

# Graded Exercise

## Operating System and Administration 20CS42P

### IV Semester Computer Science

# Certificate

**Name :** **Class: IV Semester**

**Register Number :**

**Institution :**

This is certified to be the bonafide work of the student in the **Operating System and Administration - 20CS42P** Laboratory during the academic year 202 – 202

***Course Coordinator***

***Examiner Signature***

1. \_\_\_\_\_

2. \_\_\_\_\_

## **List of Experiments**

<b>Sl. No.</b>	<b>Name of the Experiment</b>	<b>Date of Experiment</b>	<b>Remarks</b>
1	Operating system (OS) and Administration		
2	Install and configure virtual machine – Virtual box		
3	File and Directory Commands		
4	Process creation and Management		
5	Process Synchronization		
6	Memory Management		
7	Shell Programming		
8	Automation of System Tasks		
9	Network Management		
10	User Authentication		
11	System/Log monitoring commands and System Information/Maintenance Commands		
12	Domain Name Service (DNS)		
13	Storage Management		

## 01 – Operating system (OS) and Administration

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.

### 1. 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 meet 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 (**netinstall**), 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.

==== \* ====

## **2. Boot methods**

**Bootting** 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 **restarts**. 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.

==== \* ====

## **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 more 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. A computer can organize its I/O to preserve file systems with the help of a global file system.
- **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.

=== \* ===

### **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.

- **Livepatch**

Livepatch 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.

=== \* ===

**Work on Practice Session**



**Work on Practice Session**

## 02 – 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, 3D virtualization</li> <li>• Multigeneration branched snapshots.</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 Ubuntu 16.04
- [https://download.virtualbox.org/virtualbox/6.1.32/virtualbox-6.1\\_6.1.32-149290~Ubuntu~xenial\\_amd64.deb](https://download.virtualbox.org/virtualbox/6.1.32/virtualbox-6.1_6.1.32-149290~Ubuntu~xenial_amd64.deb)
  - (Choose appropriate image from repository according to current Host Linux OS distribution)
- Go to the Downloads → Select the Downloaded Virtual Box Software → Open → Check for the Compatibility (if not compatible, download the suitable image from 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 bit depending upon user system architecture
- Give RAM Memory to the VM. **Generally 1024 Mbytes**
- Select the hard disk space, **Go for 20.00 GB.**
- Create **New Hard Drive** or Select one from the list or from another location. **Go for Create a Virtual hard drive now.**
- Select the type of the file from 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**, then 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 Guest OS.

### Experiments to be conducted (If Required... ):

- **Demonstrate use of Snap Shots in Guest OS**
  - The changes made in the guest can be stored in the form of snapshots. And it can be restored (Rolled back) whenever, previous changes are required.
  - To take the snapshot, the VM that the user is working with must NOT be running.
  - User can restore a snapshot if the VM is either in a saved state or powered off.
  - To restore a state, do the following.
    1. Select the VM to work with from the left pane in the main window.
    2. Click the Snapshots button in the upper right corner.
    3. Right-click the snapshot that the user wants to restore.
    4. Click Restore Snapshot (**Figure C**).
    5. In the resulting window, uncheck the box for Create A Snapshot Of The Current Machine State.
    6. Click Restore.
    7. Allow the restore to complete.
- **Share folders between Guest and Host using Guest Additions Image...**
  1. **Settings in Host Virtual Box Manager (VBM):**
    1. Select Guest OS in Host VBM → Select “**Shared Folders**” in Settings → Select “**Add Share +**”
    2. Select the folder to be shared in Host OS in “**Folder Path**”
    3. Selected Folder will appear in “**Folder Name**”, if required check “**Read Only**”, “**Auto-Mount**”, “**Make Permanent**”. Click **Ok**

## 2. Settings in Guest Virtual Box Manager:

1. Select Guest OS → Select “**Devices**” in Menu Bar → Select “**Insert Guest Additions CD Image ...**”. Continue with running the Installation.
2. Add this Guest user to Virtualbox by executing the Command ( only in case of Linux Guest operating system, This step is NOT needed in case of Windows Guest Operating System)  
**sudo usermod -a -G vboxsf** (user login Name)
3. Restart the Guest OS.
4. Check the Shared Host Folder appear in Guest OS

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

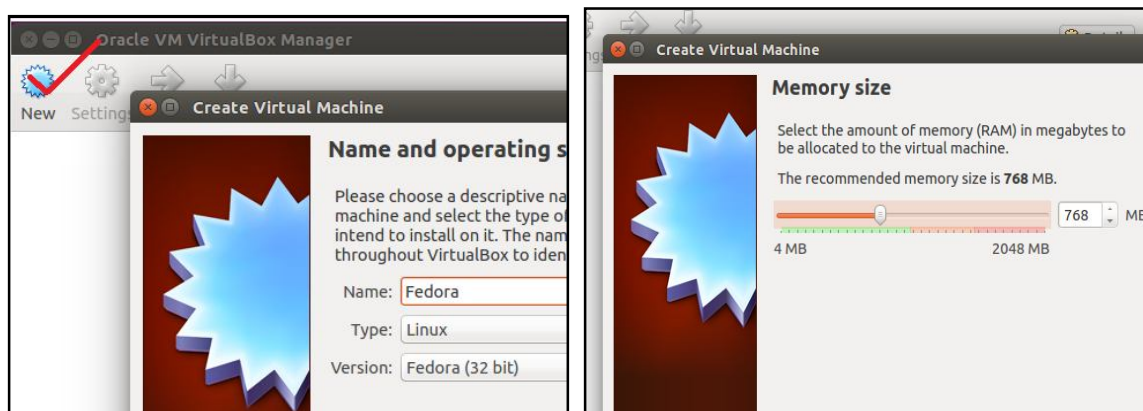
### Virtual Box Settings in Host Virtual Box Manager:

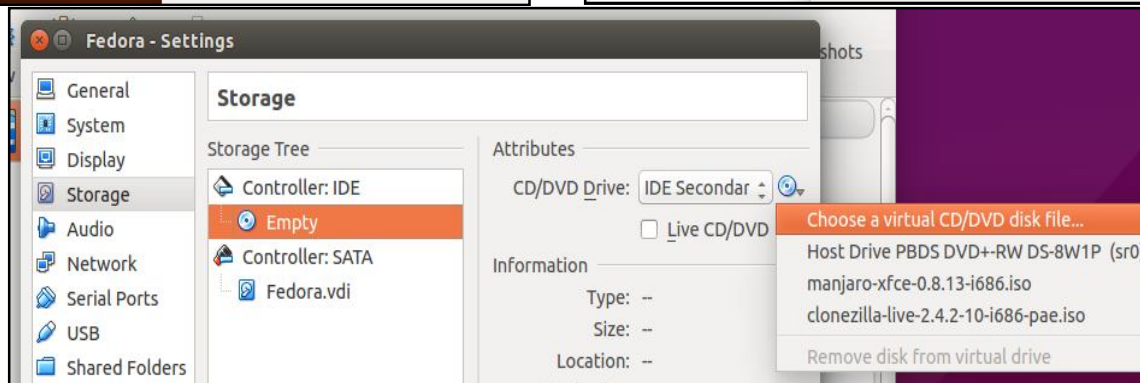
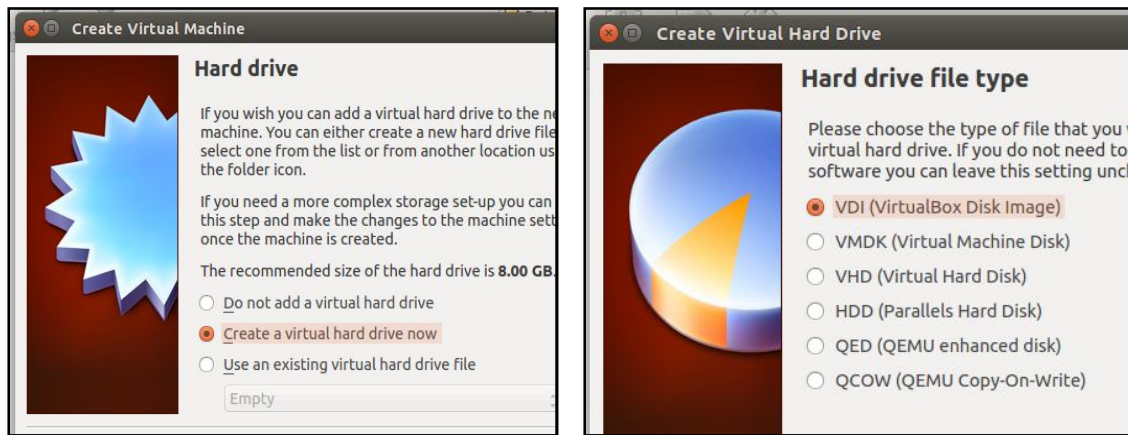
1. In Virtual Box Manager, Select “**File**”, → “**Preferences**” → Network → Host only Networks (Tab).
2. Select icon for adding “**New Host only Networks**” (See vboxnet0 gets added”
3. Change the properties of vboxnet0
4. Change the IP Address to valid Class-C Address (Say 192.100.100.5).

### Guest VBM Settings in Virtual Box:

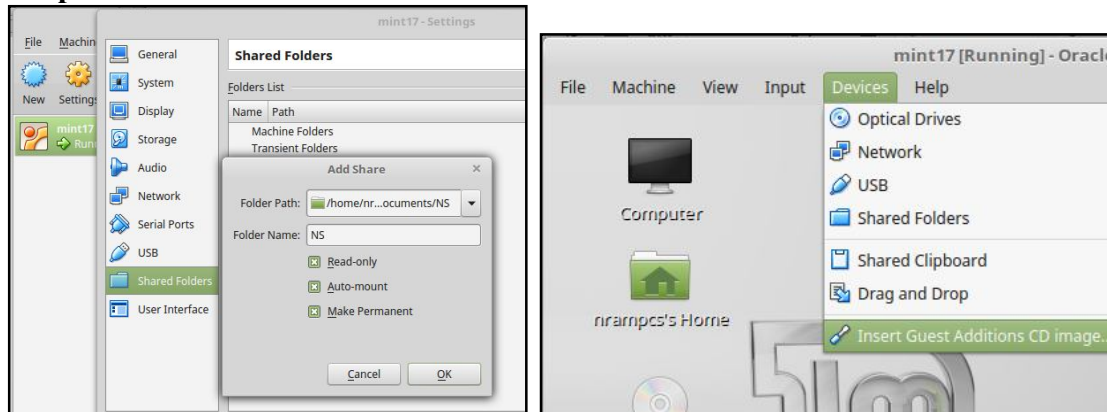
- Select Guest OS in Virtual Box. Select settings for Network.
- Change the settings for Network. - Select “**Adapter 1**” Tab – Change “**Attached to**” to “**Host-only-Adapter**”
- Select the “**Advanced**” Frame and Select “**Allow All**” in “**Promiscuous Mode**”. Click “**Ok**”
- Go to Guest OS. And assign one valid IP Address to the Guest OS (Say 192.100.100.6)
- Use Ping command to check the Communication between 192.100.100.5 and 192.100.100.6.

