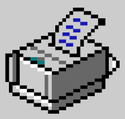
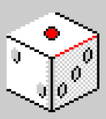
Shell Scripting

How to be a command line hacker :)

START

A presentation by
Aryan and Uchit





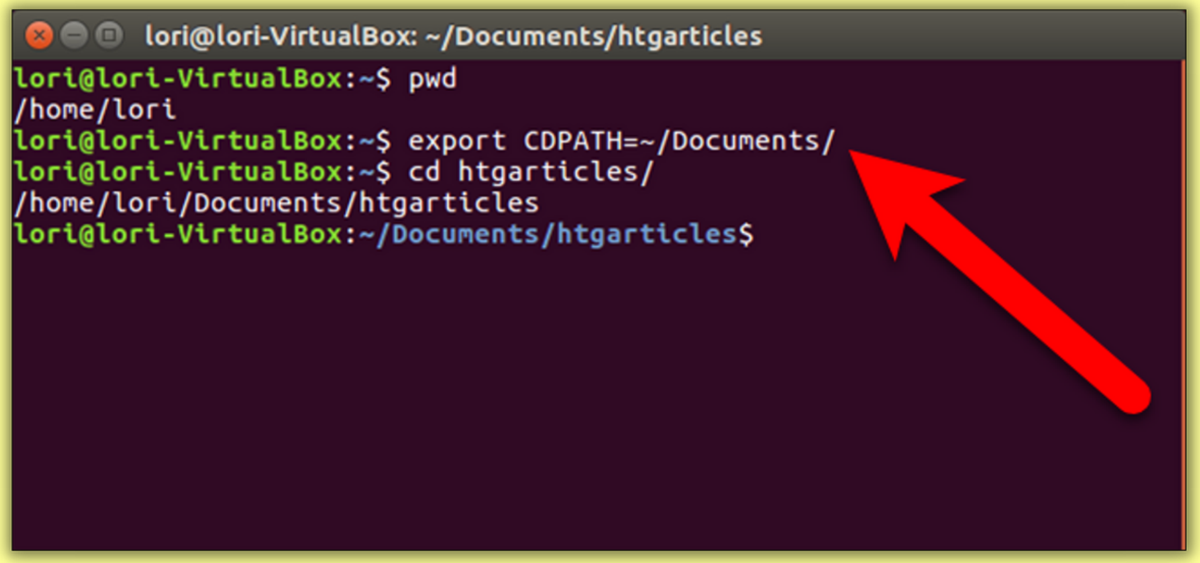
11:11PM

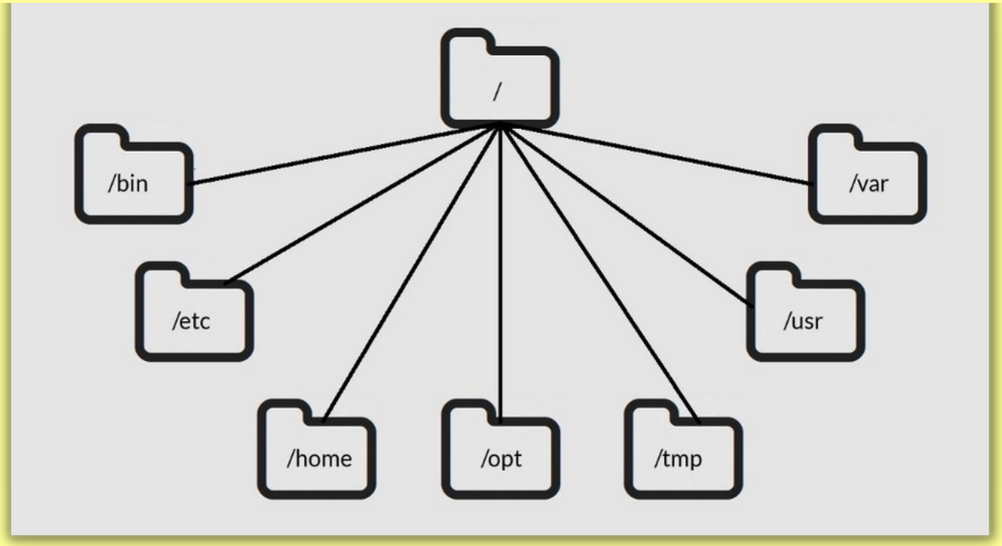
Shell

A shell is a program that takes in keyboard commands as input and passes them to the operating system to carry out or facilitate a function or a task.

