# Linux Fundamentals

**Prof. B. Thangaraju, IIITB**

# Course Overview

- Lesson 1
  - Virtualization, VirtualBox and OS Introduction
- Lesson 2
  - Working with Commands
- Lesson 3
  - Working with Files, User Management and Text Editors
- Lesson 4
  - Process, Signals, Shell Scripting and Crontab
- Lesson 5
  - Networking, Services and Firewall

# Lesson 1

- Introducing Concepts of Virtualization
  - Types of Virtualization
  - Benefits of Virtualization
- Installing Linux in VirtualBox
  - Introduction to VirtualBox
  - Linux Virtual Machine Installation
- Introduction to Linux Operating System (OS)

# Introducing Concepts of Virtualization

# Introduction to Virtualization

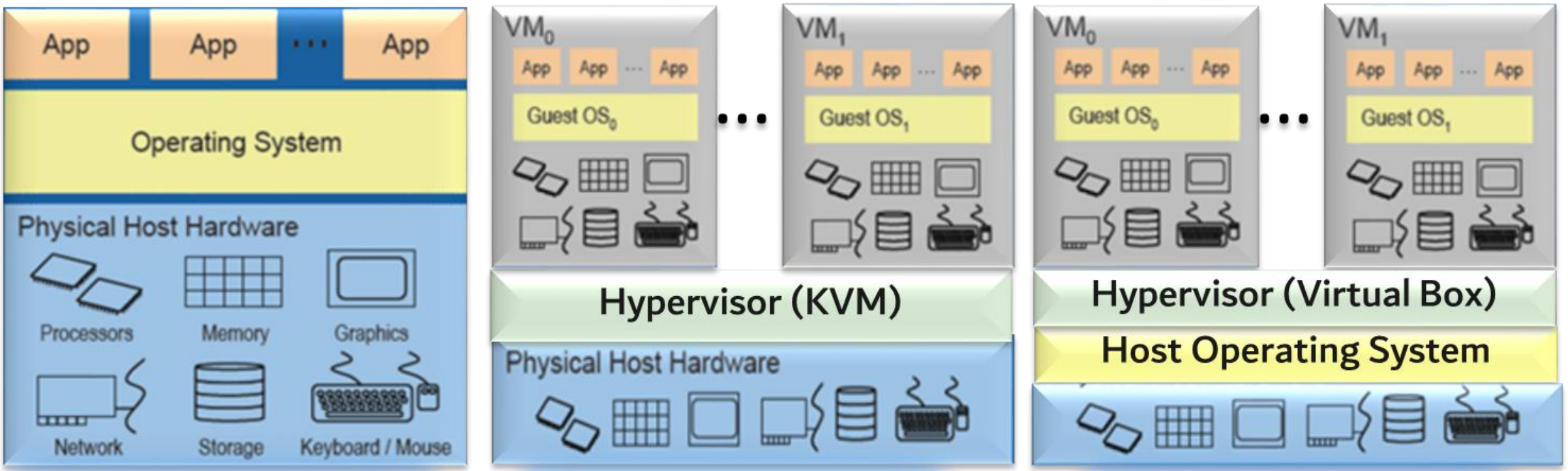
- **Software demands more** and more from operating systems to applications.
  - More : data, processing power, memory, network bandwidth and storage space.
- **Virtualization is a technology**.
  - Create multiple simulated environments from a single, physical hardware systems.
- **Software called a hypervisor** connects directly to the hardware.
  - Split one system into separate, distinct, and secure environments known as virtual machines (VMs).

# Introduction to Virtualization

The term **hypervisor** is a variant of supervisor, a traditional term for the kernel of an operating system: the *hypervisor is the* ***supervisor of the supervisors***

- The hypervisor/**VMM is the control system** at the core of virtualization.
  - Acts as the control and translation system between the VMs and the hardware.
- **Hypervisors need OS** - level components to run VMs
  - Memory manager, Process scheduler, I/O stack, DD, security manager and Network stack.
- The physical hardware, equipped with a hypervisor is called the **Host**, while the many VMs that use its resources are **Guests**.

# Types of Virtualization

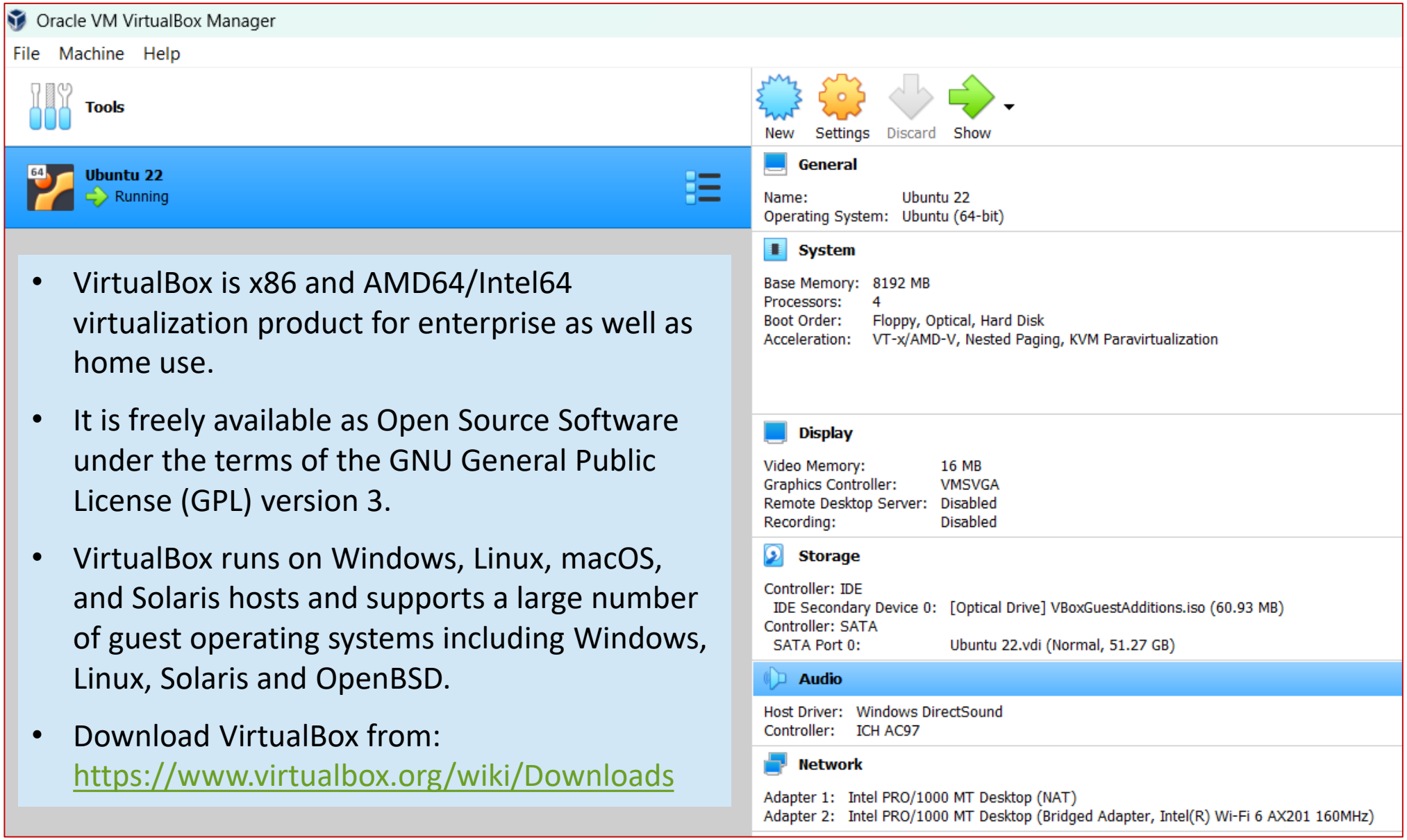


# Benefits of Virtualization

- Virtualization is being used by a growing number of organizations
  - Reduce power consumption and air conditioning needs (Move to be more Green-Friendly)
  - Trim the building space and land requirements
  - Reduce downtime and provides high availability for critical applications (Blue-Green model)
  - Simplify IT operations
  - Allow IT organizations to respond faster to changing business demands

# Installing Linux in Virtualbox

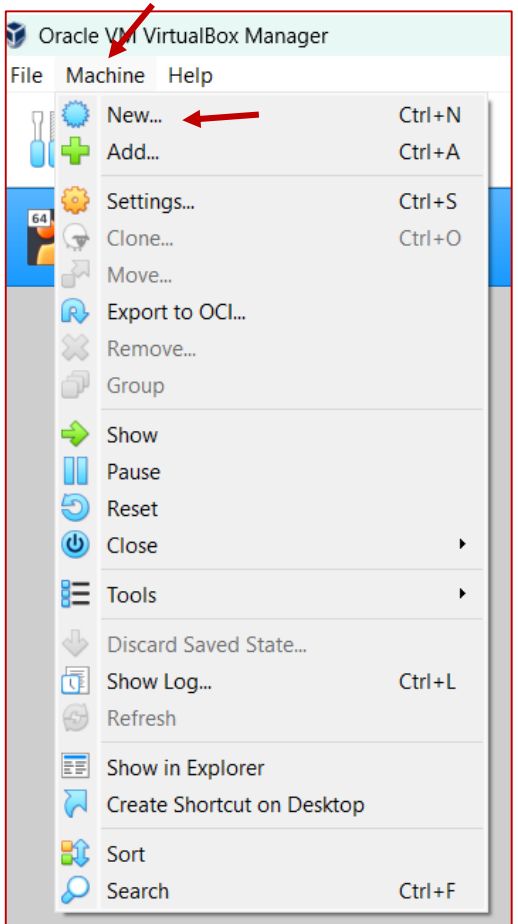
# VirtualBox

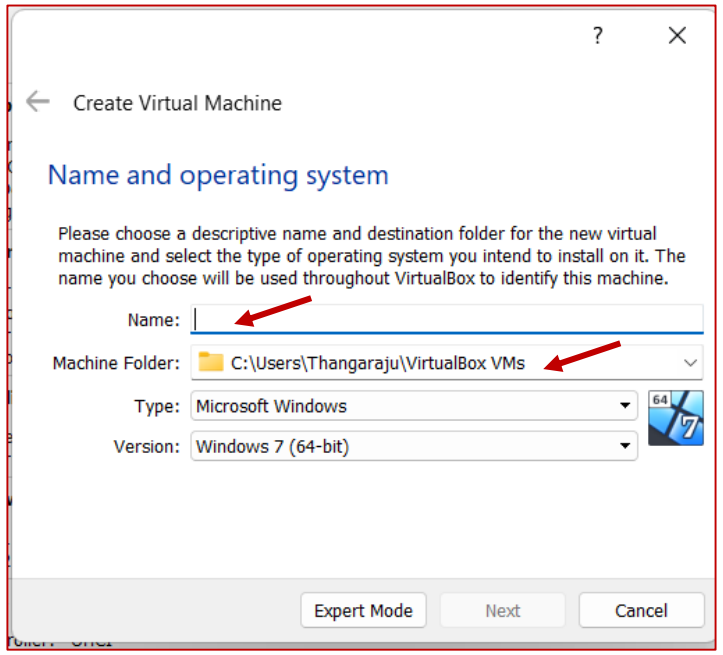


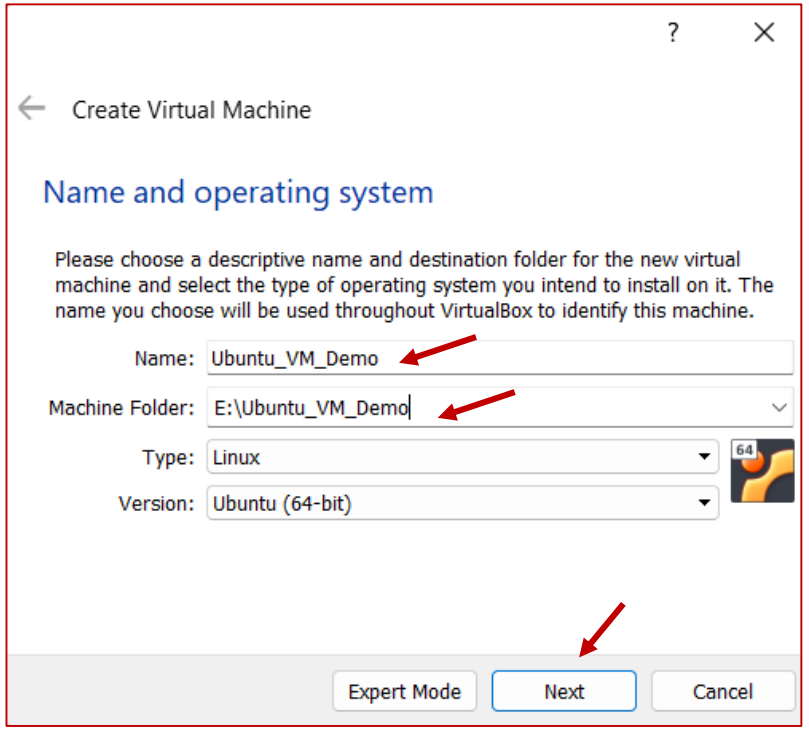

- VirtualBox is x86 and AMD64/Intel64 virtualization product for enterprise as well as home use.
- It is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 3.
- VirtualBox runs on Windows, Linux, macOS, and Solaris hosts and supports a large number of guest operating systems including Windows, Linux, Solaris and OpenBSD.
- Download VirtualBox from: https://www.virtualbox.org/wiki/Downloads

# Linux Virtual Machine Installation

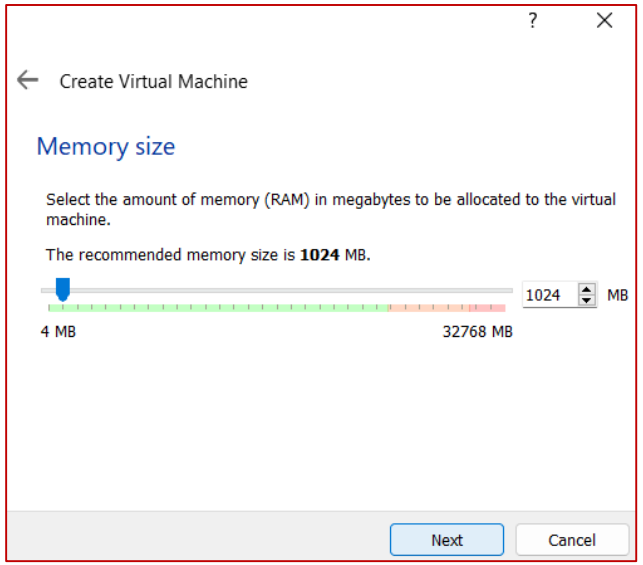
Step 1: Download virtual box and install.
Step 2: Download Ubuntu iso image from:
https://ubuntu.com/download/desktop/thank-you?version=22.04.1&architecture=amd64
Step 3: save **ubuntu-22.04.1-desktop-amd64** iso image into a desired directory.
In this demo I store the iso image and virtual machine in my E folder. E:\Ubuntu_VM_Demo

