

### Week-1

#### Aim: [1] Types of OS installation

#### Operating System -

An Operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs. In other words, an Operating System (OS) is software that acts as an interface between computer hardware components and the user.

#### Types of OS installation

##### Attended installation:

- An installation process usually **needs a user who attends** it to make choices, such as accepting or declining an end-user license agreement (EULA), specifying preferences such as the installation location, supplying passwords or assisting in product activation.
- In graphical environments, installers that offer a **wizard-based interface** are common.
- Attended installers may ask users to help mitigate the errors.
- For example, if the disk in which the computer program is being installed was full, the installer may ask the user to specify another target path or clear enough space in the disk.

##### Silent installation:

- An installation that does not display messages or windows during its progress.
- "Silent installation" is NOT the same as "unattended installation". [All silent installations are unattended but not all unattended installations are silent.]
- In bigger organizations where thousands of users work, deploying the applications becomes a typical task and for that reason silent installation is performed so that the application is installed in background without affecting the work of user.

### **Unattended installation:**

- An installation that is performed without user interaction during its progress or with no user present at all.
- One of the reasons to use this approach is to automate the installation of a large number of systems
- An unattended installation either does not require the user to supply anything or has received all necessary input prior to the start of installation. Such input may be in the form of command line switches or an answer file, a file that contains all the necessary parameters.
- For example, if the installation medium was faulty, the installer should fail the installation, as there is no user to fix the fault or replace the medium. Unattended installers may record errors in a computer log for later review.

### **Headless installation:**

- Installation performed without using a computer monitor connected.
- In attended forms of headless installation, another machine connects to the target machine (for example, via a local area network) and takes over the display output.
- Since a headless installation does not need a user at the location of the target computer, unattended headless installers may be used to install a program on multiple machines at the same time.

### **Scheduled or Automated installation:**

- An installation process that runs on a preset time or when a predefined condition meets the requirements, as opposed to an installation process that starts explicitly on a user's command.
- For example, a system administrator willing to install a later version of a computer program that is being used can schedule that installation to occur when that program is not running.
- An operating system may automatically install a device driver for a device that the user connects.

### **Clean installation:**

- A clean installation is one that is done in the absence of any interfering elements such as old versions of the computer program being installed or leftovers from a previous installation.
- The clean installation of an operating system is an installation in which the target disk partition is erased before installation.
- Since the interfering elements are absent, a clean installation may succeed where an unclean installation may fail or may take significantly longer.

### **Network installation:**

Network installation (**net install**), is an installation of a program from a shared network resource that may be done by installing a minimal system before proceeding to download further packages over the network.

This may simply be a copy of the original media but software publishers which offer site licenses for institutional customers may provide a version intended for installation over a network.

## Aim: [2] Boot Methods

### Booting: -

**Booting** is the process of starting a computer as initiated via hardware such as a button or by a software command.

### Types of Booting:

- **Warm Booting:** The Warm Booting is that in which system starts from the starting or from initial state means.
  - In the Warm Booting the system will be started from its beginning state means, first, the user will press the **Power Button**, then this will read all the instructions from the ROM and the Operating System will be automatically gets loaded into the System (RAM).
- **Cold Booting:** The Cold Booting is that in which System automatically starts when the system is in a running state.
  - For example, due to Light Fluctuation, the system will automatically **restart**. In this, chances of damaging of system are more. The system will now be start from its initial state, so some files may be damaged because they are not properly stored into the system.

## Aim: [3] File System and Formatting:

### File System:

- A file system is a process of managing how and where data on a storage disk, which is also referred to as file management or FS.
- It is a logical disk component that compresses files separated into groups, which is known as directories.
- The file system enables user to view a file in the current directory as files are often managed in a hierarchy.
- It is abstract to a human user and related to a computer; hence, it manages a disk's internal operations.
- NTFS is the most common file system in modern times (Windows OS).
- Without file management, it would be impossible for a file with the same name to exist and also impossible to remove installed programs and recover specific files.

### Examples of File Systems:

The examples of file systems are given below:

- **FAT:** FAT is a type of file system, which is developed for hard drives. It stands for **File Allocation Table**. On hard drives and other computer systems, it helps to manage files on Microsoft operating systems. In devices like digital cameras, flash memory, and other portable devices, it is also often found that is used to store file information. It also helps to extend the life of a hard drive as it minimizes the wear and tears on the hard disc. Now a days later versions of Microsoft Windows like Windows XP, Vista, 7, and 10 as use NTFS.
  - The **FAT8, FAT12, FAT32, FAT16** are all the different types of FAT (for file allocation table).
- **GFS:** A GFS is a file system, which stands for Global File System. It has the ability to make enable multiple computers to act as an integrated machine. When the physical distance of two or computers is high, and they are unable to send files directly with each other, a GFS file system makes them capable of sharing a group of files directly.

- **HFS:** HFS (Hierarchical file system) is the file system that is used on a Macintosh computer for creating a directory at the time a hard disk is formatted. Generally, its basic function is to organize or hold the files on a Macintosh hard disk. Apple is not capable of supporting to write to or format HFS disks since when OS X came on the market. Also, HFS-formatted drives are not recognized by Windows computers as HFS is a Macintosh format. With the help of WIN32 or NTFS file systems, Windows hard drives are formatted.
- **NTFS:** NTFS is the file system, which stands for NT file system and stores and retrieves files on Windows NT operating system and other versions of Windows like Windows 2000, Windows XP, Windows 7, and Windows 10. Sometimes, it is known as the **New Technology File System**. As compared to the FAT and HFS file system, it provides better methods of file recovery and data protection and offers a number of improvements in terms of extendibility, security, and performance.
- **UDF:** A UDF is a file system, stands for **Universal Disk Format** and used first developed by OSTA (Optical Storage Technology Association) for ensuring consistency among data written to several optical media. It is used with CD-ROMs and DVD-ROMs and is supported on all operating systems. Now, it is used in the process of CD-R's and CD-RW's, called packet writing.

### Formatting:

Formatting is a process of preparing the storage device to store the data. Formatting storage device will erase the earlier contents of the device.

### Week-1

#### Aim:[4]Post installation tasks

Post Installation task is the set of steps to be carried out to ensure that the installation is complete and went smoothly.

#### Post Installation Tasks for Ubuntu Operating System:

- **Online accounts**
  - The first step allows user to configure online accounts, in case user want to integrate the desktop with different services.
- **Live patch**
  - Live patch is a service that allows the installation of some updates that would generally require a system reboot, such as those of the kernel.
- **Help improve Ubuntu**
  - In this step user can choose whether or not to send data from his system to Ubuntu. The option is activated by default. The user can verify the secrecy of the data being sent beforehand. The results are used to improve Ubuntu.
- **Privacy**
  - If required, a user can enable location services so that apps can determine user geographic location. All the applications installed by default on Ubuntu are free software.
- **You are ready to start!**
  - The last screen shows some featured applications -some of which are not free software with the option to open the software center to install them.

## Week-2

### Aim: [1] Install and configure virtual machine-Virtual box

#### Virtual Box:

VirtualBox is a cross-platform virtualization application. It installs on existing operating systems and also it extends the capabilities of the existing computer so that it can run multiple operating systems (it means, inside multiple virtual machines) at the same time.

- **Host operating system (host OS):** the operating system of the physical computer on which VirtualBox was installed.
- **Guest operating system (guest OS):** the operating system that is running inside the virtual machine.
- **Virtual machine (VM).** When running, a VM is the special environment that VirtualBox creates for guest operating system.

#### VirtualBox's main features:

<ul style="list-style-type: none"> <li>• Portability</li> <li>• No hardware virtualization required</li> <li>• Guest Additions: shared folders, seamless windows, 3Dvirtualization</li> <li>• Multigeneration branchedsnapshots.</li> <li>• Clean architecture; unprecedented modularity.</li> </ul>	<ul style="list-style-type: none"> <li>• Great hardware support                             <ul style="list-style-type: none"> <li>○ Guest multiprocessing (SMP)</li> <li>○ USB 2.0 device support</li> <li>○ Hardware compatibility</li> <li>○ Full ACPI support (Advanced Configuration and Power Interface)</li> <li>○ Multiscreen resolutions</li> <li>○ Built-in iSCSI support</li> </ul> </li> <li>• Remote machine display</li> </ul>
--	--

#### Host operating systems:

- **Windows hosts** (Windows XP, Windows 7, Windows Server 2003, 2008, Vista etc.)
- **Mac OS X hosts** (Leopard 32-bit, Snow Leopard 32/64 -bit)
- **Linux hosts** (Ubuntu, Debian, Fedora, Mandriva etc.)



### Installing Virtual Box on Hosts:

- Open Terminal
- Be sure about Internet Connection is there
- Type the command **sudo apt-get install VirtualBox**
- Type **VirtualBox** in terminal or select virtual box from installed programs list.

### Installing Virtual Box from OFFLINE software:

- get the Virtual Machine image for Linux distribution from Oracle Website
- Example: for Ubuntu 18.3 – Get the Image for Ubuntu16.04
- [https://download.virtualbox.org/virtualbox/6.1.32/virtualbox-6.1\\_6.1.32-149290~Ubuntu~xenial\\_amd64.debo](https://download.virtualbox.org/virtualbox/6.1.32/virtualbox-6.1_6.1.32-149290~Ubuntu~xenial_amd64.debo)
- Go to the Downloads -> Select the Downloaded Virtual Box Software -> Open -> Check for the Compatibility (if not compatible, download the suitable image form oracle website) -> Install the Virtual Machine Software.

### Creating Virtual Machine:

- Open the VirtualBox and click on New to create a **new** VM, give name of user choice, select Linux & version as 32 bit or 64 bits depending upon user system architecture
- Give RAM Memory to the VM. **Generally, 1024Mbytes**
- Select the hard disk space, **Go for 20.00GB.**
- Create **New Hard Drive** or Select one form the list or from another location. **Go for Create a Virtual hard drive now.**
- Select the type of the file form the new virtual hard drive. **Go for VDI (Virtual Box Disk Image)**
- Choose whether the new virtual disk file should be allocated as it is used (Dynamic) or if it should be created fully allocated at its maximum size (Fixed Size). **Go for Dynamically Allocated.**
- **Summary:** Will display the Virtual disk parameters. If these settings are correct, press the **Create** button and it will create a new virtual disk file.

### Installing Guest Operating System:

- ✓ Make sure the location of Guest OS iso in host OS.
- ✓ Select the Virtual disk in Virtual Box Manager (VBM), then select **Settings**.
- ✓ Select **Storage** Tab, Select **Controller IDE**, the select ☒ **Empty**. Then click on CD symbol at **Attributes** frame, and then select the Guest OS iso File (*Choose Virtual Optical Disk File*).
- ✓ Click the Start Button on VBM toolbar.
- ✓ This wizard will help us to perform the steps necessary for installing an Operating System onto this virtual machine. Follow the steps for installing the GuestOS.

## Week-2

### Aim: [2] Download and Install a Terminal Emulator:

#### Terminal emulator:

- A Terminal emulator is a computer program that reproduces a video terminal within some other display structure (i.e., remote machine desktop will be appeared in local machine as a terminal).
- In other words, the Terminal emulator has the ability to make a dumb machine appear like a client computer networked to the server.
- The terminal emulator allows an end-user to access the console as well as its applications such as text user interface and command-line interface.
- **Examples for Terminal Emulator:** Terminator, ROXTerm, Eterm, Tilix, LXTerminal, Konsole, Kitty, st, Gnome-Terminal, Terminology, Deepin Terminal, **xterm**, LilyTerm, Extraterm, **mate-terminal**, DomTerm, TermKit

#### Example 1: xterm

- The **xterm** terminal application is a standard terminal emulator for the X Window System
- Installation:

```
sudo apt-get update
```

```
sudo apt-get install xterm
```

#### Example 2: mate-terminal

- MATE Terminal is a terminal emulation application to access a UNIX shell in the MATE environment.
- MATE Terminal also has the ability to use multiple terminals in a single window (tabs) and supports management of different configurations(profiles).
- MATE Terminal is a fork of GNOME-Terminal.
- Installation:

```
sudo apt-get update -y
```

```
sudo apt-get install -ymate-terminal
```

## Week-3

**Aim: [1] File and Directory commands: Create and delete directories and files, File movement, copy commands, Pipes (named & unnamed)**

### ls (list)

- Use **ls** without any arguments to display current directory contents.
- **ls** with the **-a** option. files all begin with a **"dot"**, which indicates they are **"hidden"** files.

**ls -a**

- **ls** with **-F**, this command is useful for distinguishing between directories, ordinary files, and executable files.

**ls -F**

- **ls** with **-l** to obtain a "long" listing of files. An explanation of the information it provides appears below.

**ls -l**

-rwxr-xr-x	1 jsmith	staff	43	Mar 23 18:14 prog1
-rw-r--r--	1 jsmith	staff	10030	Mar 22 20:41 sample. f
drwxr-sr-x	2 jsmith	staff	512	Mar 23 18:07 subdir1
drwxr-sr-x	2 jsmith	staff	512	Mar 23 18:06 subdir2
drwxr-sr-x	2 jsmith	staff	512	Mar 23 18:06 subdir3
<b>1</b>	<b>2 3</b>	<b>4</b>	<b>5</b>	<b>6 7</b>

1= accessmodes/permissions

5 = size (inbytes)

2 = numberoflinks

6 = date/time of lastmodification

3 =owner

7 = name of file

4 = group

- **ls** with **R**, Recursive listings which will display files and folders inside the folders recursively.

**ls -R**

**ls -Rl**

- Will give the present working directory path information

### **mkdir (Make Directory)**

- mkdir will create a new directory(folder)
- **Syntax: mkdir directoryname (foldername**

### **cd (ChangeDirectory)**

- cd is used to ChangeDirectory
- Change to homedirectory
  - **cd** // Change to default homedirectory
  - **cd~** // Change to default homedirectory
- Change to a subdirectory within user homedirectory
  - **cdpolytechnic/computer**
  - pwd
- Go up one level back to the current directory's parentdirectory
  - **cd..**
  - pwd
- Change to the root (top-most) directory
  - **cd/**
  - pwd
- Change to another directory
  - **cd~/polytechnic**
  - pwd
- Change to another one of subdirectories
  - **cd~/polytechnic/computer**
  - pwd

### **rmdir (Remove Directory)**

- Will remove (delete) the directory
- Make sure that folder is empty before issuing rmdir command.
- Make sure current directory is not part of the directory being deleted.
  - **rmdir civil** (assume current directory is /home/polytechnic)

### **rm (Remove File)**

- Will remove (delete) the files

**Syntax: rmFileName**

- **rmpath/Filename**

### **cp (Copy)**

- Used to create a copy of an existing file.
- Copy an existing file in current directory to another file in the current directory. Make sure that the file to be copied must be exists in specified location.
  - **Syntax: cp file1file2**
- **cp a.txt b.txt** (Make sure file a.txt exists in current directory)
  - new file b.txt is created and contents of b.txt is same as a.txt
    - **cp polytechnic/civil/a.txtpolytechnic/computer/b.txt**
  - new file b.txt will be created in polytechnic/computer (Make sure both polytechnic/civil and polytechnic/computer folder exists)
- In order to avoid accidental over writing, **-i option** can be used. It will alter the user when file2 is already exists.
  - **cp -i file1file2** (to test, make sure both files exist)
- Use the recursive option to copy an entire subdirectory to a new subdirectory and then list both directories to prove that it worked:
  - **cp -R subdir1subdir4**
- Copying a file from another location to the current directory and want its name to remain the same, user can use the shorthand "." to indicate the current directory.
  - **cp polytechnic/civil/a.txt.**

## mv (Move)

- Used to rename the file (in other words, used to move the files from one location to another location)
- It can be used to rename the Directory (Folder)also.

- **Example 1:**

- mv OldFileNameNewFileName
- ls

- **Example 2:**

- mv OldFolderNameNewFolderName
- ls

**Pipes:** The pipe command lets user sends the output of one command to another. **Piping**, as the term suggests, can redirect the standard output, input, or error of one process to another for further processing. A traditional pipe is “**unnamed**” and lasts only as long as the process.

### Named Pipe:

- In computing, a named pipe (also known as a **FIFO**) is one of the methods for inter-process communication.
- It is an extension to the traditional pipe concept on Unix.
- Usually, a named pipe appears as a file and generally processes attach to it for inter-process communication.
- **Creating a FIFO file:** In order to create a FIFO file, a function calls i.e., **mkfifo** is used.
- **Example:** a.c and b. communication

### Unnamed Pipe:

- pipe () — Create an unnamed pipe
- **Format:** Fraction of C Code for the usage of pipe ()

```
#define _POSIX_SOURCE #include
<unistd.h>

...

int pipe (int fdinfo [2]);

... pipe
()
```

## Week-3

**Aim: [2] File and Directory commands: Commands for viewing File, File comparison, File manipulation, Altering file permission, File compression and decompression.**

### File Manipulation Commands:

- **Creating File:**

- **touch:** will create empty files
  - Example: touch a.txtb.txt
- **Create file and add contents** at the end press enter key and **Ctrl+z**
  - Example:

**cat> a.txt**

Welcome to Operating System Lab

Enjoy Typing commands

**Ctrl+z**

**cat a.txt**

- **Display the Contents of Files:**

- **Syntax:** catfilename
  - cat/path/filename

- **Example:**

- **cata.txt**

- **catpolytechnic/computer/a.txt**(Assume a.txt is present in polytechnic/computer directory)

### more

- more is a filter for paging through text one screenful at a time
- Use the more command to read a file:
  - **morefilename**
- Press Space bar / Return Key to read the remaining parts of the file.
- Press q to quit viewing contents



### less

- **less** is the opposite of more command
- Use the more command to read a file:
  - lessfilename
- Searching can be done by typing **/search key**, will highlights the searched words
- Example:       **lesstest.c**
  - **/printf**

### cmp (Compare)

- compare two files byte by byte
- Syntax: cmp -b file1file2
  - Output the first/last part of files (by Default first/last 10lines)

### head and tail

- **headfilename**
- **tailfilename**

### File Permissions

- There are **three** users in Linux namely **owner(u)**, **group(g)**, **others(o)**.
  - Note: For all users can be identified with letter a
- All three uses will have three possible permissions namely **read(r)**, **write(w)**, **execute(e)**
- Numbers assigned for permissions are **read (4)**, **write (2)**, **execute (1)**
- Command used to change permission of file is **chmod**
  - **Syntax:**    **chmod**       **filePermissionPattern**       **filename**

- **PermissionTable:**

- **OctalTable:**

binary	octal	permissions
000	0	- - -
001	1	- - x
010	2	- w -
011	3	- w x

binary	octal	permissions
100	4	r - -
101	5	r - x
110	6	r w -
111	7	r w x

- **Example:command:** **ls -lh**

**Output:**

```
-rw-rw-r-- 1 admincs admincs 805 Mar 28 13:42 b.c
drwx ----- 2 admincs admincs 4.0K Mar 28 09:12 OS
-rwxrw-r-- 1 admincs admincs 1.9M Mar 28 08:45 output.pdf
drwxrwxr-x 2 admincs admincs 4.0K Jan 23 2021 Scratch Projects
```

**Explanation**

First letter (-) indicates file (in this example b.c, output.pdf are files)

First letter (d) indicates item is directory (in this example OS, Scratch Projects are Directories)

Next 9 characters indicates File Permissions for different uses

In this example, file **output.pdf** has permission **r w x r w - r - -** which indicates

- Owner has r w x Permissions, means file owner can read, write and execute the file
- Group users has r w - Permissions, means Group users can only read and write the File contents
- Other users have r - - Permissions, means Other users can only read the file contents

### Examples

- add execution permission to group and other users
  - `chmod go+xoutput.pdf`
  - `ls output.pdf-l`
  - `-rwxrwxr-x 1 admincs admincs 1.9M Mar 28 08:45output.pdf`
- remove write permissions to group users
  - `chmod g-woutput.pdf`
  - `ls output.pdf-l`
  - `-rwxr-xr-x 1 admincs admincs 1.9M Mar 28 08:45output.pdf`
- Keep only read permission to all users
  - `chmod a=routput.pdf`
  - `ls output.pdf-l`
  - `-r--r--r-- 1 admincs admincs 1.9M Mar 28 08:45output.pdf`
- Adding execution to all users
  - `ls output.pdf-l`
  - `chmod a+xoutput.pdf`
  - `-r-xr-xr-x 1 admincs admincs 1.9M Mar 28 08:45output.pdf`
- giving only writing permission to other users (all settings may be not valid)
  - `chmod o=woutput.pdf`
  - `ls output.pdf-l`
  - `-r-xr-x-w- 1 admincs admincs 1.9M Mar 28 08:45output.pdf`
- give user all permission, group users read and write, and other user only read permissions
  - `chmod u=rwx,g=rw,o=r output.pdf`
  - `ls output.pdf-l`
  - `-rwxrw-r-- 1 admincs 1.9M Mar 28 08:45output.pdf`

## Working with numbers in chmod

**Syntax: chmod chmod number filename**

**Example 1:** if user wants rwxrw-r-- format set number to 764

- 777 =rwxrwxrwx
- 765 =rwxrw-r-x
- 654 =rw-r-xr--

**Example 2:**

- chmod 777 output.pdf will set all permissions to all users
- chmod 765 output.pdf will set rwxrw-r-x to users
- chmod 654 output.pdf will set rw-r-xr-- to users

=== \* ===

## unmask

- While creating a file or directory, by default a set of permissions are applied. These default permissions are viewed by **umask** command.
- For safety reasons all Unix systems doesn't provide execution permission to newly created files.
- Adding execution permission is up to user.