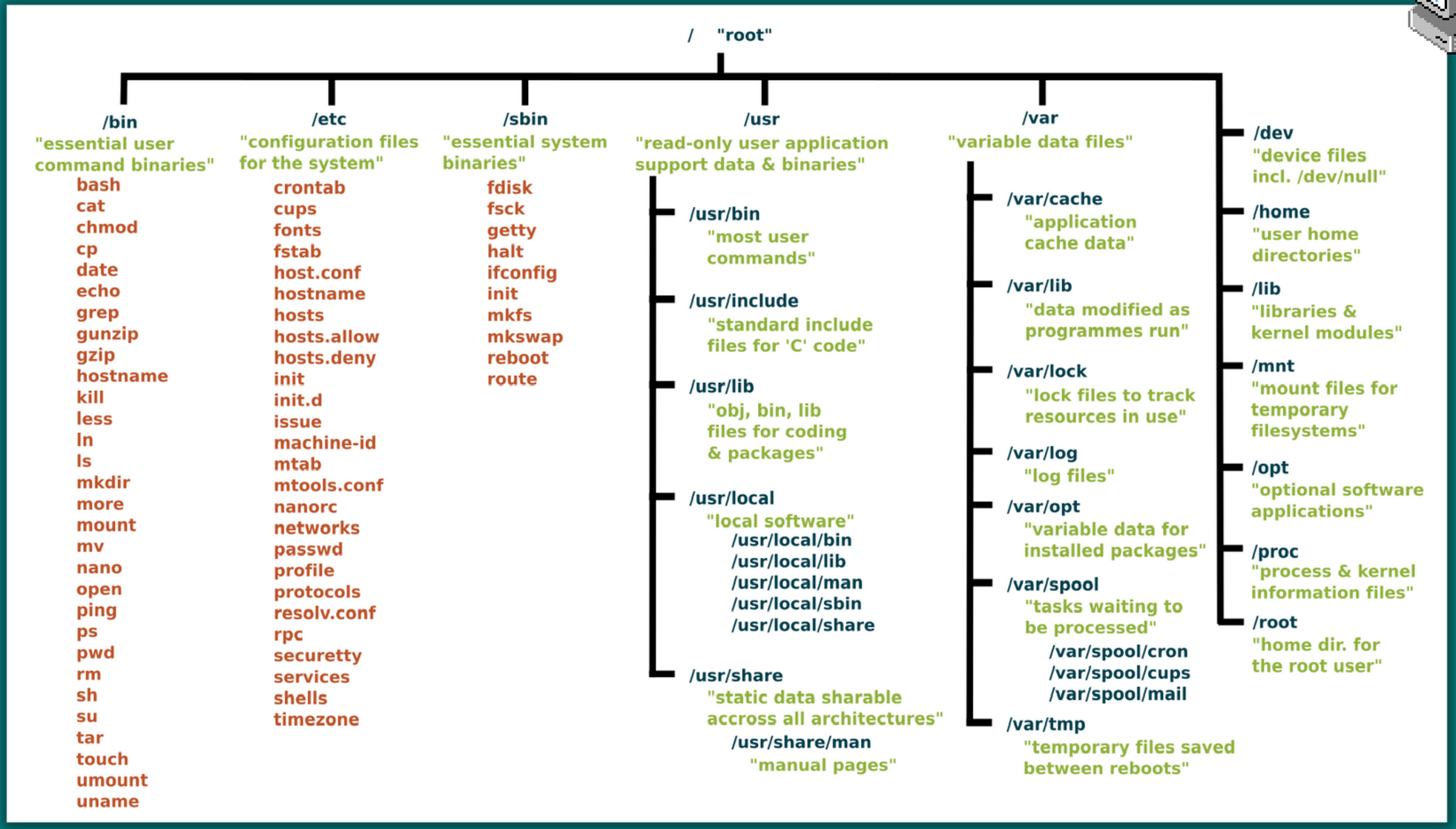
Directories in Linux

Unix uses a hierarchical structure for organizing files and directories. This structure is often referred to as a directory tree. The tree has a single root node, the slash character (/), and all other directories are contained below it.





File System in Linux



Let's learn some commands :-)

A Linux command is a program or utility that runs on the command line. A command line is an interface that accepts lines of text and processes them into instructions for your computer.