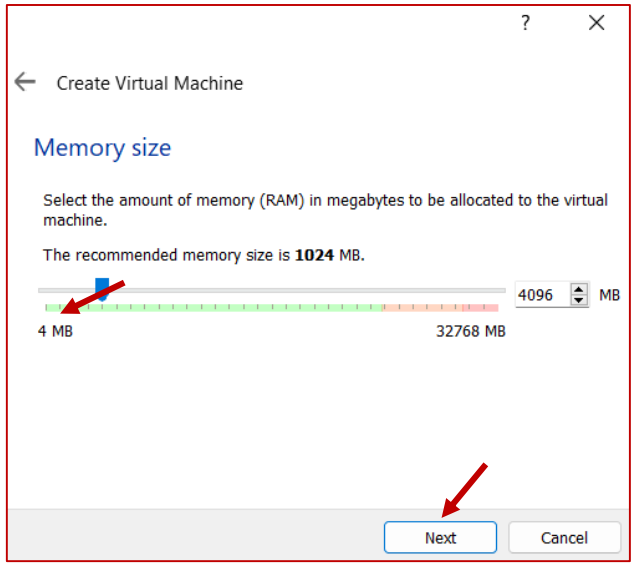


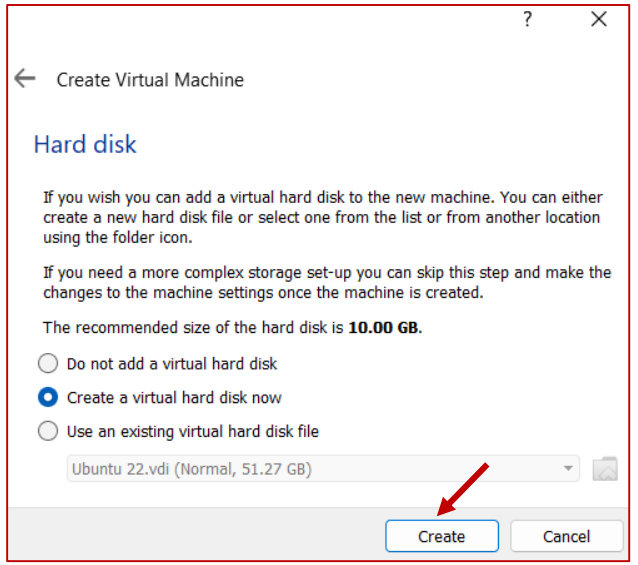



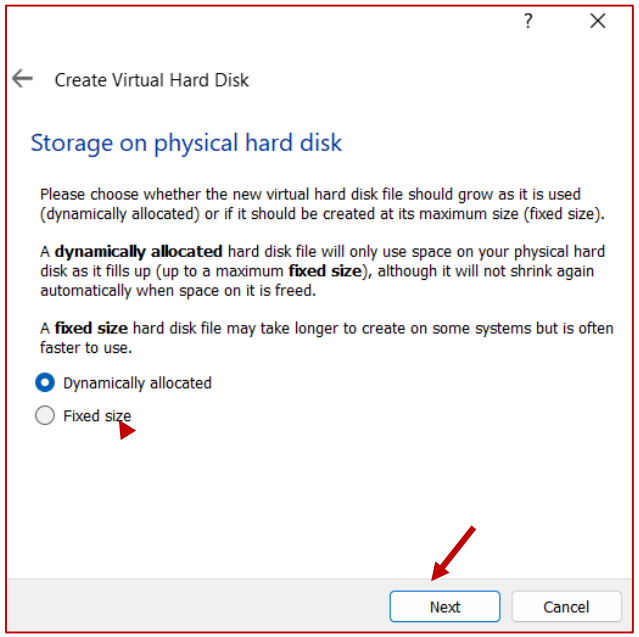
# Linux Virtual Machine Installation

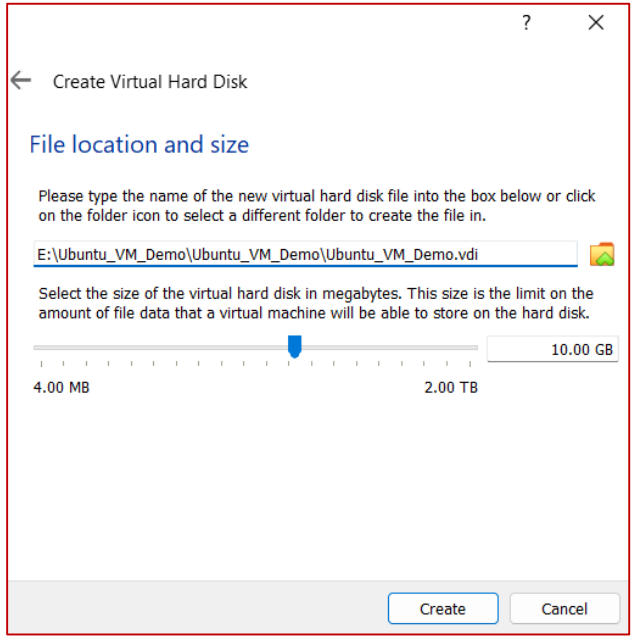


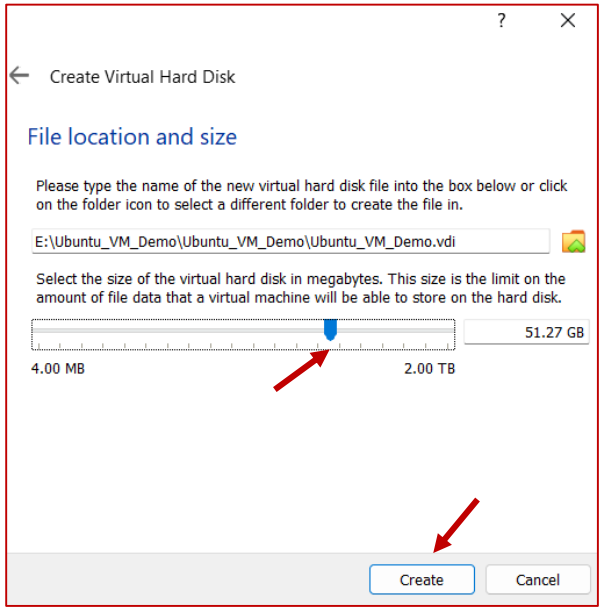



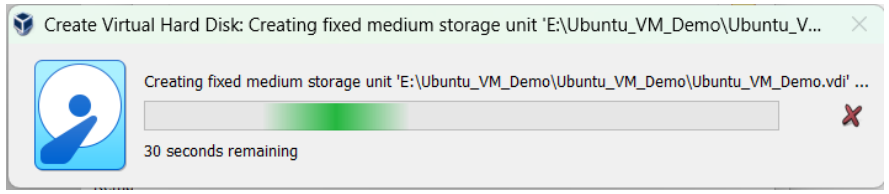


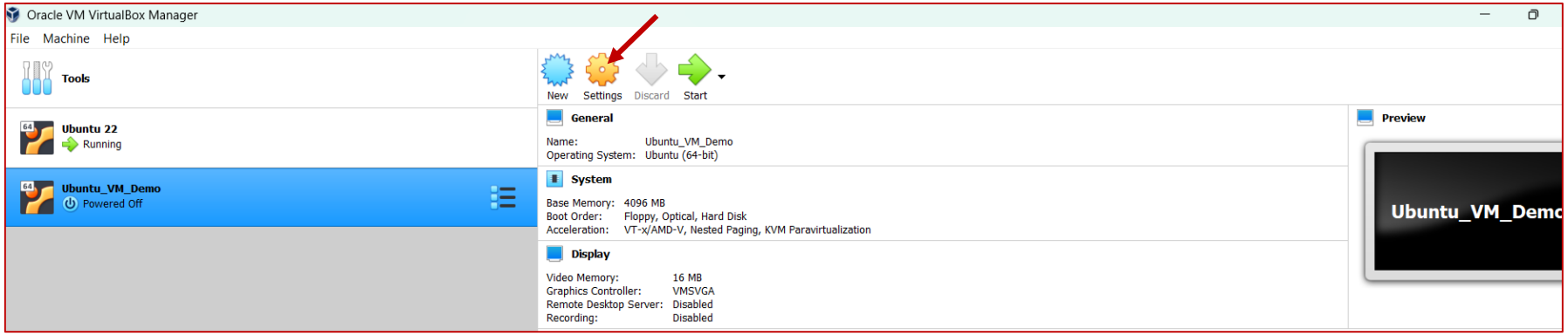


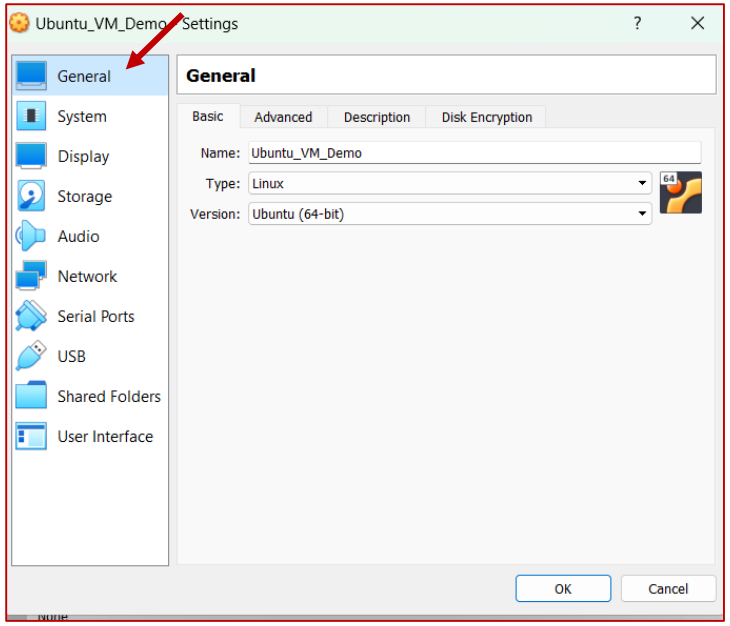



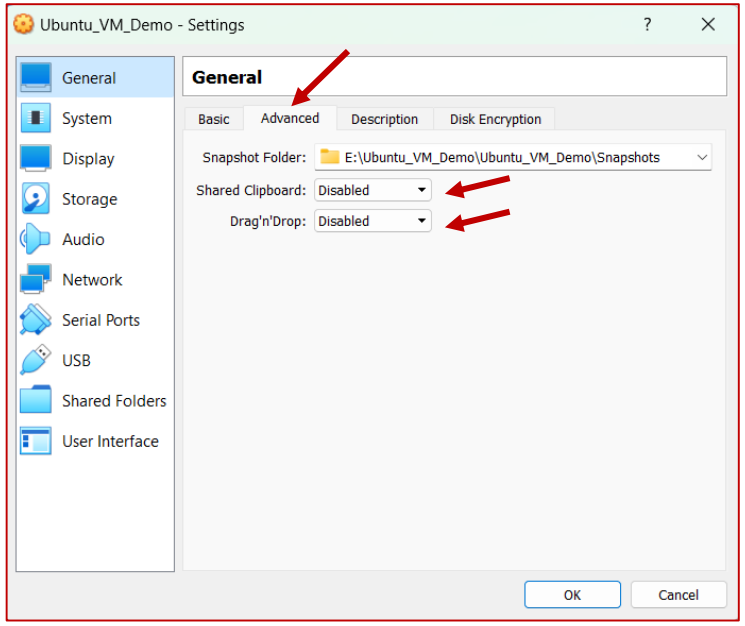
# Linux Virtual Machine Installation

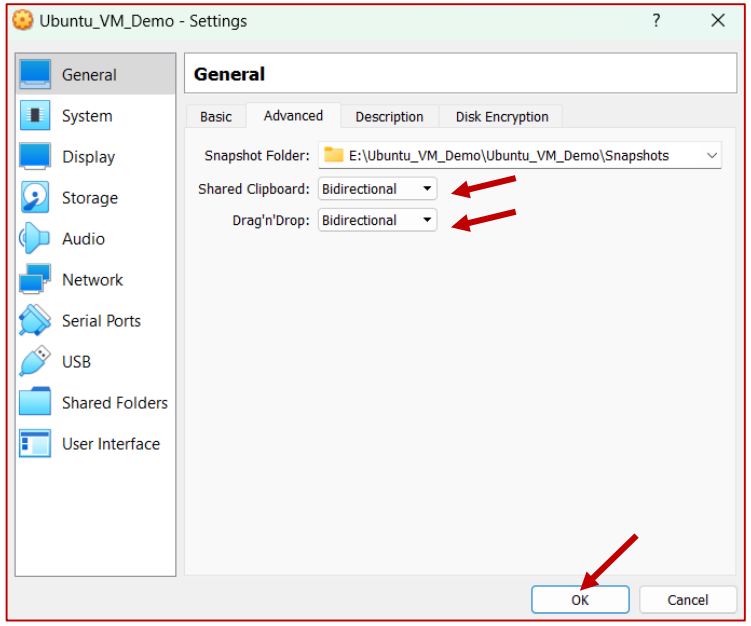


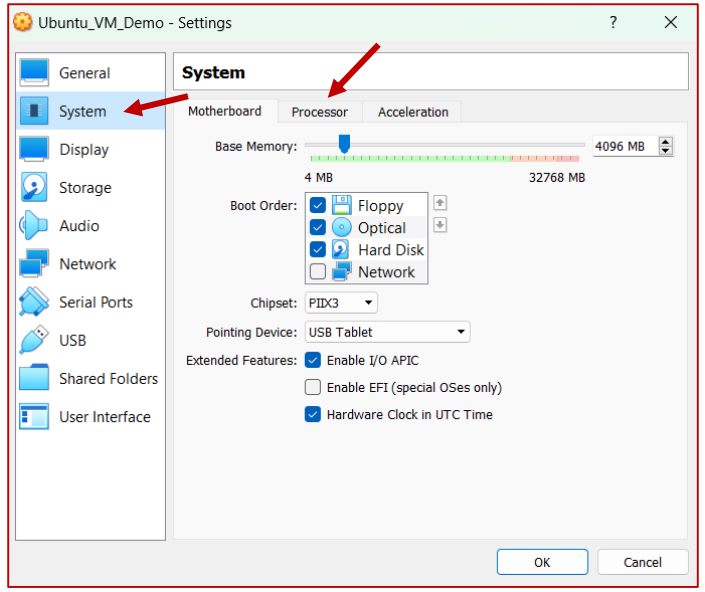



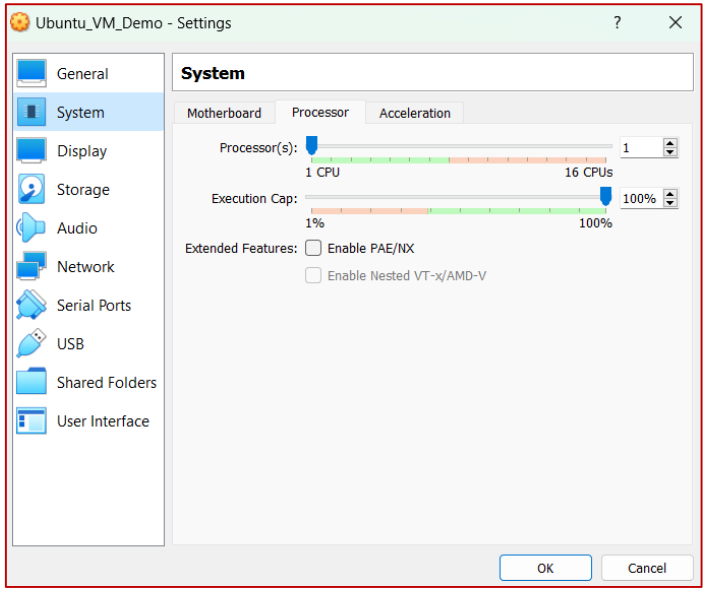



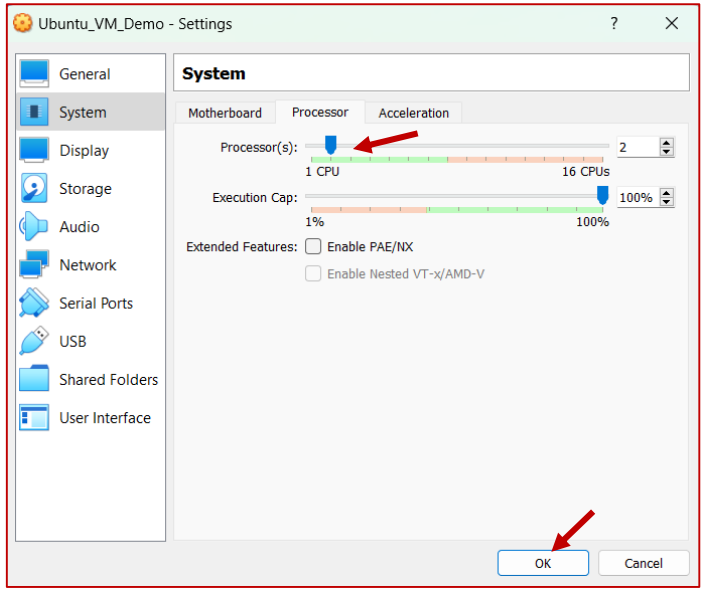



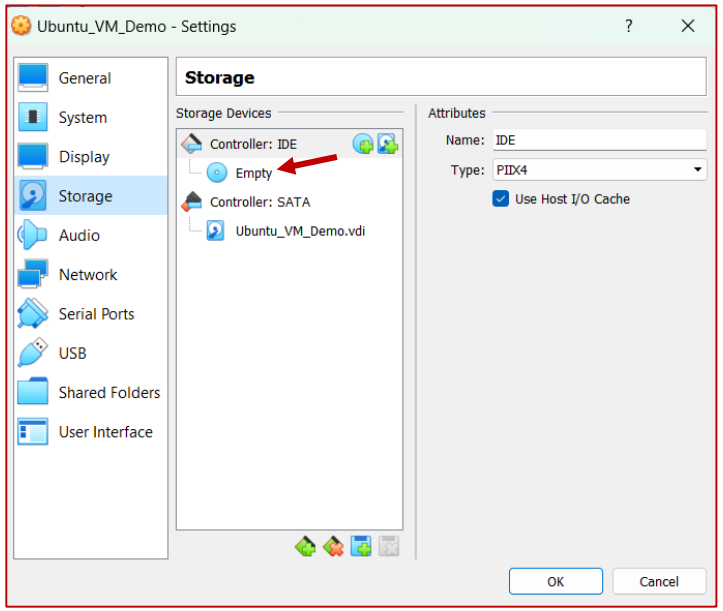



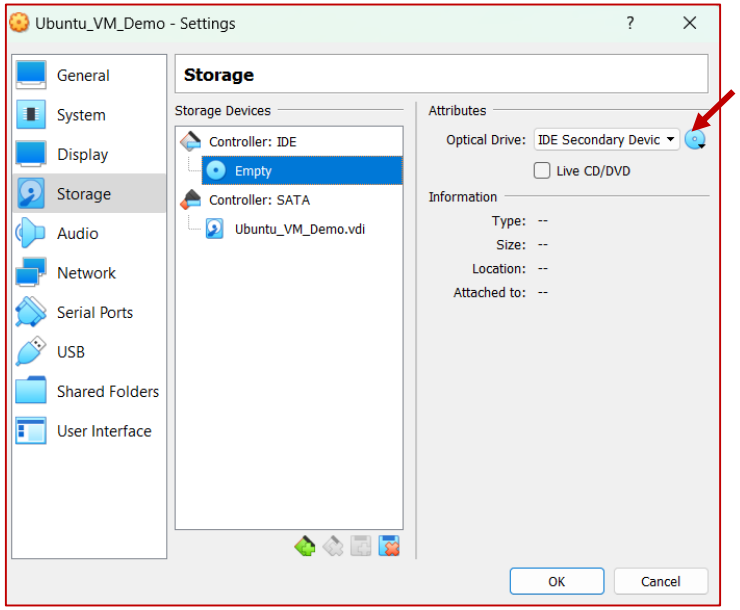
# Linux Virtual Machine Installation

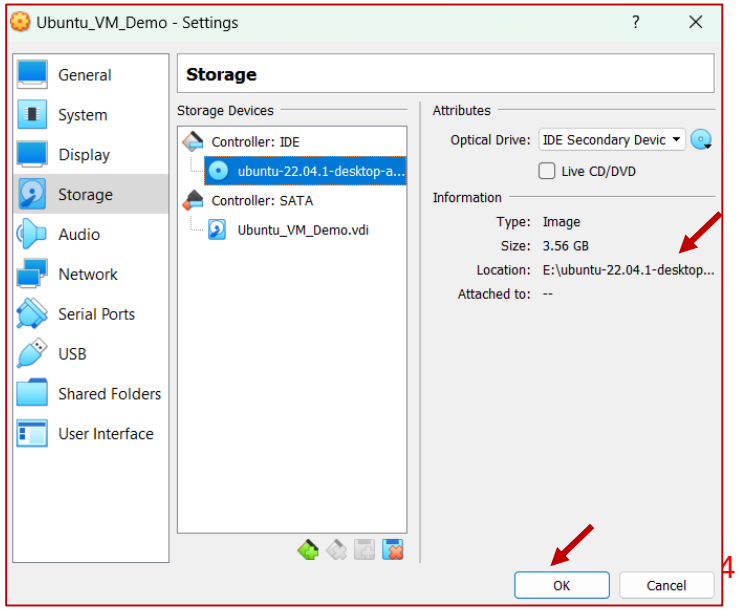


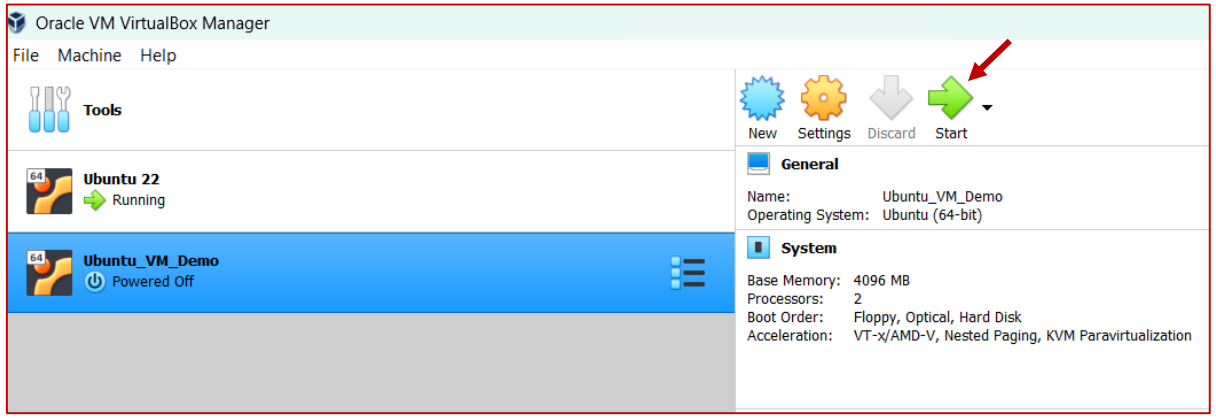



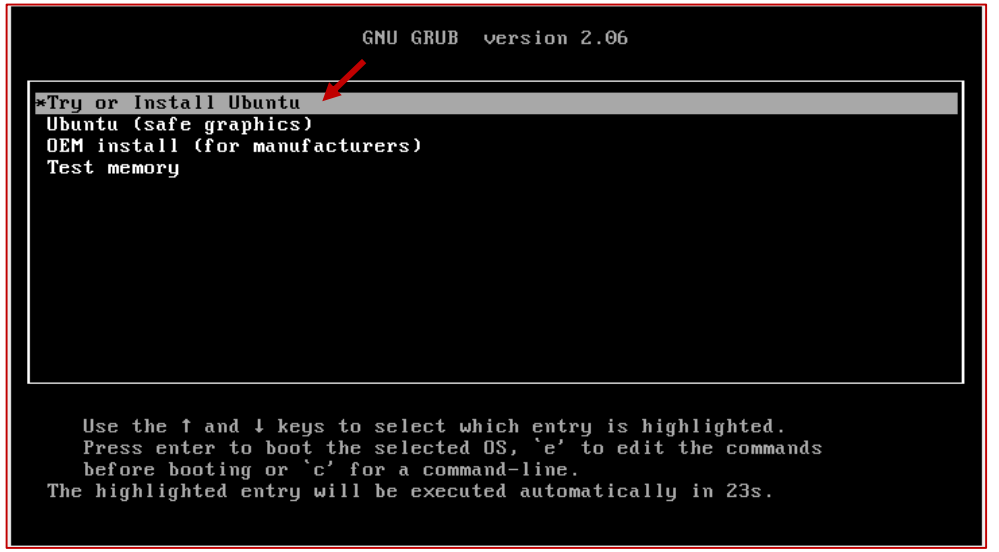



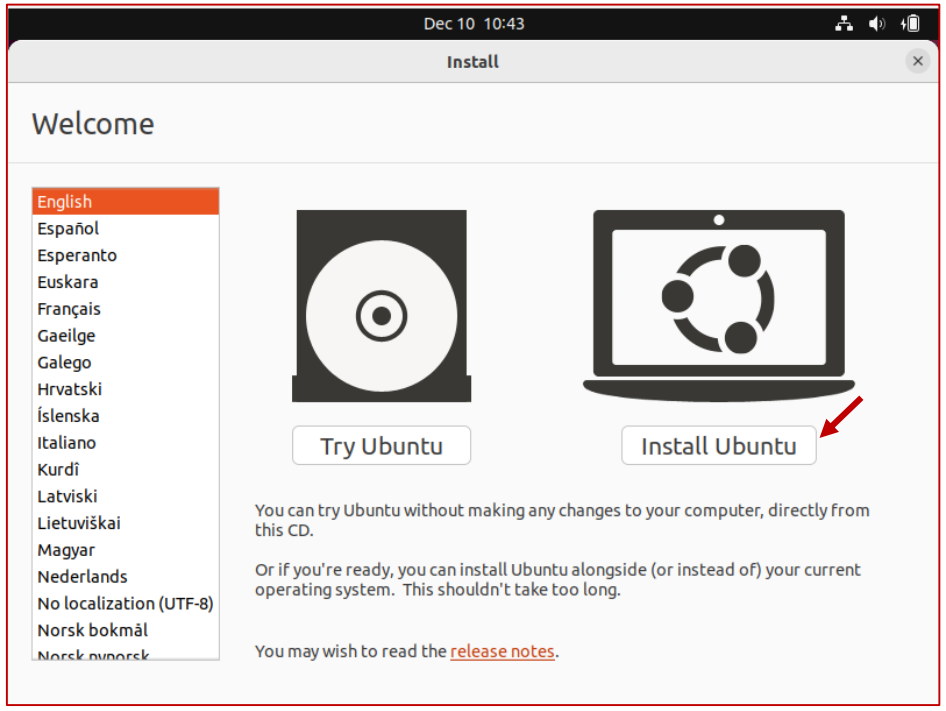



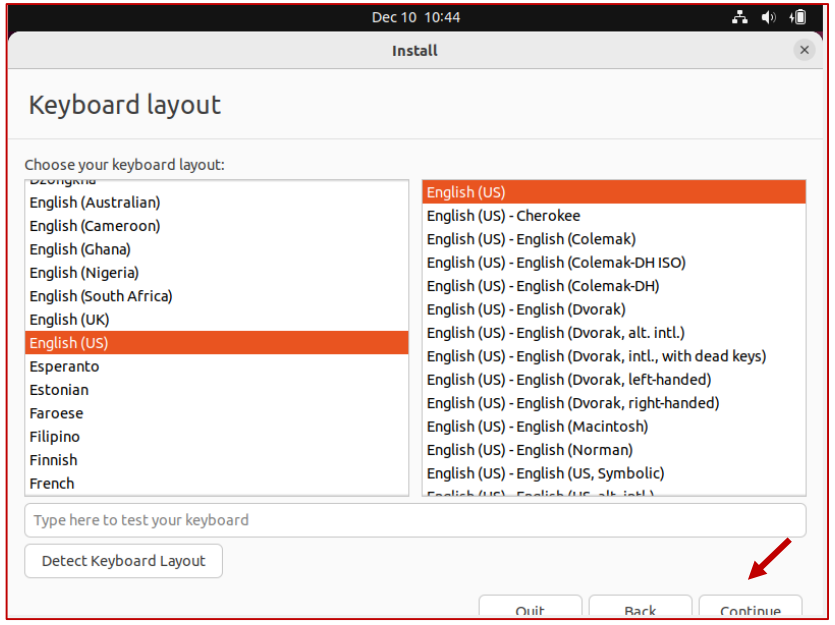



# Linux Virtual Machine Installation

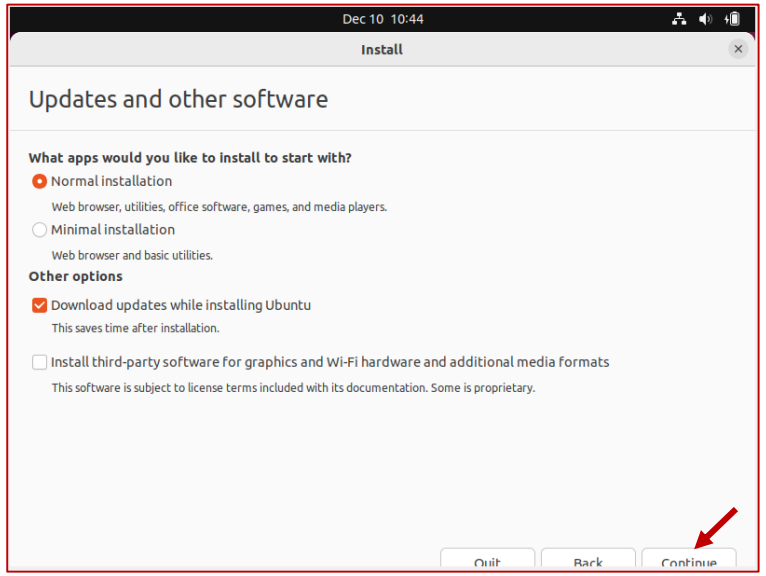


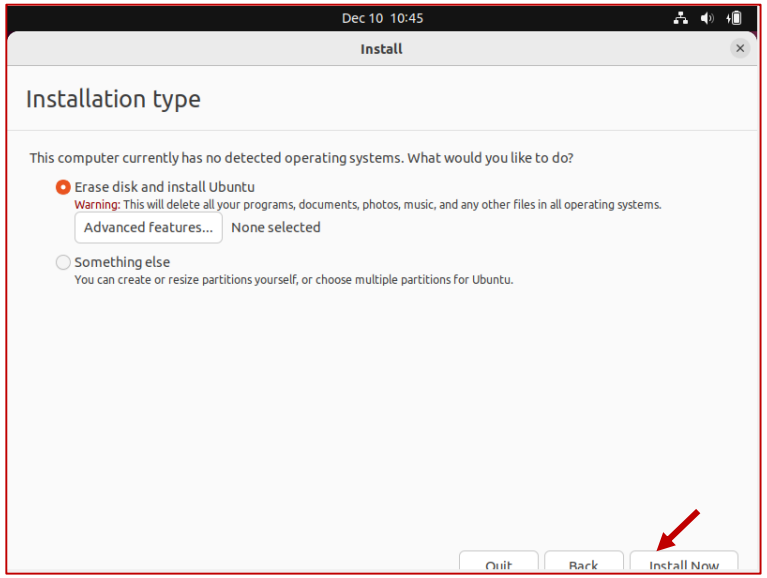



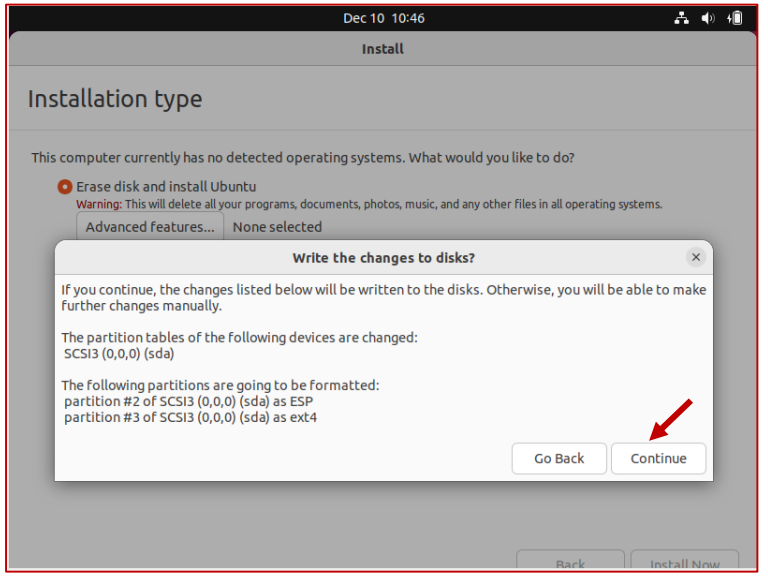



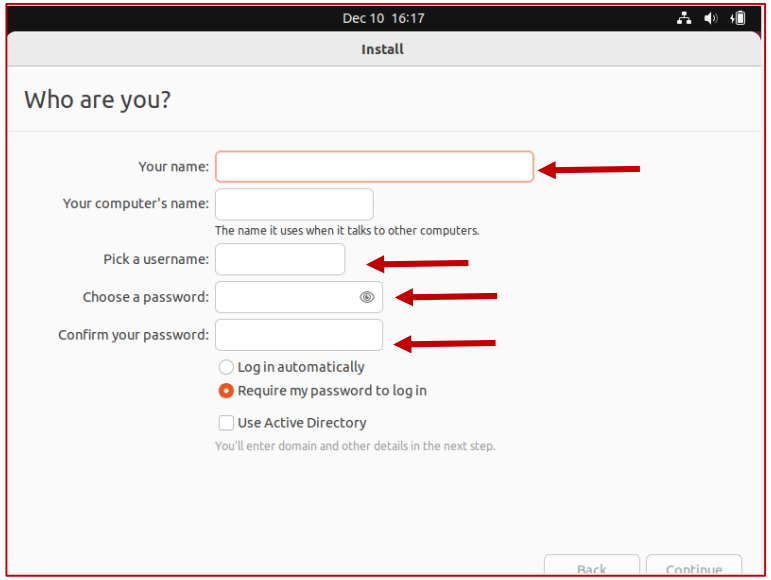
# Linux Virtual Machine Installation

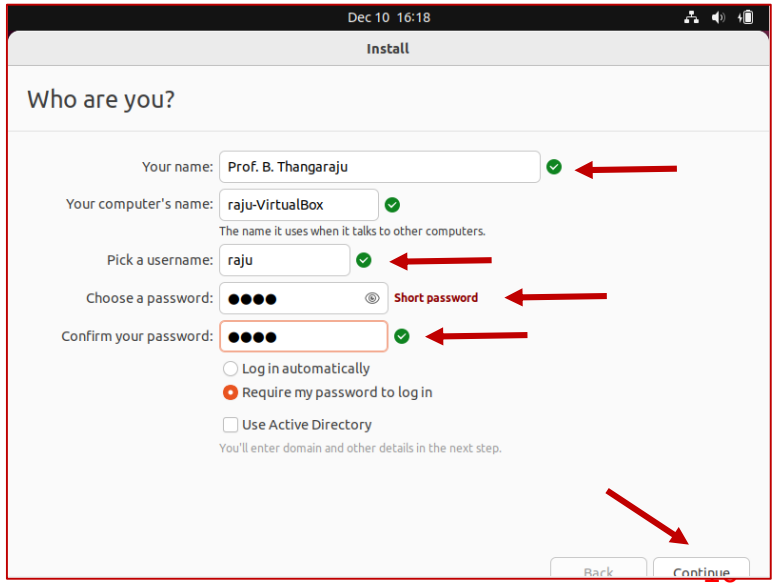


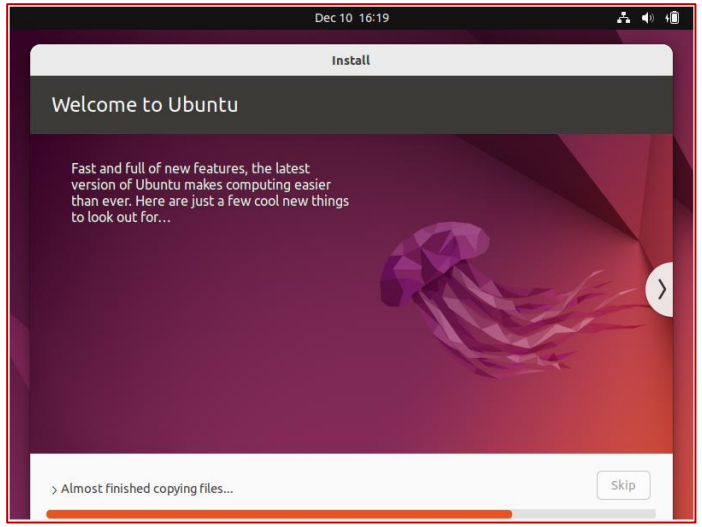



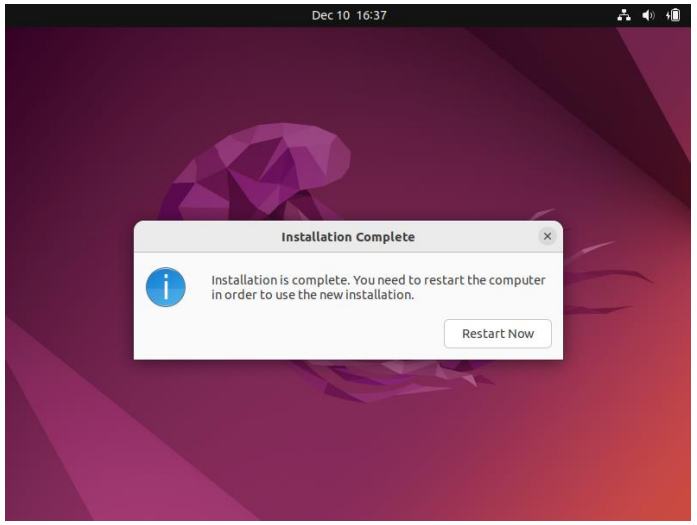


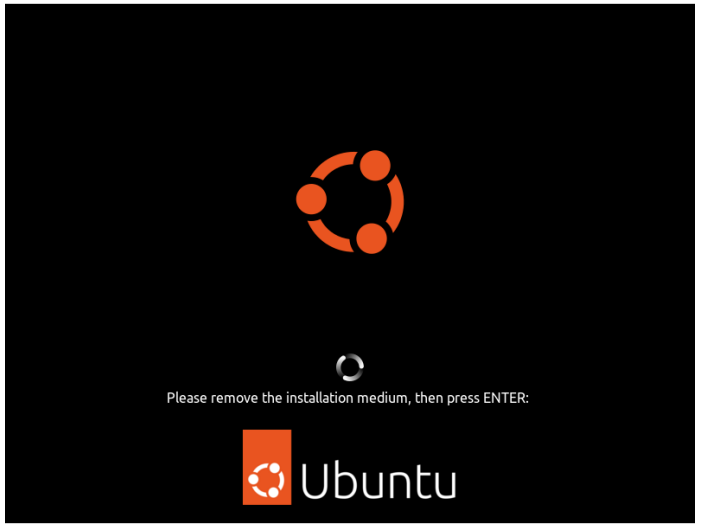


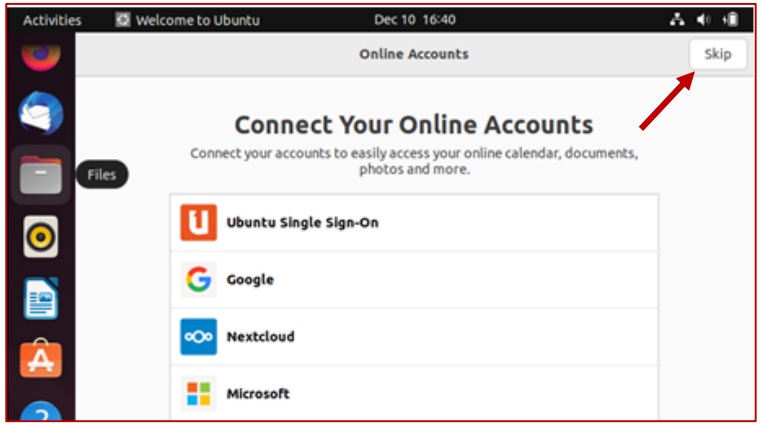
# Linux Virtual Machine Installation

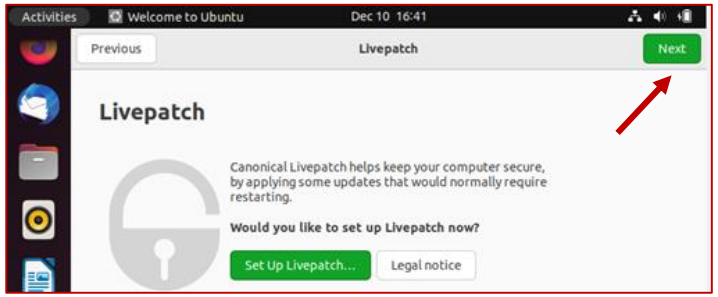


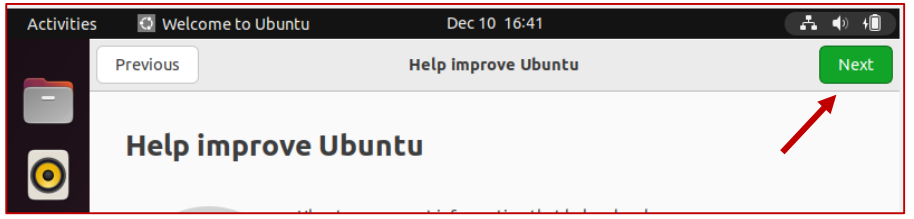



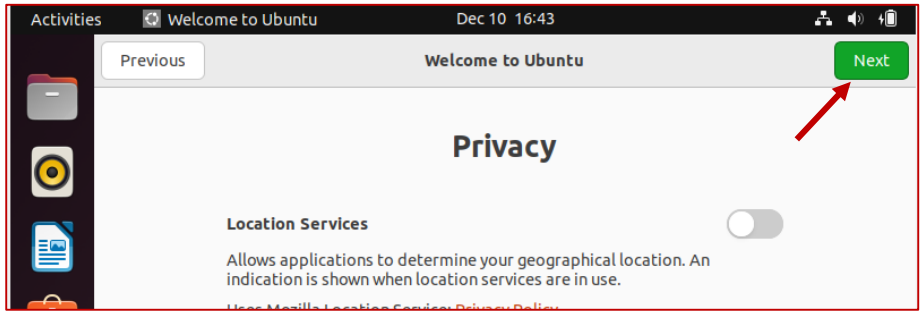


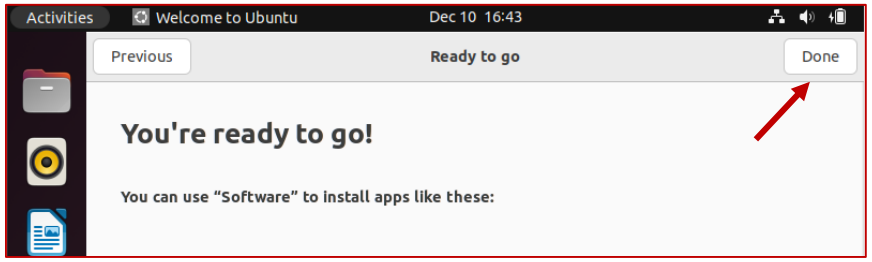


# Linux Virtual Machine Installation

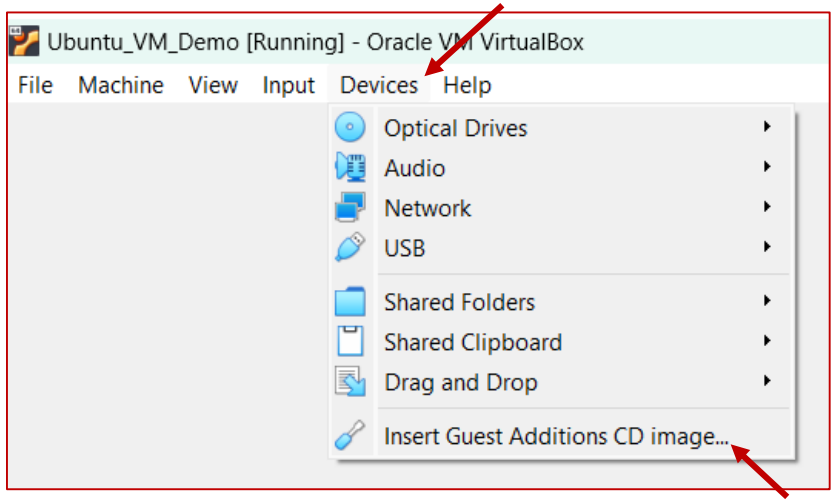


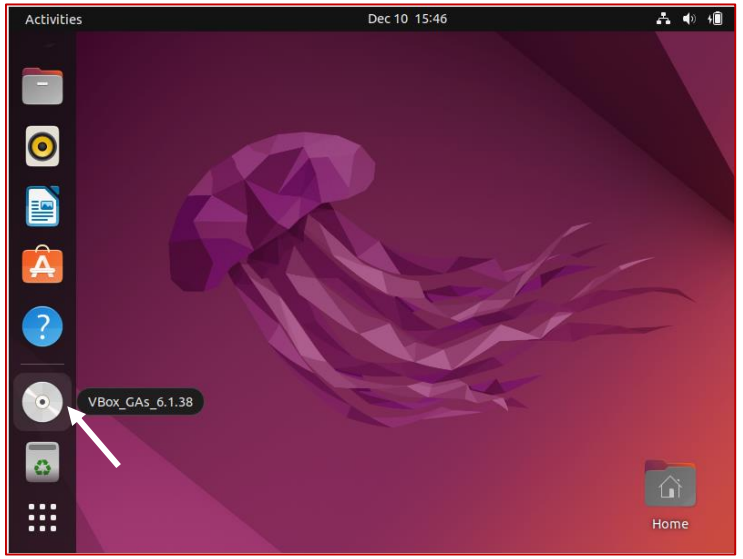



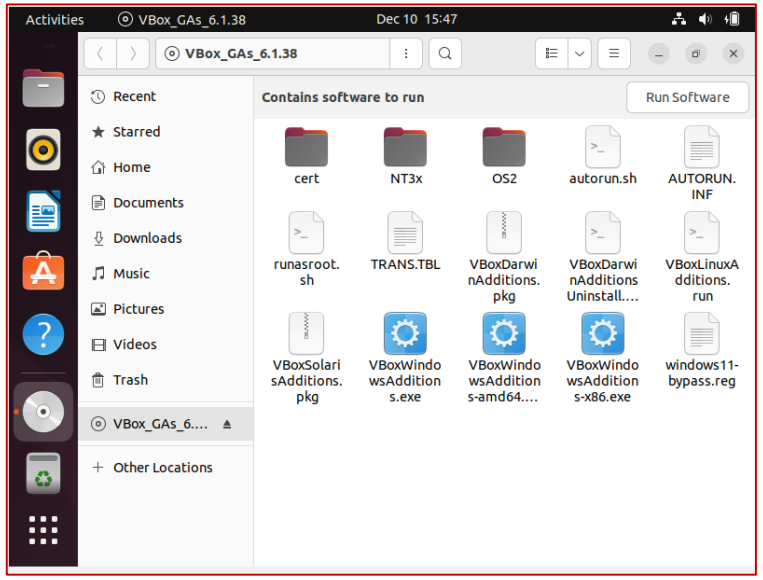

To make full screen, open a terminal and execute the following command:

**$sudo apt update**

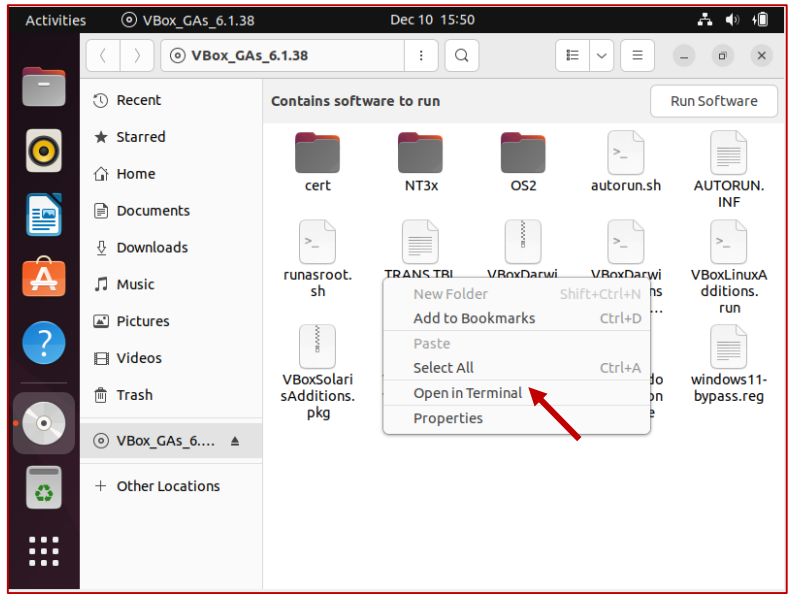
**$sudo apt install -y build-essential linux-headers-$(uname -r)**

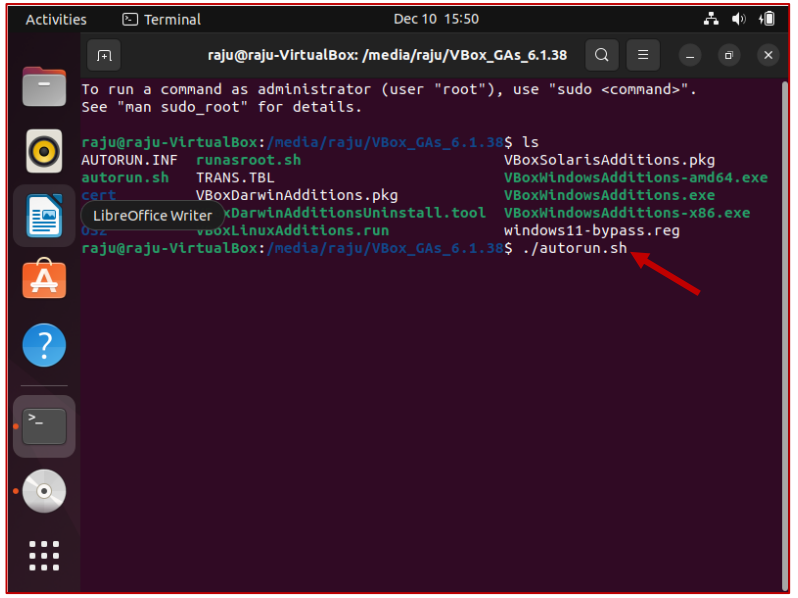


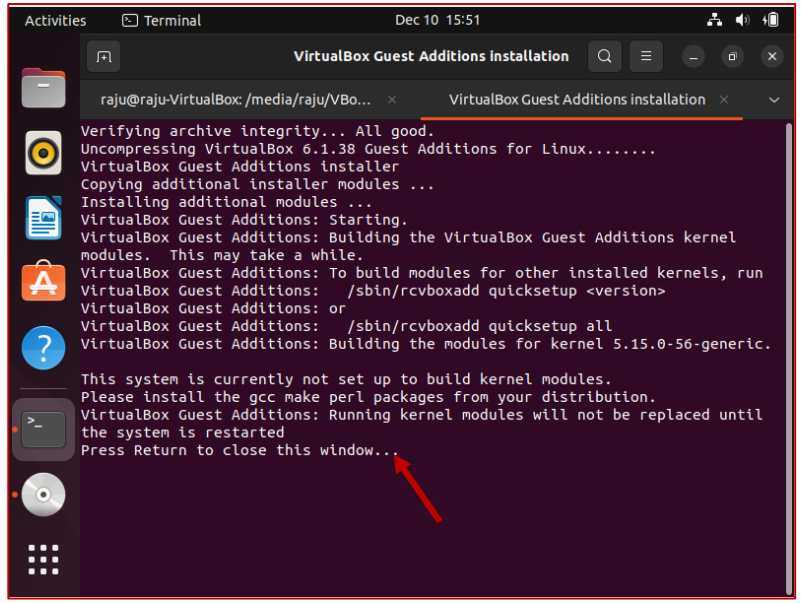



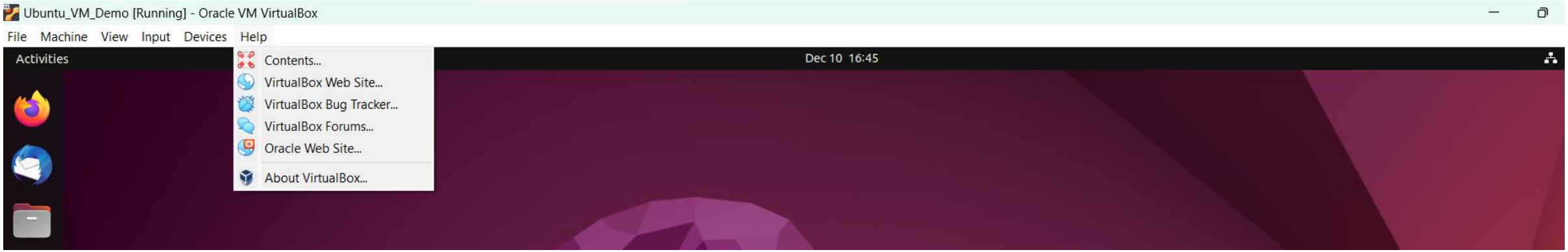



# Linux Virtual Machine Installation



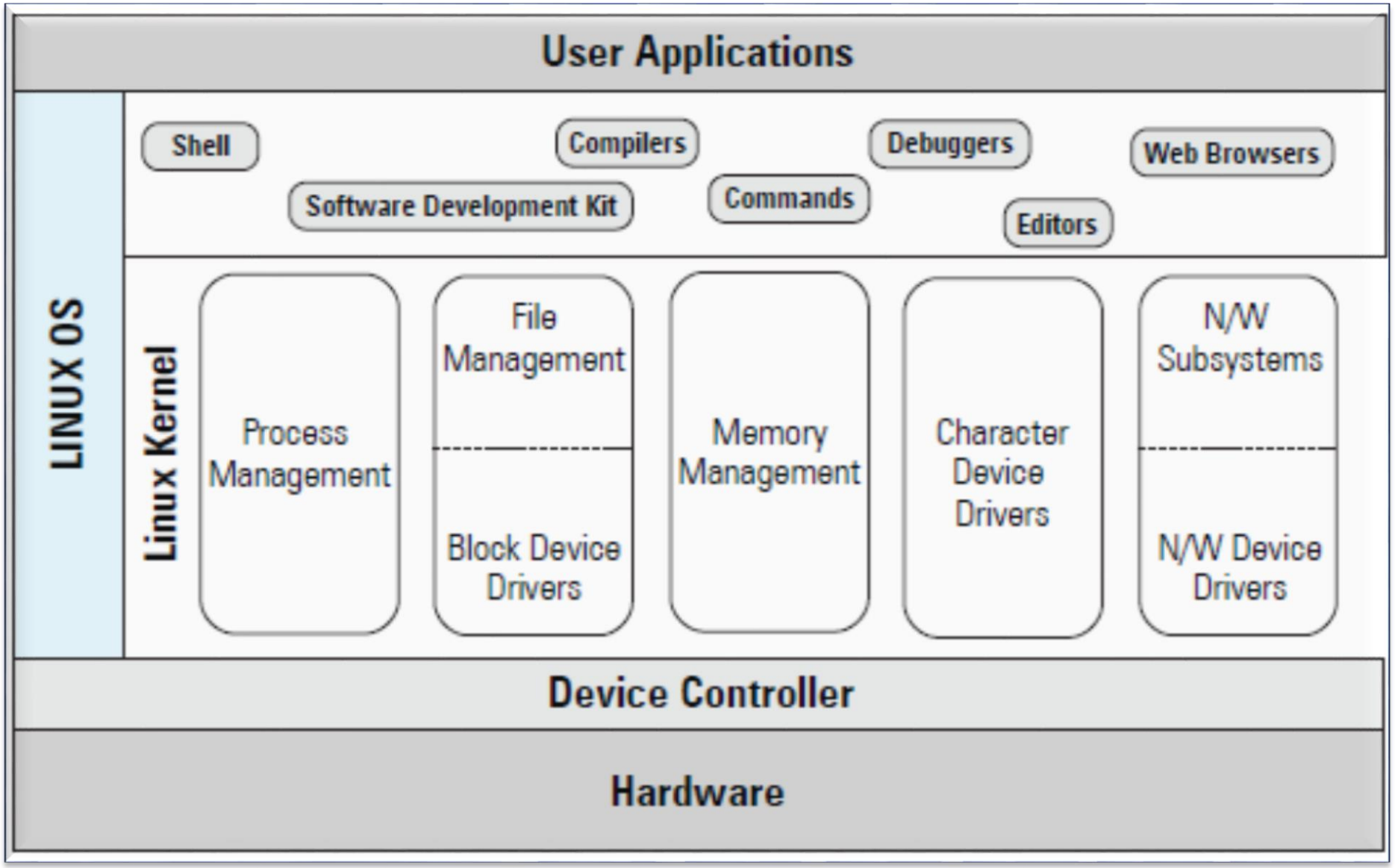



# Introduction to Linux OS

# Introduction



# Introduction

Linux Kernel is available at : https://kernel.org/


To check Ubuntu version information: $lsb_release -a

```
raju@raju-VirtualBox:/boot$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.1 LTS
Release:        22.04
Codename:       jammy
raju@raju-VirtualBox:/boot$
```

View Kernel version in a system: $**uname -r**

```
raju@raju-VirtualBox:~$ uname -r
5.15.0-56-generic
raju@raju-VirtualBox:~$
```

You can upgrade your system with the latest kernel. Need not to wait for new Ubuntu release.

```
raju@raju-VirtualBox:/boot$ uname -r
6.0.0-060000-generic
raju@raju-VirtualBox:/boot$
```

# Lesson 2 – Working with Linux Commands

- File Globbing (Regular Expressions)
- Quoting Commands
- Getting Help in the Command Line
- Working in the Shell Efficiently
- Streams, Redirects, and Pipes
- Global regular expression print (grep)
- The Sed Command
- The Awk Command

# Glob

- When working with files in bash, we often use wildcard characters to match file names.
- Bash interprets these characters and performs filename expansion - this process is called as globbing.
- **glob** - globbing pathnames
