Linux user management commands

User creation and management in Linux is done by using Linux administration commands. Here is a list of user management commands in Linux:

Command	Description
sudo adduser username	Adds a user
sudo passwd -l 'username'	Disable a user
sudo userdel -r 'username'	Delete a user
sudo usermod -a -G GROUPNAME USERNAME	Add user a to a usergroup
sudo deluser USER GROUPNAME	Remove user from a user group
finger	Gives information on all logged in user
finger username	Gives information of a particular user

- You can use both GUI or Terminal for User Administration in Linux User management
- You can create, disable and remove user accounts using Linux admin commands.
- You can add/delete a user to a usergroup.

Default actions related to user creation operation

When a new user account is added to the system, the following operations are performed.

1. His/her home directory is created (**/home/username** by default).

2. The following hidden files are copied into the user's home directory, and will be used to provide environment variables for his/her user session.

.bash_logout

.bash_profile

.bashrc

3. A mail spool is created for the user at /var/spool/mail/**username**.

4. A group is created and given the same name as the new user account.

Understanding /etc/passwd

- The full account information is stored in the **/etc/passwd** file. This file contains a record per system user account and has the following format (fields are delimited by a colon).
- **[username]:[x]:[UID]:[GID]:[Comment]:[Home directory]:[Default shell]**
- Fields **[username]** and **[Comment]** are self explanatory.
- The **x** in the second field indicates that the account is protected by a shadowed password (in **/etc/shadow**), which is needed to logon as **[username]**.
- The **[UID]** and **[GID]** fields are integers that represent the User IDentification and the primary Group IDentification to which **[username]** belongs, respectively.
- The **[Home directory]** indicates the absolute path to **[username]**'s home directory, and
- The **[Default shell]** is the shell that will be made available to this user when he or she logins the system.

Understanding /etc/group

- Group information is stored in the **/etc/group** file. Each record has the following format.
- **[Group name]:[Group password]:[GID]:[Group members]**
- **[Group name]** is the name of group.
- An **x** in **[Group password]** indicates group password is not being used.
- **[GID]**: same as in /etc/passwd.
- **[Group members]**: a comma separated list of users who are members of **[Group name]**.

❖ One more thing to note is that UID = 0 and GID = 0 are what provide the root user all the powers in the system. To prove so, rename the root to something else like Example_User and create a new root user with a new UID and GID. You will realize that the Example_User will still have elevated privileges despite not having the username root.

Options to Make changes to an existing user

- After adding an account, you can edit the following information (to name a few fields) using the **usermod** command, basic syntax of usermod is as follows.
- **# usermod [options] [username]**
- The '**usermod**' command is simple to use with lots of options to make changes to an existing user.
- **-c** = We can add comment field for the useraccount.
- **-d** = To modify the directory for any existing user account.
- **-e** = Using this option we can make the account expiry in specific period.
- **-g** = Change the primary group for a User.
- **-G** = To add a supplementary groups.
- **-a** = To add anyone of the group to a secondary group.

Pre-requisits for executing usermod command

- We must have existing user accounts to execute usermod command.
- Only superuser (root) is allowed to execute usermod command.
- The usermod command can be executed on any Linux distribution.
- Must have basic knowledge of usermod command with options

Contd.

- **-l** = To change the login name from minu to minu_admin.
- **-L** = To lock the user account. This will lock the password so we can't use the account.
- **-m** = moving the contents of the home directory from existing home dir to new dir.
- **-p** = To Use un-encrypted password for the new password. (NOT Secured).
- **-s** = Create a Specified shell for new accounts.
- **-u** = Used to Assigned UID for the user account between 0 to 999.
- **-U** = To unlock the user accounts. This will remove the password lock and allow us to use the user account.

When we execute 'usermod' command in terminal, the following files are used and affected:

- **/etc/passwd** – User account information.
- **/etc/shadow** – Secure account information.
- **/etc/group** – Group account information.
- **/etc/gshadow** – Secure group account information.
- **/etc/login.defs** – Shadow password suite configuration..

Adding Information to User Account

- The '**-c**' option is used to set a brief comment (information) about the user account. For example, let's add information on '**minu**' user, using the following command.
- **# usermod -c "This is minu" minu**
- After adding information on user, the same comment can be viewed in **/etc/passwd** file.
- **# grep -E --color 'minu' /etc/passwd**
- minu:x:500:500:**This is minu**:/home/minu:/bin/sh
- **Changing the default location of the user's home directory**
- Use the **-d**, or **–home** options, followed by the absolute path to the new home directory.
- In the above step we can see that our home directory is under **/home/minu/**

- **# usermod -d /tmp minu**
- **# grep -E --color '/tmp' /etc/passwd**
- minu:x:500:500:This is minu:**/tmp**:/bin/sh

User management

- **Setting the expiry date for an account**
- Use the -e or **–expiredate** flag followed by a date in **YYYY-MM-DD** format.
- Before, setting up an expiry date on a user, let's first check the current account expiry status using the '**chage**' (change user password expiry information) command.
- # chage -l minu
- Last password change : Nov 02, 2014
- Password expires : never
- Password inactive : never
- Account expires : **Dec 01, 2014**
- Minimum number of days between password change : 0
- Maximum number of days between password change : 99999
- Number of days of warning before password expires : 7

Contd.

- The expiry status of a '**minu**' user is **Dec 1 2014**, let's change it to **Nov 1 2014** using '**usermod -e**' option and confirm the expiry date with '**chage**' command.
- **# usermod -e 2014-11-01 minu**
- **# chage -l minu**

Last password change : Nov 02, 2014
Password expires : never
Password inactive : never
Account expires : Nov 01, 2014
Minimum number of days between password change : 0
Maximum number of days between password change: 99999
Number of days of warning before password expires :7

Contd.

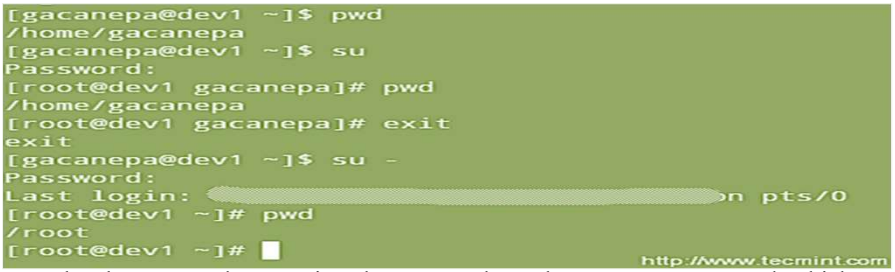
- **Change User Primary Group**
- To set or change a user primary group, we use option '**-g**' with usermod command. Before, changing user primary group, first make sure to check the current group for the user **minu_test**.
- **# id minu_test**
- uid=501(minu_test) gid=502(**minu_test**) groups=502(minu_test)
- Now, set the **babin** group as a primary group to user **minu_test** and confirm the changes.
- **# usermod -g babin minu_test**
- **# id minu_test**
- uid=501(minu_test) gid=502(**babin**) groups=502(minu_test)

Contd.

- **Adding the user to supplementary groups**
- Use the combined **-aG**, or **–append –groups** options, followed by a comma separated list of groups.
- **# usermod --append --groups root,users,minu**
- **Note**: Be careful, while adding a new groups to an existing user with '**-G'** option alone, will remove all existing groups that user belongs. So, always add the '**-a**' (append) with '**-G**' option to add or append new groups.