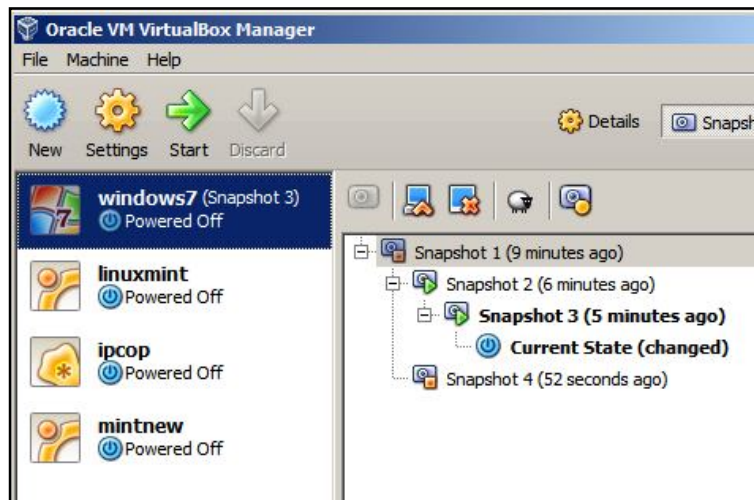
## Screen Shots (working Virtual Box in Linux Operating System – Host is Linux):





### Snap Shot:





==== \* =====

### Download and Install a 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. With it, one can run any application that is designed to run on VT102, VT220, and xterm terminals.
- 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 -y mate-terminal
```

==== \* =====

### Significance of man Command

- **man** command in Linux is used to display the **user manual** of any command that run on the terminal.
- It provides a detailed view of the command which includes NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS and SEE ALSO.
- **Every manual is divided into the following sections:**
  - Executable programs or shell commands
  - System calls (functions provided by the kernel)
  - Library calls (functions within program libraries)
  - Games
  - Special files (usually found in /dev)
  - File formats and conventions eg /etc/passwd
  - Miscellaneous (including macro packages and conventions),
  - System administration commands
- **Syntax**
  - `man [SECTION-NUM] [COMMAND NAME]`  
==== \* =====

**Work on Practice Session**



**Work on Practice Session**

### 03 – File and Directory Commands

#### 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</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>

1 = access modes/permissions      5 = size (in bytes)  
 2 = number of links                      6 = date/time of last modification  
 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 -RI**

==== \* ====

#### pwd (Present Working Directory)

- Will give the present working directory path informatoin

==== \* ====

#### mkdir (Make Directory)

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

**Example 1: mkdir polytechnic**      // will create directory with name polytechnic

**Example 2:** Create some additional subdirectories within **polytechnic**. List new directories after the command completes.

**mkdir polytechnic/mechanical polytechnic/computer polytechnic/civil**  
**ls polytechnic**

**Example 3:** Try to create a directory in a location where user doesn't have permission.

**mkdir /etc/mydir**

**Output:**      **mkdir: cannot create directory '/etc/mydir': Permission denied**

==== \* ====

#### cd (Change Directory)

- **cd** is used to Change Directory
- Change to home direcotory
  - **cd**                                      // Change to default home directory
  - **cd ~**                                      // Change to default home directory

- Change to a subdirectory within user home directory
  - **cd polytechnic/computer**
  - pwd
- Go up one level back to the current directory's parent directory
  - **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)
  - **rmdir /polytechnic/computer** (assume /polytechnic/computer folder exists)

==== \* =====

### **File Manipulation Commands:**

#### **Creating File:**

- **touch:** will create empty files
  - Example: touch a.txt b.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:** **cat filename**
  - **cat /path/filename**
- **Example:**
  - **cat a.txt**
  - **cat polytechnic/computer/a.txt** (Assume a.txt is present in polytechnic/computer directory)

==== \* =====

**rm (Remove File)**

- Will remove (delete) the files
  - **Syntax:**        **rm FileName**
    - **rm path/FileName**
  - **Example:**
    - **rm a.txt**                      // Assume a.txt is present in current directory
    - **rm polytechnic/civil/a.txt**    // Assume polytechnic/civil folder has file a.txt
    - **rm -i \***                      // will delete all the files but interactively, asking user about confirmation of deleting the files
    - **rm polytechnic/civil/\***        // will delete all files in polytechnic/civil folder
- ==== \* ====

**cp (Copy)**

- Used to create a copy of a 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 file1 file2**
  - **cp path1/file1 path2/file2**    // creates copy of file1 in current directory or in specified directory
  - **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.txt polytechnic/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 alert the user when file2 is already exists.
    - **cp -i file1 file2**                      (to test, make sure both files exists)
  - 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 subdir1 subdir4**
  - 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 OldFileName NewFileName**
    - **ls**
  - **Example 2:**
    - **mv OldFolderName NewFolderName**
    - **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.

A **named** pipe, however, can last as long as the system is up, beyond the life of the process. It can be deleted if no longer used.

### 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.
  - A FIFO file is a special kind of file on the local storage which allows two or more processes to communicate with each other by reading/writing to/from this file.
  - A FIFO special file is entered into the file system by calling **mkfifo()** in C.
  - Once a FIFO special file is created in this way, any process can open it for reading or writing, in the same way as an ordinary file. However, it has to be **open at both ends simultaneously** before it can proceed to do any input or output operations on it.
- **Creating a FIFO file:** In order to create a FIFO file, a function calls i.e. **mkfifo** is used.
  - **Example:** a.c and b.c communication

// a.c	// b.c
<pre>#include &lt;stdio.h&gt; #include &lt;string.h&gt; #include &lt;fcntl.h&gt; #include &lt;sys/stat.h&gt; #include &lt;sys/types.h&gt; #include &lt;unistd.h&gt; int main() {     int fd;      // FIFO file path     char * myfifo = "/tmp/myfifo";      // Creating the named file(FIFO)     // mkfifo(&lt;pathname&gt;, &lt;permission&gt;)     mkfifo(myfifo, 0666);      char arr1[80], arr2[80];     while (1)     {         // Open FIFO for write only         fd = open(myfifo, O_WRONLY);          // Take an input arr2ing from user.         // 80 is maximum length         fgets(arr2, 80, stdin);          // Write the input arr2ing on FIFO</pre>	<pre>#include &lt;stdio.h&gt; #include &lt;string.h&gt; #include &lt;fcntl.h&gt; #include &lt;sys/stat.h&gt; #include &lt;sys/types.h&gt; #include &lt;unistd.h&gt; int main() {     int fd1;      // FIFO file path     char * myfifo = "/tmp/myfifo";      // Creating the named file(FIFO)     // mkfifo(&lt;pathname&gt;, &lt;permission&gt;)     mkfifo(myfifo, 0666);      char str1[80], str2[80];     while (1)     {         // First open in read only and read         fd1 = open(myfifo, O_RDONLY);         read(fd1, str1, 80);          // Print the read string and close         printf("User1: %s\n", str1);         close(fd1);</pre>

<pre>// and close it write(fd, arr2, strlen(arr2)+1); close(fd);  // Open FIFO for Read only fd = open(myfifo, O_RDONLY);  // Read from FIFO read(fd, arr1, sizeof(arr1));  // Print the read message printf("User2: %s\n", arr1); close(fd); } return 0; }</pre>	<pre>// Now open in write mode and write // string taken from user. fd1 = open(myfifo, O_WRONLY); fgets(str2, 80, stdin); write(fd1, str2, strlen(str2)+1); close(fd1); } return 0; }</pre>
---	---

==== \* ====

### Unnammed 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()
```

This example creates an I/O channel. The output shows the data written into one end and read from the other.

<pre>#define _POSIX_SOURCE #include &lt;unistd.h&gt; #include &lt;string.h&gt; #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  void reverse(char *s) { // char *first, *last, temp; char temp; int i, n; n = strlen(s)-1; for(i=0; i&lt;n/2; i++) { temp = s[i]; s[i] = s[n-i]; s[n-i]=temp; } }</pre>	<pre>if (pipe(p1) != 0    pipe(p2) != 0) perror("Any one PIPE is failed");  if (fork() == 0) { close(p1[1]); close(p2[0]);  read(p1[0], buf, sizeof(buf)); reverse(buf); write(p2[1], buf, strlen(buf)+1); exit(0); } else { close(p1[0]); close(p2[1]); printf("parent is writing '%s' to pipe 1\n", original);</pre>
--	--

<pre>void main() {     char original[]="This is the original string";     char buf[80];     int p1[2], p2[2];</pre>	<pre>write(p1[1], original, strlen(original)+1); read(p2[0], buf, sizeof(buf)); printf("parent read '%s' from pipe 2\n", buf); } }</pre>
---	--

Creates a pipe, an I/O channel that a process can use to communicate with another process (in the same process or another process), or in some cases with itself. Data is written into one end of the pipe and read from the other. **fdinfo**[2] points to a memory area where pipe() can store two file descriptors. pipe() stores a file descriptor for the input end of the pipe in **fdinfo**[1], and stores a file descriptor for the output end of the pipe in **fdinfo**[0]. Thus, processes can read from **fdinfo**[0] and write to **fdinfo**[1]. Data written to **fdinfo**[1] is read from **fdinfo**[0] on a first-in-first-out (FIFO) basis.

==== \* ====

#### more

- more is a filter for paging through text one screenful at a time
- Use the more command to read a file:
  - **more filename**
- 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:
  - less filename
- Searching can be done by typing **/searchkey**, will highlights the searched words
- Example: **less test.c**
  - **/printf**

==== \* ====

#### cmp (Compare)

- compare two files byte by byte
- Syntax: **cmp -b file1 file2**

==== \* ====

#### head and tail

- Output the first/last part of files (by Default first/last 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**

==== \* ====

## 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 users 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**
- Permission Table:**
  - Octal Table:**

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 users

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 has r - - Permissions, means Other users can only read the file contents

### Examples

- add execution permission to group and other users
  - chmod go+x output.pdf
  - ls output.pdf -l
  - rwxrwxr-x 1 admincs admincs 1.9M Mar 28 08:45 output.pdf
- remove write permissions to group users
  - chmod g-w output.pdf
  - ls output.pdf -l
  - rwxr-xr-x 1 admincs admincs 1.9M Mar 28 08:45 output.pdf
- Keep only read permission to all users
  - chmod a=r output.pdf
  - ls output.pdf -l
  - r--r--r-- 1 admincs admincs 1.9M Mar 28 08:45 output.pdf
- Adding execution to all users
  - chmod a+x output.pdf



- ls output.pdf -l
- -r-xr-xr-x 1 admincs admincs 1.9M Mar 28 08:45 output.pdf
- giving only writing permission to other users (all settings may be not valid)
  - chmod o=w output.pdf
  - ls output.pdf -l
  - -r-xr-x-w- 1 admincs admincs 1.9M Mar 28 08:45 output.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 admincs 1.9M Mar 28 08:45 output.pdf

### Working with numbers in chmod

**Syntax: chmod chmodnumber 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

==== \* ====

### umask

- 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 upto user.
  - **umask**
  - Output: **0002**

**Example 1:**

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

**Example 2:**

### mkdir -m

- The 'mkdir -m' command can be used to set the mode.
- **Syntax:** mkdir -m <mode> <fileName>
- **Example:**
  - mkdir -m 777 new1
  - mkdir -m 000 new2
    - ls -ld new1 new2
    - drwxrwxrwx 2 admincs admincs 4096 Mar 28 14:37 new1
    - d- - - - - 2 admincs admincs 4096 Mar 28 14:37 new2

==== \* ====

**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 compress 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 use to compressing large file(s).</li> <li>GNU tar supports both archive 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> </ul> <p>-z: use gzip compress -j: use bzip2 compress</p>	<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>	<pre>gzip -d mydata.doc.gz gunzip mydata.doc.gz</pre>
<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 bzip2 command.</li> <li>File is restored to their original form using this command.</li> </ul>	<pre>bzip2 -d mydata.doc.bz2 gunzip mydata.doc.bz2</pre>
<b>unzip {zip file}</b>	Extract compressed files in a ZIP archive.	<pre>unzip file.zip unzip data.zip resume.doc</pre>
<b>tar -zxvf {.tgz-file}</b> <b>tar -jxvf {.tbz2-file}</b>	Untar or decompressed a file(s) that is created using tar compressing through gzip and bzip2 filter	<pre>tar -zxvf data.tgz tar -zxvf pics.tar.gz *.jpg tar -jxvf data.tbz2</pre>

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

Some time 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**

==== \* ====

## 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 sort key.
- 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.txt inputfile.txt
- **r** : sort the file contents in descending order
  - sort -r inputfile.txt
- **-n** : if the file contains numbers, based on number values, file contents can be sorted.
  - sort -n inputfile.txt
- **-k** : sort the file contents based on kth field values
  - sort -k 2 inputfile.txt // sorts file contents based on values in 2<sup>nd</sup> field.
- **-u** : will remove duplicate entries in file while sorting.

==== \* ====

#### uniq

- **uniq** filter removes duplicate lines from a sorted file. It is often seen in a pipe coupled with sort.
- **uniq -c filename**
- **removes the duplicates 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
- **Example 1: 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
- **Example 2: cut -d: -f1-4,6,8 FileName**
  - Same as above but the fields in file are separated by : (colon)

== \* ==

## 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**
- **Example 1:**

<b>head -5 a.txt</b>	<b>will display first 5 lines of a.txt</b>
<b>tail -5 a.txt</b>	<b>will display last 5 lines of a.txt</b>

==== \* =====

## grep

- A multi-purpose file search tool that uses Regular Expressions. (Global Regular Expression Print).
- **grep pattern [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:
  - ps ax
  - But search only word **swap** associated with the processes.
- **Output:**
  - **ps ax | grep swap**
  - 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/O stream:
- Important Options Used:
  - **wc -l** gives only the line count.
  - **wc -w** gives only the word count.