- $ man 7 glob –to get more details
- Globbing is the operation that expands a wildcard pattern into the list of pathnames matching the pattern.
- A string is a wildcard pattern if it contains one of the characters '?', '*' or '['.
- Matching is defined by:
- ? - matches any single character.
- * - matches any string, including the empty string.
- [ ] - matches the characters inside
- ! - excludes characters from the list
- [[:named:]] - Named character classes

# glob

```
raju@raju-VirtualBox: ~/demo
raju@raju-VirtualBox:~/demo$ ll
total 8
drwxrwxr-x  2 raju raju 4096 Nov 26 10:46  ./
drwxr-x--- 25 raju raju 4096 Nov 26 10:25  ../
-rw-rw-r--  1 raju raju    0 Nov 26 10:30  1223
-rw-rw-r--  1 raju raju    0 Nov 26 10:29  123.txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:25  abc10.txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:25  abc11.txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:25  abc12.txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:25  abc13.txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:25  abc14.txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:25  abc9.txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:33  abc_txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:46  test1
-rw-rw-r--  1 raju raju    0 Nov 26 10:46  test2
-rw-rw-r--  1 raju raju    0 Nov 26 10:46  test3
-rw-rw-r--  1 raju raju    0 Nov 26 10:46  test4.txt
-rw-rw-r--  1 raju raju    0 Nov 26 10:33 'test file'
-rw-rw-r--  1 raju raju    0 Nov 26 10:34  TESTFILE
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls -l abc*
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc10.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc11.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc12.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc13.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc14.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc9.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:33 abc_txt
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls -l abc?.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc9.txt
raju@raju-VirtualBox:~/demo$
```

# glob

```
raju@raju-VirtualBox:~/demo$ ls -l abc??.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc10.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc11.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc12.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc13.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc14.txt
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls -l test[1-3]
-rw-rw-r-- 1 raju raju 0 Nov 26 10:46 test1
-rw-rw-r-- 1 raju raju 0 Nov 26 10:46 test2
-rw-rw-r-- 1 raju raju 0 Nov 26 10:46 test3
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls -l abc[0-9]*
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc10.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc11.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc12.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc13.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc14.txt
-rw-rw-r-- 1 raju raju 0 Nov 26 10:25 abc9.txt
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls -l abc???.txt
ls: cannot access 'abc???.txt': No such file or directory
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ll test[!3]
-rw-rw-r-- 1 raju raju 0 Nov 26 10:46 test1
-rw-rw-r-- 1 raju raju 0 Nov 26 10:46 test2
raju@raju-VirtualBox:~/demo$
```