Accessing the root Account and Using sudo

- One of the ways users can gain access to the root account is by typing.
- $ su
- and then entering root's password. If authentication succeeds, you will be logged on as **root** with the current working directory as the same as you were
- before. If you want to be placed in root's home directory instead, run
- $ su - and then enter root's password.



- The above procedure requires that a normal user knows root's password, which poses a serious security risk. For that reason, the sysadmin can configure the **sudo** command to allow an ordinary user to execute commands as a different user (usually the superuser) in a very controlled and limited way. Thus, restrictions can be set on a user so as to enable him to run one or more specific privileged commands and no others.

File Managements

- All data in linux is organized into files.
- All files are organized into directories.
- These directories are organized into a tree-like structure called the filesystem.
- In linux, there are three basic types of files −
- **Ordinary Files** − An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories** − Directories store both special and ordinary files. For users familiar with Windows or Mac OS, linux directories are equivalent to folders.
- **Special Files** − Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Listing Files

- To list the files and directories stored in the current directory, use the following command −
- **$ls**

```
bin         hosts   lib       res.03
ch07        hw1     pub       test_results
ch07.bak            hw2       res.01   users
docs        hw3     res.02    work
```

- The command ls supports the -l option which would help you to get more information about the listed files −
- **$ls -l**
- total 1962188
- drwxrwxr-x 2 arnood arnood 4096 Dec 25 09:59 uml
- -rw-rw-r– 1 arnood arnood 5341 Dec 25 08:38 uml.jpg
- drwxr-xr-x 2 arnood arnood 4096 Feb 15 2006 univ
- drwxr-xr-x 2 root root 4096 Dec 9 2007 urlspedia

Contd.

- 1 2 3 4 5 6 7
- drwxr-xr-x 11 arnood arnood 4096 May 29 2007 zlib-1.2.3
- **Here is the information about all the listed columns –**
- **First Column** – Represents the file type and the permission given on the file.
- **Second Column** – Represents the number of memory blocks taken by the file or directory.
- **Third Column** – Represents the owner of the file. This is the linux user who created this file.
- **Fourth Column** – Represents the group of the owner. Every linux user will have an associated group.
- **Fifth Column** – Represents the file size in bytes.
- **Sixth Column** – Represents the date and the time when this file was created or modified for the last time.
- **Seventh Column** – Represents the file or the directory name.

In the ls -l listing example, every file line begins with a d, -, or l. These characters indicate the type of the file that's listed.

Prefix	Description
-	Regular file, such as an ASCII text file, binary executable, or hard link.
b	Block special file. Block input/output device file such as a physical hard drive.
c	Character special file. Raw input/output device file such as a physical hard drive.
d	Directory file that contains a listing of other files and directories.
l	Symbolic link file. Links on any regular file.
p	Named pipe. A mechanism for interprocess communications.
s	Socket used for interprocess communication.

Hidden Files

- An invisible file is one, the first character of which is the dot or the period character (.). Linux programs (including the shell) use most of these files to store configuration information.
- Some common examples of the hidden files include the files −
- .profile − The Bourne shell (sh) initialization script
- .kshrc − The Korn shell (ksh) initialization script
- .cshrc − The C shell (csh) initialization script
- .rhosts − The remote shell configuration file
- **To list the invisible files, specify the -a option to ls −**
- **$ ls -a**

.	.profile	docs	lib	test_results
..	.rhosts	hosts	pub	users
.	emacs	bin	hw1	res.01 work
.	exrc	ch07	hw2	res.02
.	kshrc	ch07.bak	hw3	

Creating Files

- use the vi editor to create ordinary files on any linux system
- **$ vi filename**
- The above command will open a file with the given filename. Now, press the key i to come into the edit mode. Once you are in the edit mode, you can start writing your content in the file as in the following program −
- This is linux file....I created it for the first time..... I'm going to save this content in this file.
- Once you are done with the program, follow these steps −
- Press the key esc to come out of the edit mode.
- Type wq! You will save and exit.
- You will now have a file created with filename in the current directory.

Editing Files

- You can edit an existing file using the vi editor. We will discuss in short how to open an existing file −
- **$ vi filename**
- Once the file is opened, you can come in the edit mode by pressing the key i and then you can proceed by editing the file. If you want to move here and there inside a file, then first you need to come out of the edit mode by pressing the key Esc. After this, you can use the following keys to move inside a file −
- l key to move to the right side.
- h key to move to the left side.
- k key to move upside in the file.
- j key to move downside in the file.
- So using the above keys, you can position your cursor wherever you want to edit. Once you are positioned, then you can use the i key to come in the edit mode. Once you are done with the editing in your file, press Esc and finally two keys
- Shift + ZZ together to come out of the file completely.

Display Content of a File

- You can use the cat command to see the content of a file. Following is a simple example to see the content of the above created file −
- **$ cat filename**
- This is linux file....I created it for the first time.....
- I'm going to save this content in this file.
- $
- You can display the line numbers by using the -b option along with the cat command as follows −
- **$ cat -b filename**
- 1 This is linux file....I created it for the first time.....
- 2 I'm going to save this content in this file.
- $

Counting Words in a File

- You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file.
- Following is a simple example to see the information about the file created above −
- **$ wc filename**
- 2 19 103 filename
- $
- Here is the detail of all the four columns −
- **First Column** − Represents the total number of lines in the file.
- **Second Column** − Represents the total number of words in the file.
- **Third Column** − Represents the total number of bytes in the file. This is the actual size of the file.
- **Fourth Column** − Represents the file name.
- You can give multiple files and get information about those files at a time. Following is simple syntax −
- **$ wc filename1 filename2 filename3**

Copying Files

- To make a copy of a file use the cp command. The basic syntax of the command is −
- **$ cp source_file destination_file**
- Following is the example to create a copy of the existing file filename.
- **$ cp filename copyfile**
- $
- You will now find one more file copyfile in your current directory. This file will exactly be the same as the original file filename.

Renaming Files

- To change the name of a file, use the mv command. Following is the basic syntax −
- **$ mv old_file new_file**
- The following program will rename the existing file filename to newfile.
- **$ mv filename newfile**
- $
- The mv command will move the existing file completely into the new file. In this case, you will find only newfile in your current directory.

Deleting Files

- To delete an existing file, use the rm command. Following is the basic syntax −
- **$ rm filename**
- Caution − A file may contain useful information. It is always recommended to be careful while using this Delete command. It is better to use the -i option along with rm command.
- $
- You can remove multiple files at a time with the command given below −
- **$ rm filename1 filename2 filename3**

Linux Text Editor

- **Text Editor**
- A text editor is like a word processor without a lot of features. The main use of a text editor is for writing something in plain text with no formatting so that another program can read it. Plain text can be edited in Linux by graphical GUI editors or console text editors.

A. Console Based Editor:

- **Vim**:
- Vi or Vim is one of the most popular editors. Vim is a command-line editor that's completely keyboard-based. It can be used in any OS, on any desktop environment and it won't take up a lot of system resources. This editor is ubiquitous and available on all Linux systems and is the "standard" Linux editor. While it is not intuitive and has a learning curve, it is worth learning if Linux is part of your career or future.
- **emacs**:
- This console based plain text editor supports the theory that more is better. It tries to support every feature possible.
- **pico**:
- This console based plain text editor operates with the simplicity of a GUI editor making it a favorite with Linux beginners.
- **nano**:
- This is a GNU.org clone of Pico.

Linux Text Editor contd.

