# LINUX

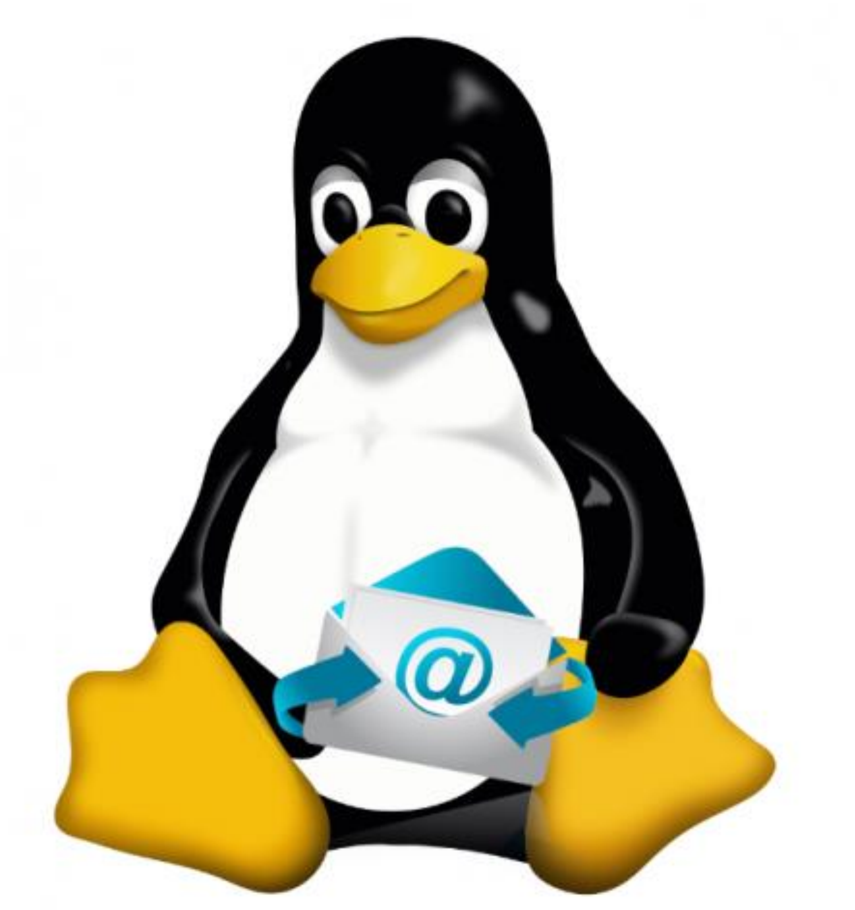


Image source: https://www.itdev.co.uk/blog/get-your-patch-merged-journey-linux-kernel-%E2%80%93-part-3

## Overview

This session will give you a good working knowledge of Linux, from both a graphical and command line perspective, allowing you to easily navigate through any of the major Linux distributions. We will discuss basic command line operations along with shell scripting.

## Agenda

We will get to know about Linux, the most popular open-source operating system. The basic architecture of Linux will be discussed, along with the command line interface and the basic Linux commands. We next discuss shell scripting where we will learn to read and write scripts.

## Content

- Operating System
- Linux Operating System
- Linux Kernel
- Basic Linux Commands
- Shell Scripting

# Operating System

- Acts as an interface between the computer hardware components and the user.
- Performs file management, process management, and memory management, handling input and output devices.
- Helps us to communicate with the computer without us having the knowledge of the computer's language.

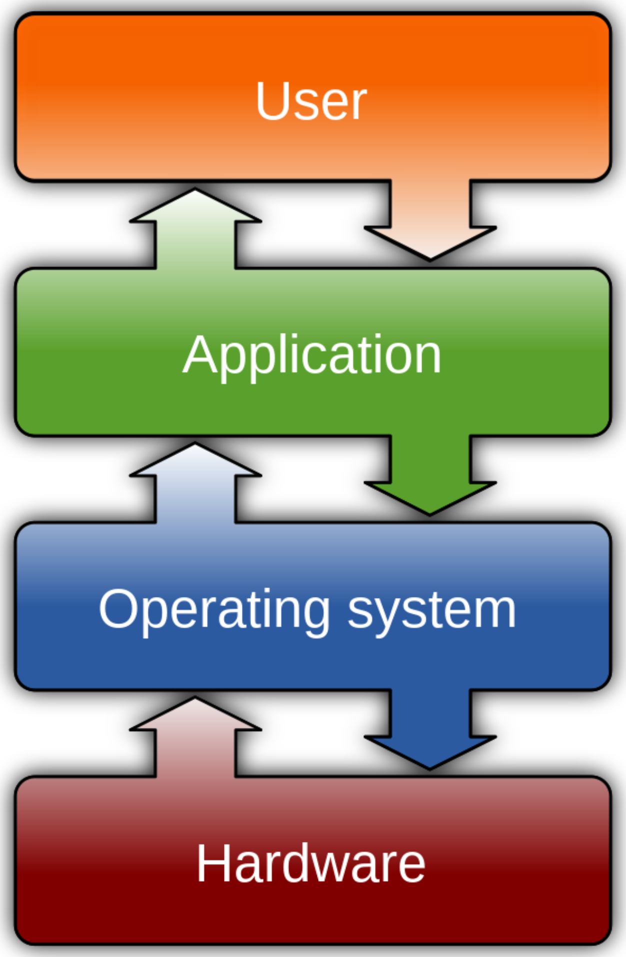


Image Source: Wikipedia

# Operating System Management Tasks

Process Management

Memory Management

Device Management

File Management

Storage Management

User-Interface Management

Security Management

Job-accounting Management

# Types of Operating System

- Batch Operating System
- Time Sharing Operating System
- Distributed Operating System
- Real Time Operating System
- Embedded Operating System

## Examples of Operating System

The three most used operating systems for personal computers:

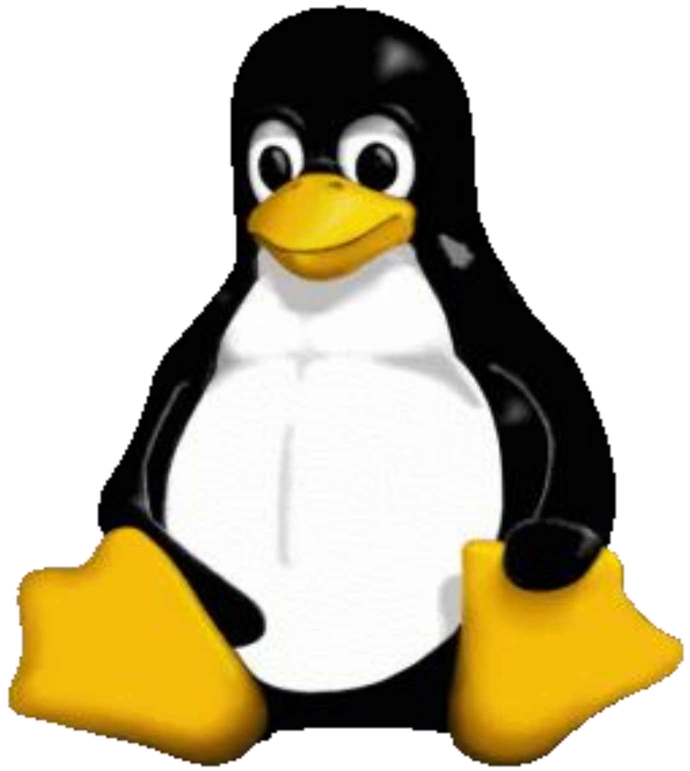
- Microsoft Windows: Created by Microsoft in mid 1980s. It comes pre-loaded on most new PCs, making it the most popular operating system.

- macOS: Created by Apple, it come pre-loaded on all the Macintosh computers.

- Linux: It belongs to the family of open-source operating systems and is different from proprietary software like Windows.

## What is Linux?

- The Linux OS is free and open-source and behaves like a Unix-like operating system.
- Linux was conceived and created in 1991 by Linus Torvalds for his i386-based PC
- Linux was adopted as the kernel for the GNU operating system, which was created as a free replacement for UNIX.

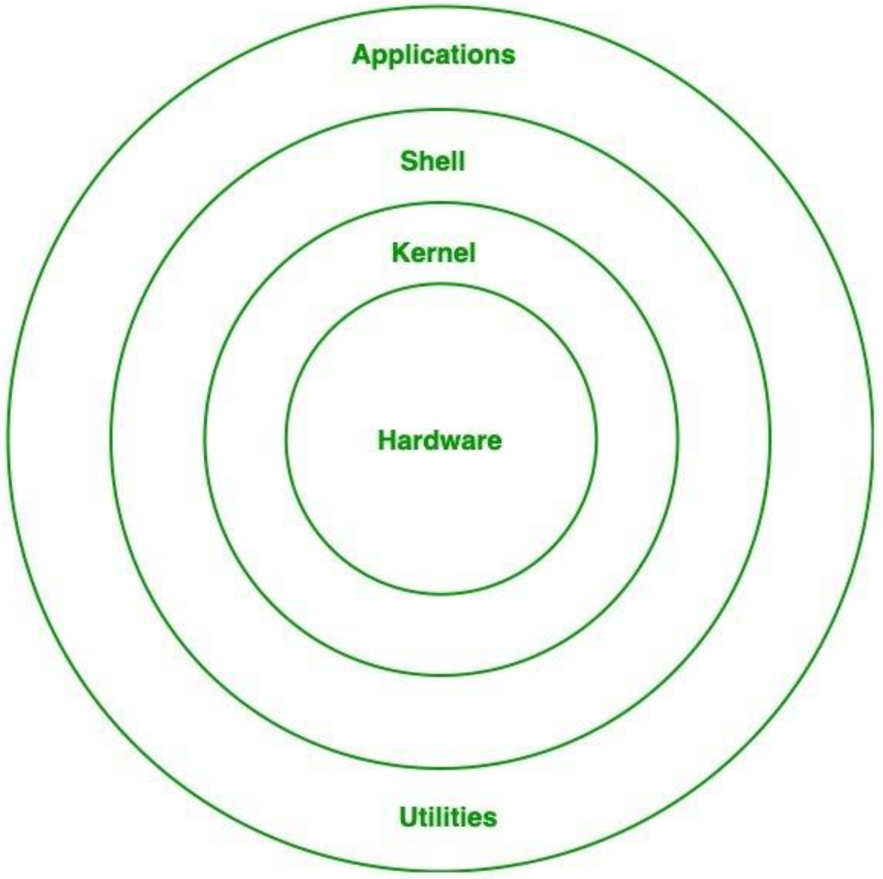

https://en.wikipedia.org/wiki/Linux_kernel

## Linux Features

- UNIX-like kernel.
- Open source.
- Preemptive multitasking
- Portable
- Security