man

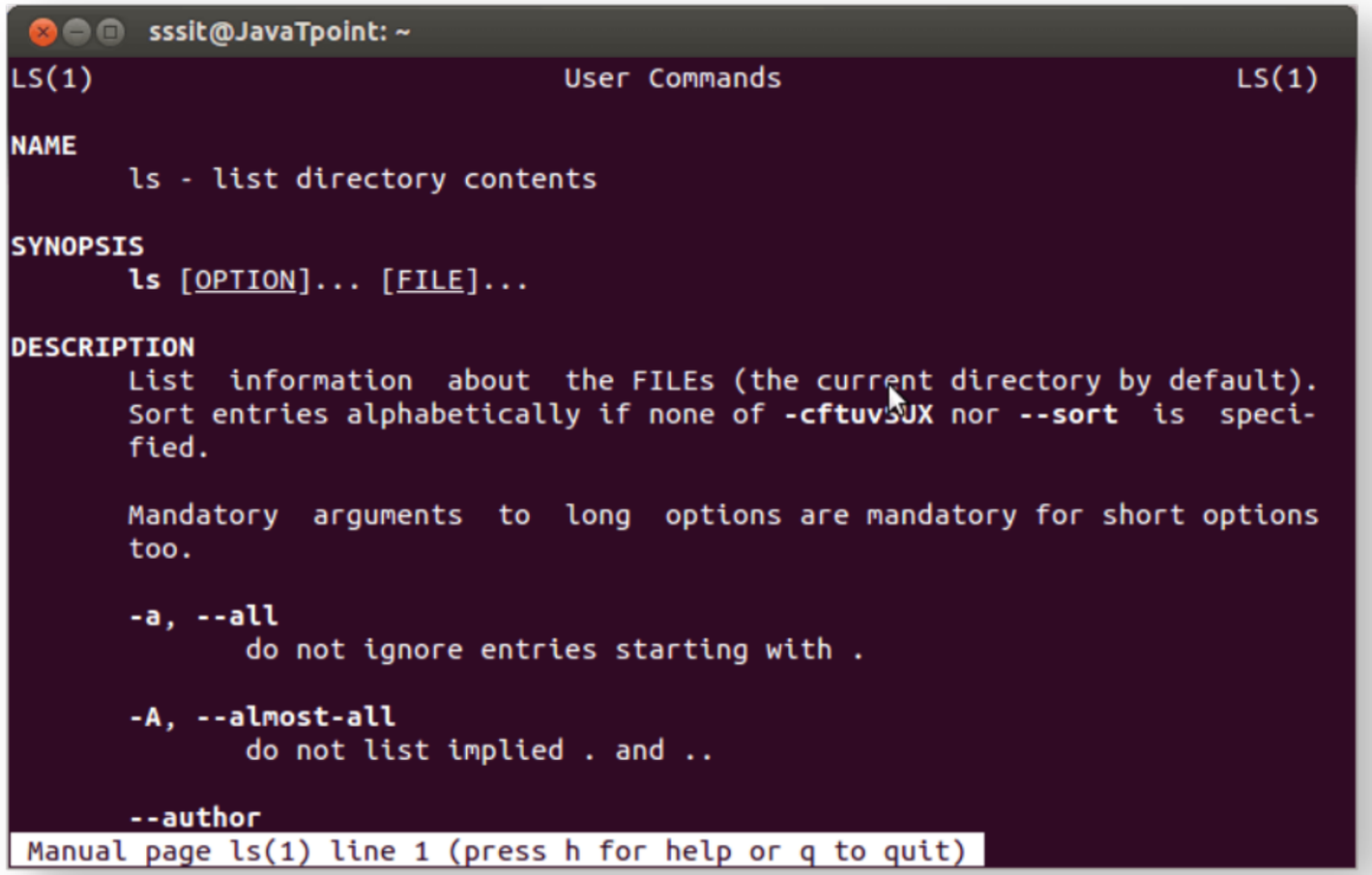
Stands for: Manual

Function: An interface to the on-line reference manuals

Syntax: man -C file -d -D --warnings=warnings -R encoding -L locale -m system,... -M path -S list -e extension -i|-I --regex|--wildcard --names-only -a -u --no-subpages -P pager -r prompt -7 -E encoding