- **umask**

Output: **0002**

**Example 1:**

- \$ touch a.txt
- \$ ls -la.txt
- **-rw-rw-r--** 1 admincs admincs 28 Mar 28 14:32a.txt
  - the default permission for file is (u,g,o) = (6-0,6-0,6-2) = (6,6,4). hence it is **rw**

**- r w - r - -**

### File Compression and Decompression:

**Compression** reduces the size of an application or document for storage or transmission. Compressed files are smaller, download faster, and easier to transport. **Decompression** or expansion restores the document or application to its original size.

### Compressing files:

Syntax	Description	Example(s)
<b>gzip {filename}</b>	<ul style="list-style-type: none"> <li>Gzip compresses the size of the given files using Lempel-Ziv coding (LZ77).</li> <li>Whenever possible, each file is replaced by one with the extension.gz.</li> </ul>	<pre>gzip mydata.doc gzip *.jpg ls -l</pre>
<b>bzip2 {filename}</b>	<ul style="list-style-type: none"> <li>bzip2 compresses files using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding.</li> <li>Compression is generally considerably better than that achieved by bzip command (LZ77/LZ78-based compressors).</li> <li>Whenever possible, each file is replaced by one with the extension.bz2.</li> </ul>	<pre>bzip2 mydata.doc bzip2 *.jpg ls -l</pre>
<b>zip {.zip-filename} {filename-to-compress}</b>	<ul style="list-style-type: none"> <li>zip is a compression and file packaging utility for Unix/Linux.</li> <li>Each file is stored in single .zip {.zip-filename} file with the extension.zip.</li> </ul>	<pre>zip mydata.zip mydata.doc zip data.zip *.doc ls -l</pre>
<b>tar -zcvf {.tgz-file} {files}</b> <b>tar -jcvf {.tbz2-file} {files}</b>	<ul style="list-style-type: none"> <li>The GNU tar is archiving utility but it can be used to compressing large file(s).</li> <li>GNU tar supports both archives compressing through gzip and bzip2.</li> <li>If user have more than 2 files then it is recommended to use tar instead of gzip or bzip2.</li> <li>-z: use gzip compress</li> <li>-j: use bzip2 compress</li> </ul>	<pre>tar -zcvf data.tgz *.doc tar -zcvf pics.tar.gz *.jpg *.png tar -jcvf data.tbz2 *.doc ls -l</pre>

### Decompressing files

Syntax	Description	Example(s)
<b>gzip -d { .gz file}</b> <b>gunzip { .gz file}</b>	<ul style="list-style-type: none"> <li>Decompressed a file that is created using gzip command.</li> <li>File is restored to their original form using this command.</li> </ul>	gzip -d mydata.doc.gz gunzip mydata.doc.gz
<b>bzip2 -d { .bz2-file}</b> <b>bunzip2 { .bz2-file}</b>	<ul style="list-style-type: none"> <li>Decompressed a file that is created using bzip2command.</li> <li>File is restored to their original form using this command.</li> </ul>	bzip2 -d mydata.doc.bz2 gunzip mydata.doc.bz2
<b>unzip { .zip file}</b>	Extract compressed files in a ZIP archive.	unzip file.zip unzip data.zip resume.doc
<b>tar -zxvf { .tgz-file}</b> <b>tar -jxvf { .tbz2-file}</b>	Unbar or decompressed a file(s) that is created using tar compressing through gzip and bzip2 filter	tar -zxvf data.tgz tar -zxvf pics.tar.gz *.jpg tar -jxvf data.tbz2

### List the contents of an archive/compressed file

Sometime user just wanted to look at files inside an archive or compressed file. Then all of the above command supports file list option.

Syntax	Description	Example(s)
<b>gzip -l { .gz file}</b>	List files from a GZIP archive	gzip -l mydata.doc.gz
<b>unzip -l { .zip file}</b>	List files from a ZIP archive	unzip -l mydata.zip
<b>tar -ztvf { .tar.gz}</b> <b>tar -jtvf { .tbz2}</b>	List files from a TAR archive	tar -ztvf pics.tar.gz tar -jtvf data.tbz2

Important options in tar manual

- **-c : create archive**
  - -x :extract
  - -j : use bzip2 compress
  - -z : use gzip compress
  - -v : verbose mode
  - -f : use archive file or device ARCHIVE

## Week-3

**Aim : [3] File and Directory commands: Text processing commands.**

### Text Processing Commands

#### Commands affecting text and text files

#### sort

- **Sort** command is used to sort a file, arranging the records in a particular order.
- By default, the sort command sorts file assuming the contents are ASCII.
- Using options in the sort command can also be used to sort numerically.
- The sort command can also sort by items not at the beginning of the line, ignore case sensitivity, and return whether a file is sorted or not.
- Sorting is done based on one or more sort keys extracted from each line of input.
- By default, the entire input is taken as the softkey.
- Blank space is the default field separator.

#### Syntax:

**sort [OPTIONS] filename**

Options used:

- -o : used to save sorted content in specified file
  - sort -o output.txtinputfile.txt
- r : sort the file contents in descending order
  - sort -rinputfile.txt
- -n :if the file contains numbers, based on number values, file contents can be sorted.
  - sort -ninputfile.txt
- -k : sort the file contents based on kth field values
  - sort -k2inputfile.txt // sorts file contents based on values in 2<sup>nd</sup> field.
- -u :will remove duplicate entries in file while sorting.

- `uniq` filter removes duplicate lines from a sorted file. It is often seen in a pipe coupled with `sort`.
  - `uniq -c filename`
- removes the duplicate lines as well as displays frequency of duplicate lines.

### cut

- A tool for extracting fields from files.
- Important options are `-d`, which specifies delimiter to separate fields, `-f` which indicates field numbers
- **Example1:**      `cut -d" " -f1-4,6,8 FileName`
  - here contains information separated by space. This command displays contents in fields 1,2,3,4 and 6 and 8

### join

- Join allows merging two files in a meaningful fashion, which essentially creates a simple version of a relational database.
- The **join** command operates on exactly two files, but pastes together only those lines with a common tagged field (usually a numerical label), and writes the result to `stdout`.
- The files to be joined should be sorted according to the tagged field for the matchups to work properly.

### head and tail

- **head/tail:** output the first/last part of files (by Default 10 lines)
  - `head filename`
  - `tail filename`
- **Example1:**      `head -5 a.txt`                      will display first 5 lines of a.txt  
                         `tail -5 a.txt`                      will display last 5 lines of a.txt



- A multi-purpose file search tool that uses Regular Expressions. (Global Regular Expression Pring).
  - **greppattern[Filename...]**
- options used with grep are:
  - The -i option causes a **case-insensitive** search.
  - The -w option matches only **whole words**.
  - The -r (recursive) option searches files in the current working directory and all subdirectories below it.
  - The -n option lists the matching lines, together with **line numbers**.
  - The -c (--count) option gives a numerical count of matches.
    - **Example 1:**
      - To see every process on the system using BSD syntax:
      - psax
      - But search only word **swap** associated with the processes.
      - **Output:**
        - **ps ax | grepswap**
        - 55?      S    0:00[kswapd0]
        - 4206 pts/0   S+    0:00 grep --color=auto **swap**

### wc

- wc gives a "word count" on a file or I/Ostream:
- Important Options Used:
  - **wc -l** gives only the line count.
  - **wc -w** gives only the wordcount.
  - **wc -c** gives only the byte count.
- **Syntax:**            **wc -lwcFileName**
  - **Example:**
    - **wc -lwctemp2.txt**

## tr

- character translation filter.
  - **Example 1: tr "A-Z" "\*" <filename**
    - will translate all the UPPER-CASE characters to \* in a file
  - **Example 2: tr "0-9" "A" <filename**
    - will translate all the DIGITS to 'A' in a file
- The -c "complement" option inverts the character set to match.
  - **Example3: echo "acfdeb123" | tr -c b-d+**
    - +c+d+b++++
  - **Example 4: echo "abcd2ef1" | tr '[:alpha:]' -**
    - .....2-1
- same as **echo "abcd2ef1" | tr 'a-zA-Z'-**
  - -----2 - 1

## fold

- A filter that wraps lines of input to a specified width. This is especially useful with the -s option, which breaks lines at word spaces.
- **Example1: fold -w 3FileName**
  - Will wrap each line for every three characters
    - **Example2: fold -s -w3**
  - Will wait for the user to enter text, after return key, the text will be wrapped to every three characters.

## nl

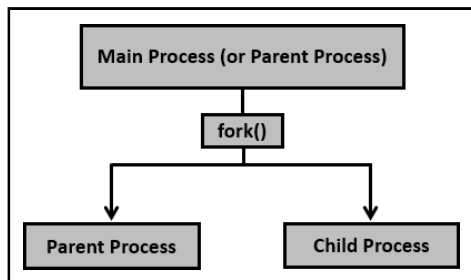
- Line numbering filter: **nl filename** lists filename to stdout, but inserts consecutive numbers at the beginning of each non-blank line.
- If filename omitted, operates on stdin.

## Week-4

**Aim : [1] Linux commands related to process creation and management- system calls fork() and exec(); bg, fg, nohup, pkill, nice, top, ps;**

### Process

- A **process** is any active (running) instance of a program. In other words, process is a program in execution.
- A new process can be created **by the fork() system call**.
- The new process consists of a copy of the address space of the original process. Fork() creates new process from existing process.
- Existing process is called the **parent process** and the process is created newly is called **child process**.



- Each process is given a unique process identification number(PID).
- PID is usually five-digit number.
  - This number is used to manage each process.
  - user can also use the process name to manage process.

### Starting a Process

A process executes in two ways they are,

- Foreground Processes
- Background Processes

### Foreground Processes

- Every process by default runs in Foreground (which means the output is printed on the screen.) The best example for the foreground process is the **ls** command which prints the output on the screen by listing the files and directories.
- When a program is running in the foreground, user cannot start another process without completing the previous process. [ It is because of this reason foreground process is considered a time-consuming process.]

### Background Processes

- When a process starts running and it is not visible on the screen it is called a background process.
- User can simultaneously run 'n' number of commands in the background process.
- To enable the background process, provide ampersand symbol (&) at the end of the command.
  - **Example :ls&**

### ps

The default output of ps is a simple list of the processes running in current terminal.

- **Example: ps**

PID	TTY	TIME	CMD
23989	pts/0	00:00:00	bash
24148	pts/0	00:00:00	ps

- Every running process (-e) and a full listing (-f) could be obtained with these options.

## Demonstrate exec( )system call

The exec() family of functions replaces the current process image with a new process image.

**exec** command in Linux is used to execute a command from the **bash** itself. This command does not create a new process it just replaces the bash with the command to be executed. If the exec command is successful, it does not return to the calling process.

**exec> output.txt**

- Redirect all output to the file **output.txt** for the current shell process.
- Redirections are a special case, and **exec** does not destroy the current shell process, but **bash** will no longer print output to the screen, writing it to the file instead.

**Example2: ( try this command in sudobash)**

- **exec3<output.txt** // no space between **3** and **<**
- Open **output.txt** for reading ("**<**") on file descriptor**3**.
- The above command is an example of explicitly opening a file descriptor.
- After running the above command, user can read a line of **output.txt** by running **read** command with the **-u** option:
  - **read -u 3mydata**
    - Here, "**-u 3**" tells **read** to get its data from file descriptor 3, which refers to **output.txt**. The contents are read, one line at a time, into the variable **my data**. This would be useful if used as part of a **while** loop, for example.
- **exec3<output.txt** // Assume contents of output.txt is **March2022**
  - **\$ read-u 3oneline**
- **\$ echo\$oneline**
  - **March2022**

**Example3: ( try this command in sudobash)**

- **exec 4> out.txt**
- The above command opens **out.txt** for writing ("**>**") on file descriptor**4**.

## Example 4:

- **exec 6>> append.txt**
- Open **append.txt** for appending (">>") as file descriptor 6.

<pre>\$ exec 6&gt;&gt; append.txt \$ echo append line one &gt;&amp;6 \$ echo append line Two &gt;&amp;6 \$ echo append line Three &gt;&amp;6</pre>	<p><b>Output:</b></p> <pre>\$ cat append.txt append line one append line Two append line Three</pre>
--	--

## bg: background

- **Bg** used to place foreground jobs in background. [used to send a process running in the foreground to the background in the current shell.]

### Syntax: bg [ job spec]

- Place the jobs identified by each *job spec* in the background, as if they have been started with '&'.
  - *Job spec* maybe
    - %n : Refer to job number.
    - %%or%+ : Refer to the current job.
    - %- : Refer to the previous job.

- **Example: sleep 500** is a command which is used to create a dummy job which runs for 500seconds.

- Use jobs command to list all jobs
  - Create a process using sleep command, get job id as 1
  - Put that process in background using id
- **\$jobs** // Display current jobs running and displayed nothing
- **\$sleep 500** // Create one job, which sleeps for 500seconds.
- ^z // Terminate the job by pressing Ctrl +z
  - [1]+ Stopped sleep500
- **\$jobs** // Display current jobs running and displayed one job stopped
  - [1]+ Stopped sleep500

- **\$bg%1** // Refer the process by job number **place it in background**
  - **[1]+ sleep 500&**
- **\$ jobs** // Display current jobs running and displayed one job which is running.
  - **[1]+ Running sleep500**

### fg: foreground

- **B gis** a command that moves a background process on current Linux shell to the foreground.

**Syntax: bg [ job spec]**

Place the jobs identified by each job spec in the foreground, making it the current job. **Example:** sleep 500 is a command which is used to create a dummy job which runs for 500 seconds.

- **\$jobs**
- **\$ sleep500**
- **^z**

### nohup

- **nohup**, short for **no hang up** is a command in Linux systems that **keep processes running even after exiting the shell or terminal**.
- nohup prevents the processes or jobs from receiving the SIGHUP (Signal Hang UP) signal.
- This is a signal that is sent to a process upon closing or exiting the terminal.
- Every command in Linux starts a process at the time of its execution, which automatically gets terminated upon exiting the terminal.
- Usually, every process in Linux systems is sent a **SIGHUP (Signal Hang UP)** which is responsible for terminating the process after closing/exiting the terminal.
- **Nohup** command prevents the process from receiving this signal upon closing or exiting the terminal/shell.
- If the output of the nohup command is redirected to some other file, **nohup. Out** file is not generated.

**Syntax: nohup command [command-argument ...]**

- nohup command can be used to run multiple commands in the background.
- Syntax: nohup bash -c 'commands'

## Example 1:

- `nohup bash -c 'cal &&ls'`
- Here, the output will be by default stored in **nohup.out**. To redirect it, type:
  - `nohup bash -c 'commands' > filename.txt`

## Example 2:

- `nohup bash -c 'cal &&ls' >output.txt`

## kill

- kill is used to send a signal to a process. The most commonly used signal is "terminate" (SIGTERM) or "kill" (SIGKILL). The full list can be shown with **kill -L**

o 1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL	5) SIGTRAP
o 6) SIGABRT	7) SIGBUS	8) SIGFPE	9) <b>SIGKILL</b>	10) SIGUSR1
o 11) SIGSEGV	12) SIGUSR2	13) SIGPIPE	14) SIGALRM	15) <b>SIGTERM</b>

- Signal number nine is SIGKILL. The default signal is 15, which is SIGTERM.
  - o Issue a command such as **kill -920896**.

## pgrep

- command used to send signal to kill process by process name.
- while issuing pgrep, make sure that selected process name is the correct process to be stopped, verify it by its full path by issuing `pgrep -fprocess name`.
- Syntax: **pgrep -fProcess Name**



### nice

- **Nice** command in Linux **helps in execution of a program/process with modified scheduling priority.**
- It launches a process with a user-defined scheduling priority.
- The **renice command** allows user to change and modify the scheduling priority of an already running process.
- Linux Kernel schedules the process and allocates CPU time accordingly for each of them.
- The kernel stores a great deal of information about processes including process priority which is simply the scheduling priority attached to a process.
- Processes with a higher priority will be executed before those with a lower priority, while processes with the same priority are scheduled one after the next, repeatedly.
- There are a total of **140** priorities and two distinct priority ranges implemented in Linux.
- The first one is a nice value (**niceness**) which ranges from -20 (highest priority value) to 19 (lowest priority value) and the default is 0.
- The other is the real-time priority, which ranges from **1 to 99** by default, then **100 to 139**

## Check Nice Value of Linux Processes

**ps -eo pid, ppid, ni, comm**

- pidprocess-id,ppidparentprocess-id,ninicenessofprocess,commcommandforthat process
- Alternatively, user can use **top** or **htop** utilities to view Linux processes nice values
- In htop command output, **PRI** – is the process's actual priority, as seen by the Linux kernel.
- **rt** value in PRI fields, means the process is running under **real-time** scheduling priority

### Important:

- If no value is provided, nice sets a priority of 10 by default.
- A command or program run without nice defaults to a priority of zero.
- Only root can run a command or program with increased or high priority.
- Normal users can only run a command or program with low priority.

### Syntax:

```
$ nice -n niceness-value [command rags] OR
$ nice -niceness-value[commandargs]      #it's confusing for negative values
OR
$ nice --adjustment=niceness-value [command args]
```

### Example:

```
$ sudo nice -n 5 tar -czf
backup.tar.gz./Documents/* OR
$ sudo nice --adjustment=5 tar -czf
backup.tar.gz./Documents/* OR
$ sudo nice -5 tar -czf backup.tar.gz./Documents/*
```

## top

- The top command has been around a long time and is very useful for viewing details of running processes and quickly identifying issues such as memory hogs.
- Its default view is shown below. **Example 1:top**

top - 11:56:28 up 1 day, 13:37, 1 user, load average: 0.09, 0.04, 0.03

Tasks: 292 total, 3 running, 225 sleeping, 0 stopped, 0 zombie

%Cpu(s): 0.1 us, 0.2 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

KiB Mem : 16387132 total, 10854648 free, 1859036 used, 3673448 buff/cache

KiBSwap: 0 total, 0 free, 0 used. 14176540 availMem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
17270	alan	20	0	3930764	247288	98992	R	0.7	1.5	5:58.22	gnome-shell
20496	alan	20	0	816144	45416	29844	S	0.5	0.3	0:22.16	gnome-terminal-
21110	alan	20	0	41940	3988	3188	R	0.1	0.0	0:00.17	top
1	root	20	0	225564	9416	6768	S	0.0	0.1	0:10.72	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.08	ksoftirqd/0

- The update interval can be changed by typing the letter **s** followed by the number of seconds the user prefers for updates.
- To make it easier to monitor required processes, user can call top and pass the PID(s) using the **-p** option.