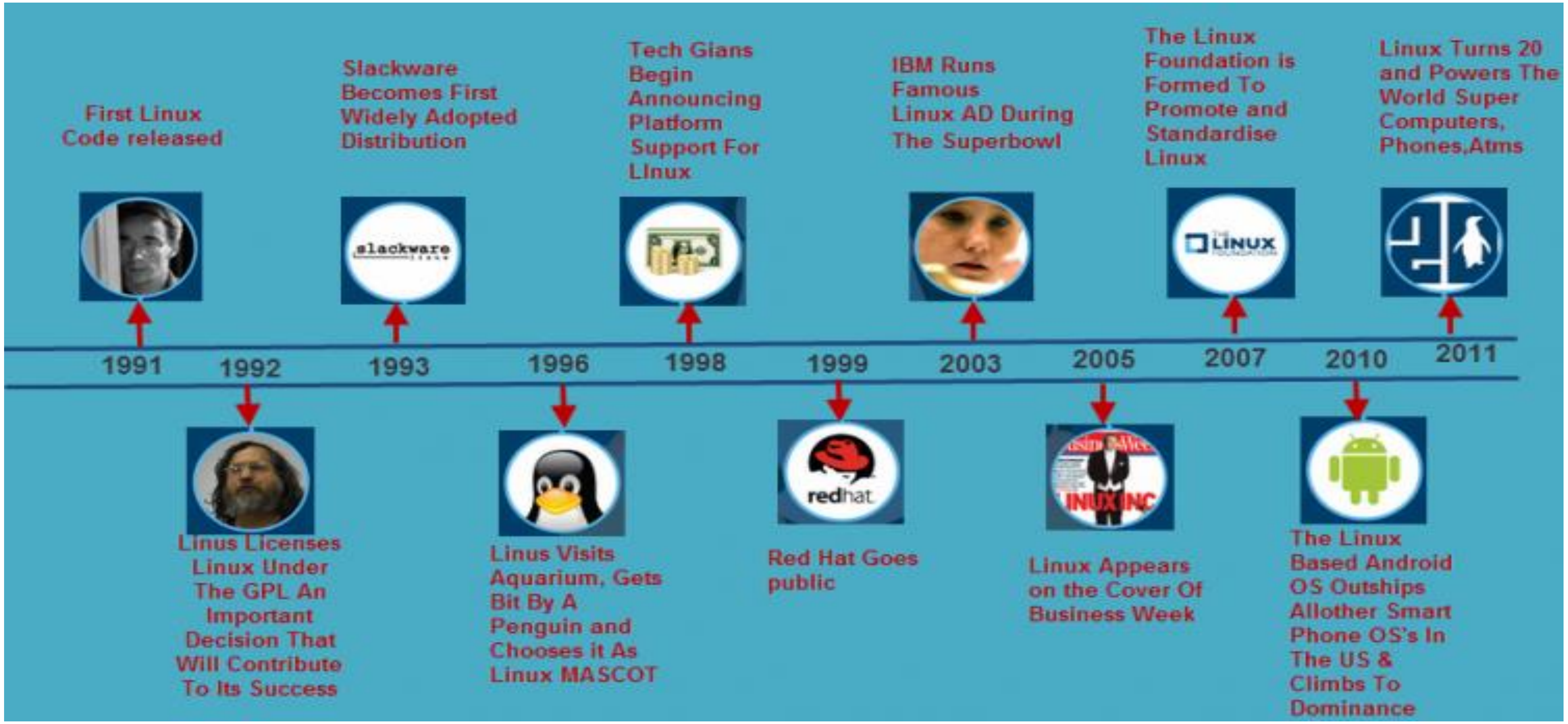
# Linux Architecture

- Hardware Layer
- Kernel
- Shell
- Utilities



https://media.geeksforgeeks.org/wp-content/uploads/20200105215737/Untitled-Diagram-215-1.jpg

# History of Linux



1991 – First Linux Code released

1992 – Linus Licenses Linux Under The GPL An Important Decision That Will Contribute To Its Success

1993 – Slackware Becomes First Widely Adopted Distribution

1996 – Linus Visits Aquarium, Gets Bit By A Penguin and Chooses it As Linux MASCOT

1998 – Tech Gians Begin Announcing Platform Support For Linux

1999 – Red Hat Goes public

2003 – IBM Runs Famous Linux AD During The Superbowl

2005 – Linux Appears on the Cover Of Business Week

2007 – The Linux Foundation is Formed To Promote and Standardise Linux

2010 – The Linux Based Android OS Outships Allother Smart Phone OS's In The US & Climbs To Dominance

2011 – Linux Turns 20 and Powers The World Super Computers, Phones,Atms

https://www.elprocus.com/wp-content/uploads/2014/10/History-OF-Linux-750x400.png

## Linux Distributions

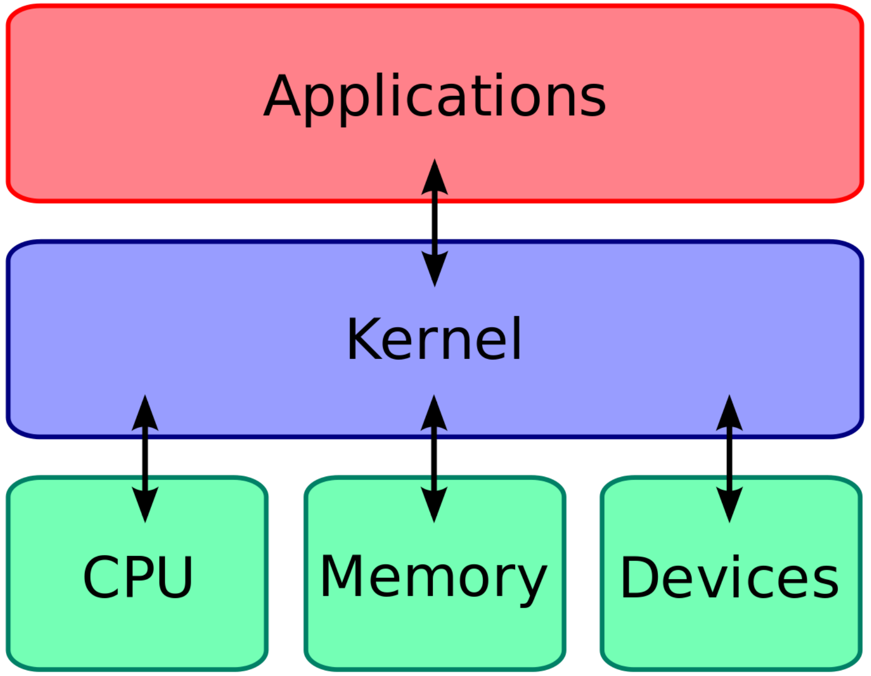
- Ubuntu
- Debian
- Fedora
- Kali Linux
- Arch Linux
- Raspberry Pi OS

# What is a Linux Kernel?

- Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

The kernel has the following jobs:

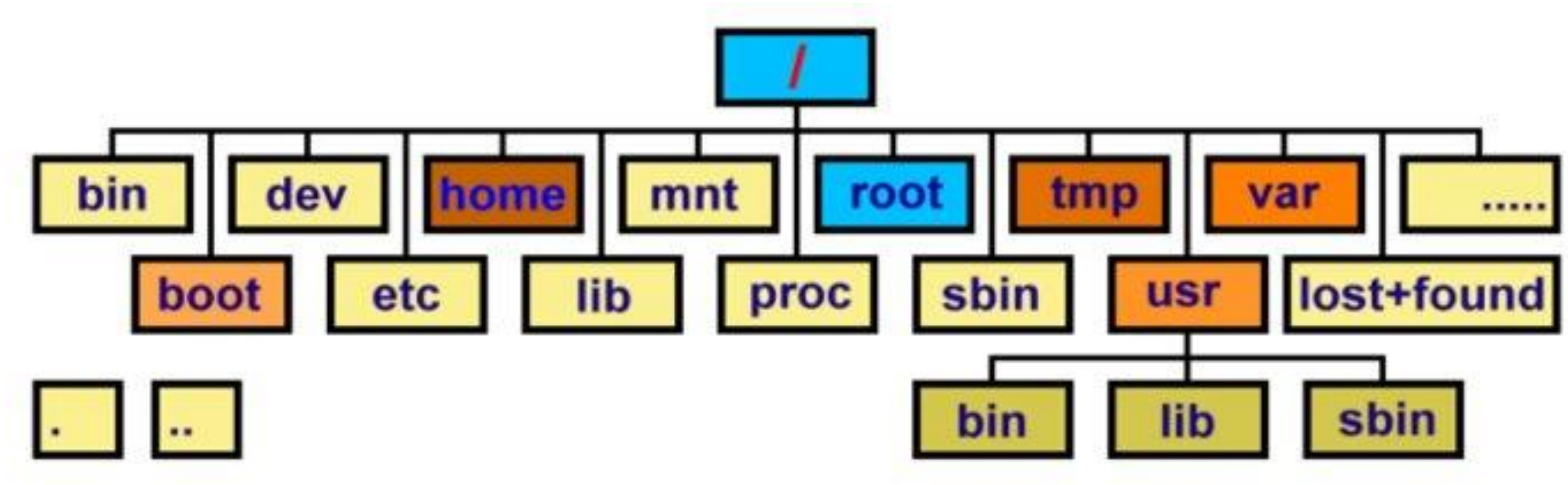
- Memory management
- Process management
- Device drivers
- System calls and security



https://blog.digilentinc.com/wp-content/uploads/2015/05/1280px-Kernel_Layout.svg_.png

## Linux File System

The kernel maintains a single hierarchical directory structure to organize all files in the system. In Microsoft Windows, each disk device has its own directory hierarchy.



| Directory | Content |
|---|---|
| /bin | Common programs, shared by the system, the system administrator and the users. |
| /boot | The startup files and the kernel, vmlinuz. In some recent distributions also grub data. Grub is the GRand Unified Boot loader. |
| /dev | Contains references to all the CPU peripheral hardware, which are represented as files with special properties. |
| /etc | Most important system configuration files are in /etc, this directory contains data similar to those in the Control Panel in Windows |
| /home | Home directories of the common users. |
| /initrd | (on some distributions) Information for booting. Do not remove! |
| /lib | Library files, includes files for all kinds of programs needed by the system and the users. |
| /lost+found | Every partition has a lost+found in its upper directory. Files that were saved during failures are here. |

| | |
|---|---|
| /misc | For miscellaneous purposes. |
| /mnt | Standard mount point for external file systems, e.g., a CD-ROM or a digital camera. |
| /net | Standard mount point for entire remote file systems |
| /opt | Typically contains extra and third-party software. |
| /proc | A virtual file system containing information about system resources. More information about the meaning of the files in proc is obtained by entering the command man proc in a terminal window. The file proc.txt discusses the virtual file system in detail. |
| /root | The administrative user's home directory. Mind the difference between /, the root directory and /root, the home directory of the root user. |
| /sbin | Programs for use by the system and the system administrator. |
| /tmp | Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work!c |
| /usr | Programs, libraries, documentation etc. for all user-related programs. |

## ls command

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

Lists the files in the current directory, in alphanumeric order, except files starting with the “.” character

## cd and pwd commands

cd <dir>
Changes the current directory to <dir>.

cd -
Gets back to the previous current directory.

pwd
Displays the current directory ("working directory").

## cp command

cp <source_file> <target_file>
Copies the source file to the target.

cp file1 file2 file3 ... dir
Copies the files to the target directory (last argument).

cp -i (interactive)
Asks for user confirmation if the target file already exists

cp -r <source_dir> <target_dir> (recursive)
Copies the whole directory.

## mv and rm commands

mv <old_name> <new_name> (move)
Renames the given file or directory.

mv -i (interactive)
If the new file already exits, asks for user confirm

rm file1 file2 file3 ... (remove)
Removes the given files.

rm -i (interactive)
Always ask for user confirm.

rm -r dir1 dir2 dir3 (recursive)
Removes the given directories with all their contents.

## Creating and removing directories

mkdir dir1 dir2 dir3 ... (make dir)
Creates directories with the given names.

rmdir dir1 dir2 dir3 ... (remove dir)
Removes the given directories

Safe: only works when directories and empty.
Alternative: rm -r (doesn't need empty directories).

# Working with Files

| **command** | **function** |
|---|---|
| **touch** filename1 fileName2 fileName3 …. | easy way to create a file |
| **cp** file1 file2 or **cp** fileName filePath | copy content of a file to other file |
| **mv** fileName1 fileName2 or **mv** fileName1 filePath/dirPath | move file content or file to other dir |
| **cp -r** dirName1 dirName2 | recursive copy |
| **cp** file1 file2 file3 file4 file5 /dirName | cp multiple files to directory |
| **head** | display first 10 lines |
| **tail** | display last 10 lines |
| **cat** | display content |
| **cat > fileName** | create file with concatenated content (ctrl+d for EOL) |

## File access rights

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

## Access right examples

-rw-r--r--

Readable and writable for file owner, only readable for others

-rw-r-----

Readable and writable for file owner, only readable for users belonging to the file group.

drwx------

Directory only accessible by its owner

-------r-x

File executable by others but neither by your friends nor by yourself. Nice protections for a trap...