- **wc -c** gives only the byte count.
- **Syntax:**        **wc -lwc FileName**
- **Example:**
  - **wc -lwc temp2.txt**
- **Output:**
  - 3 6 18 temp2.txt
  - Says there 3 Lines, 6 Words and 18 Characters altogether in a file temp2.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.
- **Example 3: echo "acdeb123" | 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.
- **Example 1: fold -w 3 FileName**
  - Will wrap each line for every three characters
- **Example 2: fold -s -w 3**
  - 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.

==== \* ====

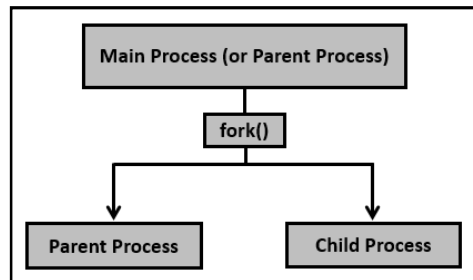
**Work on Practice Session**

**Work on Practice Session**

## 04 – Process creation and Management

### 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 fork() system call

#### Example 1:

```
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <stdio.h>
#include <sys/wait.h>
#include <stdlib.h>

int global;    // A global variable

int main(void)
{
    pid_t childPID;
    int local = 0;

    childPID = fork();

    if(childPID >= 0) // fork was successful
    {
        if(childPID == 0) // child process
        {
            local++;
            global++;
            printf("\n Child Process :: local = %d,
global %d\n", local, global);
        }
    }
}
```

```
else //Parent process
{
    local = 10;
    global = 20;

    printf("\n Parent process :: local = %d,
global %d\n", local, global);
}
else // fork failed
{
    printf("\n Fork failed, quitting!!!!!!\n");
    return 1;
}

return 0;
}
```

#### Output:

**Parent process :: local = 10, global 20**

**Child Process :: local = 1, global 1**

==== \* ====

### 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.

#### Example 1: (try this command in sudo bash)

<pre>\$ exec &gt; output.txt \$ cal \$ echo i am trying command exec \$ date \$ pwd \$ echo no commands display output on screen \$ echo enjoy exec command \$ exit exit</pre>	<p><b>Output:</b></p> <p><b>cat output.txt</b></p> <pre>March 2022 Su Mo Tu We Th Fr Sa       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 i am trying command exec</pre>
--	--

	Tue Mar 29 14:35:47 IST 2022 /home/adminics no commands display output on screen enjoy exec command
--	--

==== \* ====

### Example 2: ( try this command in sudo bash)

- **exec 3< 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 the **read** command with the **-u** option:
- **read -u 3 mydata**
  - 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 **mydata**. This would be useful if used as part of a **while** loop, for example.
- **exec 3< output.txt** // Assume contents of output.txt is **March 2022**
- **\$ read -u 3 oneline**
- **\$ echo \$oneline**
  - **March 2022**

==== \* ====

### Example 3: ( try this command in sudo bash)

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

==== \* ====

### Example 4: ( try this command in sudo bash)

- Closing read and write descriptors
- **exec 4>&-** // closing write descriptor
- **exec 3<&-** // closing read descriptor

==== \* ====

### Example 5:

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

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

==== \* ====

### 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 has been started with '**&**'.
- *Job\_spec* may be
  - **%n** : Refer to job number n.
  - **%% 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 500 seconds.
    - 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 500 seconds.
  - **^Z** // Terminate the job by pressing Ctrl + z
    - [1]+ Stopped sleep 500
  - **\$ jobs** // Display current jobs running and displayed one job stopped
    - [1]+ Stopped sleep 500
  - **\$ 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** sleep 500
- ==== \* ====

### fg: foreground

- **bg** is 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**
  - **\$ sleep 500**
  - **^Z**
    - [1]+ Stopped sleep 500
  - **\$ jobs**
    - [1]+ Stopped sleep 500
  - **\$ bg %1**
    - [1]+ sleep 500 &
  - **\$ jobs**
    - [1]+ **Running** sleep 500
  - **\$ fg %1** // Brings the background process (referred by its number) **to foreground**
  - **sleep 500** // waiting for the process to terminate.
- ==== \* ====

### 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.
- Suppose, user executing programs over SSH and if the connection drops, the session will be terminated, all the executed processes will stop, and user may face a huge accidental crisis.

- In such cases, running commands in the background can be very helpful to the user and this is where **nohup command** comes into the picture. **nohup (No Hang Up)** is a command in Linux systems that runs the process even after logging out from the shell/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.
- Once a job is started or executed using the **nohup** command, **stdin** will not be available to the user and **nohup.out** file is used as the default file for **stdout** and **stderr**.
- 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'**

```
yash9274@YASH-PC:~/GFG$ nohup bash -c 'cal && ls -l'
nohup: ignoring input and appending output to 'nohup.out'
yash9274@YASH-PC:~/GFG$ ls
nohup.out
yash9274@YASH-PC:~/GFG$ cat nohup.out
November 2020
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30

total 0
-rw----- 1 yash9274 yash9274 188 Nov 25 23:34 nohup.out
yash9274@YASH-PC:~/GFG$
```

- 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**

```
yash9274@YASH-PC:~/GFG$ nohup bash -c 'cal && ls -l' > output.txt
nohup: ignoring input and redirecting stderr to stdout
yash9274@YASH-PC:~/GFG$ ls
output.txt
yash9274@YASH-PC:~/GFG$ cat output.txt
November 2020
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30

total 0
-rw-r--r-- 1 yash9274 yash9274 188 Nov 25 23:42 output.txt
yash9274@YASH-PC:~/GFG$
```

==== \* ====

## Stopping a process

### 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**
  - 1) SIGHUP                      2) SIGINT                      3) SIGQUIT                      4) SIGILL                      5) SIGTRAP
  - 6) SIGABRT                      7) SIGBUS                      8) SIGFPE                      9) **SIGKILL**                      10) SIGUSR1
  - 11) SIGSEGV                      12) SIGUSR2                      13) SIGPIPE                      14) SIGALRM                      15) **SIGTERM**
- Signal number nine is SIGKILL. The default signal is 15, which is SIGTERM.
  - Issue a command such as **kill -9 20896**.

### pkill

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

==== \* ====

### 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** are meant for user-space.

## Check Nice Value of Linux Processes

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

- pid process-id, ppid parent process-id, ni niceness of process, comm command for that 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 args]
OR
$ nice -niceness-value [command args]      #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/*

```

**Change the Scheduling Priority of a Process**

- Linux allows dynamic priority-based scheduling. Therefore, if a program is already running, user can change its priority with the **renice command** in this form:
- \$ renice -n -12 -p 1055 //Mention process with id (-p)
- \$ renice -n -2 -u apache // Mention user process with name (-u)  
// -g for group process priority
- Output Verification:**
  - New priority for process 1055 is  $20 - 12 = 8$
  - New priority for process apache is  $20 - 2 = 18$
  - Observe new priority by issuing **top** or **htop** command.

==== \* ====

**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  
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	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 prefer for updates.

- To make it easier to monitor required processes, user can call top and pass the PID(s) using the **-p** option.

**Example 2: 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	0	12460	1628	912	S	0.0	0.0	0:00.00	nginx

==== \* ====

**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] [-L loglevel]**
- 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 cron job.
  - /etc/cron.deny
- **Note:** If neither of these files exists 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 scripted executed at a scheduled time mentioned in crontab file.

==== \* ====

### Simple Example:

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

#### Contents:

```
echo Welcome to Task Scheduler Demo
echo Try date Command
date
echo Try time Command
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/adminins/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, everyday.
  - **00 09-18 \* \* \* /home/folder/gfg-code.sh**
- Run /usr/local/bin/backup at 11:30 PM, every weekday.
  - **30 23 \* \* Mon, Tue, Wed, Thu, Fri /usr/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**

#### Example 2: Other Simple Example to Schedule user or System Tasks:

- **Note:** To schedule user task, the shell script path must be specified in place of /path/to/script
- Specify the interval at which user want to schedule the tasks using the cron job entries we specified earlier on.
- To reboot a system daily at 12:30 pm, use the syntax:
  - **30 12 \* \* \* /sbin/reboot**
- To schedule the reboot at 4:00 am use the syntax:
  - **4 \* \* \* /sbin/reboot**
- **Note:** The asterisk \* is used to match all records
- To run a script twice every day, for example, 4:00 am and 4:00 pm, use the syntax.
  - **4,16 \* \* \* /path/to/script**
- To schedule a cron job to run every Friday at 5:00 pm use the syntax:
  - **17 \* \* Fri /path/to/script** OR
  - **0 17 \* \* \* 5 /path/to/script**
- If user wish to run cron job every 30 minutes then use:
  - **\*/30 \* \* \* \* /path/to/script**
- To schedule cron to run after every 5 hours, run
  - **\*/5 \* \* \* \* /path/to/script**
- To run a script on selected days, for example, Wednesday and Friday at 6.00 pm execute:
  - **18 \* \* wed,fri /path/to/script**
- To schedule multiple tasks to use a single cron job, separate the tasks using a semicolon for example:
  - **\* \* \* \* \* /path/to/script1 ; /path/to/script2**

#### Example 3: Using special strings to save time on writing cron jobs

Some of the **cron** jobs can easily be configured using special strings that correspond to certain time intervals. For example,

- **@hourly** timestamp corresponds to **0 \* \* \* \***
  - It will execute a task in the first minute of every hour.
  - **@hourly /path/to/script**
- **@daily** timestamp is equivalent to **0 0 \* \* \***
  - It executes a task in the first minute of every day (midnight). It comes in handy when executing daily jobs.
  - **@daily /path/to/script**
- **@weekly** timestamp is the equivalent to **0 0 1 \* mon**

- It executes a cron job in the first minute of every week where a week whereby, a week starts on Monday.
- @weekly /path/to/script
- @monthly is similar to the entry 0 0 1 \* \*
- It carries out a task in the first minute of the first day of the month.
- @monthly /path/to/script
- @yearly corresponds to 0 0 1 1 \*
- It executes a task in the first minute of every year and is useful in sending New year greetings
- @monthly /path/to/script

==== \* ====

## 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**

```
File Edit View Search Terminal Help
cipers@cipers:~$ at -l
2          Wed Jun 24 03:57:00 2020 a cipers
cipers@cipers:~$ atq
2          Wed Jun 24 03:57:00 2020 a cipers
cipers@cipers:~$
```

- Schedule a job for the coming Monday at a time twenty minutes later than the current time:
  - **at Monday +20 minutes**
- Schedule a job to run at 1:45 Aug 12 2020:
  - **at 1:45 081220**
- Schedule a job to run at 3pm four days from now:
  - **at 3pm + 4 days**
- Schedule a job to shutdown the system at 4:30 today:
  - **echo "shutdown -h now" | at -m 4:30**
- Schedule a job to run five hour from now:
  - **at now +5 hours**

==== \* ====

```

cipers@cipers:~$ at sunday +20 minutes
warning: commands will be executed using /bin/sh
at> echo "Hello World"
at> <EOT>
job 11 at Sun Jun 28 05:34:00 2020
cipers@cipers:~$ at -l
11      Sun Jun 28 05:34:00 2020 a cipers
cipers@cipers:~$ echo "Hello World" | at 1pm + 2 days
warning: commands will be executed using /bin/sh
job 12 at Fri Jun 26 13:00:00 2020
cipers@cipers:~$ at -l
11      Sun Jun 28 05:34:00 2020 a cipers
12      Fri Jun 26 13:00:00 2020 a cipers
cipers@cipers:~$ mv p2.c p4.c | at 12:30 102120
warning: commands will be executed using /bin/sh
job 13 at Wed Oct 21 12:30:00 2020
cipers@cipers:~$ atq
11      Sun Jun 28 05:34:00 2020 a cipers
12      Fri Jun 26 13:00:00 2020 a cipers
13      Wed Oct 21 12:30:00 2020 a cipers
cipers@cipers:~$ gcc p1.c -o output | at now +1 hours
warning: commands will be executed using /bin/sh
gcc: error: p1.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.
job 14 at Wed Jun 24 06:16:00 2020
cipers@cipers:~$ gcc p4.c -o output | at now +1 hours
warning: commands will be executed using /bin/sh
job 15 at Wed Jun 24 06:17:00 2020
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

```

7. at -r or atrm command is used to deletes job , here used to deletes job 11.