## Week-4

### Aim : [2] Cron and at commands to schedule tasks.

#### Commands to Schedule Tasks:

##### Cron

- The **cron** is a software utility, offered by a Linux-like operating system that automates the scheduled task at a predetermined time.
- It is a **daemon process**, which runs as a background process and performs the specified operations at the predefined time when a certain event or condition is triggered without the intervention of a user.
- Dealing with a repeated task frequently is an intimidating task for the system administrator and thus he can schedule such processes to run automatically in the background at regular intervals of time by creating a list of those commands using Cron.
- It enables the users to execute the scheduled task on a regular basis unobtrusively like doing the backup every day at midnight, scheduling updates on a weekly basis, synchronizing the files at some regular interval.
- Cron checks for the scheduled job recurrently and when the scheduled time fields match the current time fields, the scheduled commands are executed.
- It is started automatically from /etc./init.d on entering multi-user run levels.
- The **crontab** (abbreviation for “cron table”) is list of commands to execute the scheduled tasks at specific time. It allows the user to add, remove or modify the scheduled tasks.
- The crontab command syntax has **six fields** separated by space where the **first five** represent the **time to run the task** and **the last one is for the command**.
  - Minute (holds a value between 0-59)
  - Hour (holds value between 0-23)
  - Day of Month (holds value between 1-31)
  - Month of the year (holds a value between 1-12 or Jan-Dec, the first three letters of the month’s name shall be used)
  - Day of the week (holds a value between 0-6 or Sun-Sat, here also first three letters of the day shall be used)
  - Command (**6<sup>th</sup>Field**)

### The rules which govern the format of date and time field as follows:

- When any of the first five fields are set to an asterisk(\*), it stands for all the values of the field. For instance, to execute a command daily, we can put an asterisk(\*) in the week's field.
- One can also use a range of numbers, separated with a hyphen(-) in the time and date field to include more than one contiguous value but not all the values of the field. For example, we can use the 7-10 to run a command from July to October.
- The comma ( , ) operator is used to include a list of numbers which may or may not be consecutive. For example, "1,3,5" in the weeks' field signifies execution of a command every Monday, Wednesday, and Friday.
- A slash character(/) is included to skip given number of values. For instance, "\* /4" in the hour's field specifies 'every 4 hours' which is equivalent to 0, 4, 8, 12, 16, 20.

### Permitting users to run cron jobs:

- The user must be listed in this file to be able to run cron jobs if the file exists.
  - /etc./cron.allow
- If the cron.allow file doesn't exist but the cron.deny file exists, then a user must not be listed in this file to be able to run the cronjob.
  - /etc./cron.deny

- **Note:** If neither of these files exist then only the superuser (system administrator) will be allowed to use a given command.

### Example to Try:

- As cron processes will run in background, so to check whether scripts are executed or not, one way to check the log file i.e., /var/log/syslog
- Search for cron in syslog using command: **cat /var/log/syslog/ | grep cron**
- User can see all the entries, which shows, the script executed at a scheduled time mentioned in crontab file.

=== \* ===

### Simple Example:

**FileName:** task.sh                      **path:** /home/admincs/(say for example)

### Contents:

```
echo Welcome to Task Scheduler Demo
echo Try dateCommand
date

echo Try timeCommand
time

echo List all file/folders in a home directory
ls

echo End of Task Scheduler Demo
```

- Change the execution permission to task.sh

**chmod 764 task.sh**

- Add the task to /etc./crontab (Task will be executed at 4.13pm, User can Change timings according to the requirements)

**13 16 \* \* \* root sh /home/admincs/task.sh > /home/admincs/output.txt**

- Save and Exit. And Wait till 4.13pm
- After that Check the output.txt by the command

**cat /home/admincs/output.txt**

- Output of the task.sh is present in output.txt.
- Alternatively, /var/log/syslog can be verified about the execution of the Task.

### Example 1:

- Run /home/folder/gfg-code.sh every hour, from 9:00 AM to 6:00 PM, every day.

- **00 09-18 \* \* \*/home/folder/gfg-code.sh**
- Run /user/local/bin/backup at 11:30 PM, every weekday.
  - **30 23 \* \* Mon, Tue, Wed, Thu, Fri/user/local/bin/backup**
- Run sample-command.sh at 07:30, 09:30, 13:30 and 15:30.
  - **30 07, 09, 13, 15 \* \* \*sample-command.sh**

### at

- **at command** is a command-line utility that is used to schedule a command to be executed at a particular time in the future.
- Jobs created with **at command** are executed only once.
- The **at command** can be used to execute any program or mail at any time in the future.
- It executes commands at a particular time and accepts times of the form HH:MM to run a job at a specific time of day.
- In the command, expression like noon, midnight, teatime, tomorrow, next week, next Monday, etc. could be used with **at command** to schedule a job.

Syntax: **at [OPTION...] runtime**

#### Working with at command

- Command to list the user's pending jobs:
  - **at-l**
- OR
- **atq**
- Schedule a job for the coming Monday at a time twenty minutes later than the current time:
  - **at Monday +20minutes**
- Schedule a job to run at 1:45 Aug 12 2020:
  - **at 1:45081220**
- Schedule a job to run at 3pm four days from now:
  - **at 3pm + 4days**
- Schedule a job to shut down the system at 4:30today:
  - **echo "shutdown -h now" | at -m4:30**

- Schedule a job to run five hours from now:
  - **at now +5hours**
  
- at-r or atrm command is used to deletes job , here used to deletes job11.

**at-r 11**

**OR**

**atrm11**

```
cipers@cipers:~$ at -l
14      Wed Jun 24 06:16:00 2020 a cipers
11      Sun Jun 28 05:34:00 2020 a cipers
12      Fri Jun 26 13:00:00 2020 a cipers
15      Wed Jun 24 06:17:00 2020 a cipers
13      Wed Oct 21 12:30:00 2020 a cipers
cipers@cipers:~$ at -r 12
cipers@cipers:~$ at -l
14      Wed Jun 24 06:16:00 2020 a cipers
11      Sun Jun 28 05:34:00 2020 a cipers
15      Wed Jun 24 06:17:00 2020 a cipers
13      Wed Oct 21 12:30:00 2020 a cipers
```

```
cipers@cipers:~$ atrm 11
cipers@cipers:~$ at -l
14      Wed Jun 24 06:16:00 2020 a cipers
15      Wed Jun 24 06:17:00 2020 a cipers
13      Wed Oct 21 12:30:00 2020 a cipers
cipers@cipers:~$ atrm 14
cipers@cipers:~$ at -l
15      Wed Jun 24 06:17:00 2020 a cipers
13      Wed Oct 21 12:30:00 2020 a cipers
cipers@cipers:~$
```

- The /etc./at.deny and /etc./at.allow files allow user to control which users can create jobs with at or batch command. The files consist of a list of usernames, one user name per line.
- By default, only the /etc./at.deny file exists and is empty, which means that all users can use the at command. If user want to deny permission to a specific user, add the username to this file.



## Week-5

**Aim : [1] Commands to exhibit thread concepts.**

### Commands to exhibit thread concepts:

- Threads are a popular programming abstraction for parallel execution on modern operating systems.
- When threads are forked inside a program for multiple flows of execution, these threads share certain resources (e.g., memory address space, open files) among themselves to minimize forking overhead and avoid expensive IPC (inter-process communication) channel.
- These properties make threads an efficient mechanism for concurrent execution.
- In Linux, threads (also called Lightweight Processes (LWP)) created within a program will have the same "thread group ID" as the program's PID.
- Each thread will then have its own thread ID (TID).
- Threads are nothing more than standard processes which happen to share certain resources.
- Classic command-line tools such as `ps` or `top`, which display process-level information by default, can be instructed to display thread-level information.

### Using the `ps`

The "-T" option for the `ps` command enables thread views.

**`ps -T -p <pid>`**

**Example:** List the threads for the **mate-terminal** process:

- First find the all process containing word **terminal**
- The "**SPID**" column represents **thread IDs**, and "CMD" column shows thread names.

### Using the top

The top command can show a real-time view of individual threads.

To enable thread views in the top output, invoke top with "-H" option. This will list all Linux threads.

User can also toggle on or off thread view mode while top is running, by pressing 'H' key.

#### Example 1: top

To restrict the top output to a particular process and check all threads running inside the process:

#### Example 2: top -H -p 1572

### Using htop

A more user-friendly way to view threads per process is via htop, a ncurses-based interactive process viewer.

This program allows user to monitor individual threads in tree views. To enable thread views in htop, launch htop, and press F2 to enter htop setup menu.

choose "Display option" under "Setup" column, and toggle on "Tree view" and "Show custom thread names" options. Press F10 to exit the setup.

## Week-6

### Aim : [1] Commands to view memory consumption

#### Memory Management

Commands to view memory consumption:

##### 1. free command

- **Syntax:** `free -m`

**Output:**

	total	used	free	shared	buffers	cached
Mem:	7976	6459	1517	0	865	2248
-/+ buffers/cache:	3344	4631				
Swap:	1951	0	1951			

- The m option displays all data in MBs.
- The total of 7976 MB is the total amount of RAM installed on the system, that is 8GB.
- The used column shows the amount of RAM that has been used by Linux, in this case around 6.4 GB. The second line tells that 4.6 GB is free. This is the free memory in first line added with the buffers and cached amount of memory (1517 + 865 + 2248 around 4631 MB=4.6GB).
- Linux has the habit of caching lots of things for faster performance, so that memory can be freed and used if needed.
- The last line is the swap memory, which in this case is lying entirely free.

##### 2. /proc/meminfo

- Check memory usage by reading the file at **/proc/meminfo**.
- Know that the /proc file system does not contain **real files**.
- They are rather **virtual files** that contain dynamic information about the kernel and the system.

**Portion of output is**

```
$ cat /proc/meminfo MemTotal: 8167848kB
MemFree: 1409696kB
Buffers: 961452kB
Cached: 2347236kB
SwapCached: 0kB
SwapTotal: 1998844kB
SwapFree: 1998844kB
```

## 3. vmstat

Syntax

- **vmstat-s**

Portion of output is

8167848 K total memory  
 7449376 K used memory  
 718472 K free memory  
 1154464 K buffer memory  
 2422876 K swap cache  
 1998844 K total swap  
 0 K used swap 1998844 K free swap

- The top few lines indicate total memory, free memory etc. and soon.

## 4. top command

- The top command is generally used to check memory and Cpu usage per process.
- It also reports total memory usage and can be used to monitor the total RAM usage.
- The header on output has the required information.
- Check the KiB Mem and KiB Swap lines on the header. They indicate total, used and free amounts of the memory. The buffer and cache information are present hereto.

Portion of outputs

```
top - 15:20:30 up 6:57, 5 users, load average: 0.64, 0.44, 0.33
Tasks: 265 total, 1 running, 263 sleeping, 0 stopped, 1 zombie
%Cpu(s): 7.8 us, 2.4 sy, 0.0 ni, 88.9 id, 0.9 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 8167848 total, 6642360 used, 1525488 free, 1026876 buffers
KiB Swap:1998844total,          0 used, 1998844 free, 2138148 cached
PIDUSER          PR NIVIRT RES SHR S%CPU %MEM          TIME+COMMAND
2986 enlighte 20 0 584m 42m 26m S 14.3 0.5 0:44.27 yakuake
1305 root        20 0 448m 68m 39m S 5.0 0.9 3:33.98Xorg
7701 enlighte 20 0 424m 17m 10m S 4.0 0.2 0:00.12 kio_thumbnail
```

## 5. htop

- Similar to the top command, the htop command also shows memory usage along with various other details.
- The header on top shows Cpu usage along with RAM and swap usage with the corresponding figures.

### RAM Information

- To find out hardware information about the installed RAM, use the dmidecode command.

#### Portion of output is

```
sudo dmidecode -t 17 # dmidecode 2.11
```

SMBIOS 2.4 present.

Handle 0x0015, DMI type 17, 27 bytes Memory Device

Array Handle: 0x0014

Error Information Handle: NotProvided Total Width: 64bits

Data Width: 64bits

**Size: 2048 MB**

Form Factor: DIMM Set: None

Locator: J1MY

Bank Locator: CHAN A DIMM 0

**Type: DDR2**

Type Detail: Synchronous

**Speed: 667 MHz**

Manufacturer: 0xFF00000000000000 Serial Number: 0xFFFFFFFF

Asset Tag: Unknown

Part Number: 0x524D32474235383443412D36344643FFFFFFFF

- Provided information includes the size (2048MB), type (DDR2) , speed(667 Mhz)etc.

## Week-7

**Aim : [1] . Write shell scripts to illustrate decision making and different types of iterations; Ex- to perform string operations; to perform file operations;**

### Shell Programming

#### 1. Write shell scripts to illustrate decision making and different types of iterations;

- **if Statement:**

The if statement allows you to specify courses of action to be taken in a shell script, depending on the success or failure of some command. It is a conditional statement that allows a test before performing another statement. The syntax for the simplest form is:

```
if [ condition ]
then
    block_of_statements
fi
```

Here,

- The **condition** in the if statement often involves a numerical or string test comparison, but it can also be any command that returns a status of 0 when it succeeds and some nonzero status when it fails.
- The **statements** that follow the then statement can be any valid UNIX command, any executable user program, any executable shell script, or any shell statement with the exception of fi.
- End every if statement with the fi statement.

#### 1.

```
#!/bin/bash
age=21
if [ $age -gt 18 ]
then
echo "You are old enough to drive in most places."
fi
```

2.

```
#!/bin/bash
name=John
if [ $name = "John" ]
then
echo "John is here !!!"
fi
```

- **if then else statement:**

---

In a conditional, you frequently have tasks to perform when the tested condition succeeds or fails. The shell can accommodate this with the if/then/else syntax. In the if/then/else form of the if statement, the block of statements after the then statement is executed if the condition succeeds. Execution continues with the statement following the fi statement. If the condition in the if statement fails, then the block of statements after the then statement is skipped, and statements following the else are executed. Execution continues with the statement following the fi statement. The syntax for if/then/else is:

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

1.

```
#!/bin/bash
total=100
if [ $total -eq 100 ]; then
echo "total is equal to 100"
else
echo "total is not equal to 100"
fi
```

- **if/then/elif/else statement:**

In the if/then/elif/else form of the if statement, the first else becomes another if statement or “elif” instead of a simple else. The shell first evaluates condition 1, then condition 2, and so on, stopping with the first condition that succeeds. The statements associated with that successful condition are then executed, followed by any statements after the fi statement. If none of the condition succeeds, then the statements following the else statement are executed. Execution then continues with any statements following the fi statement. The syntax for if/then/elif/else is:

```
if [ condition 1]
then
    block_of_statements
elif [ condition 2]
then
    block_of_statements
else
    block_of_statements
fi
```

Below is an example of if/then/elif/else form of the if loop statement.

1.

```
#!/bin/bash
total=100
if [ $total -eq 100 ]
then
echo "total is equal to 100"
elif [ $total -lt 100 ]
then
echo "total is less than 100"
else
echo "total is greater than 100"
fi
```



2.

```
#!/bin/bash
```

```
name=snoopy
```

```
if [ "$name" = "snoopy" ] then
    echo "It was a dark and stormy night."
elif [ "$name" == "charlie" ]
then
    echo "You're a good man Charlie Brown."
elif [ "$name" == "lucy" ]
then
    echo "The doctor is in."
elif [ "$name" == "schroeder" ]
then
    echo "In concert."
else
    echo "Not a Snoopy character."
fi
```

```
$ ./ex1.sh
```

Enter some text: I like the Bash shell.

var contains: I

var contains: like

var contains: the

var contains: Korn

var contains: shell.

## Different types of iterations

basically, Bash Scripts allows three different iterative statements:

- For Loop
- While Loop
- Until Loop
- **Bash For Loop:**

This statement is present in all computer languages. Of course, each language has its own implementation, but the basic idea is still the same. The For Loop repeats the execution of a group of statements over a set of items. Those items can be a sequence of numbers or text strings like the name of files in a directory.

The basic syntax of the For Loop in Bash Scripting is as follows:

```
for [Iteration Variable] in [Iteration Array]; do [Iteration
Statements]
...
Done
```

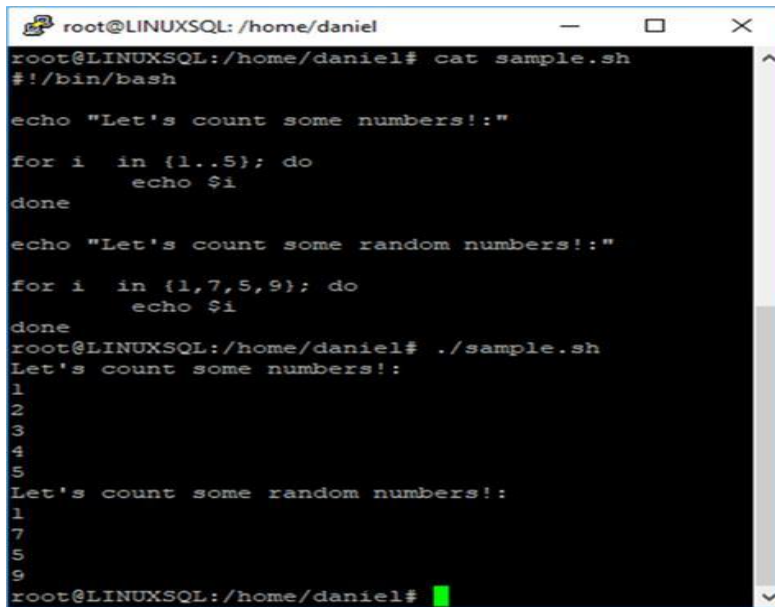
- [Iteration Variable]: This is a scalar variable (i.e., not an array) that holds one item at a time of [Iteration Array].
- [Iteration Array]: This can be either an array variable, a group of numbers or a command that returns a set of items like filenames.
- [Iteration Statements]: This contains the list of statements to be executed each iteration.

Now I will show you different examples of the For Loop. Let's consider the next case when the For Loop contains a sequence of numbers in the iteration array.

```
#!/bin/bash
echo "Let's count some numbers!" for i
in {1..5}; do
echo $i
done

echo "Let's count some random numbers!:" for

i in {1,7,5,9}; do
echo $i
done
```



```

root@LINUXSQL: /home/daniel
root@LINUXSQL:/home/daniel# cat sample.sh
#!/bin/bash

echo "Let's count some numbers!:"
for i in {1..5}; do
    echo $i
done

echo "Let's count some random numbers!:"
for i in {1,7,5,9}; do
    echo $i
done
root@LINUXSQL:/home/daniel# ./sample.sh
Let's count some numbers!:
1
2
3
4
5
Let's count some random numbers!:
1
7
5
9
root@LINUXSQL:/home/daniel#

```

- **Bash While Loop:**

The While Loop executes a set of commands as long as control condition remains true. If the control condition is a command, then its execution status must return zero for the iteration statements to execute. In the case where the control condition is an expression like a comparison of a variable with another, then the result of this comparison must return true.

Here is the basic syntax of the While Loop:

```

while [Control condition]
do
    [Iteration Statements]
done

```

You can stop the execution of a While Loop by using the "break" statement. This way even when the control condition remains true the while block stops its execution and Bash keeps running the following steps of the script. Another way to stop the execution of a While Loop in case you accidentally make an endless loop, is to send a kill signal to the process (you must use another terminal) or press CTRL+C. The difference between these methods and the "break" statement is that these methods stop the execution of the While Loop as well as the script and the "break" statement only stops the While Loop execution.

