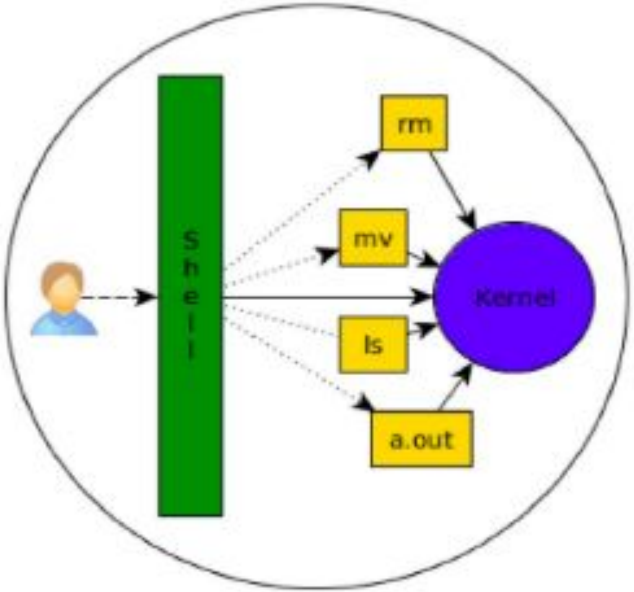
Linux Shell Command

What is a Shell ?

- Shell is a command language interpreter that executes commands read from the standard input device (keyboard) or any file and passes them to the kernel.
- Shell is not a part of system kernel but uses the kernel to execute programs.
- cat /etc/shells command will give the various shells in our system.



To install new shell

```
sudo apt-get install csh/ksh
```

Want to check which type of Shell you are using?

```
echo $SHELL
```

How to check what a particular command do?

```
man <command>
e.g. man ls
```

How to check the list of files in a folder?

ls -a <folder>	(all)Lists all the files (including .*files)
ls -l <folder>	(long) Long listing (type, date, size,owner, permissions)
ls -t <folder>	(time) Lists the most recent files first
ls -S <folder>	(size) Lists the biggest files first
ls -r <folder>	(reverse) Reverses the sort order
ls -ltr <folder>	(options can be combined) Long listing, most recent files at the end

File name pattern substitutions

ls *txt

The shell first replaces *txt by all the file and directory names ending by txt (including .txt), except those starting with ., and then executes the ls command line. This will work when you are inside the directory.