**at -r 11                      OR                      atrm 11**

```

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.

==== \* ====

**Work on Practice Session**

**Work on Practice Session**

## 05 – Process Synchronization

### 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**

```
admincs-VirtualBox ~ # ps -ef | grep terminal
admincs  2400      1  2 19:06 ?        00:00:02 mate-terminal
root     2562    2418  0 19:08 pts/0    00:00:00 grep --colour=auto terminal
admincs-VirtualBox ~ # ps -T -p 2400
  PID  SPID  TTY      TIME  CMD
  2400  2400  ?        00:00:02 mate-terminal
  2400  2402  ?        00:00:00 gdbus
  2400  2403  ?        00:00:00 dconf worker
  2400  2406  ?        00:00:00 gmain
admincs-VirtualBox ~ #
```

- 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**



```
top - 19:24:16 up 22 min, 2 users, load average: 0.36, 0.17, 0.31
Threads: 312 total, 1 running, 311 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 6.6 sy, 0.0 ni, 91.1 id, 1.7 wa, 0.0 hi, 0.3 si, 0.0 st
KiB Mem: 1025964 total, 711340 used, 314624 free, 75752 buffers
KiB Swap: 1046524 total, 0 used, 1046524 free. 401252 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1572	root	20	0	103168	36612	11884	S	2.9	3.6	0:45.92	Xorg
2664	root	20	0	5564	1568	1084	R	2.0	0.2	0:03.39	top
2400	admins	20	0	292884	22356	14896	S	0.7	2.2	0:22.41	mate-terminal
4	root	20	0	0	0	0	S	0.3	0.0	0:03.78	kworker/0:0
6	root	20	0	0	0	0	S	0.3	0.0	0:02.46	kworker/u2:0
7	root	20	0	0	0	0	S	0.3	0.0	0:03.56	rcu_sched

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

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

```
top - 19:26:00 up 24 min, 2 users, load average: 0.30, 0.18, 0.30
Threads: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 8.4 sy, 0.0 ni, 91.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1025964 total, 711124 used, 314840 free, 75784 buffers
KiB Swap: 1046524 total, 0 used, 1046524 free. 401256 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1572	root	20	0	103168	36612	11884	S	4.0	3.6	0:48.50	Xorg

### Using htop

A more user-friendly way to view threads per process is via htop, an 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. sChoose "Display option" under "Setup" column, and toggle on "Tree view" and "Show custom thread names" options. Press F10 to exit the setup.

==== \* ====

**Work on Practice Session**



**Work on Practice Session**

## 06 – Memory Management

Commands to view memory consumption:

### 1. free command

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

- **Output:**

	<b>total</b>	<b>used</b>	<b>free</b>	<b>shared</b>	<b>buffers</b>	<b>cached</b>
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:      8167848 kB
MemFree:       1409696 kB
Buffers:       961452 kB
Cached:        2347236 kB
SwapCached:        0 kB
SwapTotal:     1998844 kB
SwapFree:      1998844 kB
...
```

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

### 3. vmstat

- **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 so on.

**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 is present here too.

- **Portion of output is**

```
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: 1998844 total, 0 used, 1998844 free, 2138148 cached
  PID USER   PR NI  VIRT 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   3:33.98 Xorg
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.

**6. 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: Not Provided
  Total Width: 64 bits
  Data Width: 64 bits
  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.

==== \* ====

**Work on Practice Session**

**Work on Practice Session**

## 07 – Shell Programming

**Script 1: Biggest of Three Numbers using if-elif-fi****Output**

```
# Find biggest of Three numbers
echo Enter Three Numbers
read a b c
if (( $a>=$b && $a>=$c ))
then
big=$a
elif(( $b>=$a && $b>=$c ))
then
big=$b
elif(( $c>=$a && $c>=$b ))
then
big=$c
fi
echo Biggest among $a $b $c is $big
===== *
```

**Script 2: Count the Number of Digits, Characters and Special Characters in a given String**

```
echo Enter the String
read str
# str="aa2222 b22b c1c d22d e333e f444fff"

let i=0
let digit=0
let character=0
let other=0
let length=${#str}
while [ $i -lt $length ]
do
    x="${str:$i:1}"
    if [[ $x =~ [0-9] ]]
    then
        let digit++
    elif [[ $x =~ [a-zA-Z] ]]
    then
        let character++
    else
        let other++
    fi
    let i++
done

echo Number of Digits $digit
echo Number of Characters $character
echo Number of Special Characters $other
echo Total Length of String $length
===== *
```

**Script 3: Check whether word present in a sentence or not.****| Output**

```
echo Enter Main String Seperated by space
read str
echo Enter Search String
read key
```

```
for word in $str
do
    if [ $word == $key ]
    then
        echo Element Present
        exit
    fi
done
echo Element NOT Present
```

```
==== * ====
```

**Script 4: Print the Numbers in Matrix format****| Output**

```
# Display Numbers in Matrix Format
# -e in echo is for Enabling Back Slash
# -n in echo is for ignoring newline
```

```
read -p "Enter Rows and Columns: " r c
```

```
for((i=1; i<=r; i++))
do
    for((j=1; j<= c; j++))
    do
        echo -n "$i "
    done
    echo
done
```

```
==== * ====
```

**Script 5: Display the words typed and total words typed.****| Output**

```
let count=0
echo "Please type 'finish' to quit this demo."

# Prompt the user with "Say something> " and
# read the value into variable "x"

while read -p "Say something> " x
do
    if [ "${x}" == "finish" ]
    then
        break
    fi
    if [ "${x}" == "abort" ]
    then
        exit
    fi
```

```

# Increment the counter
let count++

# if we say "silent", then we count the word,
# but do not display it:

if [ "${x}" == "silent" ]
then
    continue
fi

echo "You said: '${x}'."
echo "You have said ${count} things."
done
echo "All done. You have said ${count} things to me."
==== * ====

```

Script 6: Whether string contains unique words or not	Output
---	--------

```

#Check Uniqueness of words in a sentence
str=(aa bb cc dd ee ff)
#str=(aa bb cc bb ee ff)
let i=0
while [ "${str[$i]}" ]
do
    let j=$i+1
    while [ "${str[$j]}" ]
    do
        if [ "${str[$i]}" == "${str[$j]}" ]
        then
            echo Words in a String are NOT Unique
            exit
        fi
        let j++
    done
    let i++
done
echo Words in a String are Unique
==== * ====

```

### File Handling Scripts:

Script 7: Read the contents of a file and display the same on Screen (Demo cat command)
---

```

read -p "Enter file name : " filename
while read line
do
    echo $line
done < $filename

==== * ====

```



**Script 8: Copy the contents of a one file to another file (Demo cp command) | Output**

```
while read line
do
    echo $line >> output
done < $filename
echo Contents of $filename
cat output
```

==== \* ====

**Script 9: Count number of lines, words, and Characters in given file. | Output**

```
echo Enter the filename
read file
c=`cat $file | wc -c`
w=`cat $file | wc -w`
l=`grep -c "." $file`
echo Number of characters in $file is $c
echo Number of words in $file is $w
echo Number of lines in $file is $l
```

==== \* ====

**Script 10: Check the permissions of a given file and display the status | Output**

```
echo -e "Enter the name of the file : \c"
read filename

echo File Permission
`ls -l $filename`

if [ -r $filename ]
then
    echo "$filename is readable"
else
    echo "$filename is not readable"
fi

if [ -w $filename ]
then
    echo "$filename is writable"
else
    echo "$filename is not writable"
fi

if [ -x $filename ]
then
    echo "$filename is executable"
else
    echo "$filename is not executable"
fi

if [ -s $filename ]
then
```

```
        echo "$filename has size>0"  
    else  
        echo "$filename has size= 0"  
    fi
```

==== \* ====

### Work on Practice Session

## 08 – 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] [-L loglevel]**
- 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 cron job.
  - **/etc/cron.deny**
- **Note:** If neither of these files exists 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 scripted executed at a scheduled time mentioned in crontab file.

==== \* ====

### Simple Example 1:

**File Name:** task.sh                      path: **/home/adminics/(say for example)**

#### Contents:

```
echo Welcome to Task Scheduler Demo
echo Try date Command
date
echo Try time Command
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/adminics/task.sh > /home/adminics/output.txt**
- Save and Exit. And Wait till 4.13pm
- After that Check the output.txt by the command  
**cat /home/adminics/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.

==== \* ====

**Simple Example 2:** Monitoring Memory Consumption and remote server connectivity Checking

**File Name:** task2.sh                      **path:** /home/admincs/(say for example)

**Contents:**

```
echo    Memory Consumption output is in memoryoutput.txt
free > /home/admincs/memoryoutput.txt
echo    Checking Connectivity output is in requestreply.txt
ping -c 4 192.100.100.5 > /home/admincs/requestreply.txt
echo    End of Task Scheduler Demo
===== *
```

- Change the execution permission to task2.sh  
**chmod 764 task2.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/task2.sh**
- Save and Exit. And Wait till 4.13pm
- After that, Check the output by the command  
**cat /home/admincs/memoryoutput.txt**  
**cat /home/admincs/requestreply.txt**
- Alternatively /var/log/syslog can be verified about the execution of the Task.

===== \*

**Extra Example 1:**

- Run /home/folder/gfg-code.sh every hour, from 9:00 AM to 6:00 PM, everyday.  
○ **00 09-18 \* \* \* /home/folder/gfg-code.sh**
- Run /usr/local/bin/backup at 11:30 PM, every weekday.  
○ **30 23 \* \* Mon, Tue, Wed, Thu, Fri /usr/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**

===== \*

**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, see Week 4 Examples**

==== \*

**Work on Practice Session**

**Work on Practice Session**



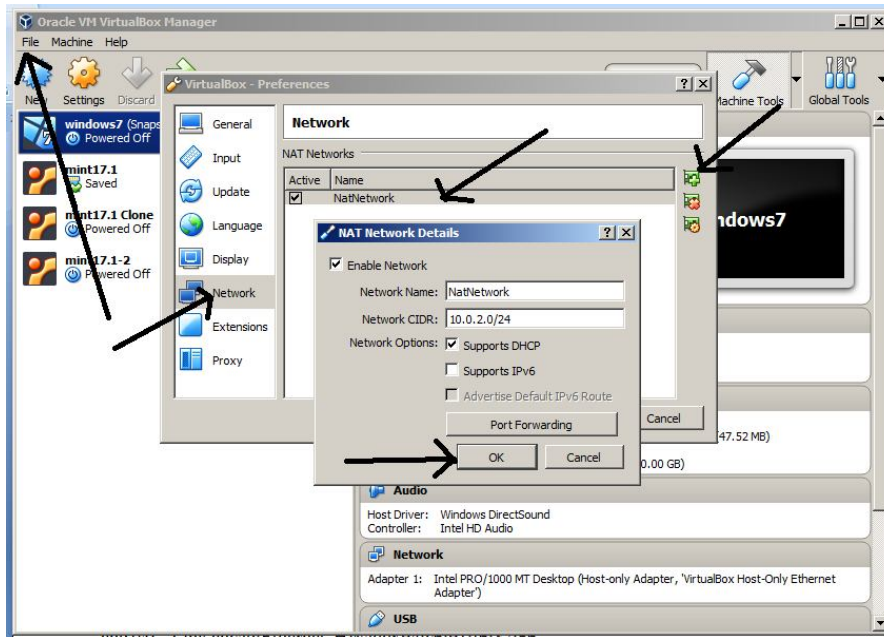
## 09 – Network Management

**Enable internet on Linux VM.**

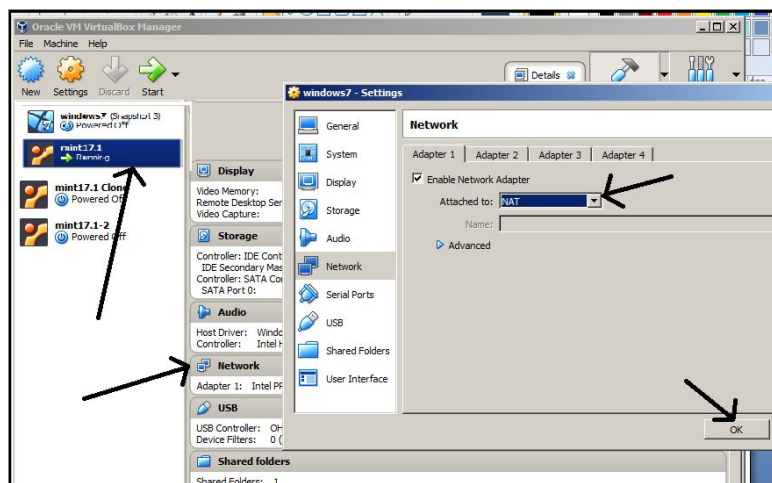
- Communicate between Guest and Host using ping command.