## SHELL SCRIPTING

## What is a Shell ?

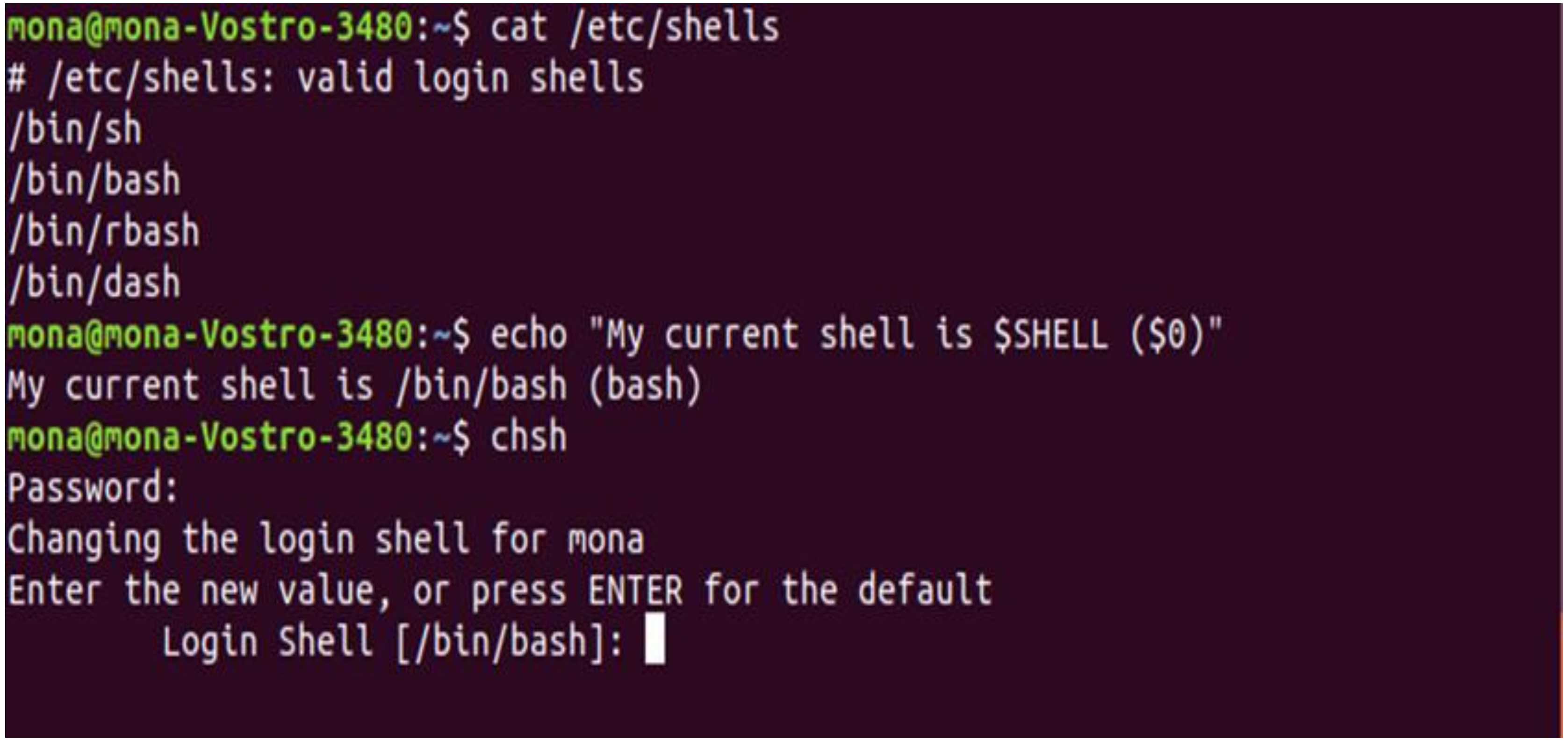
- A shell is an environment that takes commands typed by the user and calls the operating system to run those commands.
- It is a program that acts as the interface between the user and the Linux system, allowing the user to enter commands for the operating system to execute.
- Shell accepts the instruction or commands in English and translates them into computers native binary language.

## Kind of Shells

- Bourne Shell
- C Shell
- Korn Shell
- Bash Shell
- Tcsh Shell

# Changing Your Default Shell

- Command to find all available shells in your system: **$ cat /etc/shells**
- The basic Syntax to change your default shell : **$ chsh username new_default_shell**



```
mona@mona-Vostro-3480:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/bin/rbash
/bin/dash
mona@mona-Vostro-3480:~$ echo "My current shell is $SHELL ($0)"
My current shell is /bin/bash (bash)
mona@mona-Vostro-3480:~$ chsh
Password:
Changing the login shell for mona
Enter the new value, or press ENTER for the default
        Login Shell [/bin/bash]: 
```

## Shell Scripting

- Shell script is a plain text file that contains a series of command(s).
- Shell script takes input from user, file and output them on screen.
- It saves lots of time by automating routine admin tasks.

## Why to use Shell Script?

- Shell scripts can be used to automate administrative tasks.
- Encapsulate complex configuration details and get a full power of the operating system.
- The ability to combine commands allows you to create new commands, thereby adding value to your operating system.

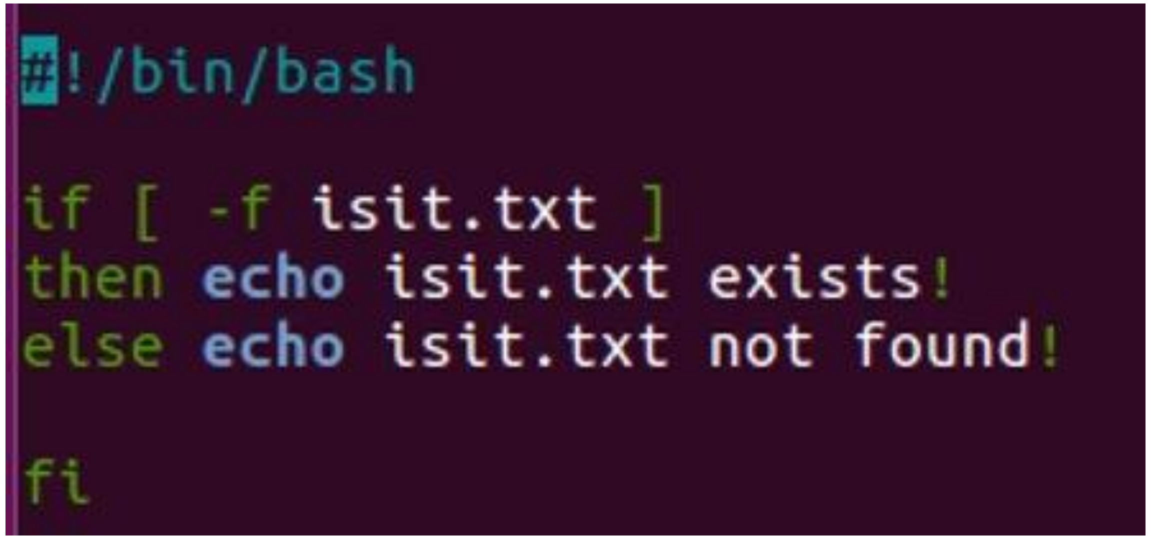
## Practical examples of Shell Scripting:

- Monitoring the Linux system.
- Data backup and creating snapshots.
- Find out what processes are eating up your system resources.
- Find out available and free memory.
- Find out all logged in users and what they are doing.
- Find out if all necessary network services are running or not.

# Practical examples:

write a shell script to use if then else

```
#!/bin/bash
if [ -f isit.txt ]
then echo isit.txt exists!
else echo isit.txt not found!
fi
```

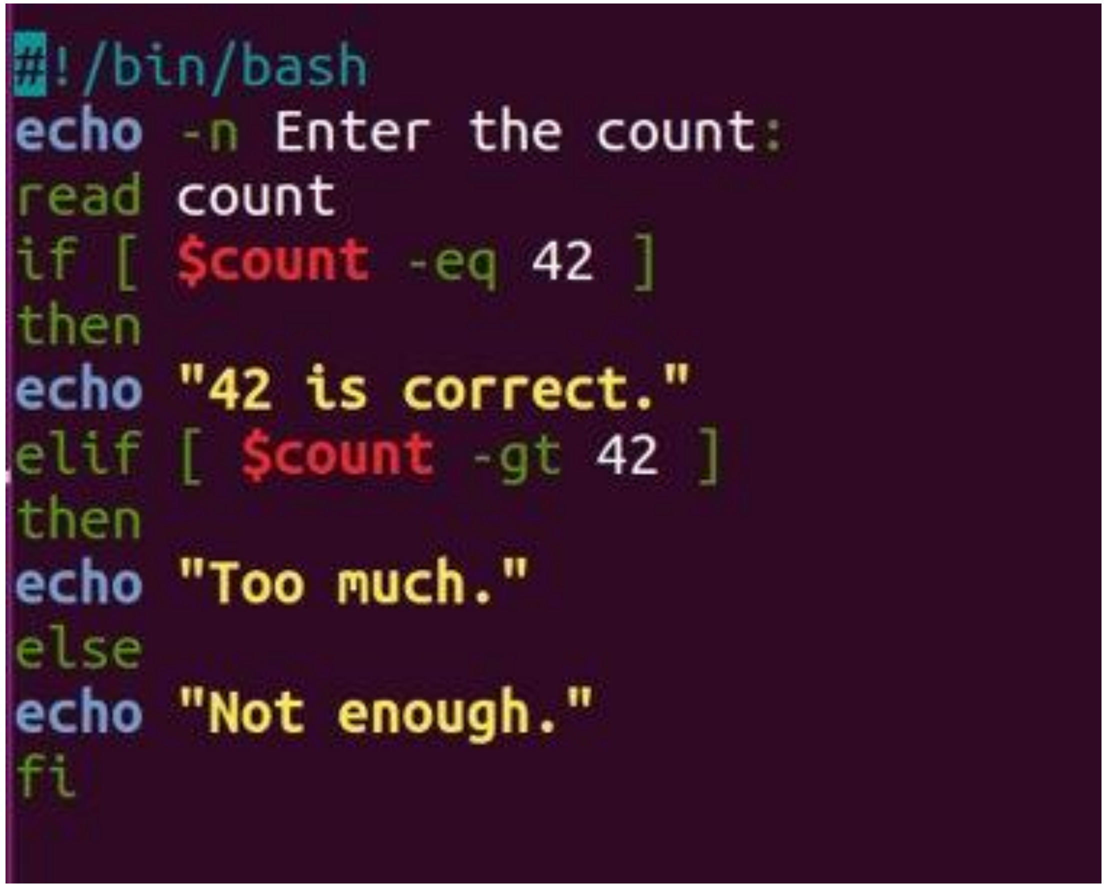


```
#!/bin/bash

if [ -f isit.txt ]
then echo isit.txt exists!
else echo isit.txt not found!

fi
```