# glob

- **[:alnum:]** - list all files having alphabets (lower and uppercases) and digits.
- **[:alpha:]** - list all files having only alphabets (lower and uppercases).
- **[:upper:]** - list all files having only upper-case letters.
- **[:lower:]** - list all files having only lower-case letters.
- **[:digit:]** - list all files having digits.
- **[:space:]** - list all files having space characters.
- **[:punct:]** - list all files having punctuation characters.
  Search for ! ” # $ % & ‘ ( ) * + , – . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~.

```
raju@raju-VirtualBox:~/demo$ ls *[[:alnum:]]
 1223         abc11.txt    abc14.txt    test1    test4.txt
 123.txt      abc12.txt    abc9.txt     test2   'test file'
 abc10.txt    abc13.txt    abc_txt      test3    TESTFILE
raju@raju-VirtualBox:~/demo$
raju@raju-VirtualBox:~/demo$ ls [[:alnum:]]*
 1223         abc11.txt    abc14.txt    test1    test4.txt
 123.txt      abc12.txt    abc9.txt     test2   'test file'
 abc10.txt    abc13.txt    abc_txt      test3    TESTFILE
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls [[:alpha:]]*
 abc10.txt    abc13.txt    abc_txt      test3        TESTFILE
 abc11.txt    abc14.txt    test1        test4.txt
 abc12.txt    abc9.txt     test2       'test file'
raju@raju-VirtualBox:~/demo$
```

# glob

```
raju@raju-VirtualBox:~/demo$ ls [[:lower:]]*
 abc10.txt   abc13.txt   abc_txt    test3
 abc11.txt   abc14.txt   test1      test4.txt
 abc12.txt   abc9.txt    test2     'test file'
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls [[:digit:]]*
1223  123.txt
raju@raju-VirtualBox:~/demo$ ls *[[:digit:]]
1223  test1  test2  test3
raju@raju-VirtualBox:~/demo$ ls *[[:digit:]]*
1223        abc11.txt  abc14.txt  test2
123.txt     abc12.txt  abc9.txt   test3
abc10.txt   abc13.txt  test1      test4.txt
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls *[[:space:]]*
'test file'
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls [[:upper:]]*
TESTFILE
raju@raju-VirtualBox:~/demo$
```

```
raju@raju-VirtualBox:~/demo$ ls *[[:punct:]]*
123.txt    abc11.txt  abc13.txt  abc9.txt   test4.txt
abc10.txt  abc12.txt  abc14.txt  abc_txt
raju@raju-VirtualBox:~/demo$
```

# Quoting Commands

**“ double quote”** – preserves the literal value of all characters within the quotes, with the exception of $, \.

**‘ Single quote’** – preserves the literal value of each character within the quotes.

**/ - escape character** - preserves the literal value of the next character that follows.

```
raju@raju-VirtualBox:~/demo$ env | grep $SHELL
SHELL=/bin/bash
raju@raju-VirtualBox:~/demo$ echo My shell is $SHELL
My shell is /bin/bash
raju@raju-VirtualBox:~/demo$ echo My   shell   is   $SHELL
My shell is /bin/bash
raju@raju-VirtualBox:~/demo$ echo "My   shell   is   $SHELL"
My   shell   is   /bin/bash
raju@raju-VirtualBox:~/demo$ echo 'My shell is $SHELL'
My shell is $SHELL
raju@raju-VirtualBox:~/demo$ echo "My   shell   is   \$SHELL"
My   shell   is   $SHELL
raju@raju-VirtualBox:~/demo$ echo ' "My   shell   is   \$SHELL" '
 "My   shell   is   \$SHELL"
```

# Getting Help in the Command Line

**/bin** : commands like ls, echo, cat etc.

**/sbin** : Commands with super user privileges required to execute, example - fdisk, mkfs, etc.

```
raju@raju-VirtualBox:~$ help | more
GNU bash, version 5.1.16(1)-release (x86_64-pc-linux-gnu)
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

 job_spec [&]                                        history [-c] [-d offset] [n] or history -anrw [f>
 (( expression ))                                    if COMMANDS; then COMMANDS; [ elif COMMANDS; the>
 . filename [arguments]                              jobs [-lnprs] [jobspec ...] or jobs -x command [>
 :                                                   kill [-s sigspec | -n signum | -sigspec] pid | j>
 [ arg... ]                                          let arg [arg ...]
 [[ expression ]]                                    local [option] name[=value] ...
 alias [-p] [name[=value] ... ]                      logout [n]
 bg [job_spec ...]                                   mapfile [-d delim] [-n count] [-O origin] [-s co>
 bind [-lpsvPSVX] [-m keymap] [-f filename] [-q na>  popd [-n] [+N | -N]
 break [n]                                           printf [-v var] format [arguments]
 builtin [shell-builtin [arg ...]]                   pushd [-n] [+N | -N | dir]
 caller [expr]                                       pwd [-LP]
 case WORD in [PATTERN [| PATTERN]...) COMMANDS ;;>  read [-ers] [-a array] [-d delim] [-i text] [-n >
 cd [-L|[-P [-e]] [-@]] [dir]                        readarray [-d delim] [-n count] [-O origin] [-s >
 command [-pVv] command [arg ...]                    readonly [-aAf] [name[=value] ...] or readonly >
 compgen [-abcdefgjksuv] [-o option] [-A action] [>  return [n]
 complete [-abcdefgjksuv] [-pr] [-DEI] [-o option]>  select NAME [in WORDS ... ;] do COMMANDS; done
 compopt [-o|+o option] [-DEI] [name ...]            set [-abefhkmnptuvxBCHP] [-o option-name] [--] [>
```

# Getting Help in the Command Line

```
raju@raju-VirtualBox:~$ grep -h
Usage: grep [OPTION]... PATTERNS [FILE]...
Try 'grep --help' for more information.
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ grep --h
Usage: grep [OPTION]... PATTERNS [FILE]...
Search for PATTERNS in each FILE.
Example: grep -i 'hello world' menu.h main.c
PATTERNS can contain multiple patterns separated by newlines.

Pattern selection and interpretation:
  -E, --extended-regexp     PATTERNS are extended regular expressions
  -F, --fixed-strings       PATTERNS are strings
```

```
raju@raju-VirtualBox:~$ grep --help
Usage: grep [OPTION]... PATTERNS [FILE]...
Search for PATTERNS in each FILE.
Example: grep -i 'hello world' menu.h main.c
PATTERNS can contain multiple patterns separated by newlines.

Pattern selection and interpretation:
  -E, --extended-regexp     PATTERNS are extended regular expressions
  -F, --fixed-strings       PATTERNS are strings
  -G, --basic-regexp        PATTERNS are basic regular expressions
  -P, --perl-regexp         PATTERNS are Perl regular expressions
  -e, --regexp=PATTERNS     use PATTERNS for matching
```

**Any command with - -h or - -help will give more details.**

# Getting Help in the Command Line

**Tab**: it will help to complete the incomplete command or list all the relevant commands.

```
raju@raju-VirtualBox:~$ ls Tab
ls              lsb_release   lsinitramfs   lslogins      lsns          lspcmcia
lsattr          lscpu         lsipc         lsmem         lsof          lspgpot
lsblk           lshw          lslocks       lsmod         lspci         lsusb
raju@raju-VirtualBox:~$ ls
```

**$ man ls**

```
LS(1)                                   User Commands                                   LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current directory by default).  Sort entries alphabetically if none of
       -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below
```

# Getting Help in the Command Line

```
MAN(1)                          Manual pager utils                          MAN(1)

NAME
       man - an interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [man options] [section] term ...
       man -f [whatis options] page ...
       man -l [man options] file ...
       man -w|-W [man options] page ...

DESCRIPTION
       man  is  the  system's manual pager.  Each page argument given to man is normally the name of a
       program, utility or function.  The manual page associated with each of these arguments is  then
       found  and  displayed.  A section, if provided, will direct man to look only in that section of
       the manual.  The default action is to search in all of the available sections following a  pre-
       defined  order  (see  DEFAULTS),  and to show only the first page found, even if page exists in
       several sections.

       The table below shows the section numbers of the manual followed by the  types  of  pages  they
       contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions, e.g. /etc/passwd
       6   Games
       7   Miscellaneous   (including   macro   packages  and  conventions),  e.g.  man(7),  groff(7),
           man-pages(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]
```

# Getting Help in the Command Line

```
SEE ALSO
       kill(1), alarm(2), kill(2), pause(2), sigaction(2), signalfd(2), sigpending(2), sigprocmask(2), sigsus□
       pend(2), bsd_signal(3), killpg(3), raise(3), siginterrupt(3), sigqueue(3), sigsetops(3), sigvec(3),
       sysv_signal(3), signal(7)
```

```
raju@raju-VirtualBox:~$ whatis ls
ls (1)               - list directory contents
raju@raju-VirtualBox:~$ whatis signal
signal (2)           - ANSI C signal handling
signal (7)           - overview of signals
raju@raju-VirtualBox:~$ whatis msgget
msgget (2)           - get a System V message queue identifier
raju@raju-VirtualBox:~$ whatis mq_send
mq_send (3)          - send a message to a message queue
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ man -k pthread | more
pthread_attr_destroy (3) - initialize and destroy thread attributes object
pthread_attr_getaffinity_np (3) - set/get CPU affinity attribute in thread attributes object
pthread_attr_getdetachstate (3) - set/get detach state attribute in thread attributes object
pthread_attr_getguardsize (3) - set/get guard size attribute in thread attributes object
```

# Getting Help in the Command Line

**apropos** - search the manual page names and descriptions

```
raju@raju-VirtualBox:~$ apropos man
HEAD (1p)             - Simple command line user agent
UPower (7)            - System-wide Power Management
aa-features-abi (1)   - Extract, validate and manipulate AppArmor feature abis
accessdb (8)          - dumps the content of a man-db database in a human readable format
aconnect (1)          - ALSA sequencer connection manager
add_key (2)           - add a key to the kernel's key management facility
alsa-info (8)         - command-line utility to gather information about the ALSA subsystem
alsabat (1)           - command-line sound tester for ALSA sound card driver
alsactl_init (7)      - alsa control management - initialization
alsaloop (1)          - command-line PCM loopback
alsaucm (1)           - ALSA Use Case Manager
amixer (1)            - command-line mixer for ALSA soundcard driver
anacron (8)           - runs commands periodically
apgbfm (1)            - APG Bloom filter management program
aplay (1)             - command-line sound recorder and player for ALSA soundcard driver
apropos (1)           - search the manual page names and descriptions
apt (8)               - command-line interface
apt-cdrom (8)         - APT CD-ROM management utility
apt-get (8)           - APT package handling utility -- command-line interface
apt-key (8)           - Deprecated APT key management utility
apt-patterns (7)      - Syntax and semantics of apt search patterns
aptd (1)              - package managing daemon proving a D-Bus interface
aptdcon (1)           - command line client for aptdaemon
arecord (1)           - command-line sound recorder and player for ALSA soundcard driver
arp (8)               - manipulate the system ARP cache
asn1parse (1ssl)      - OpenSSL application commands
axfer (1)             - command-line sound recorder and player for sound devices and nodes supported by
bash-builtins (7)     - bash built-in commands, see bash(1)
blkzone (8)           - run zone command on a device
blockdev (8)          - call block device ioctls from the command line
```

# Working in a Shell Efficiently

There are many ways you can work in the shell fast and efficiently

1. !<few char> -will execute the last command you have executed. (ex: !vim, !gcc, !cd.,)
2. !! –will execute the last commands
3. Ctrl+r –will prompt for commands you want to search: (reverse-i-search)`': for example if I give input as ls, it will display the last command starting with ls : (reverse-i-search)`ls': ls –Rl
4. In the shell prompt you can also use upper arrow key to find out the recent commands in LIFO.
5. **.bashrc** file in a user home directory will execute given commands/scripts when a user log in. You can keep all the necessary commands when you want to execute after log in. for example: start services, check system load etc.,
6. **.bash_logout** is the individual login shell cleanup file, the given commands will execute when a login shell exit. Ex: You can shut down some of the unwanted services.

```
raju@raju-VirtualBox:~$ ll .bashrc
-rw-r--r-- 1 raju raju 3792 Oct  5 15:58 .bashrc
raju@raju-VirtualBox:~$ ll .bash_logout
-rw-r--r-- 1 raju raju 220 Sep 26 10:56 .bash_logout
```

# Working in the Shell Efficiently

7. **cd -** : command will take you to the previous directory instead of typing the absolute/relative path names.
   In the below example instead of typing cd /home/raju/LF/Lesson_02 or cd ../LF/Lesson_02, you can just type cd -

```
raju@Ubuntu:~/LF/Lesson_02$ pwd
/home/raju/LF/Lesson_02
raju@Ubuntu:~/LF/Lesson_02$ cd ../Lesson_05
raju@Ubuntu:~/LF/Lesson_05$ pwd
/home/raju/LF/Lesson_05
raju@Ubuntu:~/LF/Lesson_05$ cd -
/home/raju/LF/Lesson_02
raju@Ubuntu:~/LF/Lesson_02$
```

8. **cd and cd ~:** if you are in different directory, cd command will take you to the home directory directly.
   cd ~/ is equal to cd /home/raju/ in our example.

```
raju@Ubuntu:~/LF/Lesson_02$ cd
raju@Ubuntu:~$ pwd
/home/raju
raju@Ubuntu:~$ cd LF/Lesson_05
raju@Ubuntu:~/LF/Lesson_05$ cd ~/LF/Lesson_02
raju@Ubuntu:~/LF/Lesson_02$ pwd
/home/raju/LF/Lesson_02
raju@Ubuntu:~/LF/Lesson_02$
```

# Working in the Shell Efficiently

9. **alias** – short cut for a user defined command. In the below example I can execute ***release*** instead of "lsb_release –a".

```
raju@Ubuntu:~/LF/Lesson_02$ alias release='lsb_release -a'
raju@Ubuntu:~/LF/Lesson_02$ release
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.2 LTS
Release:        22.04
Codename:       jammy
raju@Ubuntu:~/LF/Lesson_02$
```

10. **history** – will list the recently executed command list in FIFO order.

```
raju@Ubuntu:~/LF/Lesson_02$ history | tail
  187  cd LF/Lesson_05
  188  cd ~/LF/Lesson_02
  189  pwd
  190  lsb_release -a
  191  alias
  192  alias='lsb_release -a'
  193  alias
  194  alias release='lsb_release -a'
  195  release
  196  history | tail
raju@Ubuntu:~/LF/Lesson_02$
```

# Streams, Redirections and Pipes

- Streams is used for communication services in Linux System. It defines interfaces for standard input, standard output and standard error.

- 0 - Standard input - from the keyboard or normal source of input for the program
- 1 - Standard output - display on the terminal or output from the program
- 2 - Standard error - used for error messages issued by the program

- 0,1, and 2 are the standard I/O file descriptors.

- Stream are used along with re directional operators and they are essential part of working at the command line.

- > -send output to the given file; > - give input to the specific command; 2> -send error message to the given file; >> - append the file with the output.

# Streams, Redirections and Pipes

Example for Streams and redirectional operators:

- **ls –l > file_list** – it will send the output of ls –l command into file_list instead of displaying on the terminal.
- **wc –l < file_list** – we are giving input file as file_list for the wc –l command
- **ls –Rl >> file_list** – it will append the file_list file with the output of ls –Rl command
- **Test.sh 2 > error.txt** – it will store only error messages into the error.txt file.

# Streams, Redirections and Pipes

- **2>&1** -send error messages to where ever output is being redirected.
- In the below example we redirect echo output to file1.txt and file2.txt.
- The cat output of file1.txt and file2.txt is sent to output file.
- When we add file3.txt into the cat command, it displayed an error messages into the terminal since the file3.txt is not in the directory.
- **cat file1.txt file2.txt file3.txt > output 2>&1** – it has redirected the file1.txt, file2.txt output and along with the error messages into the output file.

```
raju@Ubuntu:~/LF/Lesson_02$ echo "How are you" > file1.txt
raju@Ubuntu:~/LF/Lesson_02$ echo "I am fine. " > file2.txt
raju@Ubuntu:~/LF/Lesson_02$ cat file1.txt file2.txt > output
raju@Ubuntu:~/LF/Lesson_02$ cat output
How are you
I am fine.
raju@Ubuntu:~/LF/Lesson_02$ cat file1.txt file2.txt file3.txt > output
cat: file3.txt: No such file or directory
raju@Ubuntu:~/LF/Lesson_02$ cat file1.txt file2.txt file3.txt > output 2>&1
raju@Ubuntu:~/LF/Lesson_02$ cat output
How are you
I am fine.
cat: file3.txt: No such file or directory
raju@Ubuntu:~/LF/Lesson_02$
```

# Streams, Redirections and Pipes

Pipe is used to work with two or more commands. The output of the first command acts as the input to the second command.

The symbol "**|**" is the Unix pipe symbol that is used on the command line.

***Example:***

**ls -l | grep ^d** – will display only directories. **grep ^d** will filter only the first character in all lines start with "d" from the ls -l output.

**ls -l | grep ^d | wc -l** – will find out the total number of directories in the ls -l output.

```
raju@Ubuntu:~/LF$ ls -l
total 44
-rw-rw-r-- 1 raju raju    92 Jul  3 10:45 a.c
-rwxrwxr-x 1 raju raju 14472 Jul  3 10:46 a.out
-rw-rw-r-- 1 raju raju  1812 Jul  3 10:18 copy.txt
drwxrwxr-x 2 raju raju  4096 Jul  3 10:48 Lesson_01
drwxrwxr-x 2 raju raju  4096 Jul  3 11:35 Lesson_02
drwxrwxr-x 2 raju raju  4096 Jul  3 10:49 Lesson_03
drwxrwxr-x 2 raju raju  4096 Jul  3 10:49 Lesson_04
drwxrwxr-x 2 raju raju  4096 Jul  3 10:49 Lesson_05
raju@Ubuntu:~/LF$ ls -l | grep ^d
drwxrwxr-x 2 raju raju  4096 Jul  3 10:48 Lesson_01
drwxrwxr-x 2 raju raju  4096 Jul  3 11:35 Lesson_02
drwxrwxr-x 2 raju raju  4096 Jul  3 10:49 Lesson_03
drwxrwxr-x 2 raju raju  4096 Jul  3 10:49 Lesson_04
drwxrwxr-x 2 raju raju  4096 Jul  3 10:49 Lesson_05
raju@Ubuntu:~/LF$ ls -l | grep ^d | wc
      5      45     260
raju@Ubuntu:~/LF$ ls -l | grep ^d | wc -l
5
raju@Ubuntu:~/LF$
```

# Grep - Global Regular Expression Print

- The basic usage of **grep** command is to search for a specific string in the specified file.
- The syntax for the **grep** command is: ***grep [options] pattern [files]***

| Option | |
|---|---|
| -c | Display the number of matched lines. |
| -i | Ignore case sensitivity. |
| -l | Display the filenames, but do not display the matched lines. |
| -n | Display the matched lines and their line numbers. |
| -v | Display all lines that do NOT match. |

# grep

Let us take an example greptest.txt file. The contents of the file can be seen by **cat greptest.txt** command. –n option will print the file output with line numbers.

```
raju@Ubuntu:~/LF/Lesson_02$ cat -n greptest.txt
     1  Line 0: This is a test file.
     2  Line 1: This text file contains some erors that need to be fixed.
     3  Line 2: Not all the lines are containing erors.
     4  Line 3: I hope you can spot the Erors and correct them.
     5  Line 4: Please ignore any grammatical or spelling errors you find.
     6  Line 5: It's important to review and correct any Errors before submitting.
raju@Ubuntu:~/LF/Lesson_02$
```

**$grep -i erors greptest.txt** - will search the pattern ***erors*** (also ERORS, Erors) and display the lines matching the pattern.

```
raju@Ubuntu:~/LF/Lesson_02$ grep -i erors greptest.txt
Line 1: This text file contains some erors that need to be fixed.
Line 2: Not all the lines are containing erors.
Line 3: I hope you can spot the Erors and correct them.
raju@Ubuntu:~/LF/Lesson_02$
```

# grep

**$grep -w erors greptest.txt** - search the pattern ***erors*** (exact word) and display the lines matching the pattern. The –n option will print the line number.

```
raju@Ubuntu:~/LF/Lesson_02$ grep -w erors greptest.txt
Line 1: This text file contains some erors that need to be fixed.
Line 2: Not all the lines are containing erors.
raju@Ubuntu:~/LF/Lesson_02$ grep -w -n erors greptest.txt
2:Line 1: This text file contains some erors that need to be fixed.
3:Line 2: Not all the lines are containing erors.
raju@Ubuntu:~/LF/Lesson_02$
```

**$grep -c erors greptest.txt** - search the pattern ***erors*** (exact word) and display the total number matching lines.

```
raju@Ubuntu:~/LF/Lesson_02$ grep -c erors greptest.txt
2
raju@Ubuntu:~/LF/Lesson_02$
```

# grep

**$grep -l ends greptest.txt** - search the pattern ***ends*** and print all the file names, which are matching the pattern.

```
raju@Ubuntu:~/LF/Lesson_02$ grep -l are *
file1.txt
greptest.txt
output
raju@Ubuntu:~/LF/Lesson_02$
```

**$grep -v erors greptest.txt** - - print the lines that are not matched with the pattern.

```
raju@Ubuntu:~/LF/Lesson_02$ grep -v erors greptest.txt
Line 0: This is a test file.
Line 3: I hope you can spot the Erors and correct them.
Line 4: Please ignore any grammatical or spelling errors you find.
Line 5: It's important to review and correct any Errors before submitting.
raju@Ubuntu:~/LF/Lesson_02$
```

**$grep ^This greptest.txt** - - print the lines, which start with the given pattern.

```
raju@Ubuntu:~/LF/Lesson_02$ grep ^Lin greptest.txt
Line 0: This is a test file.
Line 1: This text file contains some erors that need to be fixed.
Line 2: Not all the lines are containing erors.
Line 3: I hope you can spot the Erors and correct them.
Line 4: Please ignore any grammatical or spelling errors you find.
Line 5: It's important to review and correct any Errors before submitting.
raju@Ubuntu:~/LF/Lesson_02$
```

# grep

**$grep text.$ greptest.txt** - - print the lines, which end with the given pattern.

```
raju@Ubuntu:~/LF/Lesson_02$ grep find. greptest.txt
Line 4: Please ignore any grammatical or spelling errors you find.
raju@Ubuntu:~/LF/Lesson_02$
```

**$grep –iR –n erors <dir> greptest.txt** - Read all files under each directory recursively and print the searched pattern with file name.

```
raju@Ubuntu:~/LF/Lesson_02$ grep -iR -n are ./
./file1.txt:1:How are you
./greptest.txt:3:Line 2: Not all the lines are containing erors.
./output:1:How are you
raju@Ubuntu:~/LF/Lesson_02$
```

# grep

**Context Line control:**
-A NUM -print NUM lines of trailing context after matching lines.
-B NUM -print NUM lines of leading context before matching lines.
-C NUM -print NUM lines before and after matching lines.

**$grep –A1 erors greptest.txt** - print one line after matching lines.
**$grep –B1 erors greptest.txt** - print one line before matching lines.
**$grep –A1 erors greptest.txt** - print one line before and after matching lines.

```
raju@Ubuntu:~/LF/Lesson_02$ grep -A1 erors greptest.txt
Line 1: This text file contains some erors that need to be fixed.
Line 2: Not all the lines are containing erors.
Line 3: I hope you can spot the Erors and correct them.
raju@Ubuntu:~/LF/Lesson_02$ grep -B1 erors greptest.txt
Line 0: This is a test file.
Line 1: This text file contains some erors that need to be fixed.
Line 2: Not all the lines are containing erors.
raju@Ubuntu:~/LF/Lesson_02$ grep -C1 erors greptest.txt
Line 0: This is a test file.
Line 1: This text file contains some erors that need to be fixed.
Line 2: Not all the lines are containing erors.
Line 3: I hope you can spot the Erors and correct them.
raju@Ubuntu:~/LF/Lesson_02$
```

# grep

**$ grep -e Errors -e ends greptest.txt** – search and print all the patterns given.

```
raju@Ubuntu:~/LF/Lesson_02$ grep -e Errors -e find greptest.txt
Line 4: Please ignore any grammatical or spelling errors you find.
Line 5: It's important to review and correct any Errors before submitting.
raju@Ubuntu:~/LF/Lesson_02$
```

# SED – Stream EDitor

- A Stream EDitor is used to perform basic transformations on text read from a file or a pipe.
- The editor does not modify the original input.
- If you are facing replacement of text in a large number of files, **sed** is a great help.
- The **sed** program can perform text pattern substitutions and deletions using regular expressions
- **Sed editing commands :**
  - a\ - Append text below current line.
  - c\ - Change text in the current line with new text.
  - d - Delete text.
  - i\ - Insert text above current line.
  - p - Print text.
  - r - Read a file.
  - s - Search and replace text.
  - w - Write to a file

**Input file: test.txt**

This file has many lines in the test.txt text file.
This file has lot of erors in the text.
Because of these erors, the file is not readable.
Not all the lines are containing erors.
These lines are not containing any errors.
The file ends here.

# SED – Stream EDitor

1. To find all the lines containing our search pattern, in this case "erors". We use the **p** to obtain the result.

**$sed –n *'/erors/p'* test.txt**

**2.** Now we only want to see the lines *not* containing the search string

**$sed *'/erors/d'* test.txt**

**3.** Matching lines starting with a given pattern and ending in a second pattern

**$sed -n *'/^These.*errors.$/p'* test.txt**

**4.** we want to take out the lines containing the errors. Specify this

range to address, together with the **d** command

**$sed *'2,4d'* test.txt**

To print the file starting from a certain line until the end of the file

**$sed *'3,$d'* test.txt**

# SED – Stream EDitor

5. The following command prints the first line containing the pattern "a text", up to and including the next line containing the pattern "a line":

**$sed -n *'/a text/,/These/p'* test.txt**

6. search and replace the errors instead of only (de)selecting the lines containing the search string.