Now I will show you a few examples on how to use the While Loop.

In the next example I will show you a countdown from 5 to zero by using an endless loop

with a break statement. Notice that I have to add an IF statement in order to check the value of the variable.

```
#!/bin/bash
```

```
echo "Let's do a countdown!:"
```

```
VAR=5
```

```
while [ $VAR ];
```

```
do
```

```
echo $VAR
```

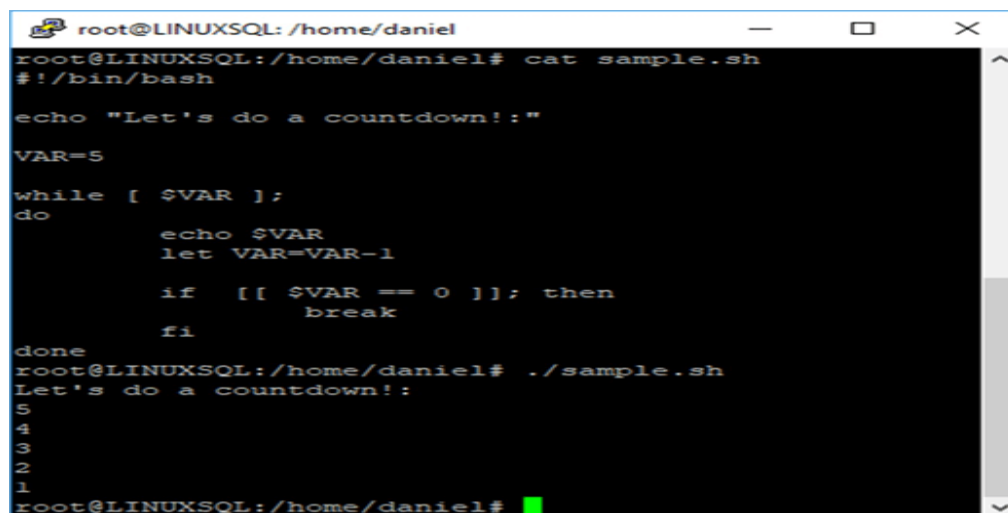
```
let VAR=VAR-1
```

```
if [[ $VAR == 0 ]]; then
```

```
break
```

```
fi
```

```
done
```

A terminal window titled 'root@LINUXSQL: /home/daniel' showing the execution of a shell script. The script content is displayed as it is being read with 'cat sample.sh'. It includes a shebang, an echo statement, a variable assignment, a while loop with an if-break condition, and a done statement. Below the script content, the output of running './sample.sh' is shown, displaying a countdown from 5 to 1. The prompt returns to the root user.

```
root@LINUXSQL: /home/daniel
root@LINUXSQL:/home/daniel# cat sample.sh
#!/bin/bash

echo "Let's do a countdown!:"

VAR=5

while [ $VAR ];
do
    echo $VAR
    let VAR=VAR-1

    if [[ $VAR == 0 ]]; then
        break
    fi
done
root@LINUXSQL:/home/daniel# ./sample.sh
Let's do a countdown!:
5
4
3
2
1
root@LINUXSQL:/home/daniel#
```

- **Bash Until Loop:**

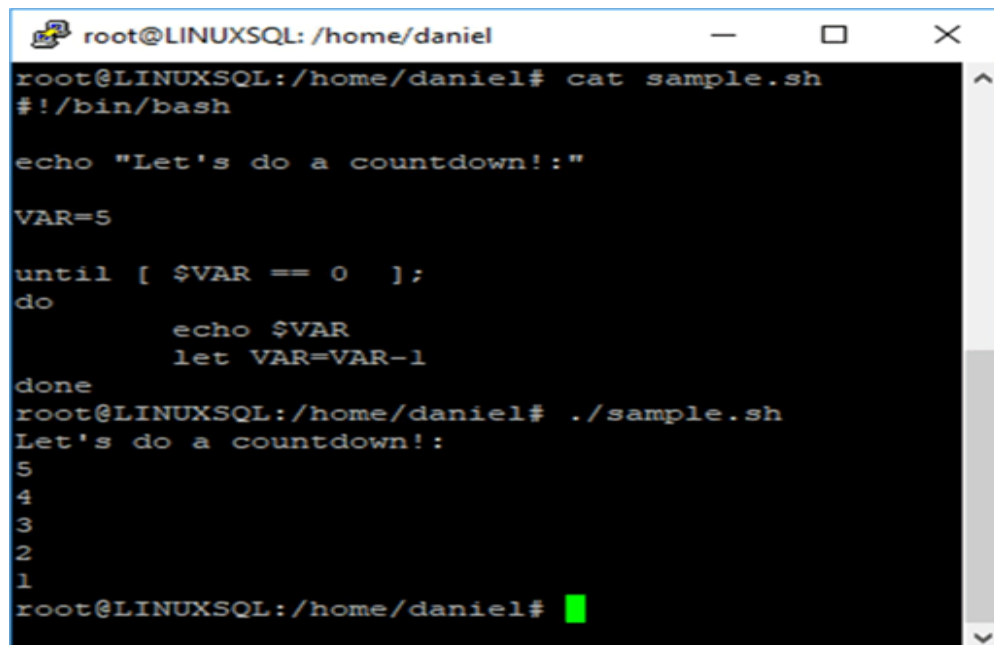
This iterative statement is almost identical to the While Loop, but it has a crucial difference. In the While Loop the execution of the iteration statements continues as long as the control condition remains true; but in the Until Loop the execution of the iteration statements continues as long as the control condition remains false.

```
Until [Control condition]
do
    [Iteration Statements]
done
#!/bin/bash
```

```
echo "Let's do a countdown!:"
```

```
VAR=5
```

```
until [ $VAR == 0 ];
do
    echo $VAR
    let VAR=VAR-1
done
```



```
root@LINUXSQL: /home/daniel
root@LINUXSQL:/home/daniel# cat sample.sh
#!/bin/bash

echo "Let's do a countdown!:"

VAR=5

until [ $VAR == 0 ];
do
    echo $VAR
    let VAR=VAR-1
done
root@LINUXSQL:/home/daniel# ./sample.sh
Let's do a countdown!:
5
4
3
2
1
root@LINUXSQL:/home/daniel#
```

## Week-8

**Aim : [1] Illustrate automation of basic tasks like monitoring memory consumption, check remote servers' connectivity, etc., at different frequencies.**

### Automation of System Tasks

**Illustrate automation of basic tasks like monitoring memory consumption, check connectivity, etc., at different frequencies.**

Sometimes, user may have tasks that need to be performed on a regular basis or at certain predefined intervals. Such tasks include backing up databases, updating the system, performing periodic reboots and so on. Such tasks in Linux are referred to as **cron jobs (Crontab)**. Cron jobs are used for **automation of tasks** that come in handy and help in simplifying the execution of repetitive and sometimes everyday tasks.

### Commands to Schedule Tasks:

#### **cron:**

- The **cron** is a software utility, offered by a Linux-like operating system that automates the  
     scheduled task at a predetermined time.
- It is a **daemon process**, which runs as a background process and performs the specified operations at the predefined time when a certain event or condition is triggered without the intervention of a user.
- Dealing with a repeated task frequently is an intimidating task for the system administrator and thus he can schedule such processes to run automatically in the background at regular intervals of time by creating a list of those commands using Cron.
- It enables the users to execute the scheduled task on a regular basis unobtrusively like doing the backup every day at midnight, scheduling updates on a weekly basis, synchronizing the files at some regular interval.
- Cron checks for the scheduled job recurrently and when the scheduled time fields match the current time fields, the scheduled commands are executed.
- It is started automatically from /etc./init.d on entering multi-user run levels.
  - **Syntax:**      **cron [-f] [-l] [-Lloglevel]**
- The **crontab** (abbreviation for “cron table”) is list of commands to execute the

scheduled tasks at specific time. It allows the user to add, remove or modify the scheduled tasks.

- The crontab command syntax has **six fields** separated by space where the **first five** represent the **time to run the task** and **the last one is for the command**
  - o Minute (holds a value between 0-59)
  - o Hour (holds value between 0-23)
  - o Day of Month (holds value between 1-31)
  - o Month of the year (holds a value between 1-12 or Jan-Dec, the first three letters of the month's name shall be used)
  - o Day of the week (holds a value between 0-6 or Sun-Sat, here also first three letters of the day shall be used)
  - o Command (**6<sup>th</sup>Field**)

#### **Extra Example 1:**

- Run /home/folder/gfg-code.sh every hour, from 9:00 AM to 6:00 PM, every day.
  - o **00 09-18 \* \* \*/home/folder/gfg-code.sh**
- Run /user/local/bin/backup at 11:30 PM, every weekday.
  - o **30 23 \* \* Mon, Tue, Wed, Thu, Fri/user/local/bin/backup**
- Run sample-command.sh at 07:30, 09:30, 13:30 and 15:30.
  - o **30 07, 09, 13, 15 \* \* \*sample-command.sh**

#### **❖ Creating cron jobs**

- To create or edit a cron job as the root user, run the command
  - **# crontab-e**
- To create a cron job or schedule a task as another user, use the syntax
  - **# crontab -u username-e**
- For instance, to run a cron job as user Pradeep, issue the command:
  - **# crontab -u Pradeep-e**
- If there is no preexisting crontab file, then user will get a blank text document. If a crontab file was existing, The -e option allows to edit the file,

**❖ Listing crontab files**

- To view the cron jobs that have been created, simply pass the -l option as shown

- #  
    **crontab**  
    -l

**❖ Deleting a crontab file**

- To delete a cron file, simply run **crontab -e** and delete or the line of the cron job that user wants and save the file.
- To remove all cron jobs, run the command:

- # **crontab -r**

**Crontab****Restrictions**

- As a Linux user, user can control who has the right to use the crontab command. This is possible using the **/etc./cron.deny** and **/etc./cron.allow** file.
- By default, only the **/etc./cron.deny** file exists and does not contain any entries. To restrict a user from using the crontab utility, simply add a user's username to the file.
- When a user is added to this file, and the user tries to run the crontab command, he/she will encounter the error like "not allowed to use this program"
- To allow the user to continue using the crontab utility, simply remove the username from the **/etc./cron.deny** file.
- If **/etc./cron.allow** file is present, then only the users listed in the file can access and use the crontab utility.
- If neither file exists, then only the root user will have privileges to use the crontab command.

For more examples refer Week4



## Week-9

### Aim : [1] Enable internet on Linux VM.

#### Network Management

##### ❖ Enable Internet on Linux VM

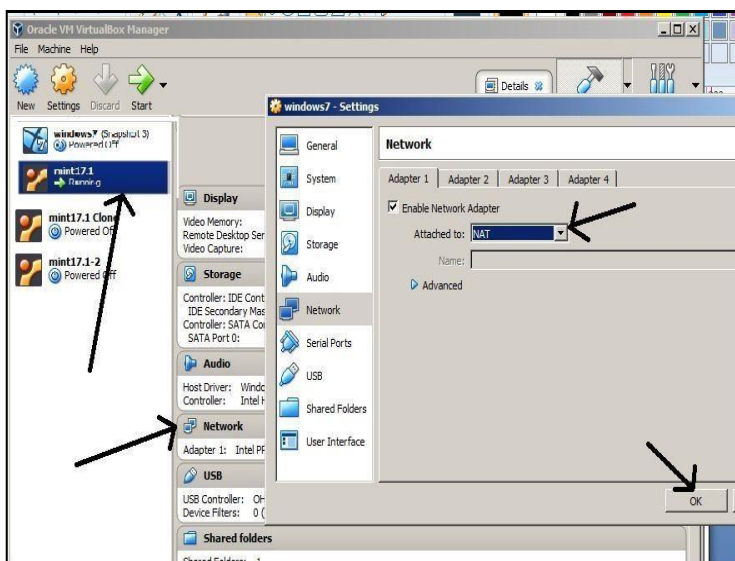
##### ➤ Communicate between Guest and Host using ping command.

##### ▪ Host Virtual Box Manager Settings:

- ◆ In Virtual Box Manager, Select “File”, → “Preferences” → Network → Host only Networks(Tab).
- ◆ Select icon for adding “New Host only Networks” (See vboxnet0 gets added)”
- ◆ Change the properties ofvboxnet0
- ◆ By default, network CIDR is 10.0.2.0/24. Click OK.

##### ▪ Guest VBM Settings in Virtual Box:

- ◆ Select Guest OS in Virtual Box. Select settings for Network.
- ◆ Change the settings for Network. - Select “Adapter 1” Tab – Change “Attached to” to “NAT”
- ◆ Go to Guest OS. And make sure that in Guest OS, setting has been made in network connection, as obtain IP address in DHCP mode..



## Week-9

**Aim : [2] . Test and manage network using following commands ifconfig, iwconfig, ethtool, arpwatch, bmon ,telnet, ssh, sendmail, mailstats, w cURL, wget, ftp, rcp, scp, rsync, sftp. netstat, ping, traceroute, iftop, nload, ss. tcpdump, dstat.**

### ifconfig

- **ifconfig**(interface configuration) is used to configure the kernel-resident network interfaces.
- It is used at the boot time to set up the interfaces as necessary.
- It is used to assign the IP address and netmask to an interface or to enable or disable a given interface.

#### Example1:

##### ifconfig-a

- **-a** :This option is used to display all the interfaces available, even if they are down.

##### Output: ifconfig-a

```
enp1s0          Link encap:Ethernet HWaddr94:c6:91:f6:37:56
inet addr:172.16.20.107 Bcast:172.16.20.255 Mask:255.255.255.0
inet6 addr: fe80::ef4b:1b1d:5c61:d9c6/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:61890 errors:0 dropped:0 overruns:0 frame:0 TX packets:44175 errors:0
dropped:0 overruns:0carrier:0 collisions:0txqueuelen:1000
RX bytes:24167103 (24.1 MB) TX bytes:6512965 (6.5 MB)
```

```
lo              Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:15869 errors:0 dropped:0 overruns:0 frame:0 TX packets:15869 errors:0
dropped:0 overruns:0carrier:0 collisions:0txqueuelen:1000
RX bytes:1729018 (1.7 MB) TX bytes:1729018 (1.7 MB)
```

## Example2:

### ifconfig-s

- **-s** :Display a short list, instead of details.

#### Output: ifconfig-s

Iface	MTU	Met	RX-OK	RX-ERR	RX-DRP	RX-OVR	TX-OK	TX-ERR	TX-DRP	TX-OVR	Fig
enp1s0	1500	0	62065	0	0	0	44236	0	0	0	BMRU
lo	65536	0	15869	0	0	0	15869	0	0	0	LRU

## Example3:

### ifconfig interface up

- **up**: This option is used to activate the driver for the given interface.

## Example4:

### ifconfig interface down

- **down**: This option is used to deactivate the driver for the given interface.

## Example5:

### ifconfig interface arpOR ifconfig interface-arp

- **[-]arp** : This option is used to enable/disable the use of ARP protocol on an interface.

### **iwconfig:**

- **Iwconfig** command in Linux is like **ifconfig** command, in the sense it works with kernel- resident network interface but it is dedicated to wireless networking interfaces only.
- It is used to set the parameters of the network interface that are particular to the wireless operation like SSID, frequencies.
- Iwconfig may also be used to display the parameters, and the wireless statistics which are extracted from /proc/net/wireless.

#### **Usage Examples:**

- **nwid:** This option sets the network ID; user may disable or enable the NetworkID.
  - **iwconfig [Interface] nwidon/off**
- **mode:** Set the operating mode of the device, which depends on the network topology. The modes can be Ad-Hoc, Managed, Master, Repeater, Secondary, and Monitor.
  - **iwconfig [Interface] mode Managed**
- **freq/channel:** This option sets the operating frequency or channel in the device.
  - **iwconfig [Interface] freq2.46000000**
- **ap:** This option forces the card to **register to the Access Point given** by the address if it is possible.
  - **iwconfig [Interface] ap00:60:1D:01:23:45**
- **rate:** This option sets bitrate in b/s in supporting cards.
  - **iwconfig [Interface] rate11M**
- **retry:** This option sets the maximum number of times the MAC can retry transmission.
  - **iwconfig [Interface] retry16**
- **commit:** This option forces the card to apply all pending changes.
  - **iwconfig [Interface]commit**

### ethtool

- ethtool utility is used to view and change the ethernet device parameters.
- Before executing the command first identify the name of the nic using command ifconfig. (Here it is, say **enp1s0**)

#### Example 1: List Ethernet Device Properties

- When user executes ethtool command with a device name, it displays the following information about the Ethernet device.

#### ethtoolenp1s0

Settings for enp1s0:

Supported ports: [ TP MII ]

Supported link modes:

10baseT/Half 10baseT/Full 100baseT/Half 100baseT/Full 1000baseT/Half 1000baseT/Full

Supported pause frame use:No

Supports auto-negotiation:Yes

Advertised link modes: 10baseT/Half 10baseT/Full 100baseT/Half 100baseT/Full 1000baseT/Full

Advertised pause frame use: Symmetric Receive-only Advertised auto-negotiation: Yes

Link partner advertised link modes: 10baseT/Half 10baseT/Full

100baseT/Half 100baseT/Full 1000baseT/Full

Link partner advertised pause frame use: No Link partner advertised auto-negotiation:Yes

**Speed: 1000Mb/s**

Duplex: Full Port: MII PHYAD: 0

Transceiver: internal

**Auto-negotiation: on** Supports Wake-on: pumbg Wake-on: g

Current message level: 0x00000033 (51)

drv probe ifdown ifup

Link detected: yes

This above ethtool output displays ethernet card properties such as speed, wake on, duplex and the link detection status. There are three types of duplexes available. **Full Duplex, Half Duplex, Auto Negotiation.**

## arpwatch:

- **Arpwatch** is an open-source computer software program that helps user to monitor **Ethernet** traffic activity (like **Changing IP** and **MAC Addresses**) on network and maintains a database of **ethernet/Ip** address pairings.
- It produces a log of noticed pairing of IP and MAC addresses information along with a timestamp, so user can carefully watch when the pairing activity appeared on the network.
- It also has the option to send reports via email to a network administrator when a pairing added or changed.
- This tool is especially useful for **Network administrators** to keep a watch on **ARP activity** to detect **ARP spoofing** or unexpected **IP/MAC** addresses modifications.

**Example 1: To watch a specific interface, type the following command with ‘-i’ and device name.**

**arpwatch -ienp1s0**

- o So, whenever a new MAC is plugged or a particular IP is changing his MAC address on the network, user can notice **syslog** entries at **‘/var/log/syslog’** or **‘/var/log/message’** file.

**Output:**

**tail -10 /var/log/syslog**

tail -10 /var/log/syslog

```
Apr 4 11:34:21 admincs-To-be-filled-by-O-E-M arpwatch: reaper: pid 4547, exit status1
Apr 4 11:38:23 admincs-To-be-filled-by-O-E-M arpwatch: new station 172.16.20.26 e2:7b:55:83:d3:a1
enp1s0 Apr 4 11:38:23 admincs-To-be-filled-by-O-E-M arpwatch: new station 172.16.20.52
e2:7b:55:83:d3:a1 enp1s0 Apr 4 11:38:23 admincs-To-be-filled-by-O-E-M arpwatch:
excl:/usr/lib/sendmail: No such file or directory Apr 4 11:38:23 admincs-To-be-filled-by-O-E-M arpwatch:
reaper: pid 4587, exit status1
Apr 4 11:38:23 admincs-To-be-filled-by-O-E-M arpwatch: excl:/usr/lib/sendmail: No such file or directory
Apr 4 11:38:23 admincs-To-be-filled-by-O-E-M arpwatch: reaper: pid 4588, exit status1
Apr 4 11:42:37 admincs-To-be-filled-by-O-E-M cinnamon-screensaver-pam-helper: pam_ecryptfs: seteuid
error Apr 4 11:43:10 admincs-To-be-filled-by-O-E-M kernel: [ 1664.202203] device enp1s0 entered
promiscuous mode
Apr 4 11:43:10 admincs-To-be-filled-by-O-E-M arpwatch: listening on enp1s0
```

- User can also check current **ARP** table, by using following command.