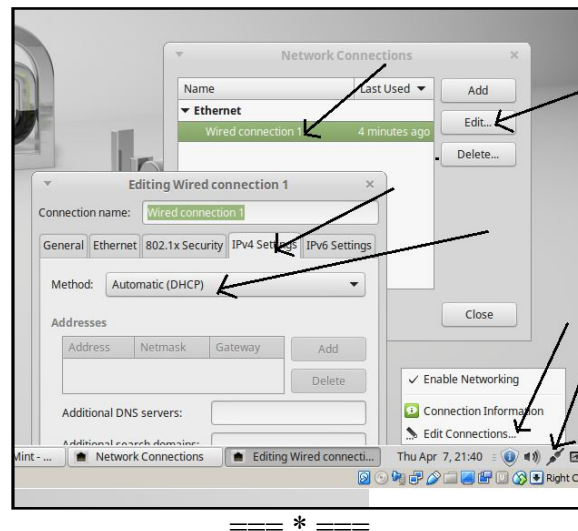
**Host Virtual Box Manager Settings:**

- 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 of vboxnet0
- 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..





==== \* ====

### 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.

### Example 1: **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 HWaddr 94: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:0 carrier:0
        collisions:0 txqueuelen: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:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:1729018 (1.7 MB) TX bytes:1729018 (1.7 MB)

```

### Example 2: **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	Flg
enp1s0	1500	0	62065	0	0	0	44236	0	0	0	BMRU
lo	65536	0	15869	0	0	0	15869	0	0	0	LRU

**Example 3: ifconfig interface up**

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

**Example 4: ifconfig interface down**

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

**Example 5: ifconfig interface arp OR 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, frequency etc.
- *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 Network ID.
    - **iwconfig [Interface] nwid on/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] freq 2.46000000**
  - **ap:** This option forces the card to **register to the Access Point** given by the address if it is possible.
    - **iwconfig [Interface] ap 00:60:1D:01:23:45**
  - **rate:** This option sets bitrate in b/s in supporting cards.
    - **iwconfig [Interface] rate 11M**
  - **retry:** This option sets the maximum number of times the MAC can retry transmission.
    - **iwconfig [Interface] retry 16**
  - **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.
- **ethtool enp1s0**  
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.**

#### Example 2: Change the Speed of Ethernet Device and set Auto Negotiation off

- **ethtool -s enp1s0 speed 100 autoneg off**
- **ethtool enp1s0**  
 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: Not reported
Advertised pause frame use: No
Advertised auto-negotiation: No
Speed: 100Mb/s
Duplex: Full
Port: MII
PHYAD: 0
Transceiver: internal
Auto-negotiation: off
Supports Wake-on: pumbg
Wake-on: g
Current message level: 0x00000033 (51)
                        drv probe ifdown ifup

```

**Example 3:** Use `ethtool -S` option to display the bytes transferred, received, errors, etc

- **ethtool -S enp1s0**  
NIC statistics:  
tx\_packets: **42257**  
rx\_packets: **57556**  
tx\_errors: **0**  
rx\_errors: **33**  
rx\_missed: **0**  
align\_errors: **0**  
tx\_single\_collisions: **0**  
tx\_multi\_collisions: **0**  
unicast: **42095**  
broadcast: **11512**  
multicast: **3949**  
tx\_aborted: **0**  
tx\_underrun: **0**

**Example 4: Make Changes Permanent after Reboot**

- If user has changed any ethernet device parameters using the `ethtool`, it will all **disappear** after the next **reboot**, unless user does the following.
- On ubuntu, user has to modify `/etc/network/interfaces` file and add all changes as shown below.
  - **nano /etc/network/interfaces**
  - **post-up ethtool -s eth2 speed 1000 duplex full autoneg off**
- The above line should be the last line of the file. This will change speed, duplex and autoneg of ethernet device permanently.

==== \* ====

#### 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 timestamps, 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 -i enp1s0**
  - 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 arptwatch: reaper: pid 4547, exit status 1
Apr  4 11:38:23 admincs-To-be-filled-by-O-E-M arptwatch: 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 arptwatch: 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 arptwatch: execl: /usr/lib/sendmail: No such file or directory
Apr  4 11:38:23 admincs-To-be-filled-by-O-E-M arptwatch: reaper: pid 4587, exit status 1
Apr  4 11:38:23 admincs-To-be-filled-by-O-E-M arptwatch: execl: /usr/lib/sendmail: No such file or directory
Apr  4 11:38:23 admincs-To-be-filled-by-O-E-M arptwatch: reaper: pid 4588, exit status 1
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 arptwatch: 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

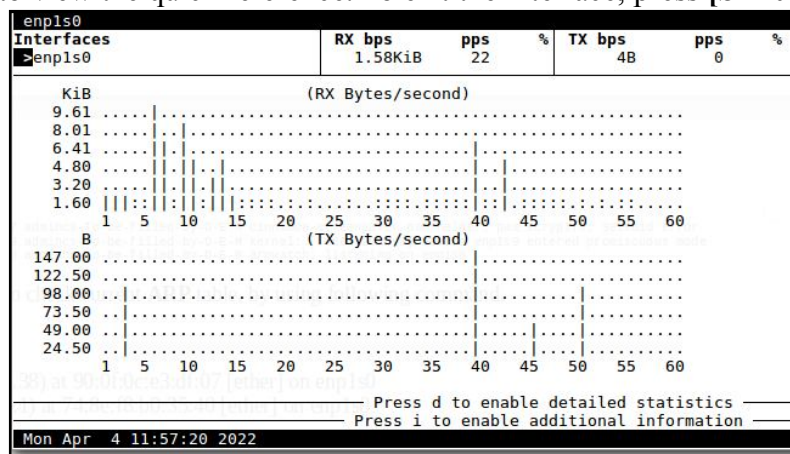
==== \* ====

**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.



==== \* ====

**Example 2: bmon -r 5 -p enp1s0 -o ascii**

- The output can be viewed in ascii mode also, the output could be collected at regular interval also.

- Output**

```

Interfaces      RX bps    pps    %    TX bps    pps    %
enp1s0          0         0      0      0         0
Interfaces      RX bps    pps    %    TX bps    pps    %
enp1s0          5.53KiB   16      0      0         0
Interfaces      RX bps    pps    %    TX bps    pps    %
enp1s0          1.85KiB   10      0      0         0
Interfaces      RX bps    pps    %    TX bps    pps    %
enp1s0          1.05KiB   10      0      0         0
Interfaces      RX bps    pps    %    TX bps    pps    %
enp1s0          4.55KiB   16      0    41B         0
===== * =====

```

**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:**
  - Install **openssh-client** and **openssh-server** through software manager
  - assign valid ip address to two machines, for connection
  - Assure, both the system have user account with password.
- Settings:**

Machine 1:	Machine 2:
<b>name:</b> admincsm1	admincsm2
<b>login:</b> admincsl1	admincsl2
<b>192.100.100.1</b>	<b>192.100.100.2</b>

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

- ssh -l admincsl2 admincsm2@192.100.100.2 21**
  - Connect to the computer with name admincsm2 and ipaddress 192.100.100.2 to the port number 21 through login admincsl2
  - It will ask for admin login password.
  - The successful connection lead to remote machine terminal. User can issue any Linux commands to handle remote machine locally.

**Example 2: create files at machine2 using ssh terminal (After Example 1, ssh already connected)**

```

cd Desktop
touch a.txt
cat > a.txt
      I am in machine 2 at filename a.txt
^Z
ls           // to view files created or not
cat a.txt   // to view file contents

```

**Example 3: copy the file a.txt at machine2 to machine 1 (local) desktop using a Command scp** (scp: **secure copy** command to copy files remotely).

- **scp a.txt admincs1@192.100.100.1:/home/admincs/Desktop/d.txt**
  - This will copy a.txt from machine 1 to machine 2's Desktop with file name d.txt

**Example 4: copy file d.txt at machine1 (local) to machine 2 (remote) Desktop**

- **scp admincs@192.100.100.1:/home/admincs/Desktop/d.txt b.txt**
  - This will copy d.txt from machine 1 to Desktop of Machine 2 with file name b.txt

**Note:** In all the examples, the files created, transferred, and its contents could be seen by exploring the desktop contents in a normal way also.

==== \* ====

### telnet:

- 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 **ssh** over telnet because ssh transfers data in encrypted form.
- **Prerequisite:**
  - Install **telnet (client)** and **telnetd (server)** through software manager
  - assign valid ip address to two machines, for connection
  - Assure, both the system have user account with password.
- **Settings:**

Machine 1:	Machine 2:
<b>name:</b> admincs1	admincs2
<b>login:</b> admincs1	admincs2
<b>192.100.100.1</b>	<b>192.100.100.2</b>

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

- **telnet -l admincs2 192.100.100.2 23**
  - Connect to the computer with ipaddress 192.100.100.2 to the port number 23 through login admincs2
  - It will ask for admin login password.
  - The successful connection lead to remote machine terminal. User can issue any Linux commands handle remote machine locally.

**Example 2: create files at machine2 using ssh terminal (ssh already connected)**

```
cd Desktop
touch a.txt
cat > a.txt
    I am in machine 2 at filename a.txt
^Z
ls           // to view files created or not
cat a.txt   // to view file contents
```



**Example 3: copy the file a.txt at machine2 to machine 1 (local) desktop using a Command scp** (scp: secure copy command to copy files remotely).

- **scp a.txt admincs1@192.100.100.1:/home/admincs/Desktop/d.txt**
  - This will copy a.txt from machine 1 to machine 2's Desktop with file name d.txt

**Example 4: copy file d.txt at machine1 (local) to machine 2 (remote) Desktop**

- **scp admincs@192.100.100.1:/home/admincs/Desktop/d.txt b.txt**
  - This will copy d.txt from machine 1 to Desktop of Machine 2 with file name b.txt

**Note:** In all the examples, the files created, transferred, and its contents could be seen by exploring the desktop contents in a normal way also.

==== \* ====

#### **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 restart vsftpd**

**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.

**Example 2: Download the file from ftp server (Connection done in Example 1)**

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

**Example 3: Upload the file from local machine to ftp server (Connection done in Example 1)**

- **put /home/admincs/index.html test.html**
- will upload the file **index.html** from current (local **/home/admincs**) user to remote ftp server with name **test.html**

**Example 4:** User can issue commands like **mkdir**, **cd**, **rmdir** etc linux basic commands to control the remote server machine. (for more commands go to **? Command** and **man ftp** 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 by Libcurl.
- 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 restart vsftpd**

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

- **-u** option: indicates username and separated with password for that account (user authenticated FTP 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)

**Example 2: Download the file from ftp Server**

- **-o** option: will download file from server with different name, here **ftp://192.100.100.2/a.txt** downloaded with the **test.txt** in current machine.
- **curl -# -u admincs:admincs -o test.txt ftp:// 192.100.100.2/a.txt**

**Example 3: Upload a file to ftp Server**

- **-T** option: used to upload File from current directory to ftp Server
- **curl -# -u admincs:admincs -T test.c ftp:// 192.100.100.2/Desktop/test.c**  
==== \* ====

**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 a webpage:
    - **wget https://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 wget command.
    - **wget http://www.example.com/filename.txt -o /home/admincs/temp.html**

- To download from ftp server:
  - options: `--ftp-user=username` and `--ftp-password=passwordstring` will be used to authenticate ftp 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:**
  - /\* using rcp command to send a file from local host to remote host \*/
    - **rcp /mydirectory/a.txt admin:remotedir/kt.txt**
  - /\* the example above is to send a file not to receive a file from remote host \*/
  - 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 **remotedir** (as path **remotedir/a.txt** is given).

**Working with rcp:**

Assume, Source machine and Destination machine have same username, so identify different machine user names with **ipaddress**.

