go as second line.
If you use > ,it will overwrite any existing contents.
I'm going to end this by pressing ctrl-D now..
```

grep has lot of option like printing lines that begin with specific text or ending with specific text etc.
check out 'man grep' or 'info grep' for all available option.(huh...Now i can move on ,If someone messes with
grep command,
i can blame him for not reading man or info pages ;))

Counting the impossible

Here is content again, the task is to count the number of lines it has ,

```
[efg@fedori ~]$ cat secretfile
This is some random data for the file
This will go as second line.
Note that we used >> to append data to the file.
If you use > ,it will overwrite any existing contents.
I'm going to end this by pressing ctrl-D now..
```

5? thats Right.pretty easy one. How about counting number of words ? thats slightly difficult. Then how about
counting number of characters ? Thats even harder. But wait there is one easy way to complete this task.
Just two word command named 'wc'.

```
[efg@fedori ~]$ wc  secretfile
  5  44 218 secretfile
```

secretfile has 5 lines , 44 words and 218 characters. Feel free to validity the result :P

How to sorting file contents:

I'm not going to say anything about this sort command - you should be able to figure out yourself.
You know original content of secretfile ,this is what i get when used sort .

```
[efg@fedori ~]$ sort secretfile
If you use > ,it will overwrite any existing contents.
I'm going to end this by pressing ctrl-D now..
Note that we used >> to append data to the file.
This is some random data for the file
```

Day - 5

How to identify file type without opening?

Do you remember ,often you get some junk mails , claiming they have sent you a 'special' attachment.
You have downloaded it but not sure whether to open it or not,because your afraid of virus (wait...virus? they
 are irrevelant with linux ..lets
try another example).

hmm..you have downloaded a file named "something.pdf" but when you tried to open with pdf reader ,it didn't op
en.
what to do now?the best way is to check to the file type before opening or playing them using "file" command.

Here comes few examples -

```
[efg@fedori C-pdf]$ file Glade-Tutorial.pdf
Glade-Tutorial.pdf: PDF document, version 1.4

[efg@fedori ~]$ file MalgudiDays.mp3
MalgudiDays.mp3: Audio file with ID3 version 2.3.0, contains: MPEG ADTS, layer
III, v1, 128 kbps, 44.1 kHz, St
ereo
```

[OT, did you heard about "Malgudi Days" by R.K.Narayan,My favorite one is "A Tiger for Maigudi" a story from t
iger's point of view. :D]

```
[efg@fedori ~]$ file extlib.ext3
extlib.ext3: Linux rev 1.0 ext3 filesystem data
```

file command don't use extention part to determine the file type. For example, we created dummy file named "e
xt2fs.pdf" like

```
[efg@fedori ~]$ echo "this file contains some random text" > ext2fs.pdf
```

we can't determine the file with 'ls' output -

```
[efg@fedori ~]$ ls -ltr ext2fs.pdf
-rw-rw-r--. 1 efg efg 36 2011-05-08 13:32 ext2fs.pdf
```

Now lets try 'file' -

```
[efg@fedori ~]$ file ext2fs.pdf
ext2fs.pdf: ASCII text
```

As you can see ,file says just a "ASCII text" file not pdf.

Its always better to verify the downloaded file type before running or opening them.

First cut it then paste and finally fold a file:

From our secretfile,if you want cut first column then try
[efg@fedori ~]$ cut -f1 -d' ' secretfile
This
This
Note
If
I'm

option f1 means cut first column and d refers to delimiter. you want to cut second column use -f2 instead of -
f1.

I created two files (1.txt and 2.txt)

[efg@fedori ~]$ cat 1.txt
file1: line1
file1: line2
[efg@fedori ~]$ cat 2.txt
file2: line1
file2: line2

In order to merge the lines of these two files we do paste -

[efg@fedori ~]$ paste 1.txt 2.txt
file1: line1   file2: line1
file1: line2   file2: line2

finally if we use fold, it will wrap each line to specified width,

below will wrap the text to 5 characters per line.

[efg@fedori ~]$ fold -w 5 1.txt
file1
: lin
e1
file1
: lin
e2

FileSystem hierarchy structure

Now its time to understand linux filesystem hierarchy structure.Its not hard, good news is you already learned
 them without realizing it :).
Remember , when you first logged in ,it checked /etc/passwd file and finds the entry "/home/efg" ?

Normally Configuration files go to /etc directory. and all Users home directory will be /home/user-name except
 admin's home which is /root.

And also remember when we searched for commands location with $PATH which gave something like-
/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/efg/bin

/usr/local/bin and /usr/bin will have user commands/appliations.

/usr/sbin will contain root user commands.

Here is more -
/tmp - any temporary files created and deleted in memory RAM. If you create some file at /tmp directory and re
boot the machine.These files won't be
there when the machine comes up.
/var - will contain log files
/proc - Real time process information stored by kernel.
/dev - will contains device files
/boot - this directory content is used for booting process.
/lib - contains system library files and kernel modules
/opt - third party softwares can be installed here.
/root - again, system administrators home directory. (try ls /root and check out your admin's secret contents
:D ,if possible.)
/mnt - where external devices can be mounted.
All above directories are subdirectory of / - the root directory

Split one big file into multiple small files :