Example: man -a intro

man ls



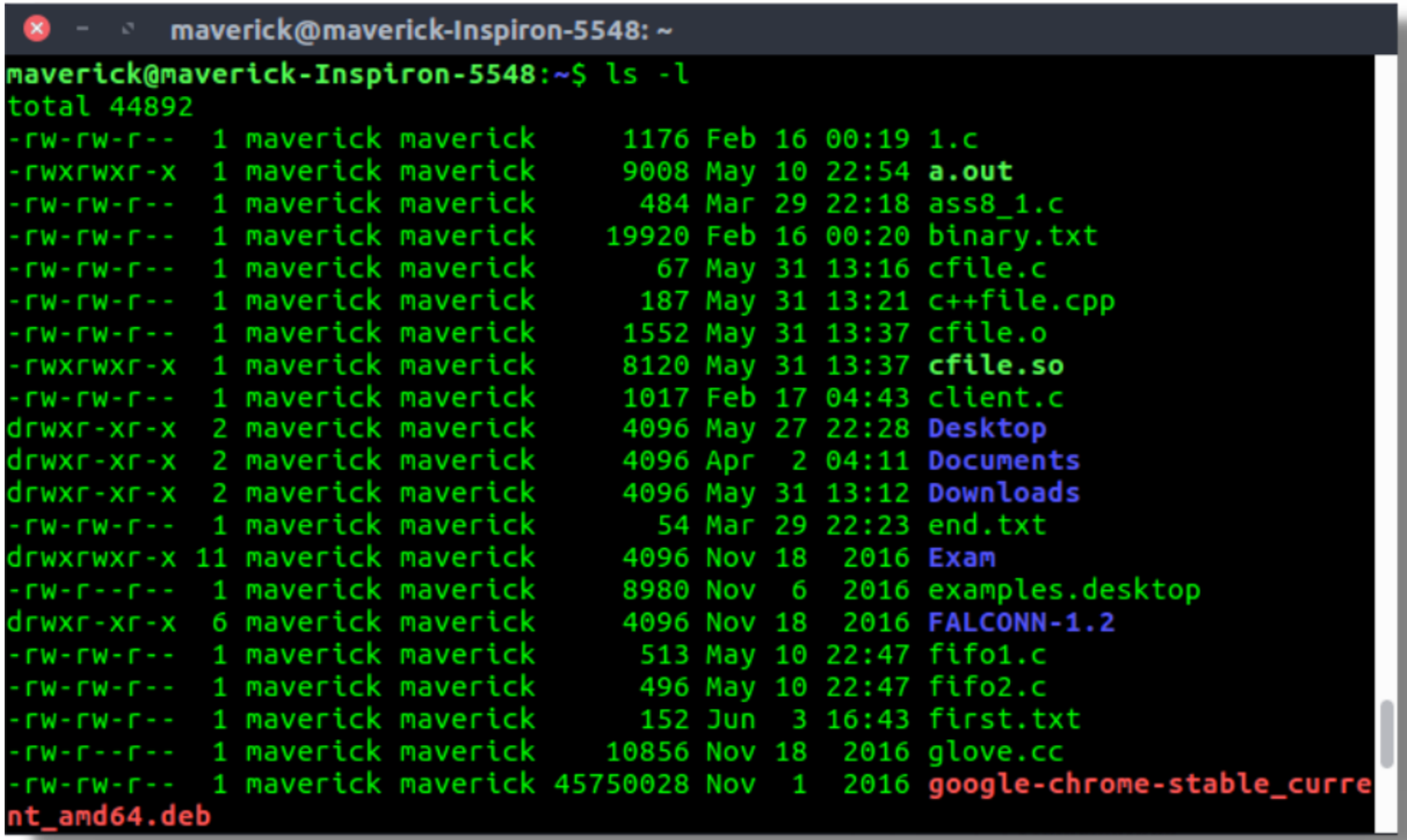
ls

Stands for: List

Function: List information about the files (the current directory by default)

Syntax: ls -1AacCdeFilnpLRrSsTtuvwxXhk FILE...

ls -l



Stands for: List

Function: List information about the files (the current directory by default)

Syntax: ls -1AacCdeFilnpLRrSsTtuvwxXhk FILE...

ls -l explained

Field	Meaning
`-rw-r--r--`	Access rights to the file. The first character indicates the type of file. Among the different types, a leading dash means a regular file, while a `d` indicates a directory. The next three characters are the access rights for the file's owner, the next three are for members of the file's group, and the final three are for everyone else. Chapter 9 discusses the full meaning of this in more detail.
`1`	File's number of hard links. See "Symbolic Links" on page 21 and "Hard Links" on page 22.
`root`	The username of the file's owner.
`root`	The name of the group that owns the file.
`32059`	Size of the file in bytes.
`2017-04-03 11:05`	Date and time of the file's last modification.
`oo-cd-cover.odf`	Name of the file.

cd

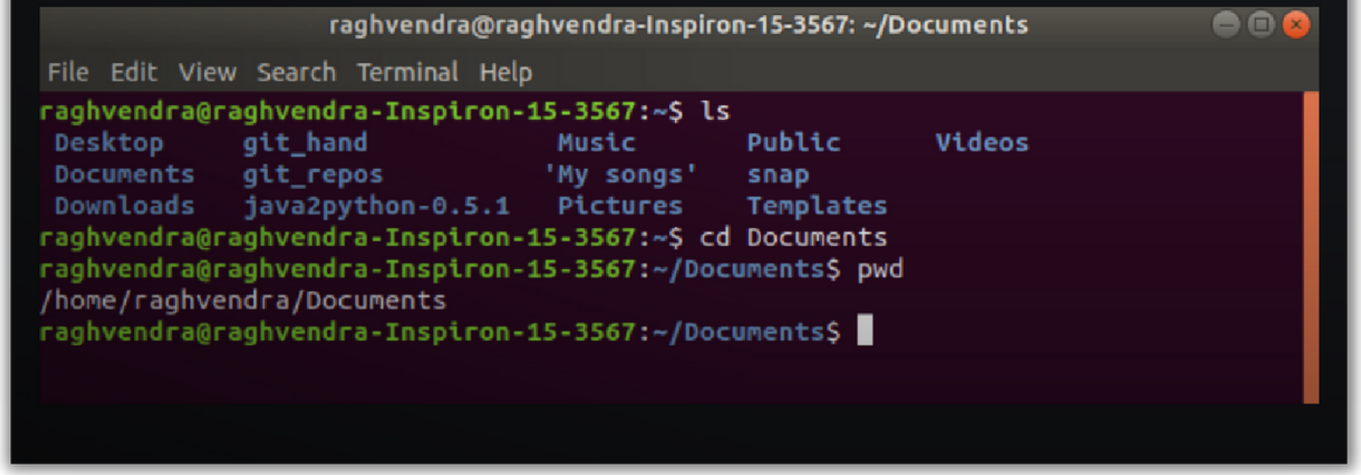
Stands for: chdir

Function: a command-line shell command used to change the current working directory in various operating systems.

Syntax: cd

To move inside a subdirectory : to move inside a subdirectory in linux we use

```
$ cd [directory_name]
```



Some other common commands

These are all quintessentials that ease our workflow helping us to analyse things better.

- **pwd** - current working dir
- **whoami** - current user(name)
- **data** - current data and time
- **history** - all shell history
- **cp** - copy
- **clear** - clear the shell interface
- **man** - manuals
- **exit** - close the session
- **who** - logged in history
- **mkdir** - make dir
- **cat** - concate files
- **mv** - move

Wildcards in commands

On Linux, Bash wildcards which allows you to define patterns to match against filenames or strings.
This process is known as globbing.