**arp -a**

? (172.16.20.38) at 90:0f:0c:e3:df:07 [ether] on enp1s0

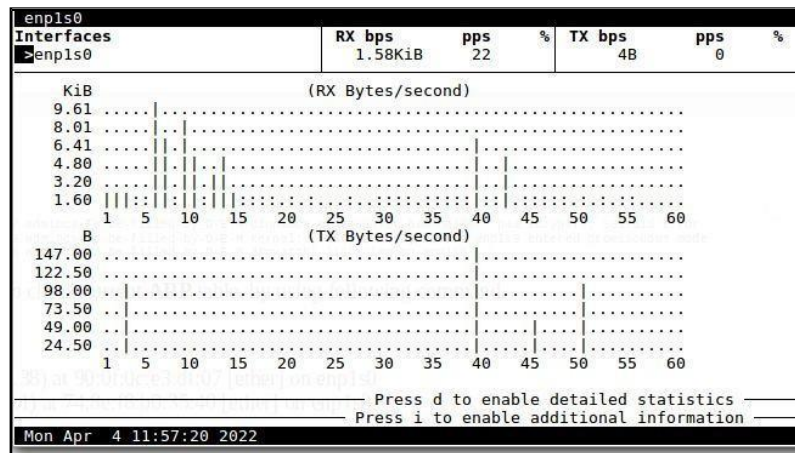
? (172.16.20.1) at 74:8e:f8:b0:35:40 [ether] on enp1s0

## bmon:

- **Bmon** is a simple yet powerful, text-based network monitoring and debugging tool for Unix-like systems, which captures networking related statistics and displays them visually in a human friendly format.
- It is a reliable and effective real-time bandwidth monitor and rate estimator.
- It can read input using an assortment of input modules and presents output in various output modes, including an interactive curses user interface as well as a programmable text output for scripting purposes.

### Example 1: bmon -p enp1s0

To view more detailed graphical statistics/information of bandwidth usage, press d key Press [Shift + ?] to view the quick reference. To exit the interface, press [Shift + ?] again.



## ssh:

- Secure Shell, sometimes referred to as **Secure Socket Shell**, is a protocol which allows user to connect securely to a remote computer or a server by using a text-based interface.
- When a secure SSH connection is established, a shell session will be started, and user will be able to manipulate the server by typing commands within the client on local computer.
  - **Prerequisite:**
    - o Install **opens-client** and **opens-server** through software manager
    - o assign valid Ip address to two machines, for connection
    - o Assure, both the system has user account with password.

## Settings:

Machine 1:	Machine 2:
<b>name:</b> admincm1	admincm2
<b>login:</b> admincl1	admincl2
<b>192.100.100.1</b>	<b>192.100.100.2</b>

## Example 1: machine 1 is connecting machine 2

```
ssh -ladmincl2      admincm2@192.100.100.2      21
```

- Connect to the computer with name admincm2 and Ip address 192.100.100.2 to the port number 21 through loginadmincl2
- It will ask for admin login password.
- The successful connection led to remote machine terminal. User can issue any

## telnet:

Linux commands to handle remote machine locally.

- The **telnet** command is used to create a remote connection with a system over a TCP/IP network.
- It allows us to administrate other systems by the terminal.
- User can run a program to conduct administration.
- It uses a TELNET protocol. However, this protocol has some security defects, but it is one of the most used networking protocols due to its simplicity.
- It is not a secure protocol because it transfers data in unencrypted form.
- Often Linux user prefers **sh** over telnet because ssh transfers data in encrypted form.
- **Prerequisite:**
  - Install **telnet (client)** and **telnet (server)** through software manager
  - assign valid Ip address to two machines, for connection
  - Assure, both the system has user account with password.

## Settings:

Machine 1:	Machine 2:
<b>name:</b> admincm1	admincm2
<b>login:</b> admincl1	admincl2
<b>192.100.100.1</b>	<b>192.100.100.2</b>



**Example 1:** machine 1 is connecting machine 2

```
telnet -ladmincl2 192.100.100.2 23
```

- Connect to the computer with Ip address 192.100.100.2 to the port number 23 through loginadmincl2
- It will ask for admin login password.
- The successful connection led to remote machine terminal. User can issue any Linux commands handle remote machine locally.

## **ftp:**

- **ftp** is the user interface to the Internet standard File Transfer Protocol.
- The program allows a user to transfer files to and from a remote network site.
- **Prerequisites:**
  - Install **Ftp** (FTP-Client) and **vsftpd** (FTP-Server) using Software Manager.
  - Change **/etc./vsftpd.conf** file to enable **write permission** (other settings can be done according to requirement.  
**nano/etc./vsftpd.conf**
  - Search for **# write\_enable=YES** (around 30<sup>th</sup>line in that file)
  - **Uncomment** and Save and Exit
  - **Restart the Service:** **systemctl restartvsftpd**

**Example 1:** Get the connection to ftp

```
ftp 192.100.100.2 21
```

- Enter user name and password for connection, this will take user to **ftp** utility and then issue ftp command sets
- For more command take the help by issuing **?and** to quit utility type **quit** command.

## **curl:**

- **Curl** is a command-line tool to transfer data to or from a server, using any of the supported protocols (HTTP, FTP, IMAP, POP3, SCP, SFTP, SMTP, TFTP, TELNET, LDAP, or FILE).
- Curl is powered byLibcurl.
- This tool is preferred for automation since it is designed to work without user interaction.
- curl can transfer multiple files at once.

**Examples below are to Upload and Download files using ftp service. Prerequisites:**

- Install **Ftp** (FTP-Client) and **vsftpd** (FTP-Server) using Software Manager.
- Change **/etc./vsftpd.conf** file to enable **write permission** (other settings can be done according to requirement.  
**nano/etc./vsftpd.conf**
- Search for **# write\_enable=YES** (around 30<sup>th</sup> line in that file)
- **Uncomment** and Save and Exit
- **Restart the Service:** **systemctl restartvsftpd**

**Example 1: Download the file from ftp Server**

- **-u** option: indicates username and separated with password for that account (user authenticated FT servers).
- **-O** option: will download file with same name as original file, here **ftp://192.100.100.2/a.txt** downloaded with the same in current machine.
- **-#** option: indicates downloading progress bar.
- **curl -# -u admincs:admincs -O ftp://192.100.100.2/a.txt**
  - Here user name and password are same(admincs)

**wget:**

- **wget** is the non-interactive network downloader which is used to download files from the server even when the user has not logged on to the system and it can work in the background without hindering the current process.
- **Examples:**
  - To simply download webpage:
    - **wgethttps://www.rediff.com/index.html**
  - To download the file in **background**
    - **wget -b https://www.rediff.com/index.html**
  - To overwrite the log while of the wgetcommand.
    - **wget http://www.example.com/filename.txt -**

o/home/admincs/temp.html

- o To download from ftp server:
  - options: --ftp-user=username and --ftp-password=password string will be used to authenticate ft server.
  - File name test.c from ftp://192.100.100.2/Desktop/ downloaded to user machine.
  - **wget --ftp-user=admincs --ftp-password=admincs ftp://192.100.100.2/Desktop/test.c**

### rcp:

- The rcp command is used to copy files between different computers without starting an FTP session or logging into the remote system explicitly.
  - It uses **ssh** for data transfer, and uses the same authentication and provides the same security as ssh.
  - To simply use the **rcp** command, just provide the source and destination to rcp command with a colon used to separate the host and the data.
  - **Syntax Example:**
- o /\* using rcp command to send a file from local host to remote host\*/
    - **rcp/mydirectory/a.txtadmin:remotedir/kt.txt**
    - o /\* the example above is to send a file not to receive a file from remote host\*/
    - o What actually happening in the above example is the file named **a.txt** whose path is given as **/mydirectory/a.txt** is getting transferred from this local path (/mydirectory) (local host) to the remote system named **admin** and the file on that system will be placed in the directory **remote Dir** (as path **remote Dir/a.txt** is given).

### Example 1: Copy file from local machine to remoter machine.

**rcp index.html admincs@192.100.100.2:/home/admincs/test.html**

- This will copy file index.html from current user to remote user admincs@192.100.100.2 and file will be copied to remote user **/home/admincs** with the file name**test.html**

### scp:

- **scp** allows user to securely copy files and directories between two **local/remote** machines.
- It uses **ssh** for data transfer, and uses the same authentication and provides the same security as ssh.

#### Working with scp:

**Note:** Assume, Source machine and Destination machine have same username, so identify different machine user names with Ip address.

#### Example 1: Copy file from local machine to remoter machine.

```
scp      index.html      admincs@192.100.100.2:/home/admincs/test.html
```

- This will copy file index.html from current user to remote user admincs@192.100.100.2 and file will be copied to remote user **/home/admincs** with the file name **test.html**

### rsync

- **Rsync** is a fast and extraordinarily versatile file copying tool.
- It can copy **locally**, to/from another host over any **remote** shell, or to/from a remote rsync daemon.
- It offers a large number of options that control every aspect of its behavior and permit very flexible specification of the set of files to be copied.
- It is famous for its delta-transfer algorithm, which reduces the amount of data sent over the network by sending only the differences between the source files and the existing files in the destination.
- Rsync is widely used for backups and mirroring and as an improved copy command for everyday use.

#### Example 1: synchronize files (copy file locally)

```
rsync/home/admincs/Documents/temp.txt
                                     /home/admincs/Desktop/test
.txt
```

- This will copy file **temp.txt** at **local/admincs/Documents** to local machine desktop with new name **test.txt**

## sftp:

- **Sftp** works on a client-server model. It is a subsystem of SSH and supports all SSH authentication mechanisms.
- To open an SFTP connection to a remote system, use the **sftp** command followed by the remote server username and the IP address or domain name (then provide password for the user):
  - o **sftpadmins@172.16.20.116**

### Example 1: Download the file from ft server

**get a.txt /home/admincs/Desktop/test.txt**

- will download the file **a.txt** from **ftp server** to current user (local user **/home/admincs/ Desktop** with file name **test.txt**

## netstat

- netstat is a command line utility for Linux that prints network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- netstat can be used to diagnose network issues and service problems.

### Important Options used:

- o **-a:** all listening and non-listening ports, **-t** tcp ports
- o **-u** udp ports, **-l** listening ports
- o **-s** Statistics of ports, **-r** Kernel Routing Information

### Example 1: netstat -at head

**// To list all tcp ports.**

```
admins-To-be-filled-by-O-E-M ~ # netstat -at | head
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:smtp          *:*                     LISTEN
tcp        0      0 localhost:submission    *:*                     LISTEN
tcp        0      0 admins-To-be-fi:domain *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
tcp        1      0 172.16.20.107:38096     172.16.20.116:ssh      CLOSE_WAIT
tcp        0      0 172.16.20.107:39984     104.18.72.113:https    ESTABLISHED
```

## ping

- **ping**(Packet Internet Groper) command is used to check the network connectivity between host and server/host.
- This command takes as input the IP address or the URL and sends a data packet to the specified address with the message “PING” and get a response from the server/host this time is recorded which is called latency.
- Fast ping low latency means faster connection.
- Ping uses **ICMP(InternetControlMessageProtocol)** to send an **ICMP Echo message** to the specified host if that host is available then it sends **ICMP reply message**.
- Ping is generally measured in millisecond.
- Important options used:
  - o -c : Number of packets to be transferred
  - o -w : deadline, with in this second, continuedly send the packets ICMP Packets.
  - o -s : Packesize

**Example1:**                    **ping -c 5 -s 100 172.16.20.116**

- o Will send 5 ICMP Packets to test wheter machine 100.172.16.116 is alive or not, and each packet size data is 100 bytes, total packet size is 108 (8 bytes of header).

## traceroute:

- o **traceroute** command in Linux prints the route that a packet takes to reach the host.
- o This command is useful when user want to know about the route and about all the hops that a packet takes.

**Example1:**    **traceroute www.google.com**

```
traceroute to www.google.com (142.250.195.196), 30 hops max, 60-
byte packets 1 172.16.20.1 (172.16.20.1) 0.761 ms 1.407 ms 1.973
ms
2 172.16.1.100 (172.16.1.100) 0.137 ms 0.137 ms 0.140 ms
3 117.236.190.194 (117.236.190.194) 7.500 ms 9.117 ms 8.384 ms
4 172.24.64.138 (172.24.64.138) 2.385 ms 2.380
ms136.232.204.173.static.jio.com
(136.232.204.173) 3.082 ms
```

```

5 * * *
6 72.14.218.250 (72.14.218.250) 19.051 ms 18.964 ms 19.297 ms
7 * * *
8 216.239.59.230 (216.239.59.230) 20.538 ms maa03s42-in-f4.1e100.net
(142.250.195.196)
17.921 ms 216.239.59.230 (216.239.59.230) 20.029 ms

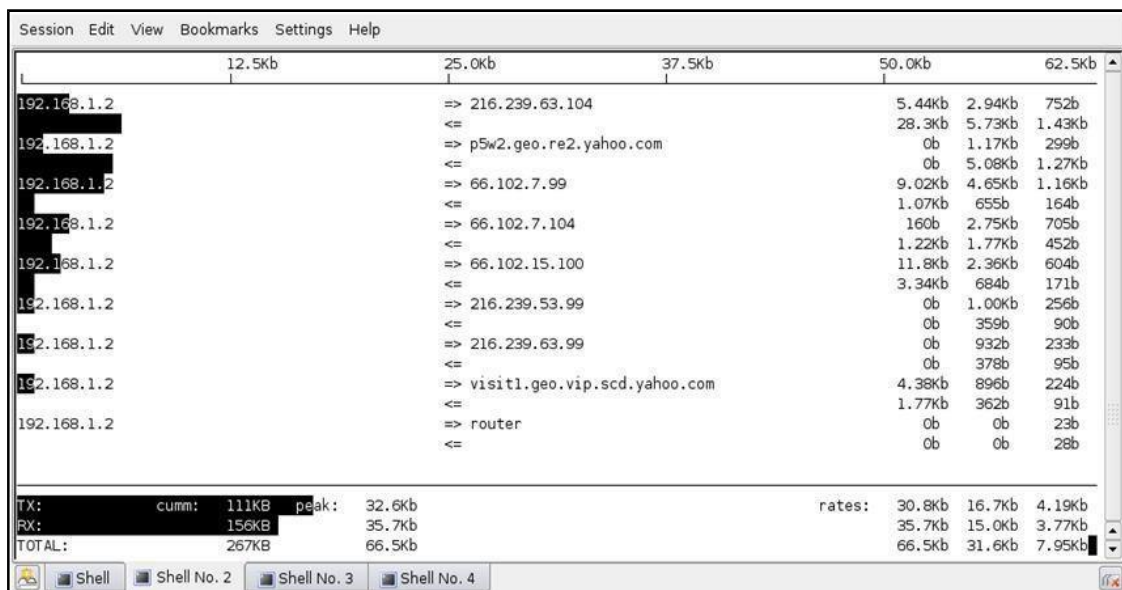
```

- The first column corresponds to the **hop count**. The second column represents **the address of that hop** and after that, three space-separated time in milliseconds.
  - traceroute command sends three packets to the hop and each of the time refers to the time taken by the packet to reach the hop.

## iftop

- The **iftop** command listens to network traffic on a named network interface, or on the first interface, it can find which looks like an external interface if none is specified, and **displays a table of current bandwidth usage by pairs of hosts**.
- The iftop is a perfect tool for remote Linux server over a ssh based session.
- If top must be run by the root or the user who has sufficient permissions to monitor all network traffic on the network interface.

### Example 1: iftop -i enp1s0



## nload

- **Nload** is a Linux command-line tool used to monitor network traffic and bandwidth usage in real time, using insightful graphs and traffic statistics.
- Output of nload is in paragraph, one for each device.
- A device is anything which sends and/or receives internet packets on the same network, but usually, it represents a network interface device.
- It does not necessarily need to be a separate physical device, but can even be on the same machine!

**nload-m**                      // -m for multiple devices.

### Output:

```
Device enp1s0 [172.16.20.107] (1/2):
=====
Incoming:                Outgoing:
Curr: 1.61kBit/s          Curr: 0.00Bit/s
Avg:1.82kBit/s            Avg: 456.00Bit/s
Min:0.00 Bit/s            Min: 0.00Bit/s
Max:34.95kBit/s           Max: 15.07kBit/s
Ttl:76.13MByte            Ttl: 8.84MByte
```

```
Device lo [127.0.0.1] (2/2):
=====
Incoming:                Outgoing:
Curr: 1.30kBit/s          Curr: 1.30kBit/s
Avg:808.00Bit/s           Avg: 808.00Bit/s
Min:0.00 Bit/s            Min: 0.00Bit/s
Max:20.16kBit/s           Max: 20.16kBit/s
Ttl:1.53 MByte            Ttl: 1.53MByte
=====
```

### Options used:

- o The -a option to set the length in seconds of the time window for average calculation. By default, nload sets this to be **300**seconds.
- o The -t interval flag sets the refresh interval of the display in milliseconds. By default, nload sets this to be **500**seconds.



## ss

- The **ss** command is a tool used to dump socket statistics and displays information in similar fashion (although simpler and faster) to **netstat**.
- The ss command can also display even more TCP and state information than most other tools.
- The ss command can also display even more TCP and state information than most other tools. Because ss is the new netstat, the ss command-line utility can display stats for the likes of PACKET, TCP, UDP, DCCP, RAW, and Unix domain sockets.
- The replacement for netstat is easier to use (compare the man pages to get an immediate idea of how much easier ss is).

### Some options used:

- |      |                      |    |                         |
|------|----------------------|----|-------------------------|
| o -t | Display TCP sockets, | -u | Display UDP sockets,    |
| o -a | All Sockets,         | -l | Only Listening Sockets, |
| o -4 | ipv4 Packets only.   |    |                         |

### Example 1: ss -t

State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
ESTAB	0	0	172.16.20.107:57818	142.250.182.46:https
ESTAB	0	0	172.16.20.107:48336	34.107.221.82:http
ESTAB	0	0	172.16.20.107:48334	34.107.221.82:http
ESTAB	0	0	172.16.20.107:49542	34.213.33.47:https

## tcpdump

- **Tcpdump** is a packet sniffing and packet analyzing tool for a System Administrator to troubleshoot connectivity issues in Linux.
- It is used to capture, filter, and analyze network traffic such as TCP/IP packets going through system.
- It is many times used as a security tool as well. It saves the captured information in a pcap file, these pcap files can then be opened through Wireshark or through the command tool itself.

### tcpdump -i enp1s0

### Important Options used:

- o -c Specific Number of Packets captured
- o -e Print the link-level header on each dump line.