**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**

**Example 2: Copy entire folder from local machine to remote machine.**

- **rcp -r /home/localuser/Documents admincs@192.100.100.2:/home/admincs/tempDocument**
- This will copy local machine entire Documents folder to remote machine (**192.100.100.2**, username **admincs**, destination directory **/home/admincs/**) with new name **tempDocument**

==== \* =====

**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 ipaddress.

**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**

**Example 2: Copy entire folder (recursively -r) from local machine to remote machine.**

- **scp -r /home/localuser/Documents admincs@192.100.100.2:/home/admincs/tempDocument**
- This will copy local machine entire Documents folder to remote machine (192.100.100.2, username **admincs**, destination directory **/home/admincs/**) with new name **tempDocument**

==== \* ====

**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**

**Example 2: synchronize Folder (copy folder locally)**

- **rsync -a /home/admincs/Documents /home/admincs/Desktop/tempDocuments**
- This will copy entire folder **local/admincs/Documents** to local machine desktop with new name **tempDocuments**
  - **-a: archive mode.**

**Example 3: Copy file from local machine to remoter machine.**

- **rsync 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**

**Example 4: Copy entire folder (recursively) from local machine to remote machine.**

- **rsync -a /home/localuser/Documents admincs@192.100.100.2:/home/admincs/tempDocument**

- This will copy local machine entire Documents folder to remote machine (192.100.100.2, username **admins**, destination directory **/home/admins/**) with new name **tempDocument**

==== \* ====

#### 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):
  - **sftp admins@172.16.20.116**

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

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

#### Example 2: Upload the file from server

- **put /home/admins/index.html test.html**
- will upload the file **index.html** from current (local **/home/admins/**) user to remote ftp server with name **test.html**

**Example 3:** User can issue commands like **mkdir**, **cd**, **rmdir** etc linux basic commands to control the remote server machine. (for more commands go to ? Command and **man sftp** command)

==== \* ====

#### 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:**
  - **-a:** all listening and non-listening ports, **-t** tcp ports
  - **-u** udp ports, **-l** listening ports
  - **-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
```

**Example 2: netstat -s // To list the statistics for all ports.****Ip:**

102865 total packets received  
 14 with invalid addresses  
 0 forwarded  
 19 with unknown protocol  
 0 incoming packets discarded  
 97399 incoming packets delivered  
 70224 requests sent out  
 24 outgoing packets dropped  
 2 dropped because of missing route

**Icmp:**

77 ICMP messages received  
 0 input ICMP message failed.  
 ICMP input histogram:  
     destination unreachable: 77  
 258 ICMP messages sent  
 0 ICMP messages failed  
 ICMP output histogram:  
     destination unreachable: 258

**Example 3: netstat -r // Kernel Routing Information****Kernel IP routing table**

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
default	172.16.20.1	0.0.0.0	UG	0	0	0	enp1s0
link-local	*	255.255.0.0	U	0	0	0	enp1s0
172.16.2.2	172.16.20.1	255.255.255.255	UGH	0	0	0	np1s0
172.16.20.0	*	255.255.255.0	U	0	0	0	enp1s0

==== \* =====

**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(Internet Control Message Protocol)** 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:
  - -c : Number of packets to be transfered
  - -w : deadline, with in this seconds, continously send the packets ICMP Packets.
  - -s : Packe size

**Example 1: ping -c 5 -s 100 172.16.20.116**

- 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).

**Example 2:           ping -w 5 172.16.20.115**

- Will continuously send the ICMP packets within 5 seconds.

==== \* =====

**traceroute:**

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

**Example 1:   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 ms  136.232.204.173.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.

**Example 2:   traceroute 172.16.2.10****//local server**

traceroute to 172.16.2.10 (172.16.2.10), 30 hops max, 60 byte packets

```

1  172.16.20.1 (172.16.20.1) 0.624 ms  1.013 ms  1.428 ms
2  172.16.2.10 (172.16.2.10) 0.212 ms !X 0.206 ms !X 0.186 ms !X

```

**Example 2: traceroute 127.0.0.1****//local machine**

traceroute to 127.0.0.1 (127.0.0.1), 30 hops max, 60 byte packets

```

1  localhost (127.0.0.1) 0.034 ms  0.010 ms  0.010 ms

```

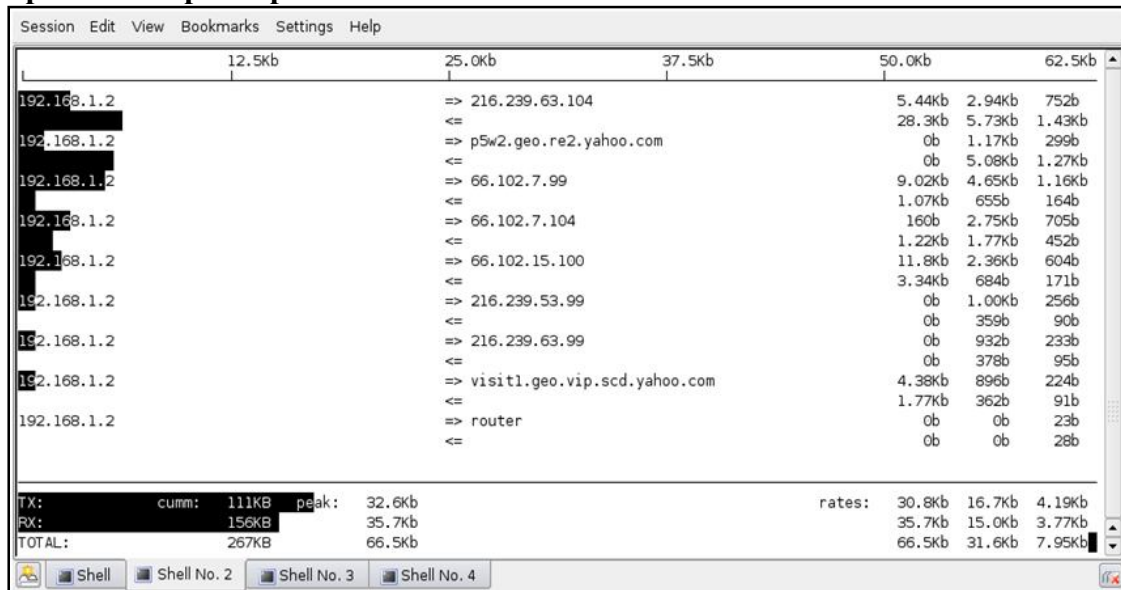
**Options used:**

- 4    Use ip version 4 i.e. use Ipv4
- 6    Use ip version 6 i.e. use Ipv6
- F    Do not fragment packet.

==== \* =====

**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 an ssh based session.
- iftop 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****Example 2: iptop -f icmp -i enp1s0**

- filters only icmp packets type ping command in some other terminal to see the effects.

**Example 3: iftop -F 192.168.1.0/24**

- Display or analyses packet flowing in and out of the 192.168.1.0/24 network:

==== \* =====

**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: 1.61 kBit/s                       Curr: 0.00 Bit/s
Avg: 1.82 kBit/s                         Avg: 456.00 Bit/s
Min: 0.00 Bit/s                         Min: 0.00 Bit/s
Max: 34.95 kBit/s                       Max: 15.07 kBit/s
Ttl: 76.13 MByte                         Ttl: 8.84 MByte
```

Device lo [127.0.0.1] (2/2):

```
=====
Incoming:                               Outgoing:
```



Curr: 1.30 kBit/s	Curr: 1.30 kBit/s
Avg: 808.00 Bit/s	Avg: 808.00 Bit/s
Min: 0.00 Bit/s	Min: 0.00 Bit/s
Max: 20.16 kBit/s	Max: 20.16 kBit/s
Ttl: 1.53 MByte	Ttl: 1.53 MByte

=====

- **Options used:**

- 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.
- 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).
- With ss, user get very detailed information about how Linux machine is communicating with other machines, networks, and services; details about network connections, networking protocol statistics, and Linux socket connections.
- **Some options used:**
  - -t Display TCP sockets,                      -u Display UDP sockets,
  - -a All Sockets,                                -l Only Listening Sockets,
  - -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:**

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

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

**tcpdump host 172.16.20.116                      // Try ping from 172.16.20.116**

**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
```

**Example 2:**

**tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'**

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp1s0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
```

**Example 3:                      tcpdump -c 5 -i enp1s0**

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp1s0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:46:05.519973 IP 172.16.20.63.51082 > 239.255.255.250.1900: UDP, length 173
13:46:05.521176 IP 172.16.20.107.42602 > dns.google.domain: 39493+ PTR?
250.255.255.239.in-addr.arpa. (46)
13:46:05.521196 IP 172.16.20.107.42602 > dns.google.domain: 39493+ PTR?
250.255.255.239.in-addr.arpa. (46)
13:46:05.521206 IP 172.16.20.107.42602 > 172.16.1.100.domain: 39493+ PTR?
250.255.255.239.in-addr.arpa. (46)
13:46:05.521400 IP 172.16.1.100.domain > 172.16.20.107.42602: 39493 NXDomain* 0/0/0 (46)
5 packets captured
17 packets received by filter
7 packets dropped by kernel
```

**Example 4:    tcpdump -c 4 'icmp[icmptype] == icmp-echo'**

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp1s0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:50:28.784739 IP 172.16.20.116 > 172.16.20.107: ICMP echo request, id 29434, seq 1, length 64
13:50:29.798343 IP 172.16.20.116 > 172.16.20.107: ICMP echo request, id 29434, seq 2, length 64
13:50:30.822415 IP 172.16.20.116 > 172.16.20.107: ICMP echo request, id 29434, seq 3, length 64
13:50:31.846476 IP 172.16.20.116 > 172.16.20.107: ICMP echo request, id 29434, seq 4, length 64
4 packets captured
4 packets received by filter
0 packets dropped by kernel
```

**Example 5: tcpdump -c 1 -XX 'icmp[icmptype] == icmp-echoreply'**

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp1s0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

```

13:53:58.768288 IP 172.16.20.107 > 172.16.20.116: ICMP echo reply, id 30066, seq 1, length 64
0x0000: f44d 30b7 dfcf 94c6 91f6 3756 0800 4500 .M0.....7V..E.
0x0010: 0054 89b1 0000 4001 6ff8 ac10 146b ac10 .T....@.o....k..
0x0020: 1474 0000 f54c 7572 0001 9efc 4b62 0000 .t...Lur...Kb..
0x0030: 0000 e80d 0400 0000 0000 1011 1213 1415 .....
0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!""$%
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 &'()*+,-./012345
0x0060: 3637                                     67

```

**1 packet captured**

1 packet received by filter

0 packets dropped by kernel

**Example 6:** To save captured packets into a file and to read captured packets from a file

```
sudo tcpdump -w captured_packets.pcap -i enp1s0
```

```
sudo tcpdump -r captured_packets.pcap
```

```
==== * ====
```

**dstat**

- **dstat** is a tool that is used to retrieve information or statistics from 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 diskset or network bandwidth for a group of interfaces, ie. 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:**
  - **-c** enable cpu stats (system, user, idle, wait, hardware interrupt, software interrupt)
  - **-d, --disk** enable disk stats (read, write)      **-g, --page** enable page stats (page in, page out)
  - **-i, --int** enable interrupt stats      **-m, --mem** enable memory stats (used, buffers, cache, free)
  - **-n, --net** enable network stats (receive, send)      **--nocolor**

- Output:

<b>dstat -n</b>	<b>dstat -c</b>	0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 &'()*+,- 0x0060: 3637 67
-net/total-	----total-cpu-usage----	1 packet captured
recv send	usr sys idl wai hiq siq	1 packet received by filter
0 0	4 1 95 0 0 0	0 packets dropped by kernel
216B 0	1 0 98 0 0 0	admins-To-be-filled-by-0-E-M ~ # dstat
336B 84B	2 0 98 0 0 0	You did not select any stats, using -cdngy by default.
632B 0	2 1 97 0 0 0^C	----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---sys
6297B 54B		usr sys idl wai hiq siq read writ recv send in out int
584B 54B		4 1 95 0 0 0 113k 103k 0 0 75B 1199B 410
697B 0^C		6 5 88 0 0 0 0 0 275B 0 0 0 467
		2 0 98 0 0 0 0 0 0 0 0 0 179
		2 1 98 0 0 0 0 0 259B 0 0 0 164
		2 0 98 0 0 0 0 0 240B 162B 0 0 222
		2 0 98 0 0 0 0 0 276B 0 0 0 235
		4 0 96 1 0 0 0 40k 582B 294B 0 0 278
		3 1 97 0 0 0 0 96k 6218B 384B 0 0 250
		2 0 98 0 0 0 0 0 216B 0 0 0 199
		2 0 98 0 0 0 0 0 579B 441B 0 0 259
		11 1 88 0 0 0 0 4096B 5879B 147B 0 0 684
		5 1 94 0 0 0 0 0 726B 660B 0 0 329
		6 1 93 0 0 0 0 0 0 0 0 0 507
		4 2 94 1 0 0 0 40k 3932B 2443B 0 0 318
		14 2 84 0 0 1 0 0 421B 54B 0 0 1035
		15 1 82 0 0 1 0 0 215B 0 0 0 1261
		4 1 95 0 0 0 0 660k 386B 171B 0 0 357
		15 3 82 0 0 1 0 0 75k 3585B 0 0 1868

==== \*

sendmail:

mailstats:

**Work on Practice Session**

**Work on Practice Session**

## 10 – User Authentication

### Work on user accounts:

#### 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:
  - It edits **/etc/passwd**, **/etc/shadow**, **/etc/group** and **/etc/gshadow** files for the newly created user accounts.
  - Creates and populates a home directory for the new user.
  - Sets permissions and ownerships to the home directory.
- **Syntax:**
  - **useradd -d /home/newusername -p passwordstring newusername**
  - This creates user with name newusername
    - -d option will create directory for new user at /home directory
    - -p option allows specifying password while creating user with the value given in place of passwordstring
  - User can change the password later by typing command
    - **passwd newusername**
    - This asks the user to enter the new password and confirm password  
==== \* ====

#### passwd

- The passwd command **changes passwords for user accounts**.
- 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:
  - **-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 passwordless.
  - **-e, --expire** Immediately expire an account's password. This in effect can force 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 neuser**
- **Important Options** used along with userdel:
  - **-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.