Character class	Meaning
[:alnum:]	Matches any alphanumeric character
[:alpha:]	Matches any alphabetic character
[:digit:]	Matches any numeral
[:lower:]	Matches any lowercase letter
[:upper:]	Matches any uppercase letter

Wildcard	Meaning
*	Matches any characters
?	Matches any single character
[*characters*]	Matches any character that is a member of the set *characters*
[!*characters*]	Matches any character that is not a member of the set *characters*
[[:*class*:]]	Matches any character that is a member of the specified *class*

diff

diff stands for difference. This command is used to display the differences in the files by comparing the files line by line. Unlike its fellow members, cmp and comm, it tells us which lines in one file have is to be changed to make the two files identical.

a : add

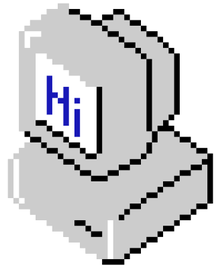
c : change

d : delete

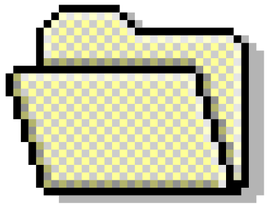
```
[root@server ~]# diff test1 test2
2c2
< asdfgh
---
> uiopas
[root@server ~]# cat test1
qwerty
asdfgh
zxcvbn
[root@server ~]# cat test2
qwerty
uiopas
zxcvbn
[root@server ~]#
```

Iteration in Bash

Topics Covered



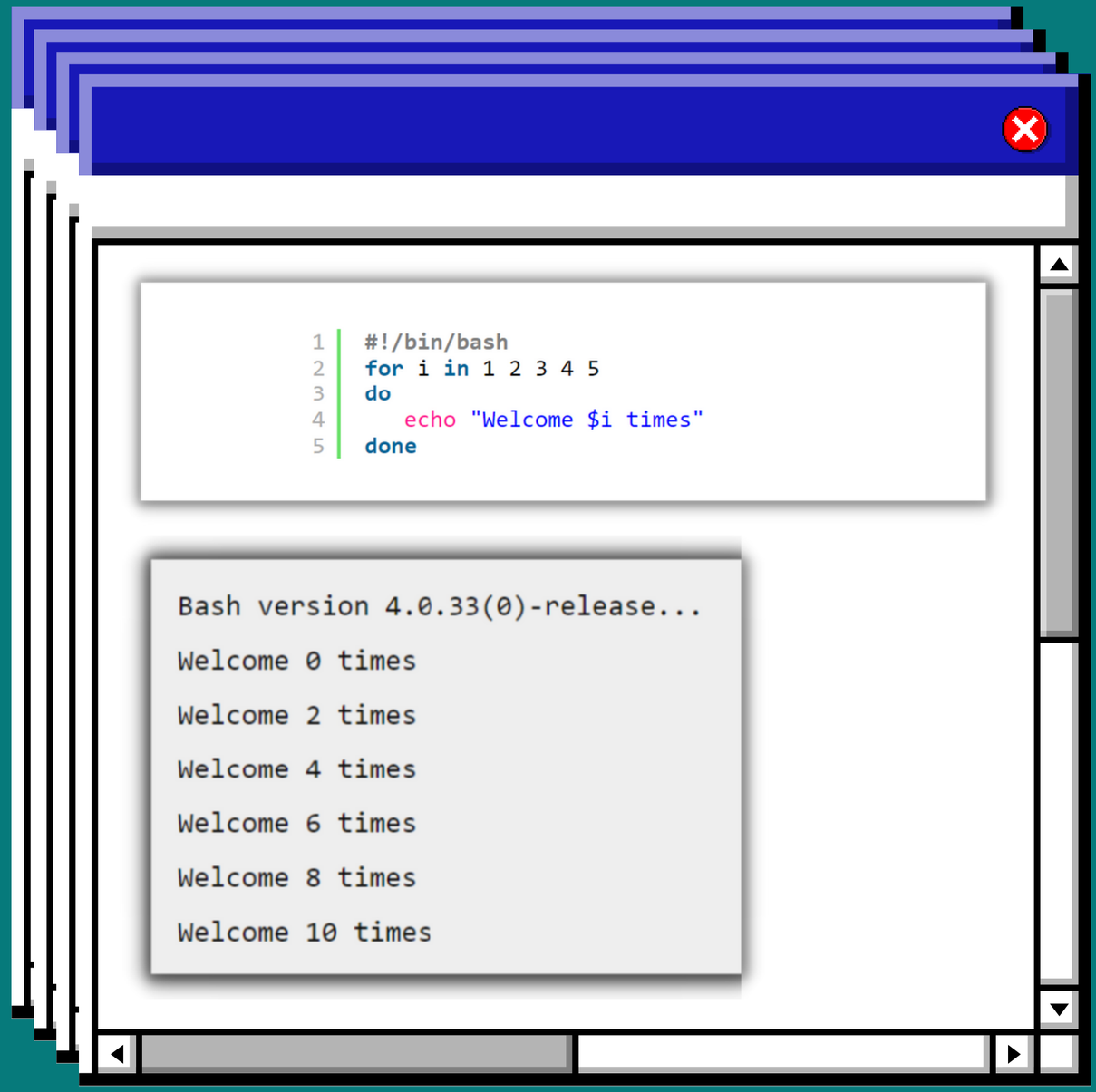
for loop()



while loop()