## Practical examples: if then elif

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
```



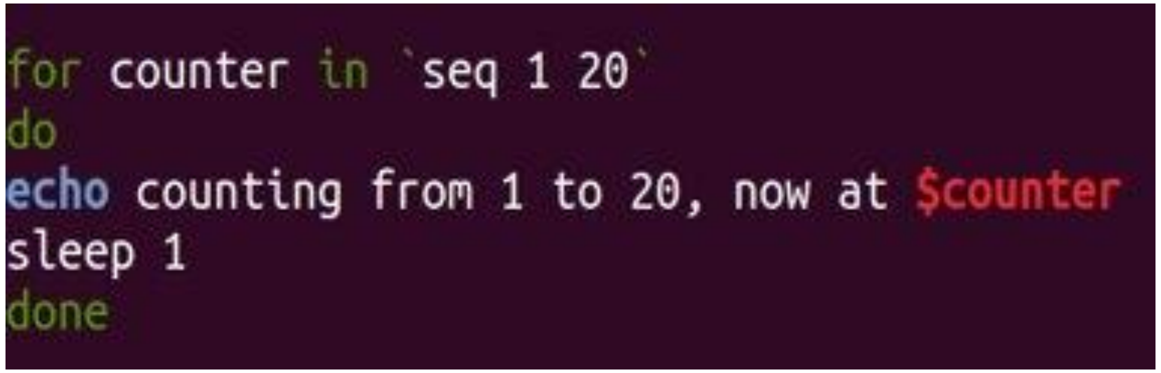
```
#!/bin/bash
echo -n Enter the count:
read count
if [ $count -eq 42 ]
then
echo "42 is correct."
elif [ $count -gt 42 ]
then
echo "Too much."
else
echo "Not enough."
fi
```

# Practical examples: for loop

**#!/bin/bash**
**for counter in `seq 1 20`**
**do**
**echo counting from 1 to 20, now at $counter**
**sleep 1**
**done**

another way:

**#!/bin/bash**
**for i in 1 2 4**
**do**
**echo $i**
**done**

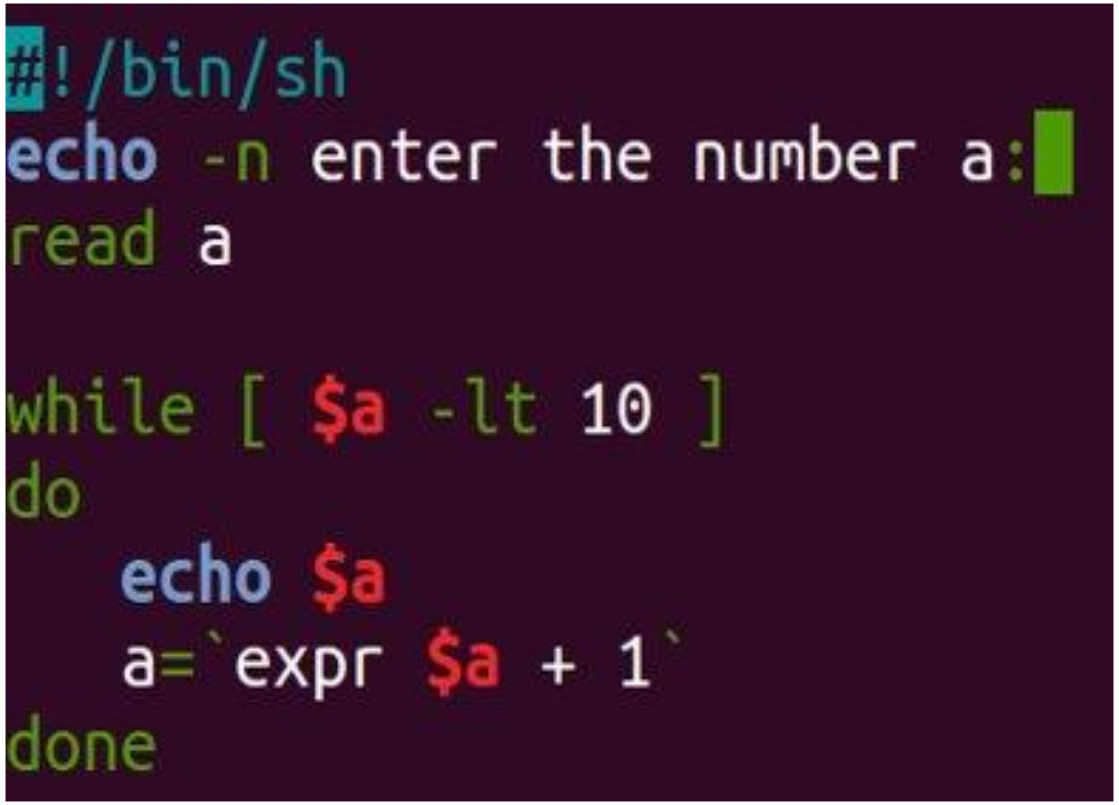


```
for counter in `seq 1 20`
do
echo counting from 1 to 20, now at $counter
sleep 1
done
```

# Practical examples: the while loop

```
while command
do
   Statement(s) to be executed if command is true
done
```

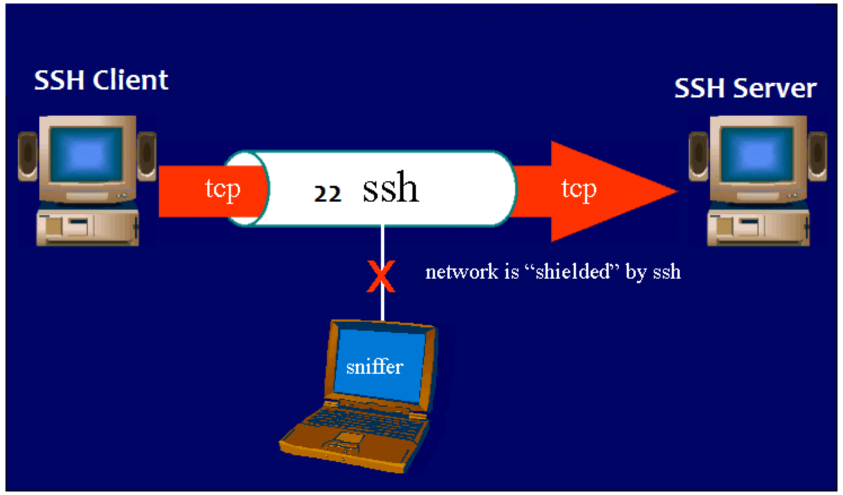
```
#!/bin/sh
a=0
while [ $a -lt 10 ]
do
   echo $a
   a=`expr $a + 1`
done
```