B. Graphical User Interface GUI based :

gedit:

This is the default text editor for the Linux Gnome desktop. It supports syntax highlighting, printing, a variety of plug-ins, multi-language spell check, tabbed for multiple files and more. It's incredibly simple to use.

jEdit:
is a mature programmer's text editor.While jEdit beats many expensive development tools for features and ease of use, it is released as free software with full source code, provided under the terms of the GPL 2.0.

Gvim:
It is GUI editor based on the vim console editor. While it provides many of the features offered by a GUI text editor, it will still require knowledge of vim to stay out of trouble.

NEdit:
The Nirvana editor is a text editor and source code editor for the X Window System. It has an interface similar to text editors on Microsoft Windows and Macintosh. This is one of the original Unix GUI editors programmed in Motif.

Geany:
It is small and fast, it only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME. Geany only requires the GTK2 runtime libraries.

Sublime:
It is a cross-platform text and source code editor with a Python API. The GUI was inspired by Vim. Its functionality is also extendable with sublime-packages; Sublime Text is not open source software nor free software, but some of the extending packages have free-software licenses and are community-built and maintained.

Directory Management

- linux uses a hierarchical structure for organizing files and directories. This structure is often referred to as a directory tree. The tree has a single root node, the slash character (/), and all other directories are contained below it.
- **Home Directory**
- The directory in which you find yourself when you first login is called your home directory.
- You can go in your home directory anytime using the following command −
- **$cd ~**
- $
- Here ~ indicates the home directory. Suppose you have to go in any other user's home directory, use the following
- command −
- $**cd ~username**
- To go in your last directory, you can use the following command −
- **$cd -**

Absolute/Relative Pathnames

- Directories are arranged in a hierarchy with root (/) at the top. The position of any file within the hierarchy is described by its pathname.
- Elements of a pathname are separated by a '/'. A pathname is absolute, if it is described in relation to root, thus absolute pathnames always begin with a /.
- Following are some examples of absolute filenames.
- /etc/passwd
- /users/sjones/chem/notes
- /dev/rdsk/Os3
- A pathname can also be relative to your current working directory. Relative pathnames never begin with /. Relative to user arnood's home directory, some pathnames might look like this −
- chem/notes
- personal/res
- To determine where you are within the filesystem hierarchy at any time, enter the command pwd to print the current working directory −
- **$pwd**
- /user0/home/arnood

Listing Directories

- To list the files in a directory, you can use the following syntax −
- **$ls dirname**
- Following is the example to list all the files contained in /usr/local directory −
- **$ls /usr/local**
- X11 bin gimp jikes sbin
- ace doc include lib share
- atalk etc info man ami

Creating Directories

- We will now understand how to create directories. Directories are created by the following command −
- **$mkdir dirname**
- Here, directory is the absolute or relative pathname of the directory you want to create. For example, the command −
- **$mkdir mydir**
- Creates the directory mydir *in the current* directory. Here is another example −
- **$mkdir /tmp/test-dir**
- This command creates the directory test-dir *in the desired* /tmp directory. The mkdir command produces no output if it successfully creates the requested directory.
- If you give *more than one directory* on the command line, mkdir creates each of the directories. For example, −
- **$mkdir docs pub**
- Creates the directories docs and pub under the current directory.

Creating Parent Directories

- Sometimes when you want to create a directory, its parent directory or the directories *might not exist*. In this case, mkdir issues an error message as follows −
- **$mkdir /tmp/arnood/test**
- mkdir: Failed to make directory "/tmp/arnood/test"; No such file or directory
- In such cases, you can *specify the -p option* to the mkdir command. It creates all the necessary directories for you. For example −
- **$mkdir -p /tmp/arnood/test**
- The above command creates all the required parent directories.

Removing Directories

- Directories can be deleted using the rmdir command as follows −
- **$rmdir dirname**
- Note − To remove a directory, make sure it is empty which means there should not be any file or sub-directory inside this directory.
- You can remove multiple directories at a time as follows −
- **$rmdir dirname1 dirname2 dirname3**
- The above command removes the directories dirname1, dirname2, and dirname3, if they are empty. The rmdir command produces no output if it is successful.

Changing Directories

- You can use the cd command to do more than just change to a home directory. You can use it to change to any directory by specifying a valid absolute or relative path. The syntax is as given below −
- **$cd dirname**
- Here, dirname is the name of the directory that you want to change to. For example, the command −
- **$cd /usr/local/bin**
- Changes to the directory /usr/local/bin. From this directory, you can cd to the directory /usr/home/arnood using the following relative path −
- **$cd ../../home/arnood**
- $

Renaming Directories

- The mv (move) command can also be used to rename a directory. The syntax is as follows −
- **$mv olddir newdir**
- You can rename a directory mydir to yourdir as follows −
- **$mv mydir yourdir**

The directories . (dot) and .. (dot dot)

- The filename . (dot) represents the current working directory; and the filename .. (dot dot) represents the directory one level above the current working directory, often referred to as the parent directory.
- If we enter the command to show a listing of the current working directories/files and use the -a option to list all the files and the -l option to provide the long listing, we will receive the following result.

```
$ls-la
drwxrwxr-x 4      teacherclass  2048  Jul16  17.56  .
drwxr-xr-x 60     root         1536  Jul13  14:18  ..
----------  1     teacherclass  4210  May1   08:27  .profile
-rwxr-xr-x 1      teacherclass  1948  May12  13:42  memo
```

Standard linux Streams

- Under normal circumstances, every linux program has three streams (files) opened for it when it starts up −
- **stdin** − This is referred to as the *standard input* and the associated file descriptor is 0. This is also represented as STDIN. The linux program will read the default input from STDIN.
- **stdout** − This is referred to as the *standard output* and the associated file descriptor is 1. This is also represented as STDOUT. The linux program will write the default output at STDOUT
- **stderr** − This is referred to as the *standard error* and the associated file descriptor is 2. This is also represented as STDERR. The linux program will write all the error messages at STDERR.
- Note: A command normally reads its input from the standard input, which happens to be your terminal by default. Similarly, a command normally writes its output to standard output, which is again your terminal by default.

Output Redirection

- The output from a command normally intended for standard output can be easily diverted(redirected) to a file with the notation > file name. Any command that normally writes its output to standard output(your terminal by default), the output of that command will be written to file instead of your terminal.

- Check the following **who** command (a tool, prints information about users who are currently logged in) which redirects the complete output of the command in the users file.
- **$ who > users**
- Notice that no output appears at the terminal. This is because the output has been redirected from the default standard output device (the terminal) into the specified file. You can check the users file for the complete content −
- **$ cat users**
- oko tty01 Sep 12 07:30
- ai tty15 Sep 12 13:32
- ruth tty21 Sep 12 10:10

Contd.

- If a command has its output redirected to a file and the file already contains some data, that data will be lost. Consider the following example −
- **$ echo line 1 > users**
- **$ cat users**
- line 1
- The content of **users** file shown in last slide is lost after the above execution.
- The remedy is, You can use >> operator to append the output in an existing file as follows −
- **$ echo line 2 >> users**
- **$ cat users**
- line 1
- line 2

Input Redirection

- Just as the output , the input of a command can be redirected from a file. As the **greater-than character >** is used for output redirection, the **less-than character <** is used to redirect the input of a command.
- The commands that normally take their input from the standard input can have their input redirected from a file in this manner. For example, to count the number of lines in the file **users** generated above, you can execute the command as follows −
- **$ wc -l users**
- 2 users
- Upon execution, you will receive the following output. You can count the number of lines in the file by redirecting the standard input of the **wc** command from the file **users** −
- **$ wc -l < users**
- 2
- Note:There is a difference in the output produced by the two forms of the wc command. In the first case, the name of the file users is listed with the line count; in the second case, it is not.
- Actually, In the first case, wc knows that it is reading its input from the file users. In the second case, it only knows that it is reading its input from standard input so it does not display file name.