**Example 1:** To print all packets arriving at or departing from sundown:

**tcpdumphost172.16.20.116**

**// Try ping from172.16.20.116**

## Output:

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode listening on enp1s0, link-type EN10MB (Ethernet), capture
size 262144 bytes
13:39:06.863886 IP 172.16.20.116 > 172.16.20.107: ICMP echo request, id 28099, seq 1,
length 64
13:39:06.863945 IP 172.16.20.107 > 172.16.20.116: ICMP echo reply, id 28099, seq 1, length
64
13:39:07.882100 IP 172.16.20.116 > 172.16.20.107: ICMP echo request, id 28099, seq 2,
length 64
13:39:07.882158 IP 172.16.20.107 > 172.16.20.116: ICMP echo reply, id 28099, seq 2, length
64
```

## **dstat**

- **Dstat** is a tool that is used to retrieve information or statistics form components of the system such as network connections, IO devices, or CPU, etc.
- It is generally used by system administrators to retrieve a handful of information about the above-mentioned components of the system. It itself performs like vmstat, netstat, iostat, etc.
- By using this tool one can even see the throughput for block devices that make up a single filesystem or storage system.
- **dstat** allows user to view all of system resources instantly, for example, user can compare disk usage in combination with interrupts from IDE controller, or compare the network bandwidth numbers directly with the disk throughput (in the same interval).
- **Dstat** also cleverly gives user the most detailed information in columns and clearly indicates in what magnitude and unit the output is displayed. Less confusion, less mistakes, more efficient.
- **Dstat** is unique in letting user aggregate block device throughput for a certain disk set or network bandwidth for a group of interfaces, i.e., user can see the throughput for all the block devices that make up a single file system or storage system.
- **Dstat** allows its data to be directly written to a CSV file to be imported and used by OpenOffice, Gnumeric or Excel to create graphs.

## Some options used:

- o -c enable cpu stats (system, user, idle, wait, hardware interrupt, software interrupt)
- o -d, --disk enable disk stats(read, write) -g, --page enable stats (page in, page out)
- o -i,--int enableinterruptstats -m,--mem enable memory stats (used, buffers, cache, free)
- o -n,--net enable network stats(receive, send) --nocolor

## Output:

<b>dstat -n</b>	<b>dstat -c</b>
-net/total-	----total-cpu-usage----
recv send	usr sys idl wai hiqsiq
0 0	4 1 95 0 0 0
216B 0	1 0 98 0 0 0
336B 84B	2 0 98 0 0 0
632B 0	2 1 97 0 00^C
6297B 54B	
584B 54B	
697B 0 ^C	

**sendmail**

**: mailstat**

## Week-10

**Aim : [1] Work on user accounts useradd, passwd, userdel, usermod, groupadd, groupmod, gpasswd, groupdel; system-config**

### User Authentication

#### useradd

- **Useradd** is a command in Linux that is used to add user accounts to system.
- It is just a symbolic link to **adduser** command in Linux and the difference between both of them is that useradd is a native binary compiled with system whereas adduser is a Perl script which uses useradd binary in the background.
- When we run the '**useradd**' command in the Linux terminal, it performs the following major things:
  - o It edits **/etc/passwd**, **/etc/shadow**, **/etc/group** and **/etc/gshadow** files for the newly created user accounts.
  - o Creates and populates a home directory for the new user.
  - o Sets permissions and ownerships to the homedirectory.

#### Syntax:

**useradd -d /home/newusername -p passwordstringnewusername**

- o This creates user with name newusername
  - -d option will create directory for new user at /homedirectory
  - -p option allows specifying password while creating user with the value given in place of passwordstring
- o User can change the password later by typing command
  - passwdnewusername
  - This asks the user to enter the new password and confirm password

#### passwd

- The passwd command **changes passwords for useraccounts**.
- A normal user may only change the password for their own account, while the superuser may change the password for any account.
- passwd also changes the account or associated password validity period.
- **Important options** used along with passwd:
  - o **-d,--delete** Delete a user's password (make it empty). This is a quick way to disable a password for an account. It will set the named account password less.
  - o **-e,--expire** Immediately expire an account's password. This in effect canforce a user to change his/her password at the user's next login

## **userdel**

- **Userdel** command in Linux system is used to delete a user account and related files.
- This command basically modifies the system account files, deleting all the entries which refer to the username LOGIN.
- It is a low-level utility for removing the users.

**Syntax:**            **sudo userdel -f -r -Z username**

**Important Options** used along with userdel:

- o **-f:** This option forces the removal of the specified user account. It doesn't matter that the user is still logged in. It also forces the userdel to remove the user's home directory and mail spool, even if another user is using the same home directory or even if the mail spool is not owned by the specified user.
- o **-r:** Whenever we are deleting a user using this option then the files in the user's home directory will be removed along with the home directory itself and the user's mail spool. All the files located in other file systems will have to be searched for and deleted manually.
- o **-Z:** This option removes any SELinux(Security-Enhanced Linux) user mapping for the user's login.

## **usermod**

- usermod command or modify user is used to change the properties of a user.
- After creating a user, we have to sometimes change their attributes like password or login directory etc.
- To change the home directory of a user
  - **usermod -d /home/newfolderexistinguser**
- To change the group of a user
  - **usermod -g newgroupnameexistinguser**
- To change user loginname
  - **usermod -l usernewnameuseroldname**
- To lock a user
  - **usermod -Lexistinguser**
- To set an unencrypted password for the user
  - **usermod -p newpasswordexistinguser**

**Groups:** Groups in Linux refer to the user groups. In Linux, there can be many users of a single system, (normal user can take uid from 1000 to 60000, and one root user (uid 0) and 999 system users (uid 1 to 999)).

## **groupadd**

- **Groupadd** command is used to create a new usergroup.
  - **Syntax:** `groupadd newgroupname`
- Every new group created is registered in the file `"/etc/group"`. To verify that the group has been created, enter the command
  - `sudo tail/etc/group`
- Adding user while creating new user
  - `useradd -d /home/newusername -p passwordstring -g existinggroupname newusername`
- Adding existing user to group // Explained once again in next topic
  - `usermod -g existinggroupname existinguser`

## **groupmod**

- **groupmod** command is used to modify or change the existing group.
- It can be handled by superuser or root user.
- Basically, it modifies a group definition on the system by modifying the right entry in the database of the group.
  - **Syntax:** `groupmod [option]group name`
    - o Change the group name to newname.
      - o `groupmod -n groupnewnamegroupoldname`

## **gpasswd**

- `gpasswd` command is used to administer the `/etc/group` and `/etc/gshadow`.
- `gpasswd` command **assigns** a user to a group with some security criteria.
- `gpasswd` command is called by a group administrator with a group name only which prompts for the new password of the group.
- System administrators can use the `-A` option to define group administrator(s) and `-M` option to define members.

**Syntax:** `gpasswd [option]group`

**Important Options:** Here only -A and -M options can be combined.

- o **-a, –add** : This option is used to add a user to the named group.
- o **-d, –delete** : It is used to remove a user from the named group.
- o **-r, –remove-password** : It is used to remove the password from the named group.
- o **-A, –administrators** : Set the list of administrators for the group.
  - o **-M, –members** : set the list of members of the group.

**Example:**

- add the user to group:
  - o **gpasswd -a existinguserexistinggroup**
- Deleting the created user from group geeks.
  - o **gpasswd -d existinguserexistinggroup**

### groupdel

- groupdel command is used to delete an existing group.
- It will delete all entry that refers to the group, modifies the system account files, and it is handled by superuser or root user.
  - **Syntax:** **groupdel –existinggroup**
- **Option used: -f –force**: It used to delete a group even if it is the primary group of a user

## Week-10

### Aim : [2] Open LDAP Installation and LDAP server and client configuration.

#### LDAP server and client configuration:

- Lightweight Directory Access Protocol (LDAP) is a standard protocol designed to manage and access hierarchical directory information over network.
- It can be used to store any kind of information, though it is most often used as a centralized authentication system or for corporate email and phone directories.
- Use phpLDAPadmin, a web interface for viewing and manipulating LDAP information.

**Prerequisites:**      **install apache and php**

### Step 1: Installing and Configuring the LDAP Server

- Install the LDAP server and some associated utilities.
- `sudo apt-get update`
- `sudo apt-get install slapd ldap-utils`
- During the installation,
  - Enter administrator password foldup.
  - Install and fill the fields asked during configuration:

**`sudo dpkg-reconfigure slapd`**

There are quite a few new questions to answer in this process. Accept most of the defaults. Questions are:

- Omit OpenLDAP server configuration? **No**
- DNS domain name?
  - This option will determine the base structure of directory path. Read the message to understand exactly how this will be implemented. This installation use **example.com**.
- Organization name?
  - For this guide, use **example** as the name of organization. (user may choose anything which is appropriate.)
- Administrator password? enter a secure password twice
- Database backend?  **MDB**
- Remove the database when slapd is purged? **No**



- Move old database? **Yes**
- Allow LDAPv2 protocol? **No**

At this point, LDAP server is configured and running. Open up the LDAP port on firewall so external clients can connect:

**sudo ufw allowldap**

Test LDAP connection with **ldapwhoami**, which should return the username as:

**ldapwhoami -H ldap://-x**

#### Output

- o **anonymous**
- o anonymous is the result expected, since running ldapwhoami without logging in to the LDAP server. This means the server is running and answering queries.

## Step 2: Installing and Configuring the phpLDAPadmin Web Interface (set up a web interface to manage LDAP data)

- Although it is very possible to administer LDAP through the command line, most users will find it easier to use a web interface. (install phpLDAPadmin, a PHP application which provides this functionality.)
- The Ubuntu repositories contain a phpLDAPadmin package. Install it with apt-get:
  - **sudo apt-get installphpldapadmin**
    - o This will install the application, enable the necessary Apache configurations, and reload Apache.
  - The web server is now configured to serve the application, but need to make some additional changes. Configure phpLDAPadmin to use user domain (example.com), and to not autofill the LDAP login information.

Begin by opening the main configuration file with root privileges in text editor:

**sudo nano/etc/phpldapadmin/config.php**

Look for the line that starts with \$servers-

**>setValue('server','name'.**

This line is a display name for LDAP server, which the web interface uses for headers and messages about the server. Choose anything appropriate here:

**/etc/phpldapadmin/config.php**

```
$servers->setValue('server','name','Example LDAP');
```

Next, move down to the \$servers->setValue('server','base' line.

This config tells phpLDAPadmin what the root of the LDAP hierarchy is. This is based on the value typed in when **reconfiguring** the **slapd** package. In above example user selected **example.com** and translate this into LDAP syntax by putting each domain component (everything **not a dot**) into a dc= notation:

**/etc/phpldapadmin/config.php**

```
$servers-
```

```
>setValue('server','base',array('dc=example,dc=com'));
```

Now find the login bind\_id configuration line and **comment** it out with a #at the beginning of the line:

**/etc/phpldapadmin/config.php**

```
#$servers->setValue('login','bind_id','cn=admin,dc=example,dc=com');
```

This option pre-populates the admin login details in the web interface. This is information user shouldn't share if phpLDAPadmin page is publicly accessible.

The last thing that needs to adjust is a setting that controls the visibility of some phpLDAPadmin warning messages. By default, the application will show quite a few warning messages about template files. These have no impact on our current use of the software. User can hide them by searching for the hide\_template\_warning parameter, **uncommenting** the line that contains it, and setting it to **true**:

**/etc/phpldapadmin/config.php**

```
$config->custom-
```

```
>appearance['hide_template_warning'] =true;
```

This is the last thing that user need to adjust. Save and close the file to finish. User NO need to restart anything for the changes to take effect.

### Step 3: Logging into the phpLDAPAdminWeb Interface

Having made the necessary configuration changes to phpLDAPAdmin, user can now begin to use it. Navigate to the application in web browser. Be sure to substitute domain for the highlighted area below:

<https://example.com/phpldapadmin>

OR

<https://localhost/phpldapadmin>

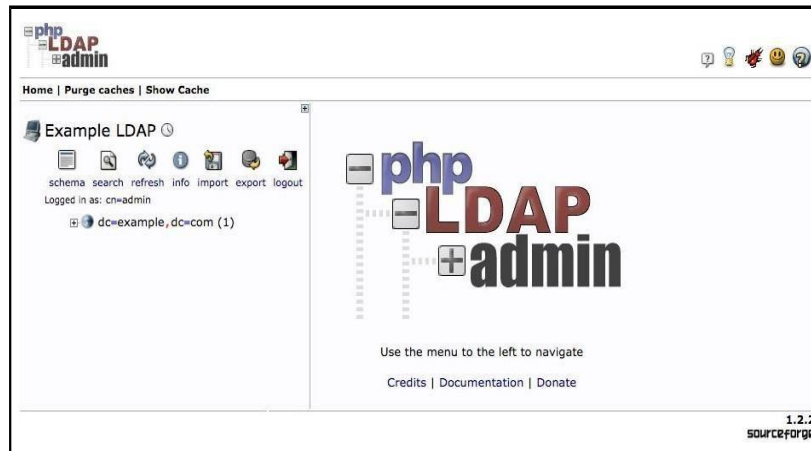
The phpLDAPAdmin landing page will load. Click on the **login** link in the left-hand menu on the page. A login form will be presented:

The **Login DN** is the **username** that user will be using. It contains the account name as a cn= section, and the domain name selected for the server broken into dc= sections as described in previous steps. The default **admin** account that is used during install is called **admin**, so type in the following:

**usrename: cn=admin,dc=example,dc=com**

After entering the appropriate string for domain, type in the admin password that the user created during configuration, then click the **Authenticate** button.

It will be displaying the main interface:



At this point, user is logged into the phpLDAPadmin interface. Here, user has the ability to add users, organizational units, groups, and relationships. User can create whatever kind of structure he would like and also create rules for how they interact.

## Creating Group and Users in using LDAP:

With web-based LDAP admin tool (phpLDAPadmin), user can more easily manage LDAP server and populate it with users.

## Creating Organizational Units

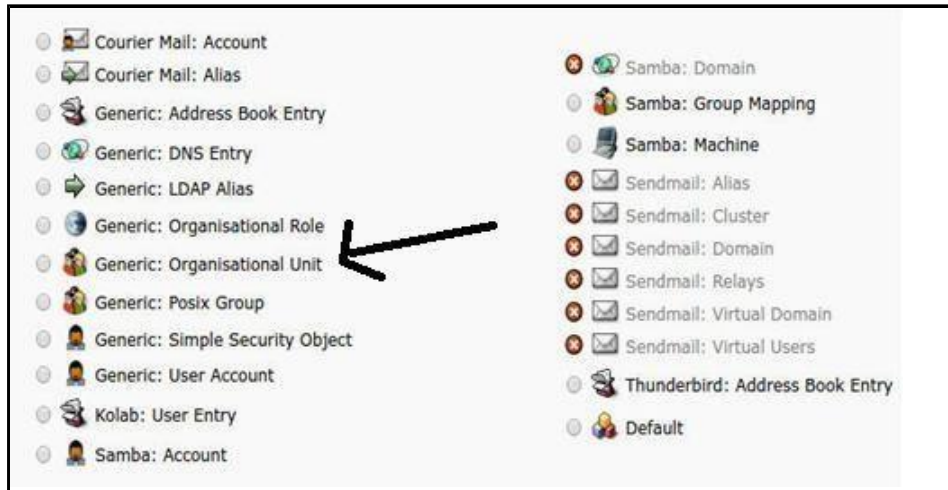
LDAP breaks everything into very specific pieces, and here focus is on two pieces: **people and groups**. Because we're creating fairly generic Organizational Units (OUs), we'll use the Generic Organizational Unit Template. To get there, log into phpLDAPadmin, click to expand server listing and then click Create New Entry Here (Figure A).

Figure A



In the right pane (Figure B), select Generic: Organizational Unit.

Figure B



First **create an OU named “groups”**. In the next window type groups and click Create Object. Commit the group by clicking Commit in the next window (**Figure C**).

**Figure C**

The 'Create LDAP Entry' dialog box has a title bar 'Create LDAP Entry' and a subtitle 'Server: My LDAP Server Container: dc=monkeypantz,dc=net'. It asks 'Do you want to create this entry?'. Below is a table:

Attribute	New Value	Skip
<b>ou=groups,dc=monkeypantz,dc=net</b>		
<b>objectClass</b>	organizationalUnit	<input type="checkbox"/>
<b>Organisational Unit</b>	groups	<input type="checkbox"/>

At the bottom are 'Commit' and 'Cancel' buttons.

Look for a new entry in the left pane called ou=groups (**Figure D**).

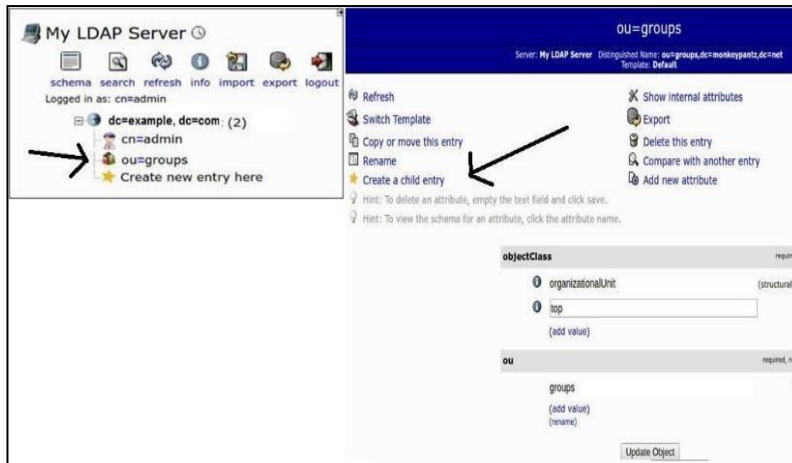
**Figure D**



**Create a new OU named “users”**. Walk through the same process as above, though name the OU “users” instead of “groups”. Look for “ou=groups” and “ou=users” in the left pane.

Creating groups

Now an OU created for groups, then add the necessary groups. Create groups for “admin”,



“developers”, and “users”. Here’s how.

1. Click the **groups OU** in the left pane.
2. In the resulting window, click **Create Child Entry (Figure E)**.
3. Click **Generic: Posix Group**.
4. Type admin into the group text area.
5. Click **Create Object**.
6. Click **Commit**.
7. Repeat the process for “developers” and “users”

**Figure E**

<b>Common Name</b>	alias, required, rdn
<input style="width: 90%;" type="text"/>	
<b>First name</b>	alias
<input style="width: 90%;" type="text"/>	
<b>GID Number</b>	alias, required, hint
<input style="width: 90%;" type="text"/>	
<b>Home directory</b>	alias, required
<input style="width: 90%;" type="text"/>	
<b>Last name</b>	alias, required
<input style="width: 90%;" type="text"/>	

## Creating users

Now groups created, then create users. To do this, follow these steps.

1. Click **ou=users** from the left pane.
2. In the resulting window, click Create a **Child Entry**.
3. Select **Generic: User Account**.
4. Fill out the required information- note that **Common Name** must be unique (**FigureF**).
5. Click **Create Object**.
6. Click **Commit**.
7. Repeat this process until necessary number of users added.

**Figure F**

The screenshot shows a 'Create Object' dialog box for a 'Generic: User Account'. The fields are as follows:

- Login shell:** A dropdown menu with a downward arrow.
- Password:** Two input fields for password and confirmation, with a 'md5' encryption dropdown and a 'Check password...' link.
- UID Number:** A text input field containing '1000'.
- User ID:** A text input field that is highlighted in yellow.
- Create Object:** A button at the bottom of the dialog.

The screenshot shows an 'Update Object' dialog box for a 'Generic: User Account'. The fields are as follows:

- cn:** A text input field containing 'developers'.
- gidNumber:** A text input field containing '501'.
- objectClass:** A list of object classes including 'posixGroup' and 'top'.
- Update Object:** A button at the bottom of the dialog.

## Adding users to groups

To add a user to a group, user's UID (named User ID in the user creation window) must be known. To find a UID go to **ou=users** | View X child (where X is the number of users) and then locate the user to be added and make note of their associated UID. Once UID is noted, then add that user to the developers group. Here 'show.

1. Expand **ou=groups**.
2. Click the **developers** group.
3. Click **Add New Attribute**.
4. From the drop-down, select **memberUID**.
5. Enter the UID for the user in the **memberUID** section (**Figure G**).
6. Click Update Object.

Figure G

cn=developers

Server: My LDAP Server Distinguished Name: cn=developers,ou=groups,dc=monkeypantz,dc=n Template: Default

Refresh Switch Template Copy or move this entry Rename Create a child entry

Show internal attributes Export Delete this entry Compare with another entry Add new attribute

Hint: To delete an attribute, empty the text field and click save.  
Hint: To view the schema for an attribute, click the attribute name.

Add Attribute

memberUID

cn required, rdn

developers (add value) (rename)

gidNumber required

501

objectClass required

posixGroup (structural)

top (add value)

Update Object

After finished adding first user, adding subsequent users is much simpler. If user click the group name (under ou=groups in the left pane), user can click **Modify Group Members** (under memberUID) and then add the users from a list.



## Week-11

**Aim : [1] System monitoring commands top, df, dmesg, iostat 1, free, cat /proc/cpuinfo, cat/proc/meminfo;**

### System/Log monitoring commands and System Information/Maintenance Commands

#### top

- The top command has been around a long time and is very useful for viewing details of running processes and quickly identifying issues such as memory hogs. Its default view is shown below.