- **-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.
- **-Z :** This option remove 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/newfolder existinguser**
- To change the group of a user
  - **usermod -g newgroupname existinguser**
- To change user login name
  - **usermod -l usernewname useroldname**
- To lock a user
  - **usermod -L existinguser** // -U for unlocking user
- To set an unencrypted password for the user
  - **usermod -p newpassword existinguser**

==== \* ====

**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 user group.
- **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 newuser
  - **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] groupname**
  - Change the group *name to new name*.
  - **groupmod -n groupnewname groupoldname**

==== \* ====



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

**Example:**

- add the user to group:
  - `gpasswd -a existinguser existinggroup`
- Deleting the created user from group geeks.
  - `gpasswd -d existinguser existinggroup`  
==== \* ====

**groupdel**

- *groupdel* command is used to delete a 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 -f existinggroup`
- **Option used: -f -force:** It used to delete a group even if it is the primary group of a user  
==== \* =====

**LDAP server and client configuration:**

- Lightweight Directory Access Protocol (LDAP) is a standard protocol designed to manage and access hierarchical directory information over a 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 for LDAP.
  - 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 allow ldap**

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

- **ldapwhoami -H ldap:// -x**
- **Output**
  - **anonymous**
  - 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 install phpldapadmin**
  - 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 need 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 phpLDAPadmin Web 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  
//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:

**username: 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 display 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.

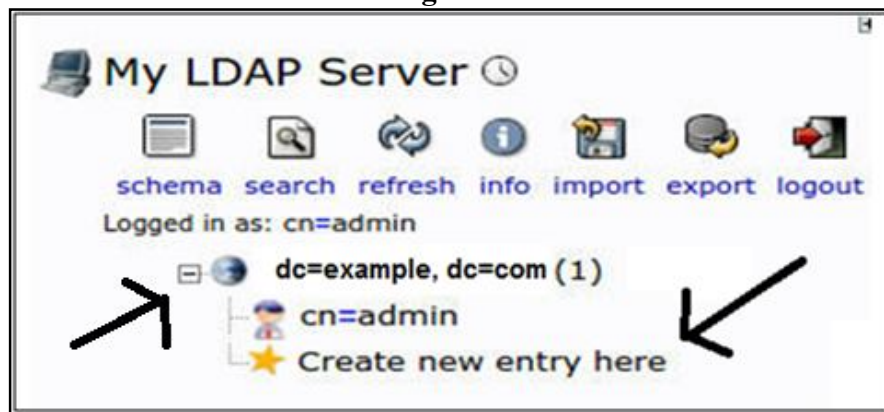
### Creating Group and Users in using LDAP:

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

### Creating Organizational Units

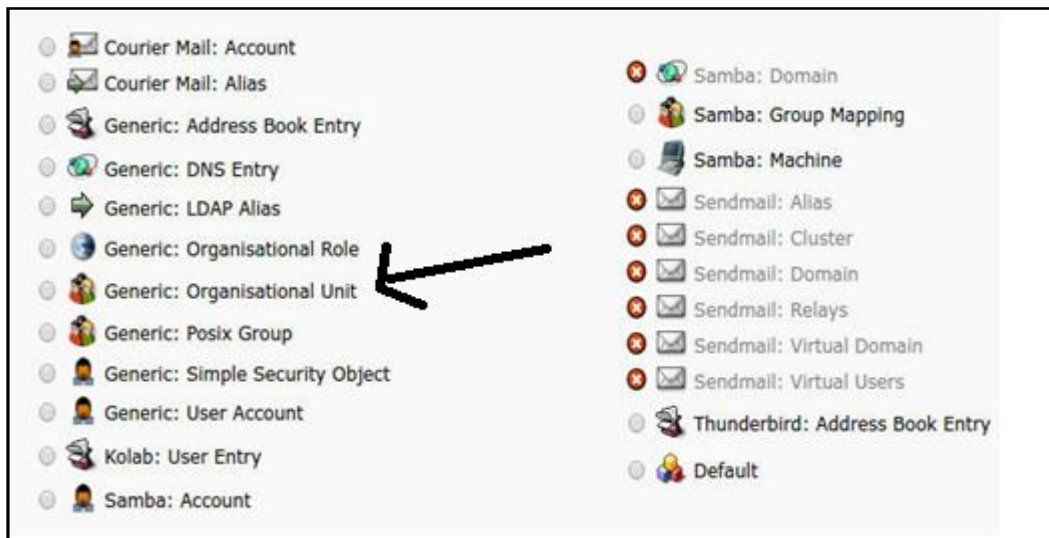
LDAP breaks everything into very specific pieces, and here focus is on two pieces: **people and groups**. Because we're creating fairly generic Organizational Units (OUs), we'll use the Generic Organizational Unit Template. To get there, log into phpLDAPadmin, click to expand server listing and then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

- Server: My LDAP Server
- Container: dc=monkeypantz,dc=net
- Do you want to create this entry?

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"/>

Buttons: Commit, Cancel

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.

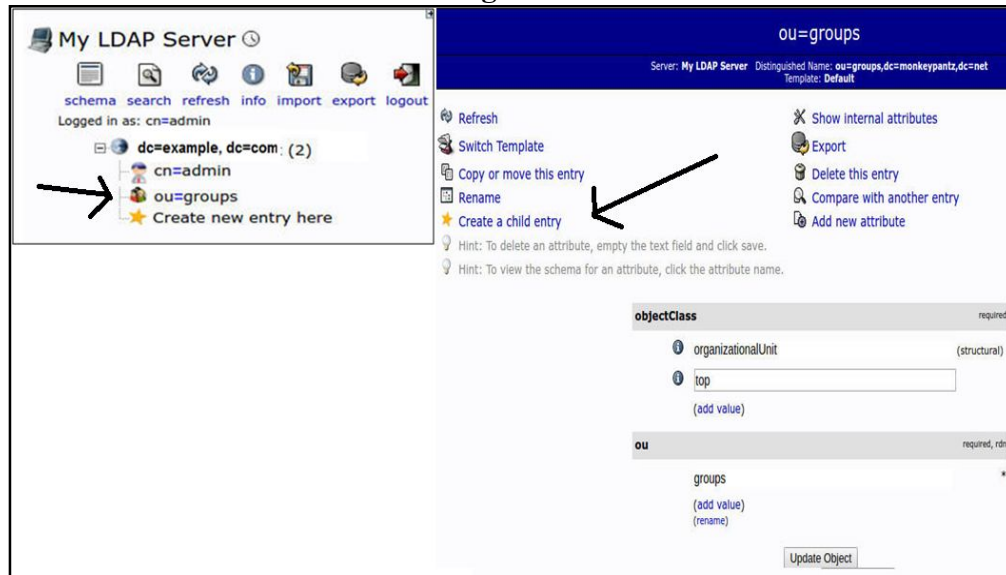
Creating groups

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” (Be careful, every time creating **child entry**, make sure First clicked on **groups OU** again, on the **left pane**).

Figure E



### 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 (Figure F).
5. Click **Create Object**.
6. Click **Commit**.
7. Repeat this process until necessary number of users added.

Figure F

The screenshot shows the 'Generic: User Account' form. It has several fields with labels and hints: 'Common Name' (alias, required, rdn), 'First name' (alias), 'GID Number' (alias, required, hint), 'Home directory' (alias, required), and 'Last name' (alias, required). The 'Common Name' field is highlighted in yellow. There are also 'add value' and 'rename' buttons for some fields, and an asterisk (\*) indicating required fields.

### 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's how.

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**

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.

==== \* =====

**Work on Practice Session**



**Work on Practice Session**

## 11 – 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     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.
- 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   0 12460 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.

#### Example 1: 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% /
```

**Example 2:** specify particular file, then it will show mount information of that particular file.  
For example: **df /home/admincs/test/test.cpp**

**Output:**

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda10	78873504	67528220	7315640	91%	/home

**Example 3: df -h /home/admincs**

- Use -h option to display size in power of 1024

**Output:**

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda10	76G	65G	7.0G	91%	/home

**Example 4: df -H /home/admincs**

- Use -H option to display sizes in power of 1000

**Output:**

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda10	81G	70G	7.5G	91%	/home

Observe the size section of two command with -h and -H option for difference.

**Example 5: df -T /home/admincs**

- Use -T option to display file type.

**Output:**

Filesystem	Type	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda10	ext4	78873504	67528128	7315732		
		=====	*	=====		

**dmesg**

===== \*

**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 sysstat package. If user doesn't have it, user need to install first. (apt-get install sysstat)
- **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 show 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/O request.
  - **%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/O request.
- The **second section** of the output contains device utilization report:
  - **Device :** The device/partition name is listed in /dev directory.
  - **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 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.

==== \* ====

### 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
cpu family    : 6
model         : 45
model name    : Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz
stepping      : 6
microcode     : 1561
cpu MHz       : 600.000
cache size    : 20480 KB
```

==== \* ====

### 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:      8167848 kB
MemFree:       1409696 kB
Buffers:       961452 kB
Cached:        2347236 kB
SwapCached:      0 kB
SwapTotal:     1998844 kB
SwapFree:      1998844 kB
...
```
- Check the values of `MemTotal`, `MemFree`, `Buffers`, `Cached`, `SwapTotal`, `SwapFree`. They indicate same values of memory usage as the `free` command.

==== \* ====

### 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 through 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 mechanisms 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 anacron) starts a cron job, it logs the information about the cron job in this file

==== \* ====

#### 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 a 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 -h 0** **// time Specification 0**
  - **shutdown now**
    - Another common command for shutting down Linux immediately:

- **Command for restarting Linux immediately**
    - **shutdown -r 0** // time Specification 0
    - **shutdown -r now**
  - **Command for Shutting Down/Restart Linux after 20 minutes**
    - shutdown -h 20
    - shutdown +20
    - shutdown -r 20
    - shutdown -r +20
    - shutdown -h 17:30 // Shutting down at 5.30pm
    - shutdown -r 17:30 // 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 command in 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, -wtm-only It will not call shutdown or the reboot system call but writes the shutdown record to /var/log/wtmp file.
    - -p, -poweroff 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 runlevel has a certain number of services stopped or started. Conventionally seven runlevel 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 runlevel 3 + display manager(X)
6	Reboot	Reboots the system

- By default most of the LINUX based system boots to runlevel 3 or runlevel 5.
- Runlevels 2 and 4 are used for user defined runlevels
- runlevel 0 and 6 are used for halting and rebooting the system.

==== \* ====

## System update & repositories

### Update the Repositories

- **sudo apt-get update**
- 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-get dist-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 Yellowdog 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 Redhat 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.

=== \* ===

**Work on Practice Session**

**Work on Practice Session**

## 12 – 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 **name servers**.
- 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-get update`
- **`sudo apt-get install bind9`**
- `sudo apt-get update`
- **`sudo apt-get install dnstools`**
- `sudo apt-get update`

### 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.

### Overview

The DNS configuration files are stored in the `/etc/bind` directory. The primary configuration file is `/etc/bind/named.conf`, which in the layout provided by the package just includes these files.

- `/etc/bind/named.conf.options`: global DNS options
- `/etc/bind/named.conf.local`: for zones
- `/etc/bind/named.conf.default-zones`: default zones such as **localhost**, its reverse, and the root hints

The root **nameservers** used to be described in the file `/etc/bind/db.root`. This is now referenced in the `named.conf.default-zones` configuration file above.

- It is possible to configure the **same** server to be a **caching name server**, **primary**, and **secondary**: it all depends on the zones it is serving.
- A server can be the Start of Authority (SOA) for one zone, while providing secondary service for another 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:

nano /etc/bind/named.conf.options	
Before	After
// forwarders { // 1.2.3.4; // 5.6.7.8; // };	forwarders { 192.168.1.10; };

**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 IP Address** and **root.localhost** to a **valid email address, but with a . instead of the usual @ symbol**, again **leaving the . at the end**. Change the comment to indicate the domain that this file is for.

**Create an A record** for the base domain, example.com. **Also, create an A record for ns.example.com**, the name server in this example:

nano /etc/bind/db.example.com	
Before	After
; ; BIND data file for loopback interface ; \$TTL 604800 @ IN SOA localhost. root.localhost. ( 2 ; Serial 604800 ; Refresh	; ; BIND data file for example.com ; \$TTL 604800 @ IN SOA example.com. root.example.com. ( 2 ; Serial

86400 ; Retry	604800 ; Refresh
2419200 ; Expire	86400 ; Retry
604800 ) ; Negative Cache TTL	2419200 ; Expire
@ IN NS localhost.com.	604800 ) ; Negative Cache TTL
@ IN A 127.0.0.1	@ IN NS <b>ns.example.com.</b>
@ IN AAAA ::1	@ IN A <b>192.168.1.10</b>
	@ IN AAAA ::1
	<b>ns IN A 192.168.1.10</b>

**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:

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

zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
};
zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
};
```