Discard the output

- Sometimes you will need to execute a command, but you don't want the output displayed on the screen. In such cases, you can discard the output by redirecting it to the file /dev/null −
- **command > /dev/null**
- Here command is the name of the command you want to execute. The file /dev/null is a special file that automatically discards all its input.

- To discard both output of a command and its error output, use standard redirection to redirect STDERR to STDOUT −
- **$ command > /dev/null 2>&1**
- Here 2 represents STDERR and 1 represents STDOUT.
- You can display a message on to STDERR by redirecting STDOUT into STDERR as follows −

- **$ echo message 1>&2**

Use of pipe (|)

- The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next. In short, the output of each process directly as input to the next on. The symbol '|' denotes a pipe like a pipeline.
- *Let us understand this with an example:-*
- When you use 'cat' command to view a file which spans multiple pages, the prompt quickly jumps to the last page of the file, and you do not see the content in the middle.
- To avoid this, you can pipe the output of the 'cat' command to 'less' which will show you only one scroll length of content at a time.
- **cat filename | less**
- i.e., it will pause after one scroll length and allow you to view scroll wise.

The **pg** and **more** Commands

- A long output can normally be zipped by you on the screen, but if you run text through **more** or use the **pg** command as a filter; the display stops once the screen is full of text.
- Let's assume that you have a long directory listing. To make it easier to read the sorted listing, pipe the output through more as follows −
- $**ls -l | grep "Aug" | more**
- -rw-rw-r-- 1 carol doc 1605 Aug 23 07:35 macros
- -rw-rw-r-- 1 john doc 2488 Aug 15 10:51 intro
- -rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07
- -rw-rw-r-- 1 john doc 14827 Aug 9 12:40 ch03
- .
- .
- .
- -rw-rw-rw- 1 john doc 16867 Aug 6 15:56 ch05
- --More--(74%)
- The screen will fill up once the screen is full of text consisting of lines. At the bottom of the screen is the **more** prompt, where you can type a command to move through the text.

Content management commands

- **head**
- Displays first few lines of a file
- **tail**
- Prints last few lines in a file
- **cmp**
- Compares the contents of two files
- **diff**
- Differential file comparator

Permission Settings

- Every file in Unix/ Linux has the following attributes −

- **Owner permissions** − The owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions** − The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions** − The permissions for others indicate what action all other users can perform on the file.
- **The Permission Indicators**
- While using **ls -l** command, it displays various information related to file permission as follows −
- **$ls -l /home/arnood**
- **-rwxr-xr--** 1 arnood users 1024 Nov 2 00:10 myfile
- **drwxr-xr--** 1 arnood users 1024 Nov 2 00:10 mydir

Permission Contd.

- **-rwxr-xr--** 1 arnood users 1024 Nov 2 00:10 myfile

1	2	3	4	5	6	7	8	9	10
-	r	w	x	r	-	x	r	-	-
	<------user--------- >			<----groups-------->			< ------others------- >		

Here, the first column represents different access modes, i.e., the permission associated with a file or a directory.
The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) −

- The first three characters (2-4) represent the permissions for the file's owner. For example, -rwxr-xr-- represents that the owner has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, -rwxr-xr-- represents that the group has read (r) and execute (x) permission, but no write permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example, -rwxr-xr-- represents that there is read (r) only permission.

Permission Contd.

- **Summery**
- MODE can be a comma separated combinations of 3 sets of symbols:
- [ugoa][+ - =][rwx]
- u --> user
- g --> group
- o --> others
- a --> all
- You can add (+), subtract (–), or exactly set (=) the mode. Examples:
- chmod u+w file # Add user (u) write (w) privileges
- chmod u-w file # Remove user (u) write privileges.
- chmod g+w file # Add group (g) write privileges
- chmod g=r file # Allow only the group read privileges, nothing else.
- chmod o+x file # Add execute (x) privileges for others.
- chmod a+x file # Add execute (x) privileges for everyone..
- chmod a=xr file # Allow read and execute only to everyone..
- chmod go-r file # Remove group and others read privileges..
- chmod ugo+w file # Same as chmod a+w file
- MODE can also be an octal representation which represents the absolute mode:
- 1 = others execute # o=x
- 2 = others write # o=w
- 4 = others read # o=r
- 10 = group execute # g=x
- 20 = group write # g=w
- 40 = group read # g=r
- 100 = user execute # u=x
- 200 = user write # u=w
- 400 = user read # u=r

Permission Contd.

- **File Access Modes**
- The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the read, write, and execute permissions, which have been described below −

- **Read**
- Grants the capability to read, i.e., view the contents of the file.

- **Write**
- Grants the capability to modify, or remove the content of the file.

- **Execute**
- User with execute permissions can run a file as a program.

Permission Contd.

- **Directory Access Modes**
- Directory access modes are listed and organized in the same manner as any other file. There are a few differences that need to be mentioned −

- **Read**
- Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.

- **Write**
- Access means that the user can add or delete files from the directory.

- **Execute**
- Executing a directory doesn't really make sense, so think of this as a traverse permission.

- A user must have execute access to the bin directory in order to execute the ls or the cd command.

Permission Contd.

- **Changing Permissions**
- To change the file or the directory permissions, you use the **chmod** (change mode) command. There are two ways to use chmod — the ***symbolic mode and the absolute mode.***
- **Using chmod in Symbolic Mode**
- The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode. With symbolic permissions you can add, delete, or specify the permission set you want by using the operators in the following table.
- **Operator** **Description**
- \+ Adds a permission to a file or directory
- \- Removes the permission
- = Sets the permission and overrides the permissions set earlier.
- **User Denotations**
- u user/owner
- g group
- o other
- a all

Permission Contd.

- Here's an example using testfile. Running ls -1 on the testfile shows that the file's permissions are as follows −
- $**ls -l testfile**
- **-rwxrwxr--** 1 arnood users 1024 Nov 2 00:10 testfile
- Then each example **chmod** command from the preceding table is run on the testfile, followed by ls –l, so you can see the permission changes −
- **$chmod o+wx testfile**
- **$ls -l testfile**
- -rwxrwxrwx 1 arnood users 1024 Nov 2 00:10 testfile
- **$chmod u-x testfile**
- **$ls -l testfile**
- -rw-rwxrwx 1 arnood users 1024 Nov 2 00:10 testfile
- **$chmod g = rx testfile**
- **$ls -l testfile**
- -rw-r-xrwx 1 arnood users 1024 Nov 2 00:10 testfile

Permission contd.

- Here's how you can combine these commands on a single line −

- **$chmod o+wx,u-x,g = rx testfile**
- **$ls -l testfile**
- -rw-r-xrwx 1 arnood users 1024 Nov 2 00:10 testfile

- **Using chmod with Absolute Permissions**
- The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file.
- Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set.

Permission contd.