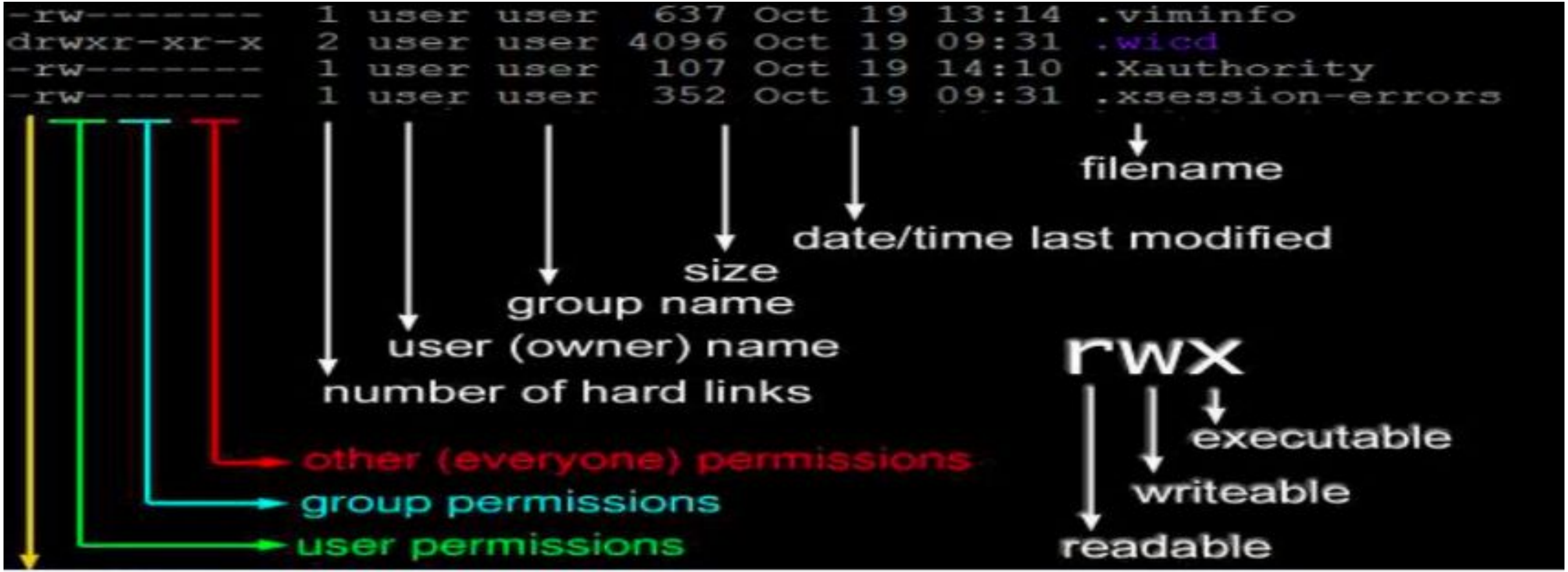
cat ?.log

Displays all the files which names start by 1 character and end by .log

Working with files

- touch
- cp
- mv
- cat
- echo

File, Users, Groups and Permissions



```
-rw-------  1 user user  637 Oct 19 13:14 .viminfo
drwxr-xr-x  2 user user 4096 Oct 19 09:31 .wicd
-rw-------  1 user user  107 Oct 19 14:10 .Xauthority
-rw-------  1 user user  352 Oct 19 09:31 .xsession-errors
```

Working with files

Use ls - l to check file access rights

3 types of access rights
Read access (r)
Write access (w)
Execute rights (x)

3 types of access levels
User (u): for the owner of the file
Group (g): each file also has a “group” attribute, corresponding to a given list of users
Others (o): for all other users

Changing permissions (chmod)

chmod <permissions> <files>

	User	Group	Others
Read(4)			
Write(2)			
Execute(1)			

R,W,E for User
W,E only for Group
R,E for Others

What is the permission?

Changing permissions (Alternate Way)

Symbolic format.

- **chmod go+r**: add read permissions to group and others.
- **chmod u-w**: remove write permissions from user.
- **chmod ax**: (a: all) remove execute permission from all.
- **chmod -R a+rX linux/**: Makes linux and everything in it available to everyone!
  - R: apply changes recursively
  - X: x, but only for directories and files already executable

ps ux	Lists all the processes belonging to the current user
kill <pids>	Sends an abort signal to the given processes. Lets processes save data and exit by themselves
pstree	display a tree of processes

~/.bashrc file

Shell script read each time a bash shell is started

You can use this file to define your default environment variables (PATH, EDITOR...).

A greeting message.

Linux Shell (bash) scripting

Different editors in Linux

Vi
Vim
Nano
Gedit
Sublime

Nano Editor

nano filename

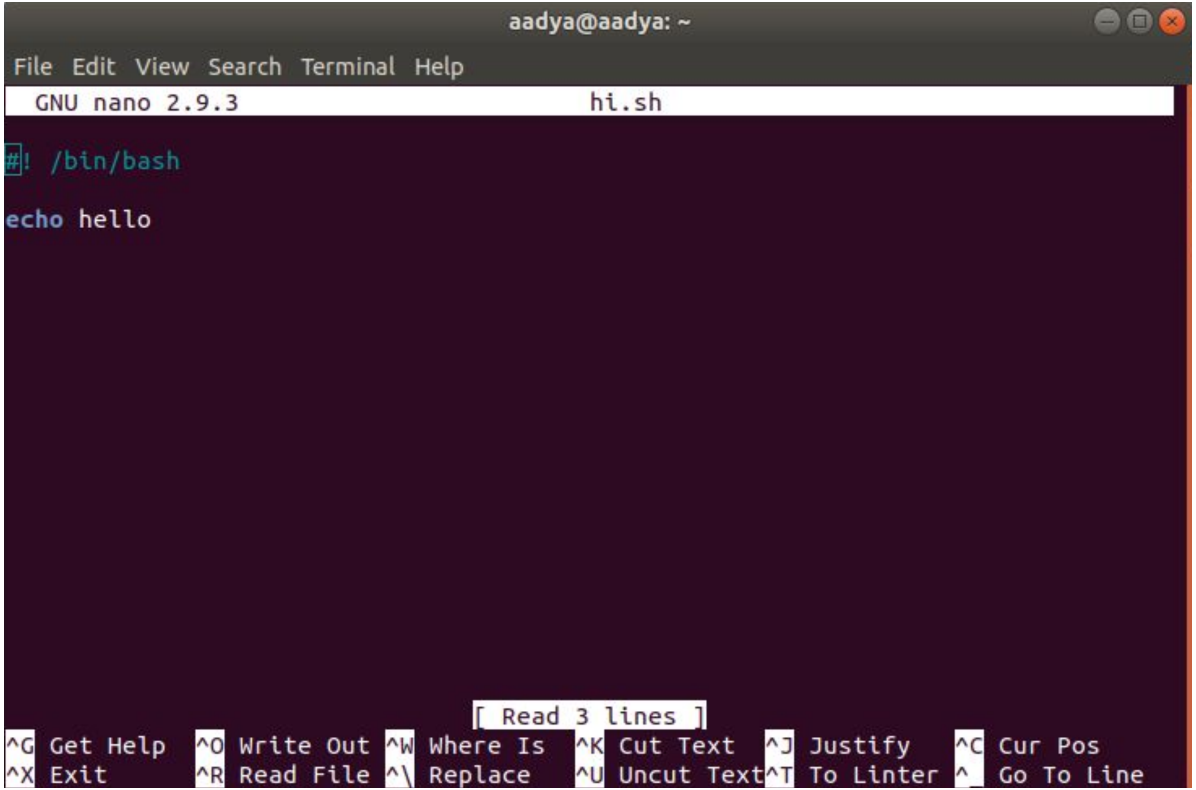
to open or create a file

Ctrl + o

Write out or save

Ctrl + x

to exit from editor



```
  GNU nano 2.9.3                              hi.sh

#! /bin/bash

echo hello




                              [ Read 3 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^T To Linter   ^_ Go To Line
```