**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:

nano /etc/bind/db.192	
Before	After
;	;
; BIND reverse data file for loopback interface	; BIND reverse data file for local
;	192.168.1.DD net
\$TTL 604800	;
@ IN SOA localhost. root.localhost. (	\$TTL 604800
2 ; Serial	@ IN SOA <b>ns.example.com.</b>
604800 ; Refresh	<b>root.example.com. (</b>

86400 ; Retry	2 ; Serial
2419200 ; Expire	604800 ; Refresh
604800 ) ; Negative Cache TTL	86400 ; Retry
;	2419200 ; Expire
@ IN NS localhost.	604800 ) ; Negative Cache TTL
1.0.0 IN PTR localhost.	;
	@ IN NS ns.
	10 IN PTR ns.example.com.

**Note:** The *Serial Number* in the Reverse zone needs to be incremented on each change as well. For each *A record* user configure in `/etc/bind/db.example.com`, that is for a different address, user need to **create a PTR record in** `/etc/bind/db.192`.

After creating the reverse zone file restart BIND9:

**sudo systemctl restart bind9.service**

### 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`:

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

zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
    allow-transfer { 192.168.1.11; };
};
zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
    allow-transfer { 192.168.1.11; };
};
```

**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:

```
nano /etc/bind/named.conf.local
This Setting on Secondary Server

zone "example.com" {
    type secondary;
    file "db.example.com";
    masters { 192.168.1.10; };
};

zone "1.168.192.in-addr.arpa" {
```

```

type secondary;
file "db.192";
masters { 192.168.1.10; };
};

```

**Note:** Replace 192.168.1.10 with the IP Address of user **Primary nameserver**.  
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 { ipaddress; };** to **/etc/bind/named.conf.local** as shown in the example below at **Primary Server**:

<b>nano /etc/bind/named.conf.local</b>
<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 hosts 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.com
```

**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
```

### Example 2: using ping

Demonstrate how applications make use of DNS to resolve a host name use the **ping** utility to send an ICMP echo request:

```
ping example.com
```

This tests if the nameserver can resolve the name `ns.example.com` to an IP Address. The command output should resemble:

```
PING ns.example.com (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.800 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=0.813 ms
    === * ===
```

### 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. This is where Primary and Secondary servers are defined.

```
@    IN    NS    ns.example.com.
@    IN    NS    ns2.example.com.
ns   IN    A     192.168.1.10
ns2  IN    A     192.168.1.11
```

==== \* ====

### FTP server on LINUX and transfer files to demonstrate its 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 restart vsftpd**

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

- **ftp 172.16.20.116 21**
  - 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.

#### Example 2: Download the file from ftp server

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

#### Example 3: Upload the file from server

- **put /home/admincs/index.html test.html**
  - will upload the file **index.html** from current (local **/home/admincs**) user to remote ftp server with name **test.html**

**Example 4:** User can issue commands like **mkdir**, **cd**, **rmdir** etc linux basic commands to control the remote server machine. (For more commands go to ? Command and **man ftp** command)

==== \* ====

### Install and configure Apache web server and create virtual hosts:

- 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 install apache2**

**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 a **public\_html** folder that will hold our actual files. This gives 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**

- Virtual host files are the files that specify the actual configuration of our 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).

<b>/etc/apache2/sites-available/example.com.conf</b>	
<b>Before</b>	<b>After</b>
<pre>&lt;VirtualHost *:80&gt;   ServerAdmin webmaster@localhost   DocumentRoot /var/www/html   ErrorLog \${APACHE_LOG_DIR}/error.log   CustomLog     \${APACHE_LOG_DIR}/access.log combined &lt;/VirtualHost&gt;</pre>	<pre>&lt;VirtualHost *:80&gt;   ServerAdmin <b>admin@example.com</b>   <b>ServerName example.com</b>   <b>ServerAlias www.example.com</b>   DocumentRoot     <b>/var/www/example.com/public_html</b>   ErrorLog \${APACHE_LOG_DIR}/error.log   CustomLog     \${APACHE_LOG_DIR}/access.log combined &lt;/VirtualHost&gt;</pre>

- 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`**

- For **example**, for the domains machine with ip address 172.16.20.107 is used then, add the following lines to the bottom of hosts file:  
127.0.0.1 localhost  
127.0.1.1 guest-desktop  
**172.16.20.107 example.com**
- This will direct any requests for example.com on user computer and send them to server at **172.16.20.107** (where above 7 steps have been done). This is what is required if we are not actually the owners of these domains in order to test our virtual hosts.
- Save and close the file.

#### Testing working of Web Server

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

=== \* ===

**Work on Practice Session**

**Work on Practice Session**

## 13 – Storage Management

### Basic Commands for Storage Partitions

#### fdisk

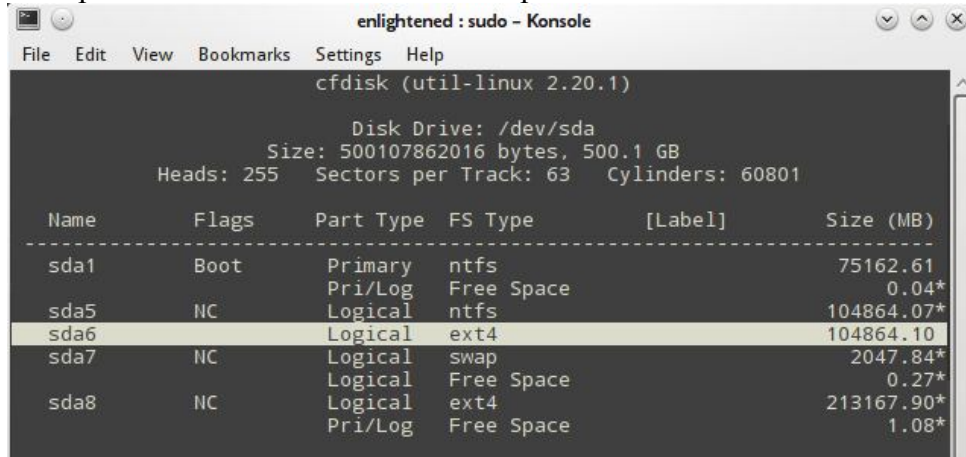
- fdisk is used to check the partitions on a disk.
- The fdisk command can display the partitions and details like file system type.
- However it does not report the size of each partitions.
  - **sudofdisk -l**
- Each device is reported separately with details about size, seconds, id and individual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



```

enlightened : sudo - Konsole
File Edit View Bookmarks Settings Help
cfdisk (util-linux 2.20.1)

Disk Drive: /dev/sda
Size: 500107862016 bytes, 500.1 GB
Heads: 255 Sectors per Track: 63 Cylinders: 60801

  Name      Flags      Part Type  FS Type      [Label]      Size (MB)
  -----
  sda1      Boot       Primary   ntfs          75162.61
             Pri/Log    Free Space 0.04*
  sda5      NC         Logical   ntfs          104864.07*
  sda6      NC         Logical   ext4          104864.10
  sda7      NC         Logical   swap          2047.84*
             Logical   Free Space 0.27*
  sda8      NC         Logical   ext4          213167.90*
             Pri/Log    Free Space 1.08*
  
```

- cfdisk works with one partition at a time.
  - So if user need 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 file systems.
- 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 a easy to read manner.
  - **pydf**



- `pydf` is limited to showing only the mounted file systems.

**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 file system type. Does not report the space on the partitions.
  - **sudoblkid**

**hwnfo**

- The `hwnfo` 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.
  - **`hwnfo --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`**

==== \* ====

**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 lvm...)
  - **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).

==== \* ====

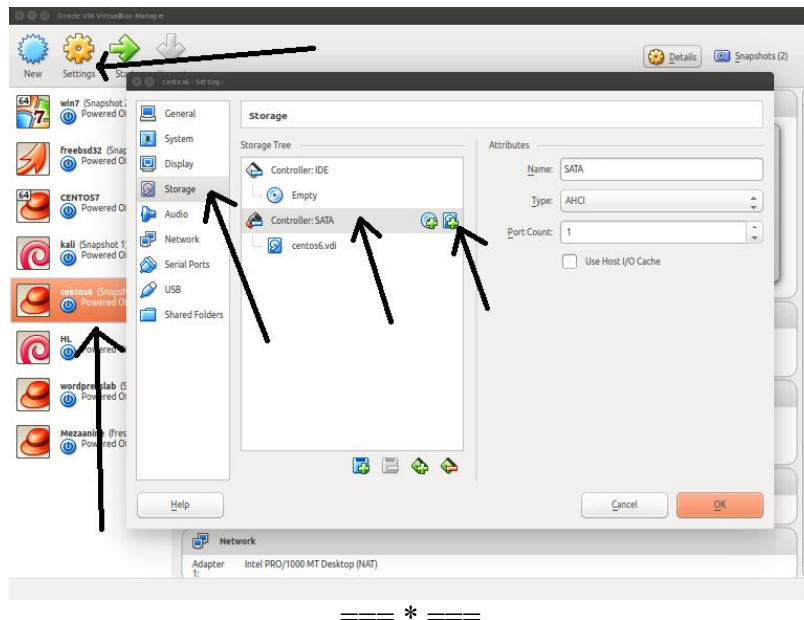
#### Initial Settings for LVM if Virtual Box Linux image is used (Better Choice for Practice):

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 **vdi** 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 [ 64.00 MiB]
```

```
/dev/sdb [ 100.00 GiB]
```

**// 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:

**sudo pvcreate /dev/sdb**

```
[ pvremove /dev/sdb // for removing Physical Volume ]
```

**Output:**

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**

[ `vgremove VolumeGroupName` // for removing Volume Group ]

#### Output:

Volume group "LVMVolGroup" successfully created

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

**sudo pvs**

#### Output:

PV	VG	Fmt	Attr	PSize	PFree
/dev/sdb	LVMVolGroup	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 a 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 db LVMVolGroup****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 **vgs** 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 **-l** flag, 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 in **100%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	LSize	LV
LVMVolGroup	1	4	0	wz--n-	100g	0	10.00g	projects
LVMVolGroup	1	4	0	wz--n-	100g	0	05.00g	www
LVMVolGroup	1	4	0	wz--n-	100g	0	20.00g	db
LVMVolGroup	1	4	0	wz--n-	100g	0	65.00g	workspace

- The “workspace” volume has been created and the “LVMVolGroup” volume group is completely allocated.

•

**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**

- [
  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 mount points:
  - `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:
  - 1. `sudo mount /dev/LVMVolGroup/projects /mnt/projects`**
  - 2. `sudo mount /dev/LVMVolGroup/www /mnt/www`**
  - 3. `sudo mount /dev/LVMVolGroup/db /mnt/db`**
  - 4. `sudo mount /dev/LVMVolGroup/workspace /mnt/workspace`**
- To make the mounts persistent, add them to `/etc/fstab` just like user would with normal block devices:
  - `sudo nano /etc/fstab`**

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 at boot.
 

====\*====

### 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.
 

==== \* =====

**Work on Practice Session**

**Work on Practice Session**