```
top - 11:56:28 up 1 day, 13:37, 1 user, load average: 0.09, 0.04, 0.03
Tasks: 292 total, 3 running, 225 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.1 us, 0.2 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 16387132 total, 10854648 free, 1859036 used, 3673448 buff/cache
KiB Swap:      0 total,      0 free,      0 used. 14176540 avail Mem
  PID USER      PR  NI   VIRT   RES   SHR S %CPU%MEM    TIME+
COMMAND
17270 alan    20   0 3930764 247288 98992 R  0.7 1.5
5:58.22 gnome-shell
20496 alan    20   0 816144 45416 29844 S  0.5 0.3 0:22.16 gnome-terminal-
21110 alan    20   0 41940  3988  3188 R  0.1 0.0 0:00.17 top
1 root      20   0 225564  9416  6768 S  0.0 0.1 0:10.72 systemd
2 root      20   0      0      0   0 S  0.0 0.0 0:00.01 kthreadd
4 root      0 -20   0      0   0 I  0.0 0.0 0:00.00 kworker/0:0H
6 root      0 -20   0      0   0 I  0.0 0.0 0:00.00 mm_percpu_wq
7 root      200   0      0   0 S  0.0 0.0 0:00.08 ksoftirqd/0
```

- The update interval can be changed by typing the letter **s** followed by the number of seconds the user prefers for updates.
- To make it easier to monitor required processes, user can call top and pass the PID(s) using the **-p** option.
- top -p20881 -p20882 -p20895 -p20896**

```
Tasks: 4 total, 0 running, 4 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.8 us, 1.3 sy, 0.0 ni, 95.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 16387132 total, 10856008 free, 1857648 used, 3673476 buff/cache
KiB Swap:      0 total,      0 free,      0 used. 14177928 avail Mem
  PID USER      PR  NI   VIRT   RES   SHR S %CPU%MEM    TIME+
COMMAND
20881 alan    20  0 12016    348    0 S  0.0 0.0 0:00.00 nginx
20882 alan    20  0 12460   1644   932 S  0.0 0.0 0:00.00 nginx
20895 alan    20  0 12016    352    0 S  0.0 0.0 0:00.00 nginx
```

```
20896 alan    20 012460    1628    912 S 0.0 0.0 0:00.00 nginx
```

## df

- The **df** command (short for disk free), is used to display information related to file systems about total space and available space.

**Syntax:** `df [OPTION]...[FILE]...`

- If no file name is given, it displays the space available on all currently mounted file systems.

**Example1:** `df`

Portion of output:

```
Filesystem    1K-blocks    Used Available Use%
Mounted on udev    3996816      0
3996816 0%/dev
tmpfs         804624  10020  794604 2%/run
/dev/sda9     68117056 18036160 46597712 28%/
```

## iostat

- The **iostat** command in Linux is used for monitoring system input/output statistics for devices and partitions.
- It monitors system input/output by observing the time the devices are active in relation to their average transfer rates.
- The iostat produce reports may be used to change the system configuration to raised balance the input/output between the physical disks.
- Note:** iostat is being included in sys stat package. If user doesn't have it, user need to install first. (apt-get installsysstat)

**Syntax:** `iostat`

**Options used:**

- x:** This command shows more details statistics information. iostat command gives I/O devices report utilization as a result. So, it's possible to extend the statistic result for a diagnose in depth with the -x option.
- c:** This command shows only the CPU statistic. It is possible to show the statistic information and report of our cpu with -c option.
- d:** This command displays only the device report. It is possible to only show the status of the device utilization with the help of -d option. It will be going to list information for each connected device.

- **-k:** This command captures the statistics in kilobytes or megabytes. By default, iostat measure the I/O system with the bytes unit.
- **-c 2 2:** To show CPU only report with 2 seconds interval and 2 times reports.
- The **first section** contains CPU report:
  - **%user** :It shows the percentage of CPU being utilization that while executing at the user level.
  - **%nice** :It shows the percentage of CPU utilization that occurred while executing at the user level with a nice priority.
  - **%system** :It shows the percentage of CPU utilization that occurred while executing at the system (kernel)level.
  - **%iowait** :It shows the percentage of the time that the CPU or CPUs were idle during which the system had an outstanding disk I/Orequest.
  - **%steal** :It shows the percentage of time being spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing by another virtual processor.
  - **%idle** :It shows the percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/Orequest.
- The **second section** of the output contains device utilization report:
  - **Device** :The device/partition name is listed in /devdirectory.
  - **tps** :The number of transfers per second that were issued to the device. Higher tps means the processor is busier.
  - **Blk\_read/s** :It shows the amount of data read from the device expressed in a number of blocks (kilobytes, megabytes) per second.
  - **Blk\_wrtn/s** :The amount of data written to the device expressed in a number of blocks (kilobytes, megabytes) per second.
  - **Blk\_read** :It shows the total number of blocks read.
  - **Blk\_wrtn** :It shows the total number of blocks written.

## free

- **free** command is used to view memory consumption

### free-m

	total	used	free	shared	buffers	cached
<b>Mem:</b>	7976	6459	1517	0	865	2248
<b>-/+buffers/cache:</b>		3344	4631			
<b>Swap:</b>	1951	0	1951			

- The m option displays all data in MBs.
- The total os 7976 MB is the total amount of RAM installed on the system, thatis8GB.
- The used column shows the amount of RAM that has been used by Linux, in this case around6.4GB.Thesecondlinetellsthat4.6GBisfree.Thisisthefreememoryinfirst line added with the buffers and cached amount of memory (1517 + 865 + 2248 around 4631 MB= 4.6GB).
- Linux has the habit of caching lots of things for faster performance, so that memory can be freed and used if needed.
- The last line is the swap memory, which in this case is lying entirely free.

## cat /proc/cpuinfo

- The file **/proc/cpuinfo** displays what type of processor the user system is running including the number of CPUs present.
- Portion of the Output: # cat/proc/cpuinfo

```
processor      0
vendor_id     :GenuineIntel
cpufamily     6
model         45
modelname     : Intel(R) Xeon(R) CPU E5-2660 0 @2.20GHz
stepping      6
microcode     1561
cpuMHz        :600.000
cachesize     : 20480KB
```

## cat /proc/meminfo

- On Linux, user can use the command cat /proc/meminfo to **determine how much memory the computer has**.
- This command displays the information stored in the meminfo file located in the /proc directory.
- The total amount of memory will be displayed as MemTotal
- Know that the /proc file system does not contain **real files**.
  - They are rather **virtual files** that contain dynamic information about the kernel and the system.

### Portion of output is

```
$ cat
/proc/meminfo
MemTotal:
           8
167848kB
MemFree:   1409696kB
Buffers:   961452kB
Cached:    2347236kB
SwapCached:      0kB
SwapTotal:  1998844kB
SwapFree:   1998844kB
...
```

- Check the values of MemTotal, MemFree, Buffers, Cached, SwapTotal, SwapFree. They indicate same values of memory usage as the free command.

## Week-11

### Aim : [2] Work on log directory - `/var/log`;

#### Work on log directory:

#### `/var/log`

- It is essential that user know where the log files are located, and what is contained in them. Such files are usually in `/var/log`. Logging is controlled by the associated `.conf` file.
- Some log files are distribution specific and this directory can also contain applications such as samba, apache, lighttpd, mail etc.
- Generated log files will be important for system administration and troubleshooting.
- Learn and understand the content of various log files, which will help user when there is a crisis and user have to look though the log files to identify the issue.

#### Few log files:

- `/var/log/messages` – Contains global system messages, including the messages that are logged during system startup. There are several things that are logged in `/var/log/messages` including mail, cron, daemon, kern, auth, etc.
- `/var/log/auth.log` – Contains system authorization information, including user logins and authentication mechanism that were used.
- `/var/log/boot.log` – Contains information that are logged when the system boots
- `/var/log/lastlog` – Displays the recent login information for all the users. This is not an ascii file. User should use `lastlog` command to view the content of this file.
- `/var/log/user.log` – Contains information about all user level logs
- `/var/log/btmp` – This file contains information about failed login attempts. Use the `last` command to view the `btmp` file. For example, “`last -f /var/log/btmp | more`”
- `/var/log/yum.log` – Contains information that are logged when a package is installed using yum
- `/var/log/cron` – Whenever cron daemon (or an acron) starts a cron job, it logs the information about the cron job in this file

## Week-11

### Aim : [3] System maintenance commands shutdown, reboot, halt, init.

System maintenance commands:

#### Shutdown

- “**Shutdown**” refers to the process of stopping and shutting down a computer or server.
- This involves cutting the power to the main components of the system using a controlled process.
- Applications are closed, active processes and protocols are saved to the hard drive, device drivers are removed, and user settings are saved in the process.
- Linux operating systems can easily be stopped, shut down, and restarted using the shutdown command and its various options.
- **Standard command for shutting down Linux**
  - **shutdown-h**
    - Linux will shut down in under a minute. The “-h” option explicitly stands for the shutting down or powering off of system.
  - **shutdown**
    - User can usually produce the same results by just entering the shutdown command on its own.
- **Standard command for restarting Linux**
  - **shutdown-r**
    - Linux will be restarted in under a minute.
    - The “-r” option stands for reboot or restart.
- **Command for shutting down Linux immediately**
  - **shutdown-h0** // time Specification0
  - **shutdown now**
    - Another common command for shutting down Linux immediately:
- **Command for restarting Linux immediately**
  - **shutdown-r0** // time Specification0
  - **shutdown -rnow**
- **Command for Shutting Down/Restart Linux after 20minutes**
  - **shutdown -h20**

- shutdown+20
- shutdown -r20
- shutdown -r +20
- shutdown-h17:30 // Shutting down at 5.30pm
- shutdown-r17:30 // Restarting at 5.30pm

### Rebooting

- Booting is starting a computer's operating system, so rebooting is to start it for a second or third time.
- Rebooting is usually necessary after a computer crashes, meaning it stops working because of a malfunction.

**sudo reboot**

**sudo systemctl reboot**

**sudo shutdown-r**

### halt

- This commanding Linux is **used to instruct the hardware to stop all the CPU functions.**
- Basically, it reboots or stops the system.

**halt[OPTION]**

**Options used:**

- -f, --force It does not invoke shutdown
- -w, --wtmp-only It will not call shutdown or the reboot system call but writes the shutdown record to /var/log/wtmpfile.
- -p, --power off To behave as poweroff

### init

- **Init** is parent of all Linux processes with PID or process ID of 1.
- It is the first process to start when a computer boots up and runs until the system shuts down.
- Init stands for initialization.
- The role of init is to create processes from script stored in the file /etc/inittab which is a configuration file which is to be used by initialization system.



- It is the last step of the kernel boot sequence.
  - Init script initializes the service. So, it responsible for initializing the system.
  - Init scripts are also called rc scripts (run command scripts)
- Run Levels is the state of init where a group of processes are defined to start at the startup of OS.
- Each run level has a certain number of services stopped or started.  
Conventionally seven run level exist numbers from zero to six.

Run Level	Mode	Action
0	Halt	Shuts down system
1	Single-User Mode	Does not configure network interfaces, start daemons, or allow non-root logins
2	Multi-User Mode	Does not configure network interfaces or start daemons.
3	Multi-User Mode with Networking	Starts the system normally.
4	Undefined	Not used/User-definable
5	X11	As run level 3 + display manager(X)
6	Reboot	Reboots the system

- By default, most of the LINUX based system boots to run level 3 or runlevel5.
- Run levels 2 and 4 are used for user defined run levels
- run level 0 and 6 are used for halting and rebooting the system.

## Week-11

### Aim : [4] System update & repositories- yum & rpm

#### System update & repositories

#### Update the Repositories

##### **sudo apt-getupdate**

- This command refreshes local list of software, making a note of any newer revisions and updates.
- If there's a newer version of the kernel, the command will find it and mark it for download and installation.

#### Run the upgrade

##### **sudo apt-getdist-upgrade**

- The “dist-upgrade” switch asks Ubuntu to handle any **dependencies** intelligently.
  - That is, if a particular software package is **dependent** on another software package to run, this command will make sure that the second package is upgraded before upgrading the first one.
- This method is a safe way to upgrade Ubuntu Linux kernel.
- The kernel updates accessible through this utility have been tested and verified to work with version of Ubuntu.

#### Packaging Manager

- Packaging manager is the software used for managing, installing, updating, upgrading  
etc. of the packages of a system.
- Linux based systems or Linux systems have a lot of such packaging managers in which two are: **yum** and **rpm**.

#### yum

- Yum and RPM are both package managers for Linux systems.
- Yum stands for Yellow dog Updater Modified. They are packaging managers for RPM- based Linux systems.
- They are a high-level front end management package managers for Linux distributions that are RPM-based.
- It can sense and resolve dependencies.

- Yum can only install the packages available in its repository.
- Yum can also scan and upgrade the packages to the latest versions. It also entirely relies on online repositories.

### **rpm**

- RPM stands for Red hat Packaging Manager.
- It can be considered one of the oldest packaging managers that do basic functions like uninstalling, updating, archiving the packages received by the Linux systems.
- It cannot sense and resolve dependencies on its own.
- It can install multiple packages with the condition that we give the correct file name with the .rpm extension.
- RPM does not depend on online repositories for any of its services and it cannot scan or upgrade itself or its packages to the latest versions.

## Week-12

**Aim : [1] Install and configure: DNS server with a domain name of your choice**

### Domain Name Service (DNS)

#### Domain Name Service (DNS)

- **DNS** is an Internet service that maps IP addresses and fully qualified domain names (FQDN) to one another.
  - In this way, DNS alleviates the need to remember IP addresses.
- Computers that run DNS are called **nameservers**.
- Ubuntu ships with **BIND** (Berkley Internet Naming Daemon), the most common program used for maintaining a name server on Linux.
  - BIND is the most common program used for maintaining a name server on Linux.

#### Installation

**Prerequisites:** Install the following packages.

`sudo apt-getupdate`

- **`sudo apt-get installbind9`**

`sudo apt-getupdate`

- **`sudo apt-get installdnsutils`**

`sudo apt-getupdate`

#### Configuration

There are many ways to configure BIND9. Some of the most common configurations are a caching **nameserver**, **primary server**, and **secondary server**.

- When configured as a caching **nameserver** BIND9 will find the answer to name queries and remember the answer when the domain is queried again.
- As a **primary server**, BIND9 reads the data for a zone from a file on its host and is authoritative for that zone.
- As a **secondary server**, BIND9 gets the zone data from another nameserver that is authoritative for the zone.

## Caching Nameserver

- The default configuration acts as a caching server. Simply **uncomment** and edit `/etc/bind/named.conf.options` to set the IP addresses of ISP's DNS servers:

Assume:

Name Server: 192.168.1.10

Primary Server: 192.168.1.10

Secondary Server 192.168.1.11

- To enable the new configuration, restart the DNS server. From a terminal prompt:
- `sudo systemctl restart bind9.service`

## Primary Server

In this section BIND9 will be configured as the **Primary server** for the domain **example.com**. Simply replace **example.com** with FQDN (Fully Qualified Domain Name).

## Forward Zone File

To add a DNS zone to BIND9, turning BIND9 into a Primary server, first edit

`/etc/bind/named.conf.local`:

```
nano /etc/bind/named.conf.local

zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
};
```

**Note:** If bind will be receiving automatic updates to the file as with DDNS, then use `/var/lib/bind/db.example.com` rather than `/etc/bind/db.example.com` both here and in the copy command below.

Now use an existing zone file as a template to create the `/etc/bind/db.example.com` file:

`sudo cp /etc/bind/db.local /etc/bind/db.example.com`

Edit the new zone file **/etc/bind/db.example.com** and change **localhost.** to the FQDN of server, leaving the **additional . at the end**. Change **127.0.0.1** to the **nameserver's A record** for the base domain, example.com. **Also, create an A record for ns.example.com**, the name server in this example:

**Note:** user must increment the *Serial Number* every time user make changes to the zone file. If user make multiple changes before restarting BIND9, simply increment the Serial once.

Now, user can add DNS records to the bottom of the zone file.

**Note:** Many admins like to use the last date edited as the serial of a zone, such as *2020012100*

which is *yyyymmddss* (where *ss* is the Serial Number)

Once made changes to the zone file BIND9 needs to be restarted for the changes to take effect:

```
sudo systemctl restart bind9.service
```

### Reverse Zone File

Now that the zone is setup and resolving names to IP Addresses, a *Reverse zone* needs to be added to allows DNS to resolve an address to a name.

edit **/etc/bind/named.conf.local** and add the following:

**Note:** Replace 1.168.192 with the **first three octets** of whatever network user are using. Also,

name the zone file **/etc/bind/db.192** appropriately. It should match the first octet of user network.

Now create the **/etc/bind/db.192** file: (copy the contents from default file **/etc/bind/db.127**)

```
sudo cp /etc/bind/db.127 /etc/bind/db.192
```

Next edit **/etc/bind/db.192** changing the same options as **/etc/bind/db.example.com**:

## Secondary Server Settings

Once a *Primary Server* has been configured a *Secondary Server* is highly recommended in order to maintain the availability of the domain should the Primary become unavailable.

**First**, on the **Primary server**, the zone transfer needs to be allowed. Add the allow-transfer option to the example Forward and Reverse zone definitions in **/etc/bind/named.conf.local**:

**Note:**

Replace 192.168.1.11 with the IP Address of user **Secondary nameserver**. Restart BIND9 on the Primary server:

```
sudo systemctl restart bind9.service
```

**Next**, on the **Secondary server**, install the bind9 package the same way as on the Primary. Then edit the **/etc/bind/named.conf.local** and add the following declarations for the Forward and Reverse zones:

Restart BIND9 on the Secondary server:

```
sudo systemctl restart bind9.service
```

In **/var/log/syslog** user should see something similar to the following (some lines have been split to fit the format of this document):

```
cat /var/log/syslog | tail -30
```

```
client 192.168.1.10#39448: received notify for zone '1.168.192.in-addr.arpa' zone 1.168.192.in-addr.arpa/IN: Transfer started.
transfer of '100.18.172.in-addr.arpa/IN' from
192.168.1.10 #53: connected using
192.168.1.11 #37531
zone 1.168.192.in-addr.arpa/IN: transferred serial 5
transfer of '100.18.172.in-addr.arpa/IN' from
192.168.1.10#53: Transfer completed: 1 messages,
6 records, 212 bytes, 0.002 secs (106000 bytes/sec)
zone 1.168.192.in-addr.arpa/IN: sending notifies (serial 5)
```

```
client 192.168.1.10#20329: received notify for zone
'example.com' zone example.com/IN: Transfer started.
```

```
transfer of 'example.com/IN' from 192.168.1.10#53:
connected using 192.168.1.11#38577
zone example.com/IN: transferred serial 5
transfer of 'example.com/IN' from 192.168.1.10#53: Transfer completed: 1
messages, 8 records, 225 bytes, 0.002 secs (112500 bytes/sec)
```

If user want to have Primary DNS notifying other Secondary DNS Servers of zone changes, user can add **also-notify { Ip address; }** to **/etc/bind/named.conf.local** as shown in the example below at **Primary Server**:

nano /etc/bind/named.conf.local
<pre>zone "example.com" {     type master;     file "/etc/bind/db.example.com";     allow-transfer { 192.168.1.11; };     <b>also-notify { 192.168.1.11; };</b> };  zone "1.168.192.in-addr.arpa" {     type master;     file "/etc/bind/db.192";     allow-transfer { 192.168.1.11; };     <b>also-notify { 192.168.1.11; };</b> };</pre>

## Troubleshooting:

The first step in testing BIND9 is to add the nameserver's IP Address to a host's resolver. The Primary nameserver should be configured as well as another host to double check things.

In the end **nameserver** line in **/etc/resolv.conf** should be pointing at 127.0.0.53 and should have a search parameter for domain. Something like this:

```
nameserver
127.0.0.53
search
example.co
m
```

**Note: User** can add the IP Address of the **Secondary nameserver** to client configuration in case the Primary becomes unavailable.



**Example 1: using dig** If user installed the **dnsutils** package user can test setup using the DNS lookup utility dig:

- After installing BIND9 use dig against the loopback interface to make sure it is listening on port 53. From a terminal prompt:

```
dig -x 127.0.0.1
```

user should see lines similar to the following in the command output:

```
;; Query time: 1 msec
```

```
;; SERVER: 192.168.1.10#53(192.168.1.10)
```

- If user have configured BIND9 as a *Caching* nameserver “dig” an outside domain to check the query time:

```
dig ubuntu.com
```

Note the query time toward the end of the command output:

```
;; Query time: 49 msec
```

After a second dig there should be improvement:

```
;; Query time: 1 msec
```

## Common Record Types used during DNS Setups:

This section covers some of the most common DNS record types.

- A record: This record maps an IP Address to a hostname.  