Number	Octal Permission Representation	Ref
0	No permission	- - -
1	Execute permission	- - x
2	Write permission	- w -
3	Execute and write permission: 1 (execute) + 2 (write) = 3	- w x
4	Read permission	r - -
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r - x
6	Read and write permission: 4 (read) + 2 (write) = 6	r w -
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	r w x

Permission contd.

- Here's an example using the **testfile**. Running **ls -l** on the testfile shows that the file's permissions are as follows −

- **$ls -l testfile**
- -rwxrwxr-- 1 arnood users 1024 Nov 2 00:10 testfile

- Then each example chmod command from the preceding table is run on the testfile, followed by ls –l, so you can see the permission changes −

- **$ chmod 755 testfile**
- **$ls -l testfile**
- -rwxr-xr-x 1 arnood users 1024 Nov 2 00:10 testfile
- **$chmod 743 testfile**
- **$ls -l testfile**
- -rwxr---wx 1 arnood users 1024 Nov 2 00:10 testfile
- **$chmod 043 testfile**
- **$ls -l testfile**
- ----r---wx 1 arnood users 1024 Nov 2 00:10 testfile

Permission contd.

- **Changing Owners and Groups**
- While creating an account on Linux, it assigns a owner ID and a group ID to each user. All the permissions mentioned above are also assigned based on the Owner and the Groups.
- Two commands are available to change the owner and the group of files −
- chown − The chown command stands for "change owner" and is used to change the owner of a file.
- chgrp − The chgrp command stands for "change group" and is used to change the group of a file.
- Changing Ownership
- The chown command changes the ownership of a file. The basic syntax is as follows −
- $ chown user filelist
- The value of the user can be either the name of a user on the system or the user id (uid) of a user on the system.
- The following example will help you understand the concept −
- $ chown arnood testfile
- Changes the owner of the given file to the user arnood.

Permission contd.

- NOTE – The super user, root, has the unrestricted capability to change the ownership of any file but normal users can change the ownership of only those files that they own.

- **Changing Group Ownership**
- The chgrp command changes the group ownership of a file. The basic syntax is as follows –

- **$ chgrp group filelist**
- The value of group can be the name of a group on the system or the group ID (GID) of a group on the system.

- Following example helps you understand the concept –

- **$ chgrp special testfile**

- Changes the group of the given file to special group.

File search commands

- **The Basic Syntax**
- The most popular command to find and filter files in a directory hierarchy on Linux is find. The basic syntax is as follows:
- **find <startingdirectory> <options> <search term>**
- It starts with the keyword find, which alerts Linux that whatever follows after will be used to find your file. The <startingdirectory> argument is the origin point of where you want to start the search. It can be replaced with several arguments, including:
- / (slash) — search the whole system.
- . (dot) — search from the folder you're currently working on (current directory).
- ~ (tilde) — to search from your home folder.
- Searching by Name
- Of course, the most common method to look for a file is using its name. To run a simple search query using the name of the file, use the find command like this:

- find . -name my-file

Search contd.

- Here **-name** option, is case sensitive in Linux, and searched for a file called my-file.
- Searching in the current directory by using the . (dot) argument at starting point.
- If you are not sure about its case-sensitivity of the particular file , use the following find command:

- **find . -iname my-file**
- You can look for multiple files with a common format like .txt as well:

- **find . -name "*.txt"**
- Lastly, if you want to find a certain file by name and remove it, use the -delete argument after the file name:

- **find . -name my-file -delete**

Search contd.

- **Searching by Type**
- Linux allows users to list all information based on their types. There are several filters that you can use:

- d – directory or folder
- f – normal file
- l – symbolic link
- c – character devices
- b – block devices
- A simple example of using a file type can be seen below:

- **find / -type d**
- This will list all of the current directories in your system since we searched from our root directory with the / (slash) symbol.

Search contd.

- **Searching by Time**
- If you want to search for files based on when they were accessed and modification time footprints. Linux keeps track of the files using these three timestamps.
- **Access Time** (**-atime**) – when the file was either read or written into.
- **Modification Time** (**-mtime**) – when the file was modified.
- **Change Time** (**-ctime**) – when the file's meta-data was updated
- This option has to be used with a number that specifies how many days passed since the file was accessed, modified or changed:
- **find / -atime 1**
- This command will show all files that were accessed a day ago starting from your current time.
- We can narrow down our queries even more by adding **plus** (+) and **minus** (–) signs preceding the number of days. For instance:
- **find / -mtime +2** It lists down all the files that have a modification time of more than two days ago.
- To find all files whose meta-data was updated less than a day ago, run the following:
- **find / -ctime -1**

Search contd.

- **Searching by Permissions**
- Users can search for files based on file *permissions* using **-perm** option. For example:
- **find / -perm 644**
- In Linux, **644** corresponds to read and write permission. That means this command will look for all the files that have only read and write permissions. You can play around with this option further:
- **find / -perm -644** With an addition of a dash symbol, it will return with all the files that have at least 644 permission.
- **Other Useful Options**
- There are other useful options that you should remember.
- For example, to look for *empty files* and folders on your system, use the following:
- **find / -empty**
- Similarly, to look for all the *executable* saved on your drive, utilize the **-exec** option:
- **find / -exec**
- To look for readable files, you can run the following command:
- **find / -read**

Search contd.

- **Using Locate Command in Linux**
- The **locate** command is a useful alternative, as it is faster than the **find** command when performing searches. That's because the former only scans your Linux database instead of the whole system. Furthermore, the syntax is relatively easier to write.
- **How to Install locate Package**
- By default, Linux does not come with the **locate** command pre-installed. To get the package, run the following commands one after another:
- **sudo apt-get update**
- **sudo apt-get install mlocate**
- **syntax:**
- **locate [my-file]**
- **Search Exact File Name**
- The basic syntax only allows you to search for files that contain the search term. If you want to get the file with the exact name, you can use the **-r** option and add dollar symbol (**$**) at the end of your search term, for example:
- **locate -r my-file$**

Search contd.

- **Count the Number of Files**
- In order to tell how many files appear on your search result, insert **-c** after the locate command.
- **locate -c my-file** Instead of listing all the files, it will give you the total number of them.
- **Ignore Case Sensitive**
- Use **-i** on your linux **locate** command to ignore case sensitive files. For instance:
- **locate -i my-file** All of the files with this name will be shown, regardless of any uppercase or lowercase symbols found.
- **Show Existing Files**
- Linux **locate** command can even show you a deleted file if you haven't updated the database. Thankfully, you can get around this problem by using **-e** option, like this:
- **locate -e my-file** By doing this, you will only get files that exist at the time you perform the **locate** command.

Search contd.

- **Disables Errors While Searching**
- **-q** option will prevent any errors from showing up when the search is being processed. To do this, simply enter:
- **locate -q my-file**
- **Limit the Number of Search Results**
- **-n <number>** will do the trick. However, remember that you need to put the option at the end of the command line. Example:
- **locate my-file n 10**
- The script will only display the first 10 files it discovers even when there are more.
- **Conclusion**
- Use **find** to search for files based on name, type, time, size, ownership and permissions, in addition to some other useful options
- Install and use Linux **locate** command to perform faster system-wide searches for files. It also allows you to filter out by name, case-sensitive, folder, and so on.