```
#!/bin/sh
echo -n enter the number a:
read a

while [ $a -lt 10 ]
do
    echo $a
    a=`expr $a + 1`
done
```

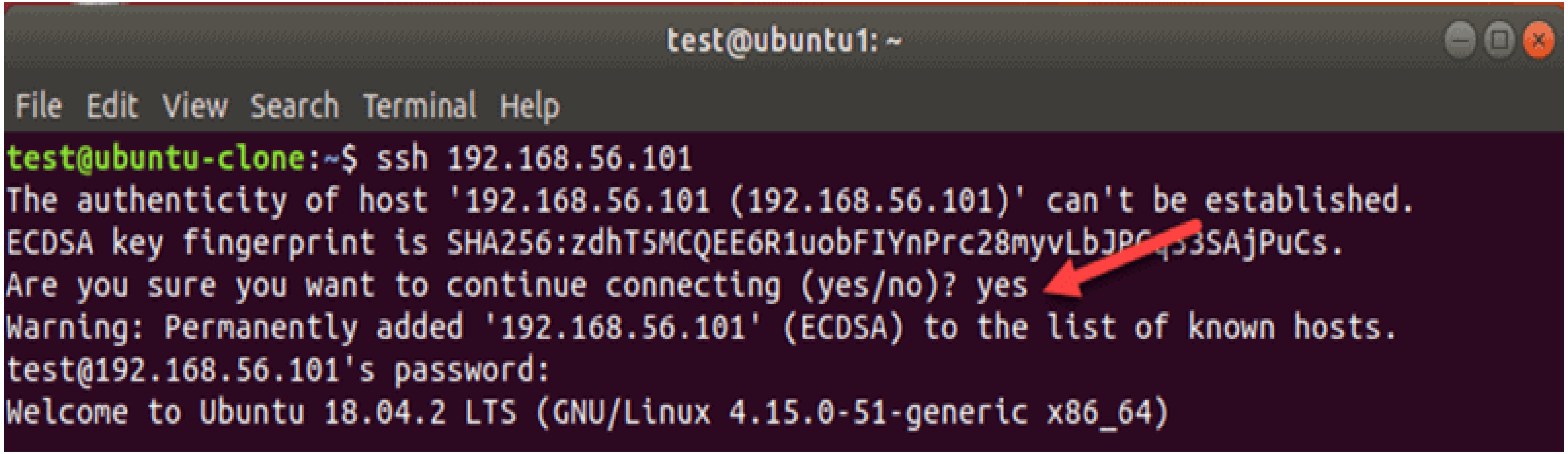
## SSH

- *SSH* stands for “**Secure Shell**”. It is a protocol used to securely connect to a remote server/system.
- *SSH* is secure in the sense that it transfers the data in encrypted form between the host and the client.
- It transfers inputs from the client to the host and relays back the output. *SSH* runs at TCP/IP port 22



***ssh user_name@host(IP/Domain_name)*** ***Example: ssh root@192.168.1.1***

## SSH



```
test@ubuntu-clone:~$ ssh 192.168.56.101
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
ECDSA key fingerprint is SHA256:zdhT5MCQEE6R1uobFIYnPrc28myvLbJPCq53SAjPuCs.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.101' (ECDSA) to the list of known hosts.
test@192.168.56.101's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-51-generic x86_64)
```

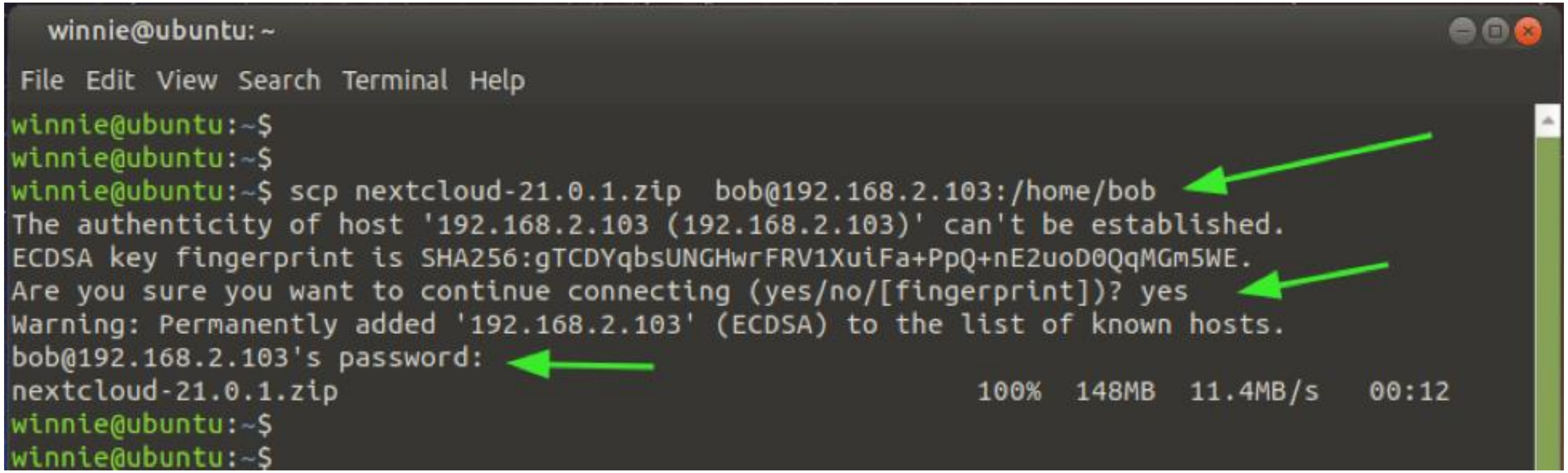
https://phoenixnap.com/kb/linux-ssh-commands

# SCP

- The scp command allows you to copy files over ssh connections.
- This is pretty useful if you want to transport files between computers
- Syntax

  *scp examplefile yourusername@yourserver:/home/yourusername/*
- *Example*

  *scp file1.pdf root@192.168.1.1:/root/Desktop*



```
winnie@ubuntu:~$
winnie@ubuntu:~$
winnie@ubuntu:~$ scp nextcloud-21.0.1.zip  bob@192.168.2.103:/home/bob
The authenticity of host '192.168.2.103 (192.168.2.103)' can't be established.
ECDSA key fingerprint is SHA256:gTCDYqbsUNGHwrFRV1XuiFa+PpQ+nE2uoD0QqMGm5WE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.2.103' (ECDSA) to the list of known hosts.
bob@192.168.2.103's password:
nextcloud-21.0.1.zip                                    100%  148MB  11.4MB/s   00:12
winnie@ubuntu:~$
winnie@ubuntu:~$
```

## Conclusion

- We know about the basic of Linux operating systems
- We are now  familiar with basic Linux commands including the Linux command line, Linux servers, file systems, and much more.
- We know about Linux Terminal, and shell scripting.

# REFERENCES

1. https://en.wikipedia.org/wiki/Linux_kernel
2. https://www.tutorialspoint.com/operating_system/os_linux.html
3. https://buildmedia.readthedocs.org/media/pdf/lym/latest/lym.pdf
4. https://phoenixnap.com/kb/linux-commands-cheat-sheet
5. https://www.guru99.com/file-permissions.html
6. https://www.hostinger.in/tutorials/linux-commands

## THANK YOU