```
www IN A
192.168.1.12
```
- CNAME record: Used to create an alias to an existing A record. User cannot create a CNAME record pointing to another CNAME record.  

```
Web IN CNAME www
```
- MX record: Used to define where email should be sent to. Must point to an A record, not a CNAME.  

```
@ IN MX 1
mail.example.com. mail
IN A
192.168.1.13
```
- NS record: Used to define which servers serve copies of a zone. It must point to an A record, not a CNAME.  

```
@ IN NS
ns.example.com. @
IN NS
ns2.example.com. ns
IN A
```

192.168.1.10

## Week-12

**Aim : [2] FTP server on LINUX and transfer files to demonstrate it's working**

### ftp

**ftp** is the user interface to the Internet standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

Examples below are to upload and download files using ftp service.

#### Prerequisites:

- Install **Ftp** (FTP-Client) and **vsftpd** (FTP-Server) using Software Manager.
- Change **/etc/vsftpd.conf** file to enable write permission (other settings can be done according to requirement).  
`nano/etc/vsftpd.conf`
- Search for **# write\_enable=YES** (around 30<sup>th</sup> line in that file)
- **Uncomment** and Save and Exit
- Restart the Service: `systemctl restartvsftpd`

#### Example 1: Get the connection to ftp

❖ **ftp 172.16.20.11621**

- Enter user name and password for connection, this will take user to ftp utility and then issue ftp set of command
- For more command take the help by issuing? And to quit utility type **quit** command.

## Week-12

### Aim : [3] Apache web server and create virtual hosts.

#### Apache web server:

- The Apache web server is the most popular way of serving web content on the internet.
- It accounts for more than half of all active websites on the internet and is extremely powerful and flexible.
- Apache breaks its functionality and components into individual units that can be customized and configured independently. The basic unit that describes an individual site or domain is called a virtual host.
- These designations allow the administrator to use one server to host multiple domains or sites off of a single interface or IP by using a matching mechanism. This is relevant to anyone looking to host more than one site off of a single server.
- Each domain that is configured will direct the visitor to a specific directory holding that site's information, never indicating that the same server is also responsible for other sites.
- This scheme is expandable without any software limit as long as server can handle the load.

#### Prerequisites:

- For all commands use **sudo**bash
- Install apache Server: **apt-get installapache2**

#### Step 1: Creating the Directory Structure

- Make a directory structure that will hold the site data that we will be serving to visitors.
- Document **root** (the top-level directory that Apache looks at to find content to serve) will be set to individual directories under the **/var/www**directory.
- Create a directory here for virtual host.
- Within**this**directory,create**public\_html**folderthatwillholdouractualfiles.Thisgi ves some flexibility in hosting.

```
mkdir -p/var/www/example.com/public_html
```

## Step 2: Granting Permissions

- Directory structure for host files owned by our root user. If user want regular user to be able to modify files in web directories, can change the ownership by doing this:
  - **chown -R \$USER:\$USER /var/www/example.com/public\_html**
- By doing this, regular user now owns the **public\_html** subdirectories where we will be storing our content.
- Also modify permissions a little bit to ensure that read access is permitted to the general web directory and all of the files and folders it contains so that pages can be served correctly:
  - **chmod -R 755/var/www**

## Step 3: Creating Demo Pages for Each Virtual Host

- Once directory structure in place then create some content to serve.
- Design a very simple Web page.
- `nano/var/www/example.com/public_html/index.html`
  - In this file, create a simple HTML document that indicates the site it is connected to. The file looks like this:

```
<html>
  <head>
    <title>Welcome to Apache Server Experiment</title>
  </head>
  <body>
    <h1>Success! The virtual host is working Fine!</h1>
  </body>
</html>
```

- Save and close the file when finished.

## Step 4: Creating New Virtual Host Files

- Virtualhostfilesarethefilesthat specify the actual configuration of four virtual hosts and dictate how the Apache web server will respond to various domain requests.
- Apache comes with a default virtual host file called 000-default.conf that can be used as a jumping off point.
- Now copy it over to create a virtual host file for domains.
- The default Ubuntu configuration requires that each virtual host file end in.conf.

### Creating the First Virtual Host File

- Start by copying the file for the domain:  
  
**`sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/example.com.conf`**
- The file will look something like this (Comments have been removed here to make the file more approachable): After removing the comments the file contents look like (First Column).
- Add few contents to the above file. And also make few changes to the file such that it finally looks like (Second Column):
- Save and close the file.

## Step 5: Enabling the New Virtual Host Files

- Now virtual host files have been created, Enable them. Apache includes some tools that allow us to do this.
- Use the **a2ensite** tool to enable each of our sites like this:
  - **`sudo a2ensite example.com.conf`**
- Next, disable the default site defined in **000-default.conf**:
  - **`sudo a2dissite 000-default.conf`**
- When above task finished, Restart Apache to make these changes take effect:
  - **`sudo systemctl restart apache2`**

### Step 6: Setting Up Local Hosts File(Optional)

- If admin have not been using actual domain names to test this procedure and **have been using** some **example domains** instead, admin can at least test the functionality of this process by temporarily modifying the hosts file on local computer.
- edit local file with administrative privileges by typing:
  - **sudo nano/etc/hosts**

### Testing working of Webserver

- Go to the other machines /etc/hosts and do the above settings (**step 6**)like
  - **172.16.20.107example.com**
- Try **http://example.com**, this will display the web page of virtual host.

## Week-13

### Aim : [1] Basic commands for storage partitions

#### Storage Management Basic Commands for Storage

##### Partitions

##### **fdisk**

- fdisk is used to check the partitions on disk.
- The fdisk command can display the partitions and details like file system type.
- However, it does not report the size of each partition.
- **sudofdisk-l**
- Each device is reported separately with details about size, sectors, id and individual partitions.

##### **sfdisk**

- sfdisk utility purpose similar to fdisk, but with more features.
- It can display the size of each partition in MB.

##### ○ **sudosfdisk -l-uM**

##### **cdisk**

- cfdisk is a Linux partition editor with an interactive user interface based on **ncurses**.
- It can be used to list out the existing partitions as well as create or modify them.
- An example of how to use cfdisk to list the partitions.
- cfdisk works with one partition at a time.
- So, if user needs to see the details of a particular disk, then pass the device name to cfdisk.

**sudocfdisk/dev/sdb**

##### **parted**

- Parted utility is to list out partitions and modify them if needed.

##### ○ **sudo parted-l**

##### **df**

- df is not a partitioning utility, but prints out details about only mounted filesystems.
- The list generated by df even includes file systems that are not real disk partitions.

- **df-h**
  - **df -h | grep ^/dev**

**Note:** df shows only the mounted file systems or partitions and not all.

## pydf

- pydf is an improved version of df, written in python.
- Prints out all the hard disk partitions in an easy-to-read manner.
- **pydf**
  - pydf is limited to showing only the mounted filesystems.

## lsblk

- Lists out all the storage blocks, which includes disk partitions and optical drives.
- Details include the total size of the partition/block and the mount point if any.
- Does not report the used/free disk space on the partitions.
- **lsblk**
  - If there is no MOUNTPOINT, then it means that the file system is not yet mounted. For cd/dvd this means that there is no disk.
  - lsblk is capable of displaying more information about each device like the label and model. Check out the man page for more information

### Display UUID and Model of device

- The "-o" option can be used to specify the columns to display.
- **lsblk -o PATH,SIZE,RO,TYPE,MOUNTPOINT,UUID,MODEL**
  - The above output has all the necessary information about all the storage devices present on the system or connected via usb.
  - This is the best command to see all information about storage devices together in one place.

## blkid

- Prints the block device (partitions and storage media) attributes like uuid and filesystem type. Does not report the space on the partitions.
- **Sudo blkid**



### **hwinfo**

- The `hwinfo` is a general purpose hardware information tool and can be used to print out the disk and partition list.
- The output however does not print details about each partition like the above commands.

- **`hwinfo --block--short`**

### **inxi**

- `inxi` command display information about various hardware components present on the system.
- To display information about the disk drives and storage devices use the "-D" option with `inxi`.

- **`inxi -D-xx`**

## Week-13

**Aim : [2] Install and configure LVM and Add Disk and CreateStandard & LVM Partition. Add virtual disk and create a new LVM partition(pvcreate, vgcreate, lvcreate) Extend disk using LVM**

### Logical Volume Management (LVM)

- **LVM**, or Logical Volume Management, is a storage device management technology that
  - gives users the power to pool and abstract the physical layout of component storage devices for easier and flexible administration.
- The main advantages of LVM are increased abstraction, flexibility, and control.
- Logical volumes can have meaningful names like “databases” or “root-backup”.
- Volumes can be resized dynamically as space requirements change and migrated between physical devices within the pool on a running system or exported easily.
- LVM also offers advanced features like snapshotting, striping, and mirroring.

### LVM Storage Management Structures

LVM functions by layering abstractions on top of physical storage devices. The basic layers that

LVM uses, starting with the most primitive, are.

- **Physical Volumes:**
  - **LVM utility prefix:**pv...

**Description:** Physical block devices or other disk-like devices (for example, other devices created by device mapper, like RAID arrays) are used by LVM as the raw

- building material for higher levels of abstraction. Physical volumes are regular storage devices. LVM writes a header to the device to allocate it for management.

- **Volume Groups:**
  - **LVM utility prefix:**vg...
  - **Description:** LVM combines physical volumes into storage pools known as volume groups. Volume groups abstract the characteristics of the underlying devices and function as a unified logical device with combined storage capacity of the component physical volumes.

- **Logical Volumes:**
- **LVM utility prefix:** lv... (generic LVM utilities might begin with lv...)

**Description:** A volume group can be sliced up into any number of logical volumes. Logical volumes are functionally equivalent to partitions on a physical disk, but with much more flexibility. Logical volumes are the primary component that users and applications will interact with.

Each volume within a volume group is segmented into small, fixed-size chunks called **extents**. The size of the extents is determined by the volume group (all volumes within the group conform to the same extent size).

### **Initial Settings for LVM if Virtual Box Linux image is used (Better Choice for Practice):**

Prior the extension is made user need to assure that user already know the actual state of the machine's hard disk.

**Vgs** Command may be helpful in finding hard disk in **vg** where Linux is installed.

### **Start from Scratch**

- Turn off Linux OS in Virtual Box (if it is already loaded)
- Click on the '**Settings**' option on the VirtualBox Manager after having selected virtual machine which user intend to perform a disk extension. In my case, it's the '**centos6**' one.
- Then, on the '**Storage**' option, next to the "**Controller: SATA**" there is an icon to "**add new hard disk**".
- Once user have click on the "**add new hard disk**" it will prompt user to "cancel" "choose existing disk" and "create new disk". Choose "**create new disk**". Of course, user can also choose an existing disk, but here we are adding a completely new fresh disk.
- Afterward, it will prompt a "create Virtual Hard Drive" box. Choose "**VDI**". Click on next, then on "**dynamically allocated**". Give a new name to hard disk. In this case, added a new **100GB** hard disk. Click on **create** and its done.
- Boot the Linux on VirtualBox. then try the **lsblk** command to see new hard disk.
- Then Carry on normal LVM settings.

## Working with LVM

- Scan the system for block devices that LVM can see and manage.

**sudo lvm diskscan**

**Output:**

```

/dev/ram0[      64.00 MiB]
/dev/ram1[      64.00 MiB]
...
/dev/ram15[     64.00MiB]
/dev/sdb [   100.00GiB]           //
physical Device 1 disks
17 partitions
0 LVM physical volume whole
disks 0 LVM physical volumes
  
```

The partitions are mostly **/dev/ram\*** partitions that are used the system as a [Ram disk](#) for performance enhancements. The disks in this example is **/dev/sdb**, which has 100G of space.

***Note: Warning:** Make sure that user double-check that the devices user intend to use with LVM do not have any important data already written to them. Using these devices within LVM will overwrite the current contents. If user already have important data on server, make backups before proceeding.*

- Now mark the **physical device as physical volumes** within LVM using the **pvcreate**

command:

**Output:**

**sudo pvcreate /dev/sdb**

```
[  pvremove/dev/sdb           // for removing PhysicalVolume  ]
```

Physical volume **"/dev/sdb"** successfully created

- This will write an LVM header to the devices to indicate that they are ready to be added to a volume group.

- Verify that LVM has registered the physical volumes by typing:

**sudo pvs**

**Output:**

PV	VG	Fmt	Attr	PSize	PFree
/dev/sdb	lvm2	---	100.00g	100.00g	

The device are present under the pv column, which stands for physical volume.

### Add the Physical Volumes to a Volume Group

- Select a good name for the volume group. Here, say volume group **LVMVolGroup** for simplicity.
- To create the volume group and add physical volumes to it in a single command, type:

**sudo vgcreate LVMVolGroup /dev/sdb**

```
[      vgremoveVolumeGroupName      // for removingVolumeGroup
]
```

**Output:**

Volume group "LVMVolGroup" successfully created

- Verify that physical volumes are now associated with new volume group using pvs command:

PV	VG	Fmt	Attr	PSize	PFree
/dev/sdb	<b>LVMVolGroup</b>	lvm2	a--	100.00g	100.00g

- See a brief summary of the volume group itself by typing:

**sudo vgs**

**Output:**

VG	#PV	#LV	#SN	Attr	VSize	VFree
LVMVolGroup	1	0	0	wz--n-	100.00g	100.00g

- Here, Volume group currently has one physical volumes, zero logical volumes,

and has the combined capacity of the underlying devices.

### Creating Logical Volumes from the Volume Group Pool

- Use the available volume group for different purposes, which will be considered as Logical Volume.
- User just need to supply the size of the logical volume and a name.
- Assume, create four separate logical volumes out of our volume group:
  - 10G “**projects**” volume
  - 5G “**www**” volume for web content
  - 20G “**db**” volume for database
  - “**workspace**” volume that will fill the remaining space
- To create logical volumes, use the **lvcreate** command.
- User must pass in the volume group to pull from, and can name the logical volume with the **-n** option.
- To specify the size directly, use the **-L** option. [If, instead, user wish to specify the size in terms of the number of extents, user can use the **-l** option.]

Create the first three logical volumes with the **-L** option like

```
this: sudo lvcreate -L 10G -n projects
LVMVolGroup sudo lvcreate -L 5G -n www
LVMVolGroup sudo lvcreate -L 20G -n
dbLVMVolGroup
```

**Output:**

Logical volume "projects" created. Logical volume "www" created.  
Logical volume "db" created.

- See the logical volumes and their relationship to the volume group by selecting custom output from the **vg** command:

```
sudo vgs -o +lv_size,lv_name
```

**Output:**

VG	#PV	#LV	#SN	Attr	VSize	VFree	LSize	LV
LVMVolGroup	1	3	0	wz--n-	100g	65.00g	10.00g	projects
LVMVolGroup	1	3	0	wz--n-	100g	65.00g	05.00g	www

```
LVMVolGroup 1 3 0 wz--n- 100g 65.00g 20.00g db
```

- Allocate the rest of the space in the volume group to the “workspace” volume using the - lflag, which works in extents.

User can also provide a percentage and a unit to better communicate our intentions.

- Here, allocate the remaining free space, so user can pass in100%FREE:

```
sudo lvcreate -l 100%FREE -n workspace LVMVolGroup
```

## Output:

Logical volume "workspace" created.

- Recheck the volume group information,  
sudo vgs -o +lv\_size,lv\_name

## Output:

VG	#PV	#LV	#SN	Attr	VSize	VFree	LVSize	LV
LVMVolGroup 1	4	0	0	wz--n-	100g	0	10.00g	projects
LVMVolGroup 1	4	0	0	wz--n-	100g	0	05.00g	www
LVMVolGroup 1	4	0	0	wz--n-	100g	0	20.00g	db
LVMVolGroup 1	4	0	0	wz--n-	100g	0	65.00g	workspace

## Format and Mount the Logical Volumes

- Now user can use logical volumes as normal block devices.
- The logical devices are available within the /dev directory just like other storage devices.
- User can access them in two places:

```
/dev/volume_group_name/logical_volume_name
```

```
[OR /dev/mapper/volume_group_name-logical_volume_name ]
```

- So, to format our four logical volumes with the Ext4 filesystem, user can type:

```
1. sudo mkfs.ext4/dev/LVMVolGroup/projects
```

```
2. sudo mkfs.ext4/dev/LVMVolGroup/www
```

```
3. sudo mkfs.ext4/dev/LVMVolGroup/db
```

## 4. `sudo mkfs.ext4/dev/LVMVolGroup/workspace`

OR user can type:

1. `sudo mkfs.ext4/dev/mapper/LVMVolGroup-projects`
2. `sudo mkfs.ext4/dev/mapper/LVMVolGroup-www`
3. `sudo mkfs.ext4/dev/mapper/LVMVolGroup-db`
4. `sudo mkfs.ext4/dev/mapper/LVMVolGroup-workspace ]`

- After formatting, user can create mountpoints:

`sudo mkdir /mnt/projects sudo mkdir -p /mnt/www sudo mkdir -p /mnt/db  
sudo mkdir -p /mnt/workspace`

- Then mount the logical volumes to the appropriate location using commands:

<b>1.sudo mount</b>	<b>/dev/LVMVolGroup/projects</b>	<b>/mnt/projects</b>
<b>2.sudo mount</b>	<b>/dev/LVMVolGroup/www</b>	<b>/mnt/www</b>
<b>3.sudo mount</b>	<b>/dev/LVMVolGroup/db</b>	<b>/mnt/db</b>
<b>4.sudo mount</b>	<b>/dev/LVMVolGroup/workspace</b>	<b>/mnt/workspace</b>

Editor will be opened for /etc/fstab. Add the below entries at the end of the file, save and exit.

`/dev/LVMVolGroup/projects /mnt/projects ext4 defaults,nofail 0 0`

`/dev/LVMVolGroup/www /mnt/www ext4 defaults,nofail 0 0`

`/dev/LVMVolGroup/db /mnt/db ext4 defaults,nofail 0 0`

`/dev/LVMVolGroup/workspace /mnt/workspace ext4 defaults,nofail 0 0`

- The operating system should now mount the LVM logical volumes automatically atboot.

## Resizing a Physical Volume

- If user wants to change the size of an underlying block device for any reason, can use the `pvresize` command to update LVM with the new size.
- User can execute this command while LVM is using the physical volume.