Linux Archiving & Compression

- Archiving is the process of combining multiple files and directories (same or different sizes) into one file. On the other hand, compression is the process of reducing the size of a file or directory. Archiving is usually used as part of a system backup or when moving data from one system to another.
- The most common programs for archiving files and directories are:
- **Tar**
- **Zip**
- There are four main modes of operation in the tar utility:-
- c – Create an archive from a file or directory
- x – Extract archive
- r – Append file to archive
- t - List the contents of the archive

- **Features of Archiving**
- Data Compression
- Encryption
- File Concatenation
- Automatic Extraction
- Automatic Installation
- Source Volume and Media Information
- File Spanning
- Checksum
- Directory Structure Information
- Other Metadata (Data About Data)
- Error discovery

- **Area of Application**
- Store Computer Files System along with Metadata.
- Useful in transferring file locally.
- Useful in transferring file over web.
- Software Packaging Application.

- **1. tar Command**
- **tar** is the standard UNIX/Linux archiving application tool. In its early stage it used to be a Tape Archiving Program which gradually is developed into General Purpose archiving package which is capable of handling archive files of every kind. tar accepts a lot of archiving filter with options.
- **tar options**
- **-A** : Append tar files to existing archives.
- **-c** : Create a new archive file.
- **-d** : Compare archive with Specified file system.
- **-j** : bzip the archive
- **-r** : append files to existing archives.
- **-t** : list contents of existing archives.
- **-u** : Update archive
- **-x** : Extract file from existing archive.
- **-z** : gzip the archive
- **–delete** : Delete files from existing archive.

Archive contd.

- **Create a new archive**
- For this guide, I will use the name ire of the folder, which contains three different types of files.
- **$ ls ire/**
- **file.odt image.png song.mp3**
- Now, let us ire create a new tar archive directory.
- **$ tar cf ire.tar ire/**
- Here, the c flag refers to the creation of a new archive, where f is the specified archive file.
- Similarly, to create an archive of a set of files in the current working directory, use the following command:
- **$ tar cf archive.tar file1 file2 file 3**
- **Extract archive**
- To extract the archive in the current directory, just do the following:
- **$ tar xf ire.tar**

Archive contd.

- We can also use the C logo (capital letter C) to extract the archive to a different directory.
- For example, the following command will extract the archive to a Downloads directory.
- **$ tar xf ire.tar -C Downloads/**
- Or, go to the Downloads folder and something like the following extract the archive.
- **$ cd Downloads/**
- **$ tar xf ../ire.tar**
- Sometimes you may want to extract a particular type of file.
- For example, the following command extracts a file of type ".png".
- **$ tar xf ire.tar --wildcards "*.png"**

Archive contd.

- **Create compressed archives in gzip and bzip format**
- By default, tar creates an archive file to .tar the end. Further, tar the command may be compression utility gzip and bzip combination.
- The end of the file to .tar use ordinary tar extension to archive files tar.gz or .tgz end use gzip archived and compressed files, files tar.bz2 or .tbz end use bzip archiving and compression.

- First, let's create a gzip archive:
- **$ tar czf ire.tar.gz ire/**
- or:
- **$ tar czf ire.tgz ire/**
- Here, we use the z flag to use gzip compression method archive.

Archive contd.

- You can use v to see the progress when creating the archive flag.
- **$ tar czvf ire.tar.gz ire/**
- **ire/**
- **ire/file.odt**
- **ire/image.png**
- **ire/song.mp3**
- Here, it v refers to the progress of the display.
- Create a gzip archive from a list of files:
- **$ tar czf archive.tgz file1 file2 file3**
- To extract the gzip archive in the current directory, use:
- **$ tar xzf ire.tgz**
- To extract to a different folder, use the -C logo:
- **$ tar xzf ire.tgz -C Downloads/**

Archive contd.

- Now let's create a bzip archive . To do this, use the following **j** logo.
- Create an archive of the directory:
- **$ tar cjf ire.tar.bz2 ire/**
- or
- **$ tar cjf ire.tbz ire/**
- Create an archive from a list file:
- **$ tar cjf archive.tar.bz2 file1 file2 file3**
- or
- **$ tar cjf archive.tbz file1 file2 file3**
- In order to show progress, the use of v signs.
- Now, in the current directory, let's extract a bzip archive. this way:
- **$ tar xjf ire.tar.bz2**
- Or, extract the archive to another directory:
- **$ tar xjf ire.tar.bz2 -C Downloads**

Archive contd.

- **Create archives of multiple directories and/or files at one time**
- This is tar another of the coolest features command. To create a gzip archive of multiple directories or files at once, use the following files:
- **$ tar czvf ire.tgz Downloads/ Documents/ ire/file.odt**
- The above command to create Downloads, Documents catalog and ire directory file.odt archives, and archived in the current working directory.
- **Skip directories and/or files when creating an archive**
- This is very useful when backing up your data. You can exclude unimportant files or directories in the backup, this is the –exclude option that can help. For example, you want to create /home an archive directory, but do not want to include Downloads, Documents, Pictures, Music these directories.
- The approach:
- **$ tar czvf ire.tgz /home/sk --exclude=/home/sk/Downloads --exclude=/home/sk/Documents --exclude=/home/sk/Pictures --exclude=/home/sk/Music**
- The above command will create a gzip archive in your HOME directory , which does not include Downloads, Documents, Pictures and Music directories. To create a bzip archive, it will be z replaced j, and the use of extensions in the above example .bz2.

dmesg' Commands for Troubleshooting and Collecting Information of Linux Systems

- The '**dmesg**' command displays the messages from the kernel ring buffer. A system passes multiple runlevel from where we can get lot of information like system architecture, cpu, attached devices, RAM etc. When computer boots up, a kernel (core of an operating system) is loaded into memory. During that period number of messages are being displayed where we can see hardware devices detected by kernel.
- The messages are very important in terms of diagnosing purpose in case of device failure. When we connect or disconnect hardware device on the system, with the help of dmesg command we come to know detected or disconnected information on the fly.
- most famous tool called 'dmesg' command. The exact syntax of dmesg as follows.

 # dmseg [options...]

dmesg contd.

- **1. List all loaded Drivers in Kernel**
- We can use text-manipulation tools i.e. '**more**', '**tail**', '**less**' or '**grep**' with dmesg command. As output of dmesg log won't fit on a single page, using dmesg with **pipe more or less** command will display logs in a single page.
- **[root@arnood.com ~]# dmesg | more**
- **[root@arnood.com ~]# dmesg | less**
- **Sample Output**
- [0.000000] Initializing cgroup subsys cpuset
- [0.000000] Initializing cgroup subsys cpu
- [0.000000] Initializing cgroup subsys cpuacct
- [0.000000] Linux version 3.11.0-13-generic (buildd@aatxe) (gcc version 4.8.1 (Ubuntu/Linaro 4.8.1-10ubuntu8)) #20-Ubuntu SMP Wed Oct 23 17:26:33 UTC 2013
- (Ubuntu 3.11.0-13.20-generic 3.11.6)
- [0.000000] KERNEL supported cpus:
- [0.000000] Intel GenuineIntel

dmesg contd.

- **2. List all Detected Devices**
- To discover which hard disks has been detected by kernel, you can search for the keyword “sda” along with “grep” like shown below.
- **[root@arnood.com ~]# dmesg | grep sda**

- [1.280971] sd 2:0:0:0: [sda] 488281250 512-byte logical blocks: (250 GB/232 GiB)
- [1.281014] sd 2:0:0:0: [sda] Write Protect is off
- [1.281016] sd 2:0:0:0: [sda] Mode Sense: 00 3a 00 00
- [1.281039] sd 2:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
- [1.359585] sda: sda1 sda2 < sda5 sda6 sda7 sda8 >
- [1.360052] sd 2:0:0:0: [sda] Attached SCSI disk
- [2.347887] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null)

dmesg contd.