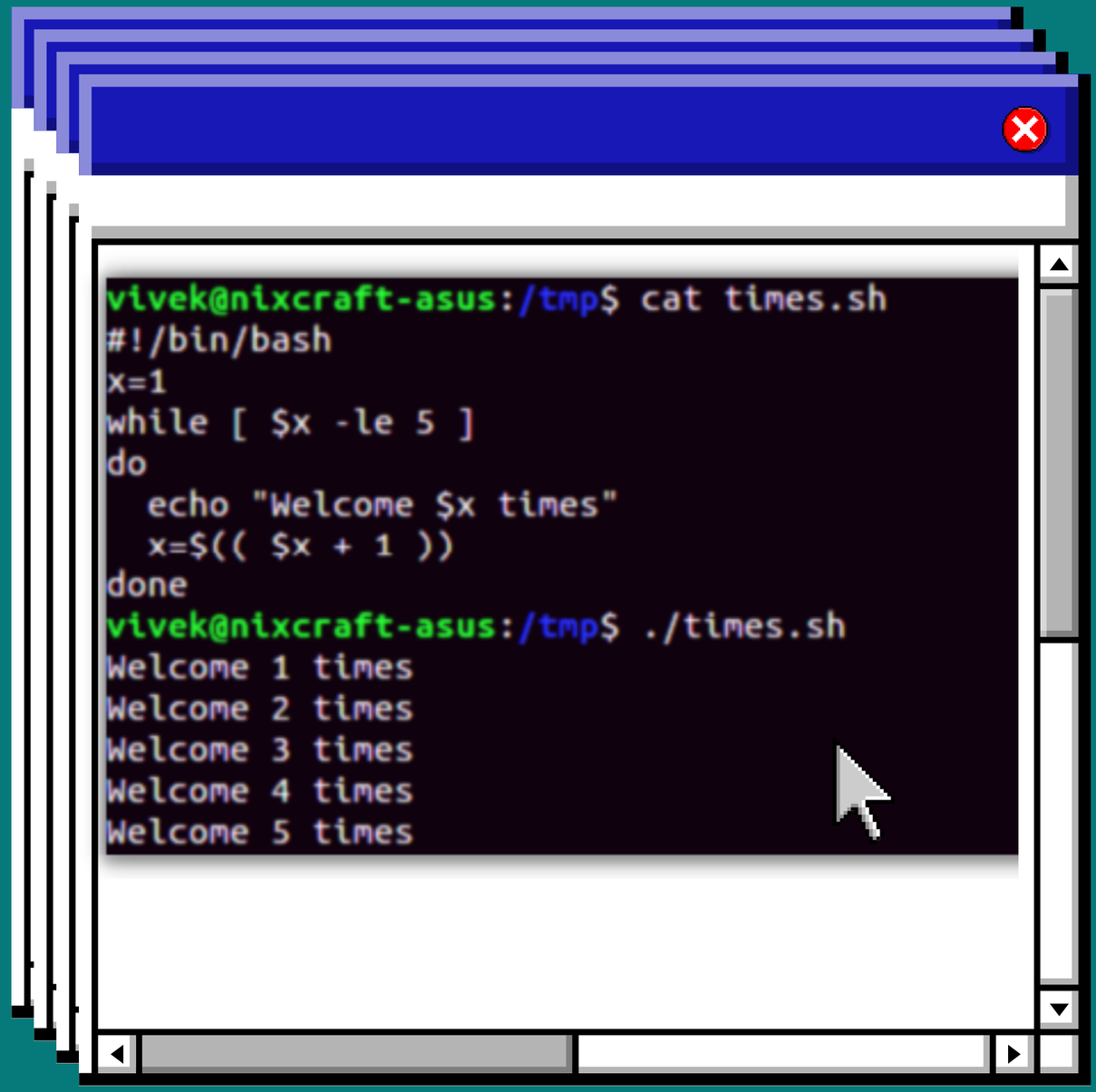
for loop()

A bash for loop is a bash programming language statement which allows code to be repeatedly executed. A for loop is classified as an iteration statement i.e. it is the repetition of a process within a bash script.



while loop()

The bash while loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition. For example, run echo command 5 times or read text file line by line or evaluate the options passed on the command line for a script.



Select Loop

```
select department in CS IT ECE EE
do
  case $department in

    CS)
      echo "I am from CS department."
      ;;

 IT)
      echo "I am from IT department."
    ;;

     ECE)
      echo "I am from ECE department."
    ;;

 EE)
      echo "I am from EE department."
    ;;

    none)
      break
    ;;

    *) echo "Invalid selection"
    ;;
  esac
done
```

Conditional Statements

The statements that perform specific functions based on certain conditions are called conditional statements. In bash scripting, we have several conditional statements like **IF, IF-ELSE, IF-ELSE-IF**, etc. Every statement has its way of working and according to the need, we use them.

if-else statements

If the condition is true, the IF statement block gets executed but if the condition is false nothing is returned or executed. If we want the program to perform certain action after the IF statement condition is false, we use the ELSE statement after the If statement.