**$sed *'s/erors/errors/g'* test.txt**

7. To insert a string at the beginning of each line of a file, for instance for quoting

**$sed *'s/^/> /'* test.txt**

8. Insert some string at the end of each line

**$sed *'s/$/EOL/'* test.txt**

9. Multiple find and replace commands are separated with individual -e options

**$sed -e *'s/erors/errors/g'* -e *'s/ends/close/g'* test.txt**

# AWK (Aho, Weinberger and Kernighan)

The basic function of **awk** is to search files for lines or other text units containing one or more patterns. When a line matches one of the patterns, special actions are performed on that line.

```
# $1 is field #1, $2 is field #2, etc.
echo one two | awk '{print $1}‘      -    # one
echo one two | awk '{print $2}‘      -    # two
echo one two | awk '{print $0}‘      -    # one two

awk '{print $3}' $filename           -    # Prints field #3 of file $filename
awk '{print $1 $5 $6}' $filename     -    # Prints fields #1, #5, and #6 of file    $filename.
awk '{print $0}' $filename           -    # Prints the entire file!
```

# AWK

**$cat personnel.txt**

| | | | | |
|---|---|---|---|---|
| 1234 | Ram | SoftEng | Insurance | ₹80000 |
| 2143 | Gopal | TechLead | Banking | ₹90000 |
| 2332 | Verma | TeamLead | Embedded | ₹100000 |
| 4554 | Krish | Manager | SalesDept | ₹70000 |
| 6754 | Siva | SysAdmin | IT_Dept | ₹60000 |
| 1356 | John | Architect | Banking | ₹200000 |

**By default Awk prints every line from the file.**

$ awk '{print;}' personnel.txt

**Print the lines which matches with the pattern**

awk '/Gopal/
> /John/' personnel.txt

**Print only specific field**

$ awk '{print $2,$5;}' personnel.txt (awk ‘{print $2, “\t”, $5;}’ personnel.txt)

$ awk '{print $2,"\t",$NF;}' personnel.txt
where $NF represents last field

# AWK

Awk has two important patterns which are specified by the keyword called BEGIN and END.

Syntax: BEGIN { Actions} {ACTION} END { Actions }

```
$ awk 'BEGIN {print "Name\tDesignation\tDepartment\tSalary";}
{print $2,"\t",$3,"\t",$4,"\t",$NF;}
END {print "Report Generated\n--------------";
 }' personnel.txt
```

**Find the employees who has employee id greater than 200**

```
$ awk '$1 >3000' personnel.txt
```

**Print the list of employees in Technology department**

```
$ awk '$4 ~/Banking/' personnel.txt
```

**Print number of employees in Technology department**

```
$ awk 'BEGIN { count=0;}
$4 ~ /Banking/ { count++; }
END { print "Number of employees in Technology Dept =",count;}'
personnel.txt
Number of employees in Technology Dept = 3
```

# Lesson 3 – Working With Files

- Navigating the Linux Filesystem
- Different File Types
- Searching for Files
- Working With File Permissions
- Setuid, Setgid and Sticky bit
- Working With Users and Groups
- Introducing Acl
- vim and nano text editors

# Navigating the Linux File System

1. **ls** - list directory contents
   options:
   -l use a long listing format
   -a do not ignore entries starting with . (hidden files)
   -d list directories themselves, not their contents (ls –ld)
   -h human readable format (ls –lh)
   -i print the index number (inode) of each file (ls –il)
   -R list subdirectories recursively (ls –Rl)
   -S sort by file size, largest first (ls –lS)

2. **touch** - create empty files. we can create multiple empty files by an appropriate command.
   example: $touch abc.txt; $touch abc{1..100}.txt

3. **cd [DIR]** - Change the shell working directory. Change the current directory to DIR.
   options:
   . -the current directory
   .. - one level above the current directory
   / -root directory
   ~ -home directory (/home/<user>)

# Navigating the Linux File System

4. **pwd** - print name of current/working directory

5. **tac** - print file in reverse order (opposite to cat command)

6. **cp** - copy files and directories
   options:
   -i, --interactive. Prompt before overwrite.
   -f, --force
   -R, -r, --recursive. Copy directories recursively (ex: cp -r ./dir1 ./dir2 or cp -R ./dir1 ./dir2)

7. **mv** - move (rename) files

8. **rm** - remove files or directories
   options:
   -f, --force. Never prompt
   -i prompt before every removal
   -r, -R, --recursive. Remove directories and their contents recursively (ex: rm -r ./dir1 - remove all the files in dir1).
   -d Remove empty directories (ex: rm –d ./dir1 –only if the dir1 is empty).

# Navigating the Linux File System

9. **mkdir** - make directories (ex: $mkdir <newdir>; $mkdir <dir1> <dir2> <dir3>).

10. **rmdir** - remove empty directories (ex: $rmdir <dir1>; $rmdir <dir1> <dir2> <dir3>).

11. **df** - report file system disk space usage.

```
raju@Ubuntu:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           794M  1.6M  793M   1% /run
/dev/sda3        24G   14G  9.5G  59% /
tmpfs           3.9G     0  3.9G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
/dev/sda2       512M  6.1M  506M   2% /boot/efi
tmpfs           794M  2.4M  792M   1% /run/user/1000
/dev/sr0         51M   51M     0 100% /media/raju/VBox_GAs_7.0.4
raju@Ubuntu:~$
```

12. **du** - estimate file space usage

```
raju@Ubuntu:~/LINUX/hands-onUSP$ du -h
52K     ./day3
76K     ./day4
68K     ./day2
64K     ./day1
264K    .
raju@Ubuntu:~/LINUX/hands-onUSP$
```

# Navigating the Linux File System

13. **head** - Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name (ex: $head <file name>; $head <file1> <file2>).
    option: -NUM - print the first NUM lines instead of the first 10 (ex: $head -3 <file1>).

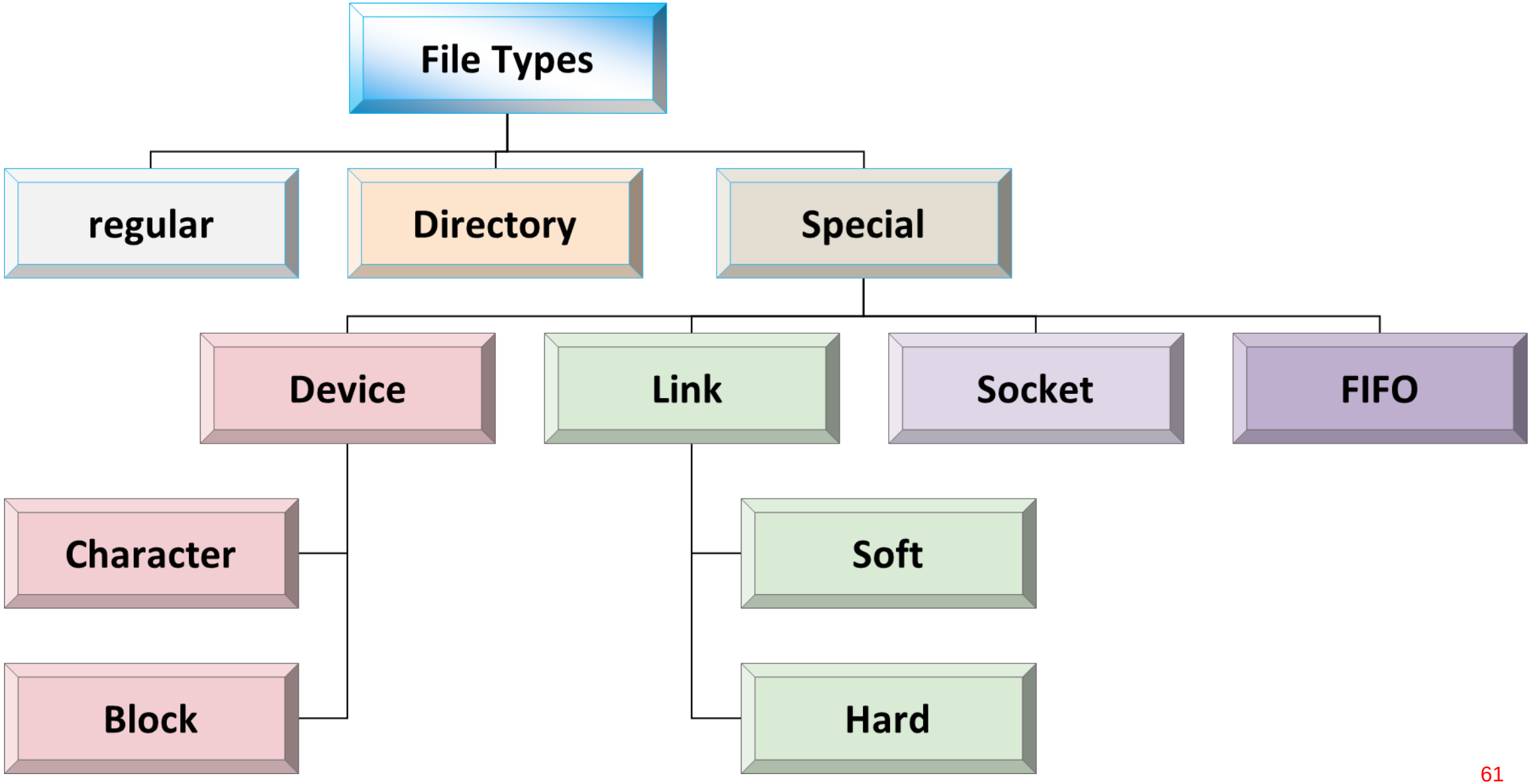
14. **tail** - Print the last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name (ex: $tail <file name>; $tail <file1> <file2>).
    option: -NUM - print the last NUM lines instead of the first 10 (ex: $tail -3 <file1>).

15. **more** - paging through text one screenfull at a time ($more <file name>).
    option: -n - Specify the number of lines per screenfull (ex: $more -3 <file1>).

16. **file** - determine file type.

```
raju@Ubuntu:~/LF/Lesson_03$ file ../*
../a.c:       C source, ASCII text
../a.out:     ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib6
ldID[sha1]=7fd9bec22828316c7e09dd06983b98d610dba97e, for GNU/Linux 3.2.0, stripped
../copy.txt:  ASCII text
../Lesson_01: directory
../Lesson_02: directory
../Lesson_03: directory
../Lesson_04: directory
../Lesson_05: directory
raju@Ubuntu:~/LF/Lesson_03$
```

# File Types


File Types
regular
Directory
Special
Device
Link
Socket
FIFO
Character
Block
Soft
Hard

# File Types

## 16. Creation of different file types:

- Regular file: ***$ls -l > regular***
- Directory file: ***$mkdir directory***
- Fifo File (pipe) : ***$mkfifo FifoFile***
- Hard Link file : ***$ln regular hrdlnk***
- Soft Link file : ***$ln -s regular sftlnk***
- Character Device file: ***$mknod char c 250 0***
- Block Device file: ***$mknod block b 250 0***

```
1220413 brw-r--r--  1 root root 250,  0 Dec  3 19:03 block
1220412 crw-r--r--  1 root root 250,  0 Dec  3 19:03 char
1219892 drwxrwxr-x  2 raju raju   4096 Dec  3 19:02 directory/
1220410 prw-rw-r--  1 raju raju      0 Dec  3 19:02 FifoFile|
1217897 -rw-rw-r--  2 raju raju    149 Dec  3 19:01 hrdlnk
1217897 -rw-rw-r--  2 raju raju    149 Dec  3 19:01 regular
1220411 lrwxrwxrwx  1 raju raju      7 Dec  3 19:03 sftlnk -> regular
```

# find command

17. **diff** - compare files line by line (ex: diff <file1> <file2>).

18. **cmp** - compare two files byte by byte (ex: cmp <file1> <file2>).

19. **find** - search for files in a directory hierarchy.

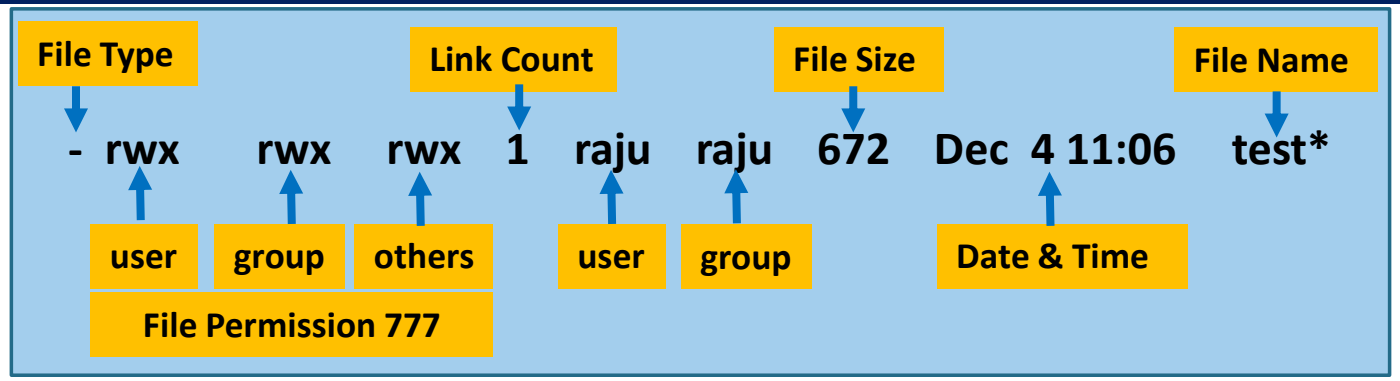
    example:

- ***$find ./ –name <file name>***
- ***$find ./LF/ -name *.txt*** (find all the txt files).
- ***$find ./ -name example -exec rm -i {} \;*** ( -exec will execute a given command)
  rm: remove regular file './example'?
- ***$find ./ -empty*** (find all empty directories and files in the entered directory or sub-directories).
- ***$find ./ -type f -empty*** (find all empty files in the current directory).
- ***$find ./ -type d -empty*** (find all empty directories in the current directory).
- ***$find ./ -perm 0775*** (find all the files in the PWD directory or sub-directory with the 0775 permissions.

# find command

- ***$find ./ -type f -name "*.txt" -exec grep 'erors' {} \;***
  (print lines which have ‘erors’ in them and ‘-type f’ specifies the input type is a file).
- ***$find ./ -type d -name temp*** (Find all directories whose name is **temp** in ./ directory).
- ***$find ./ -type f -name "*.txt“*** (Find all txt files in the current directory).
- ***$find ./ -user user1*** (find all files that belong to user **user1** in pwd).
- ***$find ./ -size 16k*** (find all files 16kb size files).
- ***$find ./ -size +2k -size -17k*** (find all files size is in between 2 to 17kb).
- ***$find ./ -type f -size +16 -exec rm -i {} \;*** (find all **>16kb** files and delete them).

# File Permission




20. **chmod** - change file mode bits (change file permission)

$chmod 711 test

-rwx--x--x 1 raju raju 672 Dec 4 11:06 test*

$chmod 744 test

-rwxr--r-- 1 raju raju 672 Dec 4 11:06 test*

$chmod 751 test

-rwxr-x--x 1 raju raju 672 Dec 4 11:06 test*

| Octal | 4 | 2 | 1 |
|---|---|---|---|
| | r | w | x |
| 0 | - | - | - |
| 1 | - | - | X |
| 2 | - | X | - |
| 3 | - | X | X |
| 4 | X | - | - |
| 5 | X | - | X |
| 6 | X | X | - |
| 7 | X | X | X |

Default permission for new a file is set by umask. $**umask** will display or set file mode mask.

-rwxrwsr-x

# Setuid, Setgid and Sticky bit

**Setuid**: changes uid, then executes command. We are able to execute root user *passwd* executable file to change our password because the setuid bit is set in the /usr/bin/passwd file.
-rwsr-xr-x 1 root root 59976 Nov 24 17:35 /usr/bin/passwd*

If you want to set setuid bit for your executable file a.out:
-rwxrwxr-x 1 raju raju 15776 Dec  3 18:34 a.out
21. $chmod u+s a.out (to set the setuid bit)
-rwsrwxr-x 1 raju raju 15776 Dec  3 18:34 a.out* ( you can see x is replaced with s: -rwsrwxr-x )
$chmod u-s a.out (to remove the setuid bit)
-rwxrwxr-x 1 raju raju 15776 Dec  3 18:34 a.out*
**22. Setgid**: $chmod g+s a.out and $chmod g-s a.out – will set (-rwxrwsr-x) and remove setgid correspondingly .

**Sticky bit**: restricted deletion flag.
If you set a sticky bit in a directory then its files can be deleted or renamed only by the owner and root.
Sticky bit can be set by using the command: `chmod +t`
drwxrwxr-x 4 raju raju 4096 Dec  3 18:51 temp
23. $chmod +t temp
drwxrwxr-t 4 raju raju 4096 Dec  3 18:51 temp ( you can see x is replaced with t for others: drwxrwxr-t ).
$chmod -t temp (to remove sticky bit)
drwxrwxr-x 4 raju raju 4096 Dec  3 18:51 temp

# User Management

**useradd** is a low level utility for adding users. On Debian, administrators should usually use **adduser**(8) instead.

```
raju@Ubuntu:~/LF/Lesson_03$ ls -l /home
total 4
drwxr-x--- 22 raju raju 4096 Jul  3 10:45 raju
raju@Ubuntu:~/LF/Lesson_03$
raju@Ubuntu:~/LF/Lesson_03$ sudo adduser user2
[sudo] password for raju:
Adding user `user2' ...
Adding new group `user2' (1001) ...
Adding new user `user2' (1001) with group `user2' ...
Creating home directory `/home/user2' ...
Copying files from `/etc/skel' ...
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
Changing the user information for user2
Enter the new value, or press ENTER for the default
        Full Name []: Thangaraju
        Room Number []: 133
        Work Phone []: 9900541576
        Home Phone []: 9900541576
        Other []: NA
Is the information correct? [Y/n] y
raju@Ubuntu:~/LF/Lesson_03$
```

```
raju@Ubuntu:~/LF/Lesson_03$ ls -l /home
total 8
drwxr-x--- 22 raju  raju  4096 Jul  3 10:45 raju
drwxr-x---  2 user2 user2 4096 Jul  3 10:53 user2
raju@Ubuntu:~/LF/Lesson_03$
```

# User Management

$**whoami** –print the user name (login id).
$**su – user2** (***su*** switch to user; **-** changes to the target user's home directory.

```
raju@Ubuntu:~/LF/Lesson_03$ whoami
raju
raju@Ubuntu:~/LF/Lesson_03$ sudo su - user2
user2@Ubuntu:~$ whoami
user2
user2@Ubuntu:~$ pwd
/home/user2
user2@Ubuntu:~$ sudo apt update
[sudo] password for user2:
user2 is not in the sudoers file.  This incident will be reported.
user2@Ubuntu:~$
```

***usermod -aG sudo user2*** (-***aG*** append the user2 in the ***sudo*** group)

```
raju@Ubuntu:~/LF/Lesson_03$ sudo usermod -aG sudo user2
raju@Ubuntu:~/LF/Lesson_03$ su - user2
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

user2@Ubuntu:~$ sudo apt update
[sudo] password for user2:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
```

# User Management

Now we can verify sudo user and normal user with id command:
User1 is a normal user and user2 is a sudo user.

```
user1@raju-VirtualBox:~$ id
uid=1001(user1) gid=1001(user1) groups=1001(user1)
user1@raju-VirtualBox:~$
```

```
user2@raju-VirtualBox:~$ id
uid=1002(user2) gid=1002(user2) groups=1002(user2),27(sudo)
user2@raju-VirtualBox:~$
```

$**passwd** - change user password

```
user2@raju-VirtualBox:~$ passwd
Changing password for user2.
Current password:
New password:
Retype new password:
passwd: password updated successfully
user2@raju-VirtualBox:~$
```

# User Management

**chown** - change file owner.

```
user2@raju-VirtualBox:~$ ls -l
total 0
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc1.mp3
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc2.mp3
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc3.mp3
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc4.mp3
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc5.mp3
user2@raju-VirtualBox:~$ sudo chown user1 abc3.mp3 abc4.mp3
[sudo] password for user2:
user2@raju-VirtualBox:~$ ls -l
total 0
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc1.mp3
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc2.mp3
-rw-rw-r-- 1 user1 user2 0 Dec  5 09:49 abc3.mp3
-rw-rw-r-- 1 user1 user2 0 Dec  5 09:49 abc4.mp3
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc5.mp3
user2@raju-VirtualBox:~$
```

**chgrp** - change group ownership

```
user2@raju-VirtualBox:~$ ls -l abc1.mp3
-rw-rw-r-- 1 user2 user2 0 Dec  5 09:49 abc1.mp3
user2@raju-VirtualBox:~$  sudo chgrp raju abc1.mp3
user2@raju-VirtualBox:~$ ls -l abc1.mp3
-rw-rw-r-- 1 user2 raju 0 Dec  5 09:49 abc1.mp3
user2@raju-VirtualBox:~$
```

# User Management

***$deluser*** - remove a user from the system. Option ***--remove-home*** - Remove the home directory of the user.

```
raju@Ubuntu:~/LF/Lesson_03$ ls -l /home
total 8
drwxr-x--- 22 raju  raju  4096 Jul  3 10:45 raju
drwxr-x---  2 user2 user2 4096 Jul  3 11:00 user2
raju@Ubuntu:~/LF/Lesson_03$ sudo deluser --remove-home user2
Looking for files to backup/remove ...
Removing files ...
Removing user `user2' ...
Warning: group `user2' has no more members.
Done.
raju@Ubuntu:~/LF/Lesson_03$
raju@Ubuntu:~/LF/Lesson_03$ ls -l /home
total 4
drwxr-x--- 22 raju raju 4096 Jul  3 10:45 raju
raju@Ubuntu:~/LF/Lesson_03$
```

# Access Control Lists (acl)

acl — POSIX Access Control Lists, which are used to define more specific set of access rights for files and directories.
ACL is used to secure business sensitive information from unauthorized users.

We have two users: raju and user1 and two directories: /confidential and /sales; root is the owner of these two directories

```
drwxr-x--- 27 raju  raju  4096 Dec  5 18:23 raju
drwxr-x---  2 user1 user1 4096 Dec  6 06:18 user1
```

```
/confidential:
total 4
-rw-r--r-- 1 root root 380 Dec  6 06:48 finance
```

```
/sales:
total 4
-rw-r--r-- 1 root root 615 Dec  6 06:49 emp_details.txt
```

Requirement: we have to give rwx permission to raju user and only r-x permission to user1 to access /confidential directory
Both the users raju and user1 will get rxw permission to access /sales directory.

**getfacl** - get file access control lists.
**setfacl** - set file access control lists.

# Access Control Lists (acl)

**Set access permission for the users**

```
root@raju-VirtualBox:/# setfacl -m raju:rwx /confidential
root@raju-VirtualBox:/# setfacl -m user1:r-x /confidential
root@raju-VirtualBox:/# setfacl -m user1:rwx /sales
root@raju-VirtualBox:/# setfacl -m raju:rwx /sales
```

**Default acl for the directories**

```
root@raju-VirtualBox:/# getfacl confidential
# file: confidential
# owner: root
# group: root
user::rwx
group::r-x
other::r-x

root@raju-VirtualBox:/# getfacl sales
# file: sales
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
```

```
root@raju-VirtualBox:/# getfacl confidential sales
# file: confidential
# owner: root
# group: root
user::rwx
user:raju:rwx
user:user1:r-x
group::r-x
mask::rwx
other::r-x

# file: sales
# owner: root
# group: root
user::rwx
user:raju:rwx
user:user1:rwx
group::r-x
mask::rwx
other::r-x
```

# Access Control Lists (acl)

**Switch to *raju* user; access /confidential directory and files. The user is allowed to create files.**

```
root@raju-VirtualBox:/# su - raju
raju@raju-VirtualBox:~$ cd /confidential
raju@raju-VirtualBox:/confidential$ ls
finance
raju@raju-VirtualBox:/confidential$ more finance
Emp.No  Name    Role          Account        Salary
1234    Ram     SoftEng       Insurance      ₹80000
2143    Gopal   TechLead      Banking        ₹90000
2332    Verma   TeamLead      Embedded       ₹100000
4554    Krish   Manager       SalesDept      ₹70000
6754    Siva    SysAdmin      IT_Dept        ₹60000
1356    John    Architect     Banking        ₹200000

raju@raju-VirtualBox:/confidential$ touch newfile
raju@raju-VirtualBox:/confidential$ ls
finance  newfile
raju@raju-VirtualBox:/confidential$
```

**Switch to *user1* user; access /confidential directory and files. The user is *not* allowed to create files.**

```
root@raju-VirtualBox:/# su - user1
user1@raju-VirtualBox:~$ cd /confidential
user1@raju-VirtualBox:/confidential$ ls
finance  newfile
user1@raju-VirtualBox:/confidential$ more finance
Emp.No  Name    Role          Account        Salary
1234    Ram     SoftEng       Insurance      ₹80000
2143    Gopal   TechLead      Banking        ₹90000
2332    Verma   TeamLead      Embedded       ₹100000
4554    Krish   Manager       SalesDept      ₹70000
6754    Siva    SysAdmin      IT_Dept        ₹60000
1356    John    Architect     Banking        ₹200000

user1@raju-VirtualBox:/confidential$ touch testfile
touch: cannot touch 'testfile': Permission denied
user1@raju-VirtualBox:/confidential$
```

***user1* is allowed to create files in sales directory.**

```
user1@raju-VirtualBox:~$ cd /sales
user1@raju-VirtualBox:/sales$ ls
emp_details.txt
user1@raju-VirtualBox:/sales$ touch testfile
user1@raju-VirtualBox:/sales$ ls
emp_details.txt  testfile
user1@raju-VirtualBox:/sales$
```

# vim - Vi IMproved, a programmer's text editor

Vim is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs. $**man vi** –will get more details. $**vimtutor** – tutorial page for vim editor and it has seven lessons.