- **3. Print Only First 20 Lines of Output**
- The ‘head’ along with dmesg will show starting lines i.e. ‘dmesg | head -20’ will print only 20 lines from the starting point.

- **[root@arnood.com ~]# dmesg | head -20**

- [0.000000] Initializing cgroup subsys cpuset
- [0.000000] Initializing cgroup subsys cpu
- [0.000000] Initializing cgroup subsys cpuacct
- [0.000000] Linux version 3.11.0-13-generic (buildd@aatxe) (gcc version 4.8.1 (Ubuntu/Linaro 4.8.1-10ubuntu8)) #20-Ubuntu SMP Wed Oct 23 17:26:33 UTC 2013 (Ubuntu 3.11.0-13.20-generic 3.11.6)
- [0.000000] KERNEL supported cpus:
- [0.000000] Intel GenuineIntel
- [0.000000] AMD AuthenticAMD
- [0.000000] NSC Geode by NSC

dmesg contd.

- **4. Print Only Last 20 Lines of Output**
- The 'tail' along with dmesg command will print only 20 last lines, this is useful in case we insert removable device.
- **[root@arnood.com ~]# dmesg | tail -20**

- parport0: PC-style at 0x378, irq 7 [PCSPP,TRISTATE]
- ppdev: user-space parallel port driver
- EXT4-fs (sda1): mounted filesystem with ordered data mode
- Adding 2097144k swap on /dev/sda2. Priority:-1 extents:1 across:2097144k
- readahead-disable-service: delaying service auditd
- ip_tables: (C) 2000-2006 Netfilter Core Team
- nf_conntrack version 0.5.0 (16384 buckets, 65536 max)
- NET: Registered protocol family 10
- lo: Disabled Privacy Extensions
- e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
- Slow work thread pool: Starting up
- Slow work thread pool: Ready
- FS-Cache: Loaded

dmesg contd.

- **5. Search Detected Device or Particular String**
- It's difficult to search particular string due to length of dmesg output. So, filter the lines with are having string like 'usb' 'dma' 'tty' and 'memory' etc. The '-i' option instruct to grep command to ignore the case (upper or lower case letters).
- **[root@arnood.com log]# dmesg | grep -i usb**
- **[root@arnood.com log]# dmesg | grep -i dma**
- **[root@arnood.com log]# dmesg | grep -i tty**

- **[root@arnood.com log]# dmesg | grep -i memory**

- [0.000000] Scanning 1 areas for low **memory** corruption
- [0.000000] initial **memory** mapped: [mem 0x00000000-0x01ffffff]
- [0.000000] Base **memory** trampoline at [c009b000] 9b000 size 16384
- [0.000000] init_**memory**_mapping: [mem 0x00000000-0x000fffff]
- [0.000000] init_**memory**_mapping: [mem 0x37800000-0x379fffff]

dmesg contd.

- **6. Clear dmesg Buffer Logs**
- Yes, we can clear dmesg logs if required. It will clear dmesg ring buffer message logs immediatetly you executed the command below. Still you can view logs stored in '/var/log/dmesg' files. If you connect/disconnect any device will generate dmesg output.

- **[root@arnood.com log]# dmesg –c**
- **7. Monitoring dmesg in Real Time**
- Some distro allows command 'tail -f /var/log/dmesg' as well for real time dmesg monitoring.

- **[root@arnood.com log]# watch "dmesg | tail -20"**

Linux mount and umount

- The mount command mounts a storage device or filesystem, making it accessible and attaching it to an existing directory structure.

- The umount command "unmounts" a mounted filesystem, informing the system to complete any pending read or write operations, and safely detaching it. Description: mount
- All files accessible in Unix, or a Unix-style system such as Linux, are arranged in one big tree: the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command attaches a filesystem, located on some device or other, to the file tree. Conversely, the umount command will detach it again.

- The standard form of the mount command is:

- **mount -t type device dir**
- This tells the kernel to attach the filesystem found on device (which is of type type) at the directory dir. The previous contents (if any), owner, and mode of dir become invisible, and as long as this filesystem remains mounted, the pathname dir refers to the root of the filesystem on device.

mount contd.

- If only directory or device is given, for example:

- **mount /dir**

then mount looks for a corresponding mountpoint (and then, if not found, for a corresponding device) entry in the /etc/fstab file, and attempts to mount it.
Listing Mounts And Getting Help
Three forms of the mount command do not actually mount anything:

mount -h
prints a help message, and exits;

mount -V
prints mount's version information, and exits;

mount [-l] [-t type]
lists all mounted filesystems (of type type). The option -l adds labels to this listing.

mount contd

- SYNTAX
- **mount -a [-fFnrsvw] [-t vfstype]**
- **mount [-fnrsvw] [-o options [,...]] device | dir**
- **mount [-fnrsvw] [-t vfstype] [-o options] device dir**
- **mount [-hV**]
- OPTIONS

- -a Mount all filesystems (of the given types) mentioned
- in fstab.

- -F (Used in conjunction with -a.) Fork off a new
- incarnation of mount for each device. This will do
- the mounts on different devices or different NFS
- servers in parallel. This has the advantage that
- it is faster; also NFS timeouts go in parallel. A
- disadvantage is that the mounts are done in unde-
- fined order. Thus, you cannot use this option if
- you want to mount both /usr and /usr/spool.

mount contd

- -f Causes everything to be done except for the actual system call; if it's not obvious, this "fakes" mounting the file system. This option is useful in conjunction with the -v flag to determine what the mount command is trying to do. It can also be used to add entries for devices that were mounted earlier with the -n option.

- -n Mount without writing in /etc/mtab. This is necessary for example when /etc is on a read-only file system.

- -s Tolerate sloppy mount options rather than failing. This option exists for support of the Linux autofs-based automounter.

- -r Mount the file system read-only. A synonym is -o ro

- -w Mount the file system read/write. This is the default. A synonym is -o rw.

mount contd

- -o Several -o options can be specified in a comma separated string ... see info for more

- async All I/O to the file system should be done asynchronously.

- atime Update inode access time for each access. This is the default.

- noatime Do not update inode access times on this file system (e.g, for faster access on the news spool to speed up news servers).

- auto Can be mounted with the -a option.

- noauto Can only be mounted explicitly (i.e., the -a option will not cause the file system to be mounted).

- ro Mount the file system read-only.

- rw Mount the file system read-write.

- suid Allow set-user-identifier or set-group-identifier bits to take effect.

- sync All I/O to the file system should be done synchronously.

mount contd

- -L label
- Mount the partition that has the specified label.
- -U uuid
- Mount the partition that has the specified uuid.
- -t vfstype
- The argument following the -t is used to indicate
- the file system type.
- -h Print a help message.
- -V Output version.
- -v Verbose mode.
- **umount syntax**
- umount [-hV]
- umount -a [-dflnrv] [-t vfstype] [-O options]
- umount [-dflnrv] {dir|device}...

File system maintenance: fsck &df

- Filesystems are responsible for organizing how data is stored and recovered. One way or another, with time, filesystem may become corrupted and certain parts of it may not be accessible. If your filesystem develops such inconsistency it is recommend to verify its integrity.

- This can be completed via system utility called fsck (*file system consistency check*). This check can be done automatically during boot time or ran manually.
- **When to Use fsck in Linux**
- There are different scenarios when you will want to run fsck. Here are few examples:

- The system fails to boot.
- Files on the system become corrupt (often you may see input/output error).
- Attached drive (including flash drives/SD cards) is not working as expected.

fsck&df contd.