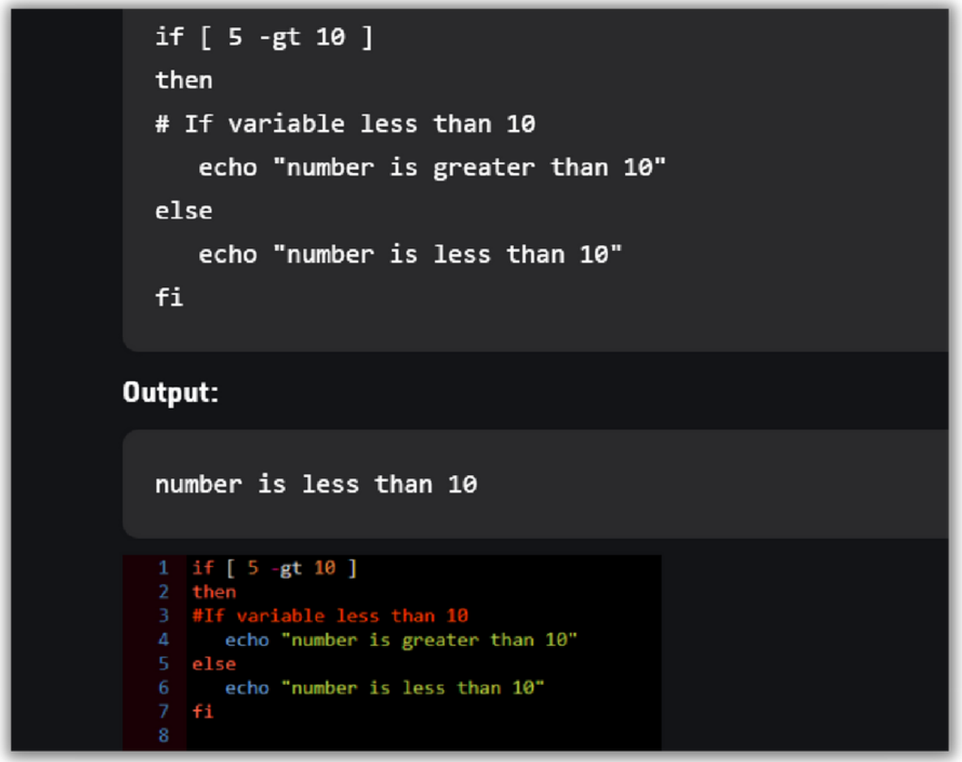
```
if [condition ]
then
    If statement
else
    ELSE statement
fi
```

- If the condition is true: the IF statement will get executed.
- If the condition is false: the ELSE statement will get executed.

```
if [ 5 -gt 10 ]
then
# If variable less than 10
   echo "number is greater than 10"
else
   echo "number is less than 10"
fi
```

Output:

```
number is less than 10
```



if-elif-else statements

```
if [ condition1 ]
then
        statement1
 elif [ condition2 ]
then
        statement2
 elif [condition3 ]
then
        statement3
else
       statement_n
fi
```

ELIF is the keyword used for the ELSE IF statement in bash scripting. If in a loop if more than two conditions exist which can not be solved only by using IF-ELSE statement then ELIF is used. Multiple ELIF conditions can be defined inside one if-else loop.

Permission Attributes

File systems use **permissions** and **attributes** to regulate the level of interaction that system processes can have with files and directories.

Attribute	Files	Directories
r	Allows a file to be opened and read.	Allows a directory's contents to be listed if the execute attribute is also set.
w	Allows a file to be written to or truncated; however, this attribute does not allow files to be renamed or deleted. The ability to delete or rename files is determined by directory attributes.	Allows files within a directory to be created, deleted, and renamed if the execute attribute is also set.
x	Allows a file to be treated as a program and executed. Program files written in scripting languages must also be set as readable to be executed.	Allows a directory to be entered, e.g., `cd directory`.

-rwx------	A regular file that is readable, writable, and executable by the file's owner. No one else has any access.
-rw-------	A regular file that is readable and writable by the file's owner. No one else has any access.
-rw-r--r--	A regular file that is readable and writable by the file's owner. Members of the file's owner group may read the file. The file is world-readable.
-rwxr-xr-x	A regular file that is readable, writable, and executable by the file's owner. The file may be read and executed by everybody else.
-rw-rw----	A regular file that is readable and writable by the file's owner and members of the file's group owner only.
lrwxrwxrwx	A symbolic link. All symbolic links have "dummy" permissions. The real permissions are kept with the actual file pointed to by the symbolic link.
drwxrwx---	A directory. The owner and the members of the owner group may enter the directory and create, rename, and remove files within the directory.
drwxr-x---	A directory. The owner may enter the directory and create, rename, and delete files within the directory. Members of the owner group may enter the directory but cannot create, delete, or rename files.