What is difference between command and shell scripting?

command is (computing) a directive to a computer program acting as an interpreter of some kind, in order to perform a specific task while script is (computing) a file containing a list of user commands, allowing them to be invoked once to execute in sequence.

Comparison

For integer comparison we have the following

-eq	: equal to
-ne	: not equal to
-lt	: less than
-gt	: greater than
-le	: less than or equal to
-ge	: greater than or equal to

For string comparison we have

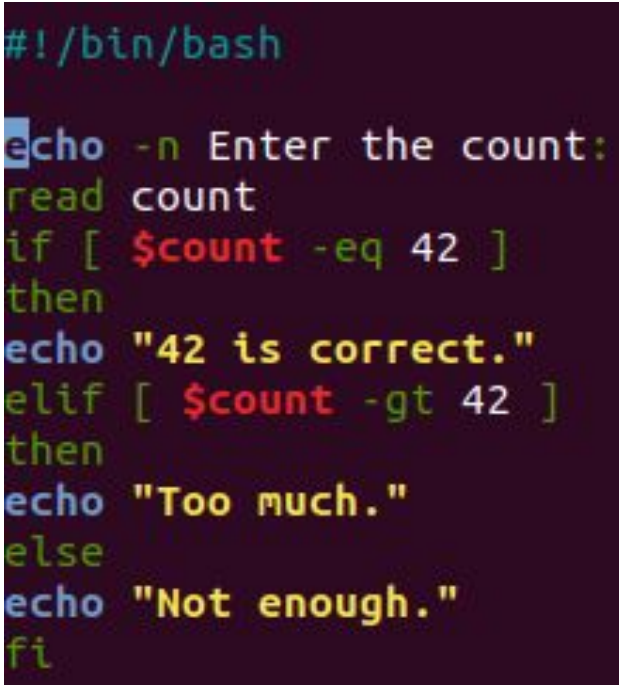
=	: equal to
~=	: not equal to

For logical operators

-a	: AND
-o	: OR

You can nest a new if inside an else with elif.

```
#!/bin/bash
echo -n Enter the count:
read count
if [ $count -eq 42 ]
then
echo "42 is correct."
elif [ $count -gt 42 ]
then
echo "Too much."
else
echo "Not enough."
fi
```



```
#!/bin/bash

echo -n Enter the count:
read count
if [ $count -eq 42 ]
then
echo "42 is correct."
elif [ $count -gt 42 ]
then
echo "Too much."
else
echo "Not enough."
fi
```

f3.sh

Execute the file

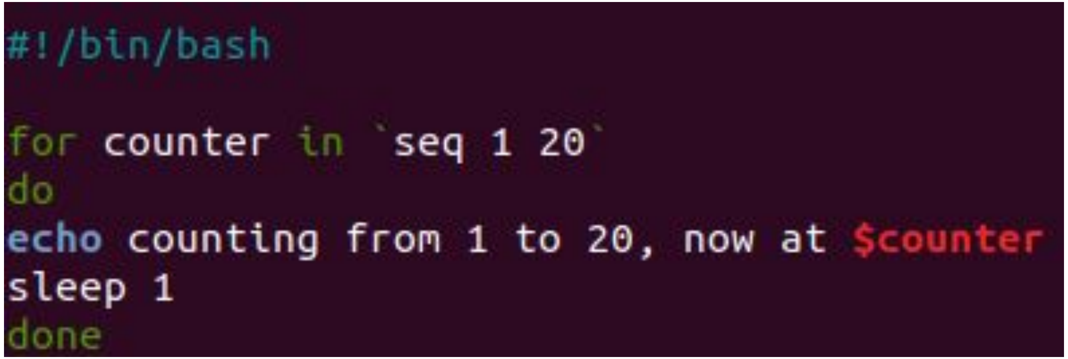
sh f3.sh

./f3.sh (if the file has execution permission)

For loop

```
#!/bin/bash

for counter in `seq 1 20`
do
echo counting from 1 to 20, now at $counter
sleep 1
done
```



```
#!/bin/bash

for counter in `seq 1 20`
do
echo counting from 1 to 20, now at $counter
sleep 1
done
```

Execute the file

```
sh f4.sh
./f4.sh (if the file has execution permission)
```

f4.sh