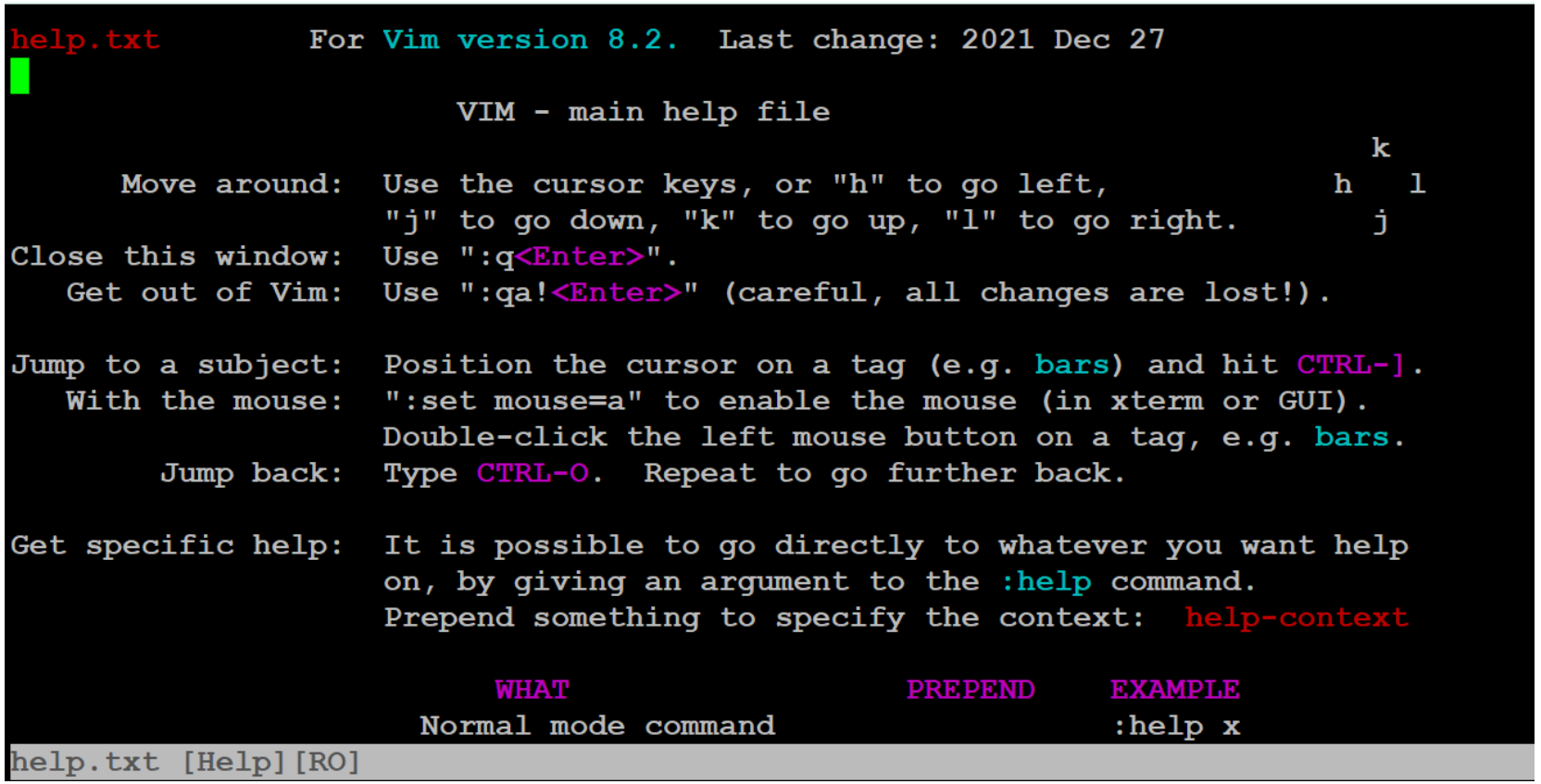
```
raju@Ubuntu:~/LF/Lesson_03$ vim --v
VIM - Vi IMproved 8.2 (2019 Dec 12, compiled Jun 22 2023 04:08:04)
Unknown option argument: "--v"
More info with: "vim -h"
raju@Ubuntu:~/LF/Lesson_03$
```

$vi -will give the following page

```
                VIM - Vi IMproved

                 version 8.2.4919
             by Bram Moolenaar et al.
    Modified by team+vim@tracker.debian.org
  Vim is open source and freely distributable

             Sponsor Vim development!
  type  :help sponsor<Enter>     for information

  type  :q<Enter>                to exit
  type  :help<Enter>  or  <F1>   for on-line help
  type  :help version8<Enter>    for version info
```

# vim - Vi IMproved, a programmer's text editor

:help (inside vim will give basic command details.




**:w** –write changes into the file (save)
**:q** –will exit from the editor **(:q!** –will exit without saving)
**:wq** – save and exit
**:x** –save and exit (**:x!** –exit without saving)

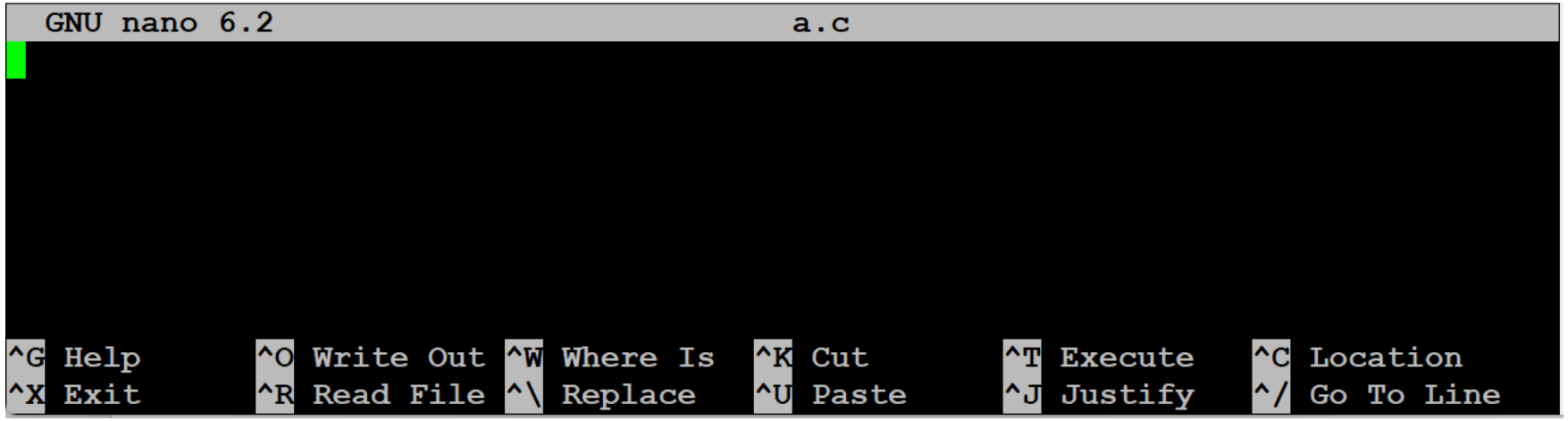
# vim - Vi IMproved, a programmer's text editor

- vim will execute in **two modes**: **command** and **insert**.
- Command mode - user can execute commands.
- Insert mode – enter insert mode by pressing **“I”** or **“a”** key here user can enter or edit text.
- To change from insert to command mode, press **esc** key.
- In the command mode:
  - **i** - change to insert mode and enter text before the current position.
  - **a** - change to insert mode and enter text after the current position.
  - **o** - insert text before the line from the current line.
  - **O** - insert text after the line from the current line.
  - **w** - move from word to word (<n>w for ex: 5w - to move cursor 5 words forward).
  - **$** - move the cursor to end of the line (<n>$ for ex: 5w - to move cursor 5 lines forward).
  - **x** - to delete a character (<n>x for ex: 5w -to delete 5 characters from the cursor position).
  - **dw** - delete a word (d3w -delete 3 words).
  - d**$** -delete the line from the cursor position (d3$ -delete 3 lines).
  - **gg** - move the cursor to beginning of the file.
  - **G** - move the cursor to end of the file.
- **:/word** - to search a word (press n to the next word).
- **:s/erors/errors** - to search the word erors and replace it with errors.
- **:%s/erors/erros/g** - to search the word erors and replace it with errors in the whole file (g for globally).
- **:se nu** -set number in each line.

You can also use arrow keys to move the cursor: front, back, up and down.

# nano Text Editor

**nano** - Nano's ANOther editor, inspired by Pico. nano  is a small and friendly editor. It has several features, such as: opening multiple files, scrolling per line, undo/redo, syntax coloring, line numbering, copy and paste etc.,




$**nano <filename.c>** (it will open existing file for editing or open a new file with a given file ane.
You can work with nano just like notepad editor (since it doesn't have two modes: command & insert)

**^o** - save the file
**^w** -search a keyword
**^** -replace (it will prompt which word you want to replace and prompt to select whether one instance or all).
**Alt+c** -will show the line number
**^k** - cut a line. if you wan to cut more than a line: press ^6 then select the text and press ^k.
**^u** -paste the line(s). **^u** – undo.
**^x** -exit

# Lesson 4

# Essential Linux Commands

- Processes Management
- Signaling Mechanisms
- How to Work With Bash Shell Variables
- Introduction to Bash Shell Scripting
- How to Automate Script Execution

# Linux Process Management

- **Process is a program in execution** for example, if you execute any command, it will become a process.
- **Process is identified by its process id** (pid). Each process has its unique pid.
- **Process Types:**
  - foreground, background, parent, child, orphan and daemon process.
- **Process States:**
  - Running (R), Sleeping (S), Stopped (T) and Zombie (Z).
- **Process Scheduling Priority:**
  - 0-139 process priority range; 0 – 99 real time priority and 100 – 139 normal priority.
- **Viewing running Process:**
  - ps command:  ps displays information about a selection of the active processes.
- **Process Monitoring Tool:**
  - top, mpstat, System Monitor(GUI).

# Process Types

If you execute any program (including command, shell script or java program) then it will be foreground process. User can directly interact with the program. Example:

```
raju@raju-VirtualBox:~/test$ sleep 10
```

If you execute any program  to include an **&** at the end then it will run the process in background, leaving the terminal free for other work. You can view the process information by executing ***ps*** or ***jobs*** command.

$**fg** command will bring the background process into the foreground.

```
raju@raju-VirtualBox:~/test$ sleep 10 &
[1] 583021
raju@raju-VirtualBox:~/test$ ps
    PID TTY          TIME CMD
 490350 pts/0    00:00:00 bash
 583021 pts/0    00:00:00 sleep
 583022 pts/0    00:00:00 ps
raju@raju-VirtualBox:~/test$ jobs
[1]+  Running                 sleep 10 &
raju@raju-VirtualBox:~/test$ fg
sleep 10
```

# Process Types

if you have multiple background processes, by default ***fg*** will bring the last process to the foreground.

```
raju@raju-VirtualBox:~/test$ sleep 101 &
[1] 583131
raju@raju-VirtualBox:~/test$ sleep 102 &
[2] 583132
raju@raju-VirtualBox:~/test$ sleep 103 &
[3] 583133
raju@raju-VirtualBox:~/test$ jobs
[1]   Running                 sleep 101 &
[2]-  Running                 sleep 102 &
[3]+  Running                 sleep 103 &
raju@raju-VirtualBox:~/test$ fg
sleep 103
```

if you need any specific process to bring into the foreground, you have to execute $***fg*** <***job number***>

```
raju@raju-VirtualBox:~/test$ fg 2
sleep 102
```

# Process Types

In the below case: **bash** is the parent process, **sleep** and **ps** are the child processes.

```
raju@raju-VirtualBox:~/test$ sleep 10 &
[1] 583274
raju@raju-VirtualBox:~/test$ ps
    PID TTY          TIME CMD
 490350 pts/0    00:00:00 bash
 583274 pts/0    00:00:00 sleep
 583275 pts/0    00:00:00 ps
```

$**pstree** shows running processes as a tree. In the below example, we executed pstree with –p option to print pid of the processes. Here **pstree** is a child process, **bash** is the parent of **pstree** and the grand parent is **systemd**, the first process.

```
raju@raju-VirtualBox:~/test$ pstree -p
systemd(1)─┬─ModemManager(758)─┬─{ModemManager}(825)
           │                   └─{ModemManager}(837)
           │                   └─{snapd}(582366)
           ├─sshd(802)─┬─sshd(1077)───sshd(2025)───bash(2028)
           │           └─sshd(490161)───sshd(490349)───bash(490350)───pstree(583215)
```

# Process Types

**Orphan process**: if **parent** exits before **child** completes its execution then child will become orphan process and attached with the **systemd** process.

**Daemon process**: it is a orphan process, running at the background and not associated with any terminal type (tty). Daemons processes are usually started when the system starts and it will run until the system stops. Usually system services are running as daemon processes.

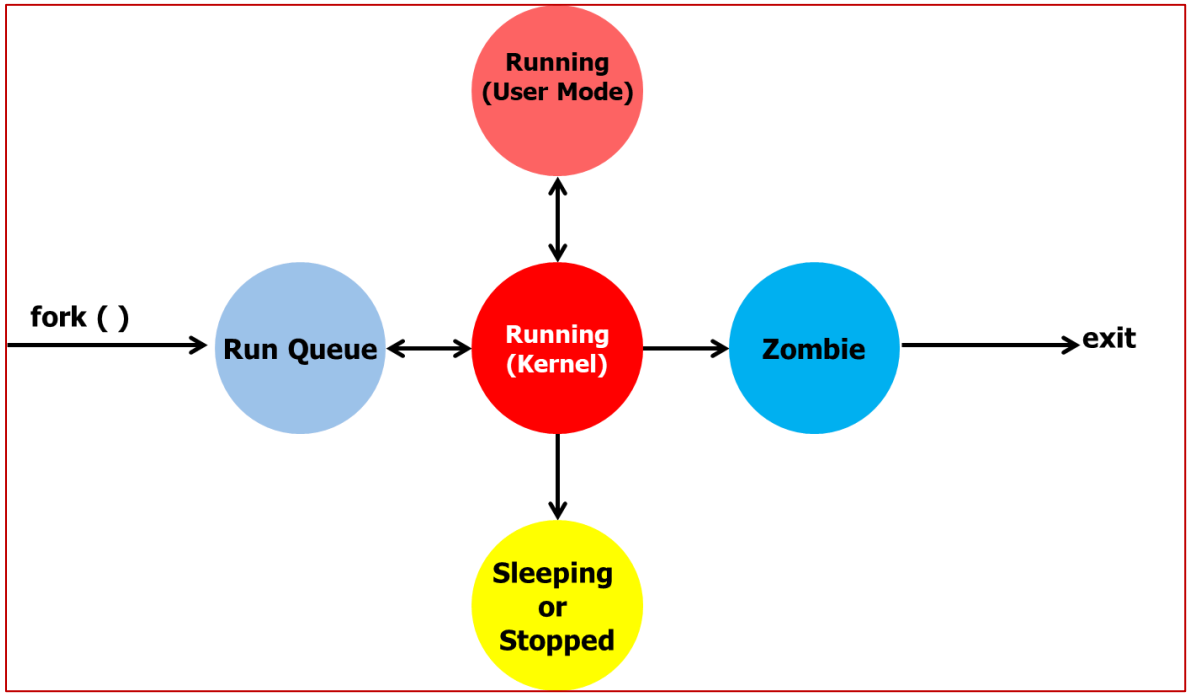
The below screen shot showes the details of the **ssh** service. **sshd** is a daemon process, it is not associated with the terminal type (**tty -?**), it is an orphan process: **PPID is 1** (attached with the **systemd** process.

```
raju@raju-VirtualBox:~/test$ ps -f 802
UID          PID    PPID  C STIME TTY          STAT   TIME CMD
root         802       1  0 08:04 ?            Ss     0:00 sshd: /usr/sbin/sshd -D [listener]
raju@raju-VirtualBox:~/test$
```

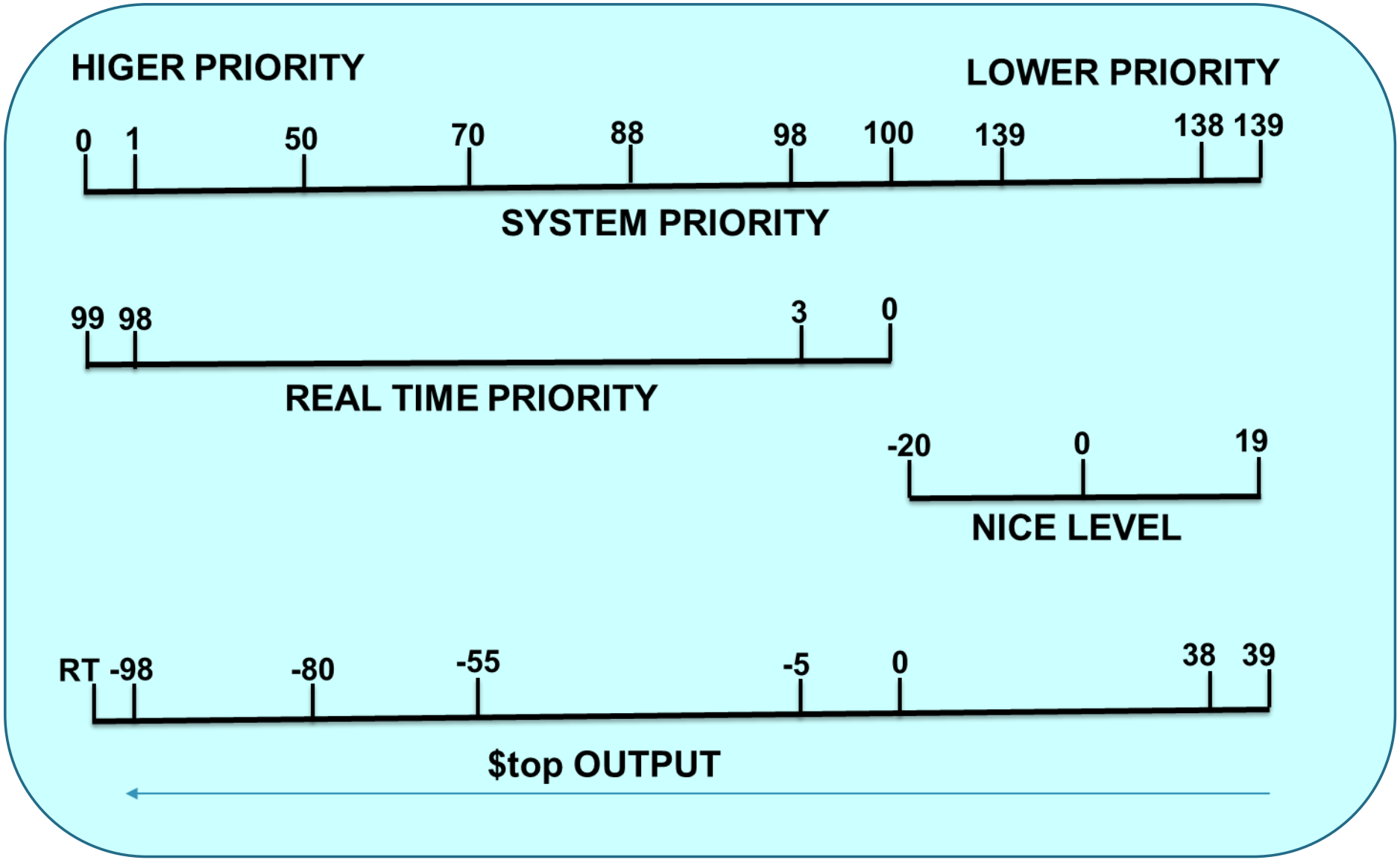
```
raju@raju-VirtualBox:~/test$ service sshd status
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-12-06 06:11:02 IST; 7h ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 755 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 802 (sshd)
      Tasks: 1 (limit: 9450)
     Memory: 7.4M
        CPU: 410ms
```

# Process States

**D** uninterruptible sleep (usually IO)
**R** running or runnable (on run queue)
**S** interruptible sleep (waiting for an event to complete)
**T** stopped by job control signal (if you press: ^z)
**Z** defunct ("zombie") process, terminated but not reaped by its parent



# Process Scheduling Priority



# Changing Priority

**nice** - run a program with modified scheduling priority.
We can get the pid of the process by executing $***sleep 1000 &*** command.

```
raju@raju-VirtualBox:~$ sleep 1000 &
[1] 583987
```

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 583987 raju      20   0   17024   1016    924 S   0.0   0.0   0:00.00 sleep
```

$**nice** <priority> executable command/file.
We can also renice (changing current running process priority by using renice command).
**renice** - alter priority of running processes. Ex: $**renice -n -20 -p 583987**

```
raju@raju-VirtualBox:~$ sudo renice -n -20 -p 583987
[sudo] password for raju:
583987 (process ID) old priority 0, new priority -20
```

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 583987 raju       0 -20   17024   1016    924 S   0.0   0.0   0:00.00 sleep
```

To change from normal to real time priority of the process, use **chrt** command.
**chrt** - manipulate the real-time attributes of a process. **$chrt -p <pid>.**
Example**: $chrt -r -p 99 <pid>  - r** option set scheduling policy to SCHED_RR.

# Viewing Running Processes

- **ps** displays information about a selection of the active processes.
- Syntax: **ps** [options]
- There are many options, user can refer man pages for the various option details.
- We can also customize the required print column format with the –aN option.

```
raju@raju-VirtualBox:~$ ps -aN --format cmd,user,pid,ppid | head
CMD                         USER          PID    PPID
/sbin/init splash           root            1       0
[kthreadd]                  root            2       0
[rcu_gp]                    root            3       2
[rcu_par_gp]                root            4       2
[slub_flushwq]              root            5       2
[netns]                     root            6       2
[kworker/0:0H-events_highpr root            8       2
[kworker/0:1H-events_highpr root           10       2
[mm_percpu_wq]              root           11       2
raju@raju-VirtualBox:~$
```

# Process Monitoring Tools

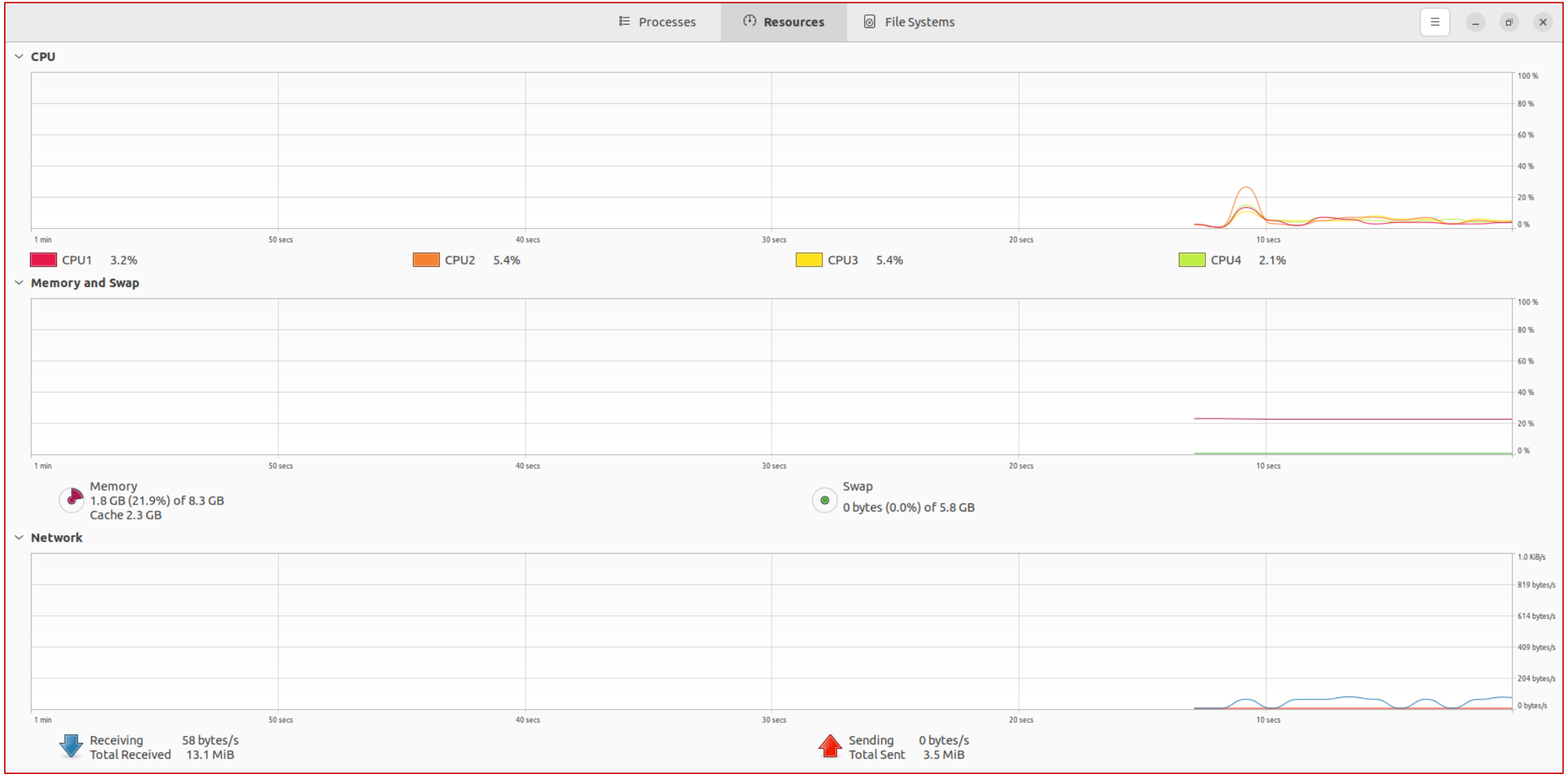
$**top** - provides a dynamic real-time view of a running system.

```
top - 15:47:28 up  6:36,  3 users,  load average: 0.16, 0.13, 0.10
Tasks: 217 total,   1 running, 216 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7949.0 total,   4342.4 free,   1384.6 used,   2222.1 buff/cache
MiB Swap:   5500.0 total,   5500.0 free,      0.0 used.   6284.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    585 systemd+  20   0   14824   6228   5440 S   0.3   0.1   0:28.36 systemd-oomd
 581164 raju      20   0  162220   2752   2360 S   0.3   0.0   0:20.23 VBoxClient
 583915 raju      20   0   17544   8140   5676 S   0.3   0.1   0:00.36 sshd
      1 root      20   0  166856  12032   8308 S   0.0   0.1   0:11.11 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.08 kthreadd
```

$**mpstat** - Report processors related statistics. The option 1 will display the output in every one second.

```
raju@raju-VirtualBox:~$ mpstat 1
Linux 6.0.0-060000-generic (raju-VirtualBox)    06/12/22        _x86_64_        (4 CPU)

03:49:42 PM IST  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
03:49:43 PM IST  all    0.00    0.00    0.00    0.00    0.00    0.24    0.00    0.00    0.00   99.76
03:49:45 PM IST  all    0.19    0.00    0.00    0.19    0.00    0.00    0.00    0.00    0.00   99.63
03:49:46 PM IST  all    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
03:49:47 PM IST  all    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
03:49:48 PM IST  all    0.00    0.00    0.00    0.00    0.00    0.24    0.00    0.00    0.00   99.76
03:49:49 PM IST  all    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
```

# System Monitor



# Signal

- Linux supports both standard and real-time signals.
- Signals are a fundamental method for inter process communication.
- A signal is generated when
  - an event occurs (timer expires, alarm, etc.,)
  - a user quota exceeds (file size, no of processes etc.,)
  - an I/O device is ready
  - encountering an illegal instruction
  - a terminal interrupt like Ctrl-C or Ctrl-Z.
  - some other process send ( kill -9 pid)

- **Default Action:**
  - **Term** Default action is to terminate the process.
  - **Ign** Default action is to ignore the signal.
  - **Core** Default action is to terminate the process and dump core (see core(5)).
  - **Stop** Default action is to stop the process.
  - **Cont** Default action is to continue the process if it is currently stopped.