Permission Attributes

File systems use **permissions** and **attributes** to regulate the level of interaction that system processes can have with files and directories.

Notation	Meaning
`u+x`	Add execute permission for the owner.
`u-x`	Remove execute permission from the owner.
`+x`	Add execute permission for the owner, group, and world. This is equivalent to `a+x`.
`o-rw`	Remove the read and write permissions from anyone besides the owner and group owner.
`go=rw`	Set the group owner and anyone besides the owner to have read and write permissions. If either the group owner or the world previously had execute permission, it is removed.
`u+x,go=rx`	Add execute permission for the owner and set the permissions for the group and others to read and execute. Multiple specifications may be separated by commas.

Evaluation operation and syntaxes

To compliment our conditional statements, bash provides us with a set of bitwise operations.

-lt - less than

-mt - more than

-eq - equality

-ne - not equal

-a / && - logical and

-o / || - logical or

To evaluate a mathematical expr :

$((< var > <opertaion> <var>))

Eg: c=$((${a}+${b}))

OR

`expr <valid operation>`

File test operators

File test operators let you test various aspects of a file like if a file exists, is readable/writable, etc. Using file test operators can help to prevent errors in your Bash scripts

-d : check if given path/object is dir

-f : check if given path/object is file

-e :check if given path/object exist

*__True__ on success and __False__ on failure

```
device0="/dev/sda2"    # /   (root directory)
if [ -b "$device0" ]
then
  echo "$device0 is a block device."
fi

# /dev/sda2 is a block device.



device1="/dev/ttyS1"   # PCMCIA modem card.
if [ -c "$device1" ]
then
  echo "$device1 is a character device."
fi

# /dev/ttyS1 is a character device.
```

Piping

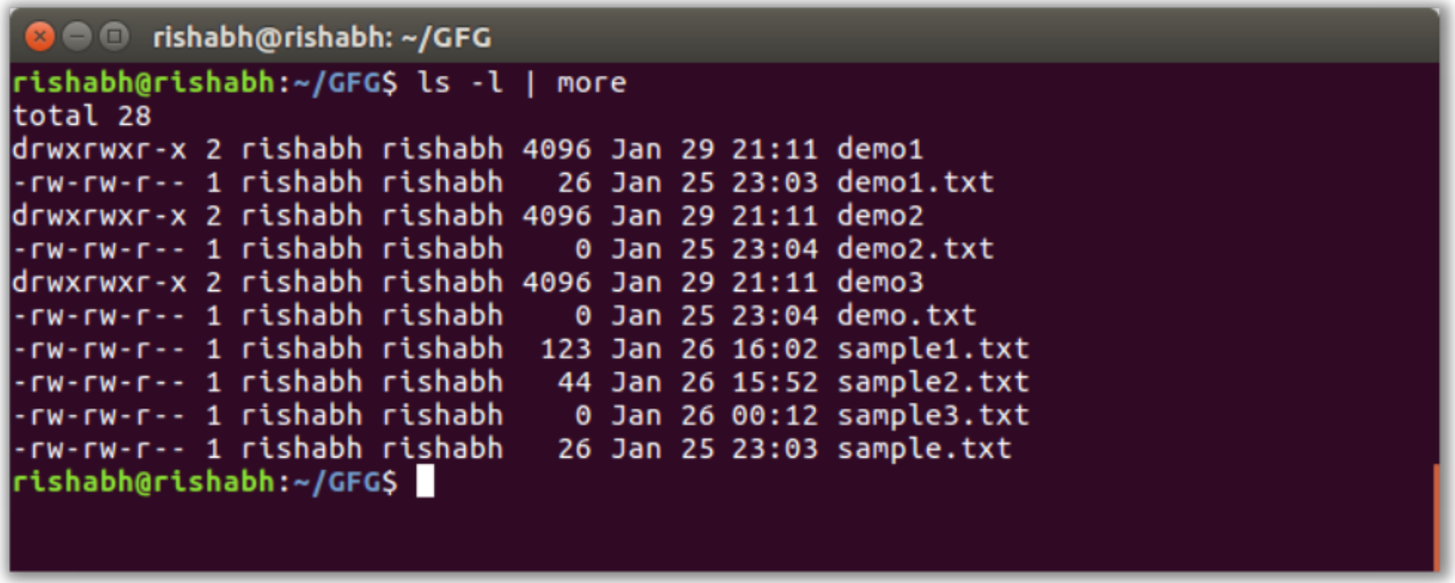
A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing.

Example :

Listing all files and directories and give it as input to more command.

```
$ ls -l | more
```

Output :



Standard Redirection

Types:

- Standard input redirection
- Standard output redirection
- Standard error redirection

Standard input redirection

To redirect standard input from another file instead from the screen, we use the '<' redirection operator followed by the name of the file/source.

Standard output redirection

I/O redirection allows us to redefine where standard output goes. To redirect standard output to another file instead of the screen, we use the '>' redirection operator followed by the name of the file.

Standard error redirection

To redirect standard error to another file instead of the screen, we use the '>' redirection operator followed by the name of the file.


Thank you!
Kavya for your wonderful guidance throughout the course :)


PESU