```
[efg@fedori ~]$ ls -l secretfile
-rw-------. 1 efg efg 218 2011-05-08 12:27 secretfile
```

the file has 218 bytes as size. If we want to split this file to 2 files of 109 bytes each with command like
-

```
[efg@fedori ~]$ split -b 109 secretfile
[efg@fedori ~]$ ls -l
```

If you do a 'ls' you will find two new files with perfix 'x' with 109 bytes each
-

```
-rw-------.  1 efg efg           218 2011-05-08 12:27 secretfile
-rw-rw-r--.  1 efg efg           109 2011-05-16 10:04 xab
-rw-rw-r--.  1 efg efg           109 2011-05-16 10:04 xaa
```

Lets check out their contents

```
[efg@fedori ~]$ cat xaa
This is some random data for the file
This will go as second line.
Note that we used >> to append data to the
```

second file content is -

```
[efg@fedori ~]$ cat xab
 file.
If you use > ,it will overwrite any existing contents.
I'm going to end this by pressing ctrl-D now..
```

as you can see secretfile content is now placed in two files.This split command will be very useful when you w
ant to
transfer one big file from one machine to another.

Anybody has questions ? No ..oh..yeah someone there thinking how to merge these contents again? something like
 'merge' command?
hmm..good question..but wait..I just checked 'man merge',it won't fit into this.
the simple way is to use cat and append files , like

```
[efg@fedori ]$ cat xa* > newsecretfile
[efg@fedori ]$ cat newsecretfile
This is some random data for the file
This will go as second line.
Note that we used >> to append data to the file.
If you use > ,it will overwrite any existing contents.
I'm going to end this by pressing ctrl-D now..

that's it.
```

How to check for free disk fedori?

On windows , probably you mighti do somethin like 'right-click' on D: or C: and choose properties and there yo
u can see the size.
From linux command line is a matter of just two letters - yeah 'df'

```
[efg@fedori ~]$ df  -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             140G  123G  9.6G  93% /
/dev/sda5             194M  100M   85M  55% /boot
```

As you can see ,I have used upto 93% of root partition and boot partition occupies 55%.
I have passed -h option to df command to make it more readable.

In order to check for free fedori on RAM, do run "free" command.

Remember this !!!!

No one (no not even linus :D) would remember all options of all commands.So try and make some guess for examp
le
-h may be human readable format.
-a may be run all
-r may be recursive
-i may be ...what ? hmm..install or interactive
-s may be size
-b may be block
-d may be directory
-l may be links
-v may be verbose or version

So If first-time interviewer asks you about 'how do you do that with command X' - Tell him bluntly, "First che
ck man commandname and then use it.No one can remember all options of commands."

How long the machine is up and running?

uptime gives you the answer.

```
[efg@fedori ~]$ uptime
 20:30:50 up 21 min,  3 users,  load average: 0.24, 0.07, 0.09
```

First column denotes current time , next field shows its running for "21 minutes" (yeah...Just came back from
office and start the machine)

and the 3 users currently logged in. and approximate load average of the system in past 1,5,15 minutes.

Who is it?

I like this song from MJ :D..no not that,I was thinking about "how to find out 3 users reported by uptime outp
ut."
and the command is ..who..whoo..who..who (I like that song too :))

```
[efg@fedori ~]$ who
efg     tty1         2011-05-17 20:10
efg     pts/0        2011-05-17 20:11 (:0.0)
efg     pts/1        2011-05-17 20:15 (:0.0)
```

It was me ,opened up different terminals.Not a intruders.
there is one more way for doing the same - use 'w'

it combines the output for uptime and also lets us know what each user currently doing

```
[efg@fedori ~]$ w
 20:41:18 up 31 min,  3 users,  load average: 0.08, 0.16, 0.14
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
efg     tty1     -                20:10   30:42   0.04s  0.01s /bin/sh
/usr/bin/startx
efg     pts/0    :0.0             20:11    3.00s  2.90s  7.81s gnome-terminal
efg     pts/1    :0.0             20:15    0.00s  0.04s  0.00s w
```

What's your hostname and version

Okay,So far you have done all this on which machine and what OS or kernel version ?
Yeah..Its was pretty late, we didn't think about it till now , right? Anyway its better late than never.

To find hostname

```
[efg@fedori ~]$ hostname
fedori
```

to find OS and kernel details run :

```
[efg@fedori ~]$ uname -a
Linux fedori 2.6.32.21-168.fc12.x86_64 #1 SMP Wed Sep 15 16:12:07 UTC 2010 x86_64 x86_64 x86_64 GNU/Linux
```

I'll let you to decode that uname output. Check man uname for any help.
So far we talked about file related stuffs mainly,next we will focus on process.