- Two signals that cannot be caught or ignored: **SIGSTOP** and **SIGKILL.**

# Signal

- **$kill** - send a signal to a process. -l option will list all the signal numbers with signal names.
- 1 – 31: – standard signals and 32 - 64: – real time signals.
- Ctrl + c – will send SIGINT (terminate the process).
- Ctrl + z – will send SIGSTOP (suspend/stop the process).
- $**kill -9 <pid>** - send SIGKILL signal to the given process id (also called **sure kill** since SIGKILL (9) can't be caught or ignored).
- **killall** - kill processes by name, example: $**killall a.out**

```
raju@raju-VirtualBox:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
raju@raju-VirtualBox:~$
```

# Work with Bash Shell Variables

- Variables are a way of passing information from the shell to programs when you run them.
- Standard UNIX variables are split into two categories:
  - environment variables and shell variables
- shell variables apply only to the current instance of the shell and are used to set short term working conditions.
- environment variables set at login are valid for the duration of the session.
- By convention, environment variables have UPPER CASE and shell variables have lower case names.
- Shell variable can be a character, number or string. No data type so no need to declare a variable.

# Work with Bash Shell Variables

**To set normal variable**

- Value=500
- Name="B Thangaraju"
- NAME=Thangaraju
- echo $NAME (To print value of the variable)

To store a command output in a variable use back quote " ` ".
Example: **totdir=`ls –Rl`**
**echo $totdir**

**To set environment variable (Bash)**

- export NAME
- To check environment variables : $env

**unset** <variable name> : Unset values and attributes of shell variables and functions.

```
raju@raju-VirtualBox:~/demo$ name="B Thangaraju"
raju@raju-VirtualBox:~/demo$ echo $name
B Thangaraju
raju@raju-VirtualBox:~/demo$ unset name
raju@raju-VirtualBox:~/demo$ echo $name

raju@raju-VirtualBox:~/demo$
```

# Shell Scripting

The shell script language, like other programming languages, has constructs for:

- Conditional execution (if-then-else)
- Iterative execution (for loop; while loop)
- A switch statement

Let us start with the **hello world** example:

```
raju@raju-VirtualBox:~/test$ cat hw.sh
#! /bin/bash
# Hello World shell script example
echo "Hello World"
raju@raju-VirtualBox:~/test$
```

The first line is **#! /bin/bash** –shebang. It will tell the kernel which shell interpreter should be used to run the program irrespective of your default shell.

The second line starts with # -it is a comment statement for documentation.

Unlike compiler, shell is the interpreter and it will execute program line by line but a compiler compiles the entire source code and createS a single executable file.

# Shell Scripting

We can execute the shell program by $**sh hw.sh**. But this operation was costly since the ***sh*** command will create a child shell and execute the program.

```
raju@raju-VirtualBox:~/test$ sh hw.sh
Hello World
raju@raju-VirtualBox:~/test$
```

If we try to execute by **$./hw.sh**. It will throw an error as *permission denied*. The reason was the file hw.sh doesn't have executable permission.

```
raju@raju-VirtualBox:~/test$ ./hw.sh
-bash: ./hw.sh: Permission denied
```

```
raju@raju-VirtualBox:~/test$ ll hw.sh
-rw-rw-r-- 1 raju raju 32 Dec  7 15:07 hw.sh
```

So, we need to give executable permission for the user and then we can execute by **./hw.sh**.

```
raju@raju-VirtualBox:~/test$ chmod u+x hw.sh
raju@raju-VirtualBox:~/test$ ll hw.sh
-rwxrw-r-- 1 raju raju 32 Dec  7 15:07 hw.sh*
raju@raju-VirtualBox:~/test$ ./hw.sh
Hello World
```

# Shell Scripting

**Command Line Argument:**

Passing value to the variable when execute a shell script will allow you to make a script more flexible.

- The variables **$1, $2, $3** etc. Refer to the arguments given in the order.
- The variables **$@** refers to the complete string of arguments.
- The variable **$#** will give the number of arguments given.

```
raju@raju-VirtualBox:~/test$ cat cmd.sh
#! /bin/bash
# Command Line Arguments example.
echo "The first command Line Arguement is : $1"
echo "The second command Line Arguement is : $2"
echo "The third command Line Arguement is : $3"
echo "The complete command Line Arguments are: $@"
echo "Total number of command Line Argument is: $#"
raju@raju-VirtualBox:~/test$
```

# Shell Scripting

If we use command line arguments, you have to give input when you execute the shell script. Here the inputs are: First, Second and Third.

```
raju@raju-VirtualBox:~/test$ ./cmd.sh First Second Third
The first command Line Arguement is : First
The second command Line Arguement is : Second
The third command Line Arguement is : Third
The complete command Line Arguments are: First Second Third
Total number of command Line Argument is: 3
```

Input may be: character, string or number. If the string contains blank space, we have to use “ “.

```
raju@raju-VirtualBox:~/test$ ./cmd.sh "Bala Kumar" 300 "Siva Kumar"
The first command Line Arguement is : Bala Kumar
The second command Line Arguement is : 300
The third command Line Arguement is : Siva Kumar
The complete command Line Arguments are: Bala Kumar 300 Siva Kumar
Total number of command Line Argument is: 3
```

# Shell Scripting

For example, we have two shell script files and three directories in the PWD

```
raju@raju-VirtualBox:~/test$ ls
cmd.sh  hw.sh  test1  test2  test3
raju@raju-VirtualBox:~/test$ ls | wc -l
5
raju@raju-VirtualBox:~/test$ ls -l | grep ^d | wc -l
3
raju@raju-VirtualBox:~/test$
```

The below shell script shows how to execute command inside the script.
–n option will not output the trailing newline.

```
raju@raju-VirtualBox:~/test$ cat total.sh
#! /bin/bash
# Print total number of files and directories.

echo "List all the files in the $PWD"
ls

echo -n "Total Number of files in the $PWD:"
ls | wc -l

echo "Print only directories in the $PWD"
ls -l | grep ^d

echo -n "Total Number of directories in the $PWD:"
ls -l | grep ^d | wc -l
raju@raju-VirtualBox:~/test$
```

Output of total.sh shell script.

```
raju@raju-VirtualBox:~/test$ ./total.sh
List all the files in the /home/raju/test
cmd.sh  hw.sh  test1  test2  test3  total.sh
Total Number of files in the /home/raju/test:6
Print only directories in the /home/raju/test
drwxrwxr-x 2 raju raju 4096 Dec  7 17:31 test1
drwxrwxr-x 2 raju raju 4096 Dec  7 17:31 test2
drwxrwxr-x 2 raju raju 4096 Dec  7 17:31 test3
Total Number of directories in the /home/raju/test:3
raju@raju-VirtualBox:~/test$
```

# Shell Scripting

We can also assign the output of a command in a variable.

```
#! /bin/bash
# Print total number of files and directories.
files=`ls | wc -l`
dir=`ls -l | grep ^d | wc -l`
echo  "Total Number of files in the $PWD is:$files."
echo  "Total Number of directories in the $PWD is:$dir."
```

Output of total.sh shell script.

```
raju@raju-VirtualBox:~/test$ ./total.sh
Total Number of files in the /home/raju/test is:6.
Total Number of directories in the /home/raju/test is:3.
raju@raju-VirtualBox:~/test$
```

## Conditional Statements

```
if [ condition ]
then
  statement
fi
```

```
raju@raju-VirtualBox:~/test$ cat if_01.sh
#!/bin/bash
 age=29
  if [ $age -lt 30 ]
    then
     echo "you are still under 30"
  fi
raju@raju-VirtualBox:~/test$ ./if_01.sh
you are still under 30
raju@raju-VirtualBox:~/test$
```

```
raju@raju-VirtualBox:~/test$ cat if_01.sh
#!/bin/bash
 age=29
  if test $age -lt 30
    then
     echo "you are still under 30"
  fi
raju@raju-VirtualBox:~/test$ ./if_01.sh
you are still under 30
raju@raju-VirtualBox:~/test$
```

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

```
#!/bin/bash
echo -n "Enter the first number: "
read first
echo -n "Enter the second number: "
read second

if test $first -eq $second
  then
     echo "You have entered the same number."
else
        echo "You haven't entered the same number."
  fi
```

```
raju@raju-VirtualBox:~/test$ ./if_02.sh
Enter the first number: 100
Enter the second number: 100
You have entered the same number.
raju@raju-VirtualBox:~/test$ ./if_02.sh
Enter the first number: 100
Enter the second number: 200
You haven't entered the same number.
raju@raju-VirtualBox:~/test$
```

## Conditional Statements

```
if [ condition ]
then
   statement
elif [ condition ]
then
   statement
else
   statement
fi
```

```
#! /bin/bash
age=$1
if [ $age -lt 40 ]
then
        echo "You are still under 40."
elif [ $age -ge 40 -a $age -le 50 ]
then
        echo "You are in your 40's."
else
        echo "You are 50 or over."
fi
```

```
raju@raju-VirtualBox:~/test$ ./if_03.sh 30
you are still under 40
raju@raju-VirtualBox:~/test$ ./if_03.sh 43
you are in your 40s
raju@raju-VirtualBox:~/test$ ./if_03.sh 52
you are 50 or over
raju@raju-VirtualBox:~/test$
```

```
STRING1 = STRING2
       the strings are equal

STRING1 != STRING2
       the strings are not equal

INTEGER1 -eq INTEGER2
       INTEGER1 is equal to INTEGER2

INTEGER1 -ge INTEGER2
       INTEGER1 is greater than or equal to INTEGER2

INTEGER1 -gt INTEGER2
       INTEGER1 is greater than INTEGER2

INTEGER1 -le INTEGER2
       INTEGER1 is less than or equal to INTEGER2

INTEGER1 -lt INTEGER2
       INTEGER1 is less than INTEGER2

INTEGER1 -ne INTEGER2
       INTEGER1 is not equal to INTEGER2
```

# Shell Scripting

## Looping Statements

```
for (( looping instruction ))
do
        statement
done
```

```
#!/bin/sh

for FILE in $PWD/*.sh
do
   echo $FILE
done
```

```
raju@raju-VirtualBox:~/shell_script$ ./for2.sh
/home/raju/shell_script/a.sh
/home/raju/shell_script/cmd.sh
/home/raju/shell_script/for1.sh
/home/raju/shell_script/for2.sh
/home/raju/shell_script/hw.sh
/home/raju/shell_script/if_01.sh
/home/raju/shell_script/if_02.sh
/home/raju/shell_script/if_03.sh
/home/raju/shell_script/total.sh
raju@raju-VirtualBox:~/shell_script$
```

# Shell Scripting

## Looping Statements

```
for (( looping instruction ))
do
        statement
done
```

```
#! /bin/bash
# for loop execution example

count=0
for i in 1 2 3
do
        echo "i is $i"
        count=`expr $count + 1`
done
echo "The loop was executed $count times."
```

```
raju@raju-VirtualBox:~/shell_script$ ./for1.sh
i is 1
i is 2
i is 3
The loop was executed 3 times.
```

```
raju@raju-VirtualBox:~/shell_script$ cat for3.sh
#! /bin/bash
# for loop execution example

for (( i=1; i<=3; i++))
do
        echo "i is $i"
done
echo "The loop was executed $i times."
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ ./for3.sh
i is 1
i is 2
i is 3
The loop was executed 4 times.
raju@raju-VirtualBox:~/shell_script$
```

# Shell Scripting

```
raju@raju-VirtualBox:~/shell_script$ cat for1.sh
#! /bin/bash
# for loop execution example
count=0
for i in 1 2 3
do
        echo "i is $i"
        count=`expr $count + 1`
done
echo "The loop was executed $count times."
```

**for loop syntax example: 1**

```
raju@raju-VirtualBox:~/shell_script$ ./for1.sh
i is 1
i is 2
i is 3
The loop was executed 3 times.
```

```
raju@raju-VirtualBox:~/shell_script$ cat for3.sh
#! /bin/bash
# for loop execution example
count=0
for (( i=1; i<=3; i++ ))
do
        echo "i is $i"
        count=`expr $count + 1`
done
echo "The loop was executed $count times."
raju@raju-VirtualBox:~/shell_script$
```

**for loop syntax example: 2**

```
raju@raju-VirtualBox:~/shell_script$ ./for3.sh
i is 1
i is 2
i is 3
The loop was executed 3 times.
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ cat for4.sh
#! /bin/bash
# for loop execution example
count=0
for i in {1..3}
do
        echo "i is $i"
        count=`expr $count + 1`
done
echo "The loop was executed $count times."
raju@raju-VirtualBox:~/shell_script$
```

**for loop syntax example: 3**

```
raju@raju-VirtualBox:~/shell_script$ ./for4.sh
i is 1
i is 2
i is 3
The loop was executed 3 times.
raju@raju-VirtualBox:~/shell_script$
```

# Shell Scripting

```
raju@raju-VirtualBox:~/shell_script$ cat mul.sh
#! /bin/bash
# Print a multiplication table.
echo -n "Enter a number:  "
read x
for I in {1..10}
do
   echo "$I X $x = `expr $I \* $x`"
done
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ ./mul.sh
Enter a number:  16
1 X 16 = 16
2 X 16 = 32
3 X 16 = 48
4 X 16 = 64
5 X 16 = 80
6 X 16 = 96
7 X 16 = 112
8 X 16 = 128
9 X 16 = 144
10 X 16 = 160
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ bash -x mul.sh
+ echo -n 'Enter a number:  '
Enter a number:  + read x
21
+ for I in {1..10}
++ expr 1 '*' 21
+ echo '1 X 21 = 21'
1 X 21 = 21
+ for I in {1..10}
++ expr 2 '*' 21
+ echo '2 X 21 = 42'
2 X 21 = 42
```

```
+ for I in {1..10}
++ expr 10 '*' 21
+ echo '10 X 21 = 210'
10 X 21 = 210
raju@raju-VirtualBox:~/shell_script$
```

$**bash –x mul.sh** –is a debugging mode.
It will execute step by step instruction.
+ sign is for source code instruction.
++ sign is replace variable with the value.

# Shell Scripting

```
while [ condition ]
do
        statement
done
```

**Count total number of lines in a file using While loop**

```
raju@raju-VirtualBox:~/shell_script$ cat while.sh
#!/bin/bash
# While loop example
count=0
while read line
do
   count=`expr $count + 1`
done < sw.sh
echo "Total number of lines in the sw.sh file is: $count."
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ ./while.sh
Total number of lines in the sw.sh file is: 20.
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ cat sw.sh | wc -l
20
raju@raju-VirtualBox:~/shell_script$
```

Multiplication Table using While loop

```
raju@raju-VirtualBox:~/shell_script$ cat mul_while.sh
#! /bin/bash
# Print a multiplication table.
echo -n "Enter a number:  "
read x
i=1
while [ $i -le 10 ]
do
   echo "$i X $x = `expr $i \* $x`"
   i=`expr $i + 1`
done
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ ./mul_while.sh
Enter a number:  9
1 X 9 = 9
2 X 9 = 18
3 X 9 = 27
4 X 9 = 36
5 X 9 = 45
6 X 9 = 54
7 X 9 = 63
8 X 9 = 72
9 X 9 = 81
10 X 9 = 90
raju@raju-VirtualBox:~/shell_script$
```

# Shell Scripting

**switch – case :** Decision Making, will create menu driven program

```
case    value in

      pattern 1)
            command
            command
      ;;

      pattern 2)
            command
            command
      ;;

      pattern *)
            command
      ;;

esac
```

**case** – starting block, **value** is a pattern value, **;;** is a break statement, ***)** is a default case, if the value is not matching with any pattern then this case will be executed. **esac** is the end of block.

```
raju@raju-VirtualBox:~/shell_script$ cat sw.sh
#! /bin/bash
# switch case example
echo " Enter 1 for listing all the shell script files"
echo " Enter 2 for listing all the directories"
echo -n "Enter your choice: "

read choice
case $choice in
        1)
            ls -l *.sh
            ;;

        2)
            ls -l | grep ^d
            ;;

        *)
           echo "You have entered a wrong choice"
           ;;
esac
raju@raju-VirtualBox:~/shell_script$
```

# Shell Scripting

```
raju@raju-VirtualBox:~/shell_script$ ./sw.sh
 Enter 1 for listing all the shell script files
 Enter 2 for listing all the directories
Enter your choice: 1
-rwxrw-r-- 1 raju raju 184 Dec  7 18:49 a.sh
-rwxrw-r-- 1 raju raju 295 Dec  7 15:28 cmd.sh
-rwxrw-r-- 1 raju raju 157 Dec  8 08:35 for1.sh
-rwxrw-r-- 1 raju raju  55 Dec  8 08:09 for2.sh
-rwxrw-r-- 1 raju raju 167 Dec  8 08:31 for3.sh
-rwxrw-r-- 1 raju raju 158 Dec  8 08:32 for4.sh
-rwxrw-r-- 1 raju raju  93 Dec  8 08:55 for5.sh
-rwxrw-r-- 1 raju raju  67 Dec  7 17:26 hw.sh
-rwxrw-r-- 1 raju raju  92 Dec  7 18:21 if_01.sh
-rwxrw-r-- 1 raju raju 245 Dec  7 18:36 if_02.sh
-rwxrw-r-- 1 raju raju 208 Dec  7 18:47 if_03.sh
-rwxrw-r-- 1 raju raju 141 Dec  8 08:51 mul.sh
-rwxrw-r-- 1 raju raju 319 Dec  8 10:33 sw.sh
-rwxrw-r-- 1 raju raju 222 Dec  7 17:53 total.sh
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ ./sw.sh
 Enter 1 for listing all the shell script files
 Enter 2 for listing all the directories
Enter your choice: 2
drwxrwxr-x 2 raju raju 4096 Dec  7 17:31 test1
drwxrwxr-x 2 raju raju 4096 Dec  7 17:31 test2
drwxrwxr-x 2 raju raju 4096 Dec  7 17:31 test3
raju@raju-VirtualBox:~/shell_script$
```

```
raju@raju-VirtualBox:~/shell_script$ ./sw.sh
 Enter 1 for listing all the shell script files
 Enter 2 for listing all the directories
Enter your choice: 3
You have entered a wrong choice
raju@raju-VirtualBox:~/shell_script$
```

$sw**.sh** – execution of switch case shell script.

**Demo on Complex Shell Script**

- Write a menu driven program to create Employee Record.
- The Record should contains: Employee Name, Employee Number, and Telephone Number.
- The script should be able to delete a record, search and display a specific employee details, sort the record and list the complete record.

# Automate Script Execution

A crontab file contains instructions to the cron daemon of the general form: "**run this command at this time on this date**".

The Time and Date fields are:

```
        field          allowed values
        -----          --------------
        minute         0-59
        hour           0-23
        day of month   1-31
        month          1-12 (or names, see below)
        day of week    0-7 (0 or 7 is Sun, or use names)

A field may be an asterisk (*), which always stands for ``first-last''.
```

```
root@raju-VirtualBox:~# cat backup.sh
#!/bin/bash
# Take Backup in every Friday at 11:59 pm.

file=`date | awk '{print $2 $3}'`
tar -cvf backup.tar /home/raju/shell_script/*
gzip backup.tar
mkdir /backup/$file
mv backup.tar.gz /backup/$file/
root@raju-VirtualBox:~#
```

$**crontab -e** –you can edit the cron job.
$**crontab -l** will list the cron jobs

# Automate Script Execution

```
root@raju-VirtualBox:~# crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

59 23 * * FRI /root/backup.sh
root@raju-VirtualBox:~#
```

***50 23 * * FRI /root/backup.sh***
Take backup in every Friday at 11:59 pm.

We can execute any tasks (command, shell script, etc.,) to a specific date and time in crontab.

You can also see few examples in $**man 5 crontab**

# Lesson 5

- Basic Networking Commands
- Install New Software and Update the System
- Introduction to Services
- Basic System Troubleshooting and Firewalling

# Basic Networking Commands

# ifconfig and nmcli

```
raju@raju-VirtualBox:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:8a:9f:4c:dd  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::3317:cf88:8643:dc6a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:2e:20:0a  txqueuelen 1000  (Ethernet)
        RX packets 2941  bytes 3704043 (3.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1655  bytes 129698 (129.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.174  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::aa69:6bf5:5cb4:43a4  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:23:83:47  txqueuelen 1000  (Ethernet)
        RX packets 3536  bytes 426733 (426.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1207  bytes 213432 (213.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 379  bytes 32152 (32.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 379  bytes 32152 (32.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

raju@raju-VirtualBox:~$
```

From the **ifconfig** command output, we can get an ip address of our system. This ip address is needed to connect to other systems through ssh or copy file to and from the system.

**nmcli** is a command-line tool for controlling Network Manager and reporting network status.

**nmcli** is used to create, display, edit, delete, activate, and deactivate network connections, as well as control and display network device status.

**$man nmcli-examples** - manual page is to provide you with various examples and usage scenarios of nmcli.

$**nmcli devices** –gives the status of Network connections.

```
raju@raju-VirtualBox:~$ nmcli device
DEVICE    TYPE      STATE                   CONNECTION
enp0s3    ethernet  connected               Wired connection 1
enp0s8    ethernet  connected               Wired connection 2
docker0   bridge    connected (externally)  docker0
lo        loopback  unmanaged               --
raju@raju-VirtualBox:~$
```

# nmap

- **nmap** - Network exploration tool and security / port scanner.
- **nmap** ("Network Mapper") is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts.
- $**man nmap**.
- The below screen shots showed **nmap scan report** for the three network interfaces.

```
raju@raju-VirtualBox:~$ nmap 10.0.2.15
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-09 10:03 IST
Nmap scan report for raju-VirtualBox (10.0.2.15)
Host is up (0.000076s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ nmap 192.168.1.174
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-09 10:03 IST
Nmap scan report for raju-VirtualBox (192.168.1.174)
Host is up (0.00010s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ nmap 127.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-09 10:03 IST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000074s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
631/tcp  open  ipp
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
```

# ping

**ping** - send echo request to a given network hosts.
This will help us to understand whether we can reach a specific machine/website or not.

```
raju@raju-VirtualBox:~$ ping 192.168.1.174
PING 192.168.1.174 (192.168.1.174) 56(84) bytes of data.
64 bytes from 192.168.1.174: icmp_seq=1 ttl=64 time=0.042 ms
64 bytes from 192.168.1.174: icmp_seq=2 ttl=64 time=0.036 ms
64 bytes from 192.168.1.174: icmp_seq=3 ttl=64 time=0.040 ms
^C
--- 192.168.1.174 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2330ms
rtt min/avg/max/mdev = 0.036/0.039/0.042/0.002 ms
```

```
raju@raju-VirtualBox:~$ ping google.com
PING google.com (142.250.192.142) 56(84) bytes of data.
64 bytes from bom12s18-in-f14.1e100.net (142.250.192.142): icmp_seq=1 ttl=118 time=44.1 ms
64 bytes from bom12s18-in-f14.1e100.net (142.250.192.142): icmp_seq=2 ttl=118 time=46.8 ms
64 bytes from bom12s18-in-f14.1e100.net (142.250.192.142): icmp_seq=3 ttl=118 time=43.8 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 43.825/44.879/46.754/1.329 ms
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ ping 10.0.2.11
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data.
From 10.0.2.15 icmp_seq=1 Destination Host Unreachable
From 10.0.2.15 icmp_seq=2 Destination Host Unreachable
From 10.0.2.15 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.2.11 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss
```

# scp — OpenSSH secure file copy

- **scp** copies files between hosts on a network. It uses ***ssh*** for data transfer, and uses the same authentication and provides the same security as a login session.
- **$scp copy.txt raju@10.0.2.15:/home/raju/LF**
- In this example a file from (local system) /home/raju/copy.txt is copied from 192.168.1.174 to 10.0.2.15 and stored into the /home/raju/LF (remote system) directory. raju is the user name in the local system.
- It will prompt for : Yes/no –type yes and enter. It will ask password for raju and after enter the password, the file will be copied into the given remote system's directory.

```
raju@Ubuntu:~$ scp copy.txt raju@10.0.2.15:/home/raju/LF
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.
ED25519 key fingerprint is SHA256:cxtSU6/znyQgKGYTsbqmWvlegqfdpyihH+mrr0NpUIo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.15' (ED25519) to the list of known hosts.
raju@10.0.2.15's password:
copy.txt                                      100% 1812    693.8KB/s   00:00
raju@Ubuntu:~$
raju@Ubuntu:~$ ls -l /home/raju/LF
total 4
-rw-rw-r-- 1 raju raju 1812 Jul  3 10:18 copy.txt
```

# Install New Software and Update the System

# Advanced Packaging Tool (APT)

- **apt** provides a high-level command line interface for the package management system.
- It is intended as an end user interface and enables some options better suited for interactive usage by default compared to more specialized APT tools like ***apt-get***.
- ***Syntax***: apt command [packages]  (example: $**apt install openjdk-11-jdk**)

**Frequently Used Commands:**

- ***list*** - list packages based on package names
- ***search*** - search in package descriptions
- ***show*** - show package details
- ***install*** - install packages
- ***reinstall*** - reinstall packages
- ***remove*** - remove packages
- ***autoremove*** - Remove automatically all unused packages
- ***update*** - update list of available packages
- ***upgrade*** - upgrade the system by installing/upgrading packages
- ***full-upgrade*** - upgrade the system by removing/installing/upgrading packages

# Show Information About a Package

**Show information about Java package:**
**$sudo apt show openjdk-11-jdk**

```
raju@raju-VirtualBox:~$ apt show openjdk-11-jdk
Package: openjdk-11-jdk
Version: 11.0.17+8-1ubuntu2~22.04
Priority: optional
Section: java
Source: openjdk-lts
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: OpenJDK Team <openjdk@lists.launchpad.net>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 1,618 kB
Provides: java-compiler, java-sdk, java10-sdk, java11-sdk, java2-sdk, java5-sdk, java6-sdk, java7-sdk, java8-sdk, java
9-sdk
Depends: openjdk-11-jre (= 11.0.17+8-1ubuntu2~22.04), openjdk-11-jdk-headless (= 11.0.17+8-1ubuntu2~22.04), libc6 (>=
2.34)
Recommends: libxt-dev
Suggests: openjdk-11-demo, openjdk-11-source, visualvm
Conflicts: openjdk-11-jre-headless (<< 11~19-2)
Replaces: openjdk-11-jre-headless (<< 11~19-2)
Homepage: https://openjdk.java.net/
Download-Size: 1,547 kB
APT-Manual-Installed: yes
APT-Sources: http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages
Description: OpenJDK Development Kit (JDK)
 OpenJDK is a development environment for building applications,
 applets, and components using the Java programming language.
```

# Install a Package

**Install Java package:**
**$sudo apt install openjdk-11-jdk**

```
raju@raju-VirtualBox:~$ sudo apt install openjdk-11-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev
  xorg-sgml-doctools xtrans-dev
Suggested packages:
  libice-doc libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-11-d
The following NEW packages will be installed:
  libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev
  xorg-sgml-doctools xtrans-dev
0 upgraded, 13 newly installed, 0 to remove and 95 not upgraded.
Need to get 218 MB of archives.
After this operation, 233 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Press **Y** and enter. It will start to install the packages. It may take 10 to 20 minutes.

**Verify Java version:**

```
raju@raju-VirtualBox:~$ java --version
openjdk 11.0.17 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu222.04, mixed mode, sharing)
raju@raju-VirtualBox:~$
```

# List and Update Packages

```
raju@raju-VirtualBox:~$ apt list | grep openjdk-11-jdk

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

openjdk-11-jdk-headless/jammy-updates,jammy-security,now 11.0.17+8-1ubuntu2~22.04 amd64 [installed,automatic]
openjdk-11-jdk-headless/jammy-updates,jammy-security 11.0.17+8-1ubuntu2~22.04 i386
openjdk-11-jdk/jammy-updates,jammy-security,now 11.0.17+8-1ubuntu2~22.04 amd64 [installed]
openjdk-11-jdk/jammy-updates,jammy-security 11.0.17+8-1ubuntu2~22.04 i386
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ sudo apt update
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:4 https://pkg.jenkins.io/debian-stable binary/ Release
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Hit:7 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu jammy InRelease
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [20.0 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [13.3 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [94.9 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [258 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [11.7 kB]
Fetched 723 kB in 4s (187 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

# Remove a Package

**$sudo apt remove openjdk-11-jdk :** will uninstall the java package
$**sudo apt purge openjdk-11-jdk :** will remove a package and java related configuration files.

```
raju@raju-VirtualBox:~$ sudo apt remove openjdk-11-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev libxdmcp-dev
  xorg-sgml-doctools xtrans-dev
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  openjdk-11-jdk
0 upgraded, 0 newly installed, 1 to remove and 96 not upgraded.
After this operation, 1,618 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 265574 files and directories currently installed.)
Removing openjdk-11-jdk:amd64 (11.0.17+8-1ubuntu2~22.04) ...
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ sudo apt purge openjdk-11-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'openjdk-11-jdk' is not installed, so not removed
The following packages were automatically installed and are no longer required:
  libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev
  xorg-sgml-doctools xtrans-dev
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 96 not upgraded.
raju@raju-VirtualBox:~$
```

# Update the System

$**sudo apt update** is used to download package information from all configured sources.

```
raju@raju-VirtualBox:~$ sudo apt update
[sudo] password for raju:
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:4 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu jammy InRelease
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Hit:6 https://pkg.jenkins.io/debian-stable binary/ Release
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [20.1 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [13.3 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [758 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [391 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [95.1 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [761 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [555 kB]
Get:16 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [257 kB]
Get:17 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]
Get:18 http://in.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [11.6 kB]
Fetched 3,187 kB in 14s (225 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

# Upgrade the System

$**sudo apt upgrade** is used to install available upgrades of all packages currently installed on the system from the sources configured. New packages will be installed if required to satisfy dependencies.

```
raju@raju-VirtualBox:~$ sudo apt upgrade
[sudo] password for raju:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2
Use 'sudo apt autoremove' to remove them.
The following packages have been kept back:
  xserver-common xserver-xephyr xserver-xorg-core xserver-xorg-legacy
The following packages will be upgraded:
  alsa-ucm-conf ansible ansible-core apport apport-gtk distro-info-data firmware-sof-signed fonts-opensymbol fwupd
  gdb gir1.2-gdkpixbuf-2.0 gir1.2-gnomedesktop-3.0 gir1.2-mutter-10 gnome-desktop3-data gnome-shell
  gnome-shell-common gstreamer1.0-pipewire libfprint-2-2 libfwupd2 libfwupdplugin5 libgdk-pixbuf-2.0-0
  libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libglib2.0-0 libglib2.0-bin libglib2.0-data libgnome-bg-4-1
  libgnome-desktop-3-19 libgnome-desktop-4-1 libmutter-10-0 libnetplan0 libpam-sss libpipewire-0.3-0
  libpipewire-0.3-common libpipewire-0.3-modules libpulse-mainloop-glib0 libpulse0 libpulsedsp libpython3.10
  libpython3.10-minimal libpython3.10-stdlib libreoffice-base-core libreoffice-calc libreoffice-common
  libreoffice-core libreoffice-draw libreoffice-gnome libreoffice-gtk3 libreoffice-help-common
  libreoffice-help-en-us libreoffice-impress libreoffice-math libreoffice-ogltrans libreoffice-pdfimport
  libreoffice-style-breeze libreoffice-style-colibre libreoffice-style-elementary libreoffice-style-yaru
  libreoffice-writer libsmbclient libspa-0.2-modules libuno-cppu3 libuno-cppuhelpergcc3-3 libuno-purpenvhelpergcc3-3
  libuno-sal3 libuno-salhelpergcc3-3 libwbclient0 linux-firmware mutter-common netplan.io openvpn pipewire
  pipewire-bin pulseaudio pulseaudio-module-bluetooth pulseaudio-utils python3-apport python3-distupgrade
  python3-problem-report python3-uno python3.10 python3.10-minimal rsync runc samba-libs sudo ubuntu-advantage-tools
  ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk uno-libs-private ure
91 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
5 standard security updates
Need to get 397 MB/405 MB of archives.
After this operation, 28.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 rsync amd64 3.2.3-8ubuntu3.1 [404 kB]
```

# Introduction to Services

# systemctl – Service Manager

- A service is a program or application that runs in the background.
- **systemctl** - Control the service manager.
- To list status of all the services: $**systemctl list-unit-files --type service -all**

| UNIT FILE | STATE |
|---|---|
| accounts-daemon.service | enabled |
| acpid.service | disabled |
| alsa-restore.service | static |
| alsa-state.service | static |
| alsa-utils.service | masked |
| anacron.service | enabled |
| apparmor.service | enabled |
| apport-autoreport.service | static |
| apport-forward@.service | static |
| apport.service | generated |
| apt-daily-upgrade.service | static |
| apt-daily.service | static |
| autovt@.service | alias |
| avahi-daemon.service | enabled |
| bluetooth.service | enabled |
| bolt.service | static |
| brltty-udev.service | static |
| brltty.service | disabled |
| colord.service | static |
| configure-printer@.service | static |
| console-getty.service | disabled |

- Start a service: **systemctl start <service-name>**
- Stop a service: **systemctl stop <service-name>**
- Restart a service: **systemctl restart <service-name>**
- Status of a service: **systemctl status <service-name>**

- **service** command is also useful but **systemctl** is more powerful than service command.
- $**service <service-name> command**
- To check all the services:
  $**service --status-all**

# systemctl – Service Manager

To check whether the Network service is running or not: $**systemctl status NetworkManager.service**

```
raju@raju-VirtualBox:~$ sudo systemctl status NetworkManager.service
[sudo] password for raju:
● NetworkManager.service - Network Manager
     Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2022-12-08 18:27:55 IST; 21h ago
       Docs: man:NetworkManager(8)
   Main PID: 669 (NetworkManager)
      Tasks: 3 (limit: 9450)
     Memory: 11.8M
        CPU: 1.930s
     CGroup: /system.slice/NetworkManager.service
             └─669 /usr/sbin/NetworkManager --no-daemon

Dec 09 12:41:13 raju-VirtualBox NetworkManager[669]: <info>  [1670569873.7973] dhcp4 (enp0s8): activation:
```

To check the service name: $ **systemctl list-unit-files --type service -all | grep ssh**

```
raju@raju-VirtualBox:~$ systemctl list-unit-files --type service -all | grep ssh
ssh.service                                enabled         enabled
ssh@.service                               static          -
sshd.service                               alias           -
raju@raju-VirtualBox:~$
```

# SSH configuration

- ssh — OpenSSH remote login client.
- ssh is a program for logging into a remote machine and for executing commands on a remote machine. It is intended to provide secure encrypted communications between two untrusted hosts over an insecure network.
- Package Name**: openssh-server**
- Service Name: **ssh**; Port Number: **22**
- Install the ssh package: $**sudo apt-get install openssh-server**
- Start the service: $**systemctl start ssh.service**
- Check the status of the service: $**systemctl status ssh.service**

```
raju@raju-VirtualBox:~$ systemctl status ssh.service
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2022-12-08 18:27:55 IST; 22h ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 807 (sshd)
      Tasks: 1 (limit: 9450)
     Memory: 13.0M
        CPU: 588ms
     CGroup: /system.slice/ssh.service
             └─807 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
```

# SSH configuration

To connect a remote system through ssh:

$ssh <username>@<ipaddress>

Example: $**ssh raju@10.0.2.15**

It will prompt for password then it will connect the remote system through ssh.

```
raju@raju-VirtualBox:~$ ssh raju@10.0.2.15
raju@10.0.2.15's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 6.0.0-060000-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

88 updates can be applied immediately.
5 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Fri Dec  9 16:18:13 2022 from 127.0.0.1
raju@raju-VirtualBox:~$
```

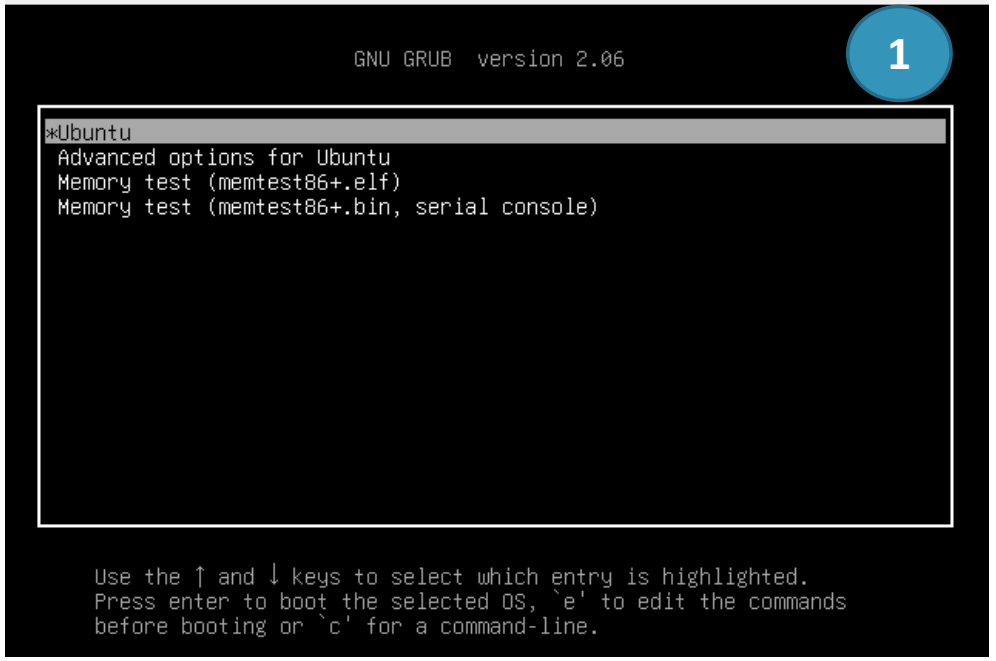
# Basic System Troubleshooting and Firewalling

# Basic System Troubleshooting

If some application or service or network is not working:

- Check whether the Network service is running or not.
- Check all the network interfaces are up and running (use ifconfig, nmcli or ping commands).
- If some application is not working, check the status of its service.
- Update the system.
- Upgrade the system.
- Reboot (most of the time, it will work).

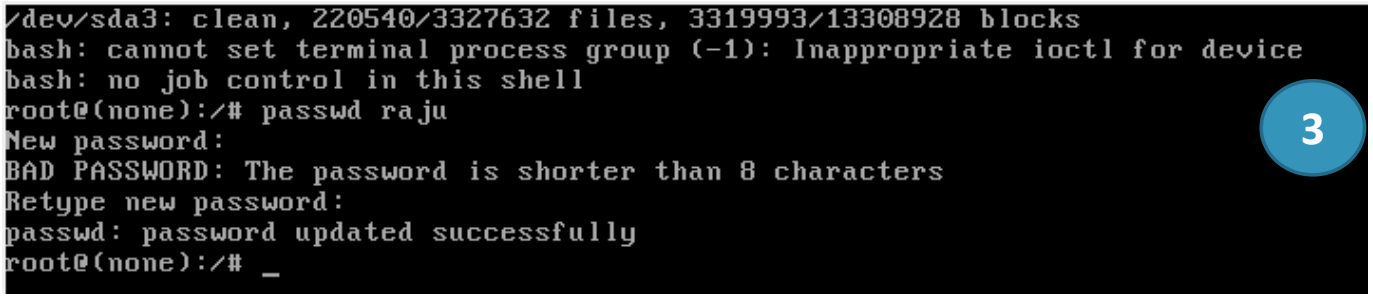
# Forgot your password?




1


2

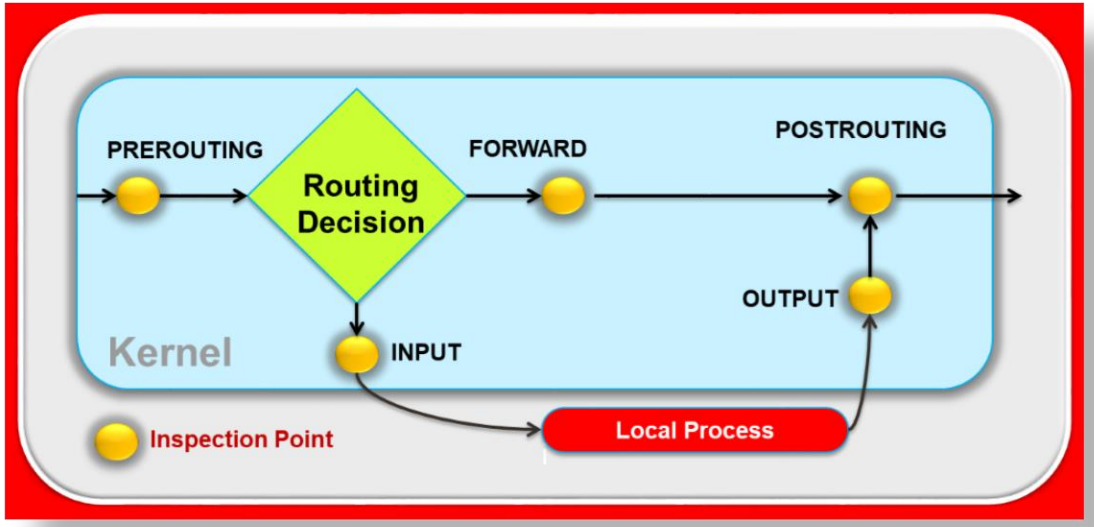
- Reboot your computer.
- Hold Shift key during boot to start GRUB menu.
- Press e to edit. (1)
- Append rw init=/bin/bash at the end of Linux line.
- Press Ctrl + X to boot. (2)
- It will show root prompt without asking passwd.
- Type passwd raju (username). (3)
- Reset the passwd and reboot.




3

# Firewall

- This Linux based **firewall is controlled by** the program called **iptables** to handles filtering for IPv6.
- **iptables** — administration tool for packet filtering and Network Address Translation (NAT).
- Iptables are used to set up, maintain, and inspect the tables of packet filter rules in the Linux kernel.
- Understanding how to setup and configure iptables will help you manage your Linux firewall effectively.




| | |
|---|---|
| **INPUT** | — Alters network packets targeted for the host. |
| **OUTPUT** | — Alters locally-generated n/w packets before they are sent out. |
| **FORWARD** | — Alters network packets routed through the host. |
| **PREROUTING** | — Alters incoming network packets before they are routed. |
| **POSTROUTING** | — Alters network packets before they are sent out. |

**ufw** - **U**ncomplicated **F**ire**w**all. The default firewall configuration tool for Ubuntu is **ufw**.

This program is for managing a Linux firewall and aims to provide an easy to use interface for the user. It is consisting of a small number of simple commands, and uses **iptables** for configuration.

# ufw

ufw Package Details

```
raju@raju-VirtualBox:~$ apt show ufw
Package: ufw
Version: 0.36.1-4build1
Priority: standard
Section: admin
Origin: Ubuntu
Maintainer: Jamie Strandboge <jdstrand@ubuntu.com>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 850 kB
Depends: iptables, lsb-base (>= 3.0-6), ucf, python3:any, debconf (>= 0.5) | debconf-2.0
Suggests: rsyslog
Homepage: https://launchpad.net/ufw
Task: standard
Download-Size: 162 kB
APT-Manual-Installed: no
APT-Sources: http://in.archive.ubuntu.com/ubuntu jammy/main amd64 Packages
Description: program for managing a Netfilter firewall
 The Uncomplicated FireWall is a front-end for iptables, to make managing a
 Netfilter firewall easier. It provides a command line interface with syntax
 similar to OpenBSD's Packet Filter. It is particularly well-suited as a
 host-based firewall.
```

ufw Service Status

```
raju@raju-VirtualBox:~$ systemctl status ufw.service
● ufw.service - Uncomplicated firewall
     Loaded: loaded (/lib/systemd/system/ufw.service; enabled; vendor preset: enabled)
     Active: active (exited) since Sat 2022-12-10 12:14:22 IST; 2min 7s ago
       Docs: man:ufw(8)
    Process: 523 ExecStart=/lib/ufw/ufw-init start quiet (code=exited, status=0/SUCCESS)
   Main PID: 523 (code=exited, status=0/SUCCESS)
        CPU: 1ms

Dec 10 12:14:22 raju-VirtualBox systemd[1]: Starting Uncomplicated firewall...
Dec 10 12:14:22 raju-VirtualBox systemd[1]: Finished Uncomplicated firewall.
raju@raju-VirtualBox:~$
```

# iptables

```
raju@raju-VirtualBox:~$ iptables --version
iptables v1.8.7 (nf_tables)
raju@raju-VirtualBox:~$
```

$**sudo iptables -F** : Flush the selected chain (all the chains in the table if none is given). This is equivalent to deleting all the rules one by one.

$**sudo iptables -L** : List all rules in the selected chain. If no chain is selected, all chains are listed.

$**sudo iptables –S** : Print all rules in the selected chain.

```
raju@raju-VirtualBox:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

```
raju@raju-VirtualBox:~$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ sudo iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
raju@raju-VirtualBox:~$
```

# iptables

- **I** – insert rule on top;
- **INPUT / OUTPUT / FORWARD**;
- **s** – source;
- **p** – protocol;
- **--dport** – destination port;
- **!** – apart from given network address.
- **-j** - jump; (This specifies the target of the rule; i.e., what to do if the packet matches it.)
- Port numbers: **80 – httpd**; **21 – ftp**; **22 – ssh**; **25 – mail server**;

To check all the port numbers: **/etc/services**

```
raju@raju-VirtualBox:~$ sudo iptables  -I INPUT   -s 192.168.0.0/255.255.255.0 -p  tcp --dport 80 -j  ACCEPT
raju@raju-VirtualBox:~$ sudo iptables  -I INPUT ! -s 192.168.0.0/255.255.255.0 -p  tcp --dport 21 -j  REJECT
raju@raju-VirtualBox:~$ sudo iptables  -I INPUT   -s 192.168.0.0/255.255.255.0 -p  tcp --dport 25 -j  REJECT
raju@raju-VirtualBox:~$ sudo iptables  -I INPUT   -s 192.168.0.0/255.255.255.0 -p  tcp --dport 22 -j  ACCEPT
```

```
raju@raju-VirtualBox:~$ sudo iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.0.0/24       anywhere             tcp dpt:ssh
REJECT     tcp  --  192.168.0.0/24       anywhere             tcp dpt:smtp reject-with icmp-port-unreachable
REJECT     tcp  -- !192.168.0.0/24       anywhere             tcp dpt:ftp reject-with icmp-port-unreachable
ACCEPT     tcp  --  192.168.0.0/24       anywhere             tcp dpt:http
raju@raju-VirtualBox:~$
```

# iptables

- If you want to block all ping request from the server to remote server. The Internet Control Message Protocol (ICMP) is a supporting protocol in the Internet protocol suite. ICMP is the message protocol used for the ping command.

- In the below example a rule is inserted to drop all the ping requests from the server.

```
raju@raju-VirtualBox:~$ sudo iptables -A OUTPUT -p icmp -j DROP
```

```
raju@raju-VirtualBox:~$ sudo iptables -L OUTPUT
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere
raju@raju-VirtualBox:~$
```

```
raju@raju-VirtualBox:~$ ping google.com
PING google.com (142.250.196.78) 56(84) bytes of data.
^C
--- google.com ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3074ms

raju@raju-VirtualBox:~$
```

# iptables

In the below example: the system will block any incoming ping request to the system.

```
raju@raju-VirtualBox:~$ sudo iptables -A INPUT -p icmp -j DROP
raju@raju-VirtualBox:~$ ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
^C
--- localhost ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2050ms
```

```
raju@raju-VirtualBox:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere
DROP       icmp --  anywhere             anywhere
DROP       icmp --  anywhere             anywhere
raju@raju-VirtualBox:~$
```

# iptables

- We flush the iptable rules with –F option
- We can verify the empty iptable.

```
raju@raju-VirtualBox:~$ sudo iptables -F
raju@raju-VirtualBox:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

- Now outgoing ping request is working.

```
raju@raju-VirtualBox:~$ ping -c 1 google.com
PING google.com (142.250.193.110) 56(84) bytes of data.
64 bytes from maa05s24-in-f14.1e100.net (142.250.193.110): icmp_seq=1 ttl=59 time=14.3 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 14.342/14.342/14.342/0.000 ms
raju@raju-VirtualBox:~$
```

- The incomfing ping request is working.

```
raju@raju-VirtualBox:~$ ping -c 1 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.012 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.012/0.012/0.012/0.000 ms
raju@raju-VirtualBox:~$
```

# Thank You