- fsck Available options
- Fsck command needs to be run with superuser privileges or root. You can use it with different arguments. Their usage depend on your specific case. Below you will see some of the more important options:

- -A – Used for checking all filesystems. The list is taken from /etc/fstab.
- -C – Show progress bar.
- -l – Locks the device to guarantee no other program will try to use the partition during the check.
- -M – Do not check mounted filesystems.
- -N – Only show what would be done – no actual changes are made.
- -P – If you want to check filesystems in parallel, including root.
- -R – Do not check root filesystem. This is useful only with '-A'.
- -r – Provide statistics for each device that is being checked.
- -T – Does not show the title.
- -t – Exclusively specify the filesystem types to be checked. Types can be comma separated list.
- -V – Provide description what is being done.

fsck&df contd.

- In order to run fsck, you will need to ensure that the partition you are going to check is not mounted. For the purpose, let's use second drive /dev/sdb and suppose, is mounted in /mnt.

- Here is what happens if we try to run fsck when the partition is mounted.
- **# fsck /dev/sdb**

- To avoid this unmount the partition using.
- **# umount /dev/sdb**
- Then **fsck** can be safely ran with.

fsck&df contd.

- **# fsck /dev/sdb**

- **Understanding fsck exit codes**
- After running fsck, it will return an exit code. These cods can be seen in fsck's manual by running:
- **# man fsck**
- **0 No errors**
- **1 Filesystem errors corrected**
- **2 System should be rebooted**
- **4 Filesystem errors left uncorrected**
- **8 Operational error**
- **16 Usage or syntax error**
- **32 Checking canceled by user request**
- **128 Shared-library error**

fsck&df contd.

- **Repair Linux Filesystem Errors**
- Sometimes more than one error can be found on a filesystem. In such cases you may want fsck to automatically attempt to correct the errors. This can be done with:

- **# fsck -y /dev/sdb**
- The -y flag, automatically "yes" to any prompts from fsck to correct an error.

- Similarly, you can run the same on all filesystems (**without root**):
- **$ fsck -AR -y** (-**A**)ll file system without (-**R**)oot

fsck&df contd.

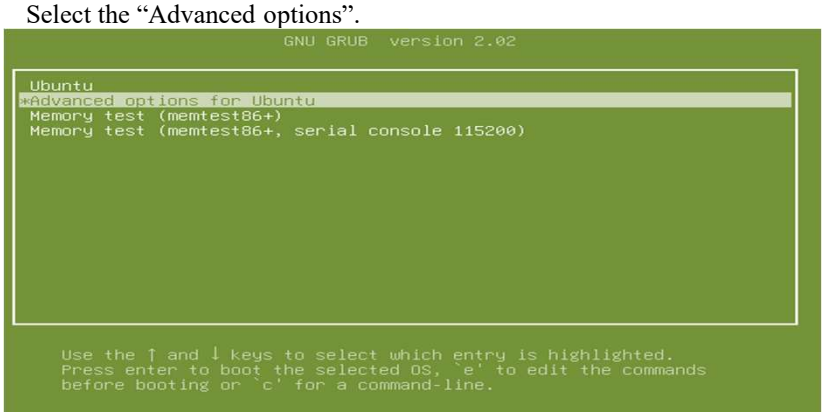
- **How to Run fsck on Linux Root Partition**

In some cases, you may need to run fsck on the root partition of your system. Since you cannot run fsck while the partition is mounted, you can try one of these options:

- Force fsck upon system boot
- Run fsck in rescue mode
- **Force fsck Upon System Boot**
- This is relatively easy to complete, the only thing you need to do is create a file called forcefsck in the root partition of your system. Use the following command:
- **# touch /forcefsck**
- Then you can simply force or schedule a reboot of your system. During the next bootup, the fsck will be performed. If downtime is critical, it is recommended to plan this carefully, since if there are many used inodes(an **inode** is a data structure on a traditional Unix-style **file system** such as ext3 or ext4. storing the properties of a **file** and directories) on your system, fsck may take some extra time.
- After your system boots, check if the file still exists:
- **# ls /forcefsck**
- If it does, remove it in order to avoid fsck on every system boot.

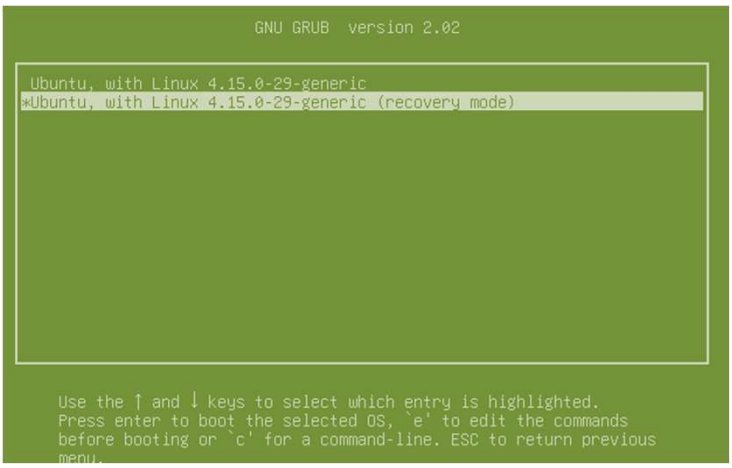
fsck&df contd.

- **Run fsck in Rescue Mode**
- Running fsck in rescue mode requires few more steps. First prepare your system for reboot. Stop any critical services like MySQL/MariaDB etc and then type.
- **# reboot**
- During the boot, hold down the **shift** key so that the grub menu is shown. Select the "Advanced options".



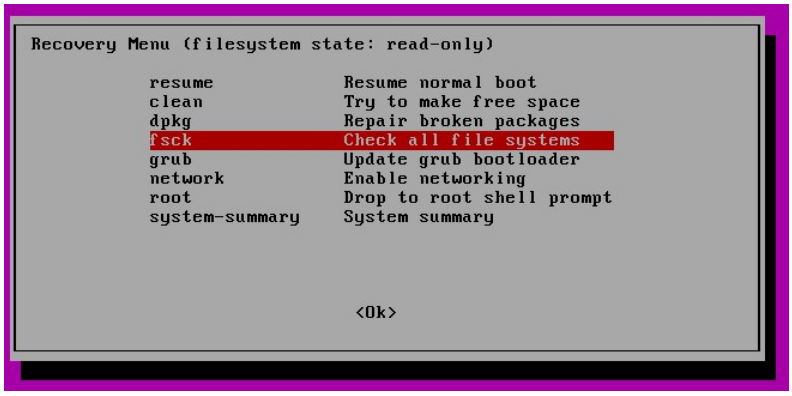
fsck&df contd.

- Then choose "**Recovery mode**".



fsck&df contd.

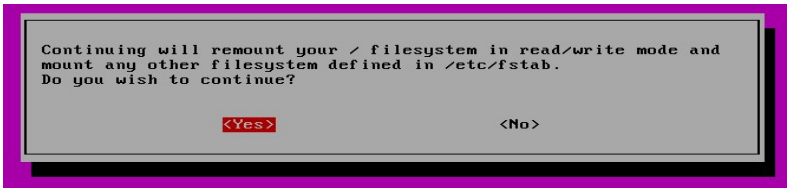
- In the next menu select "**fsck**".



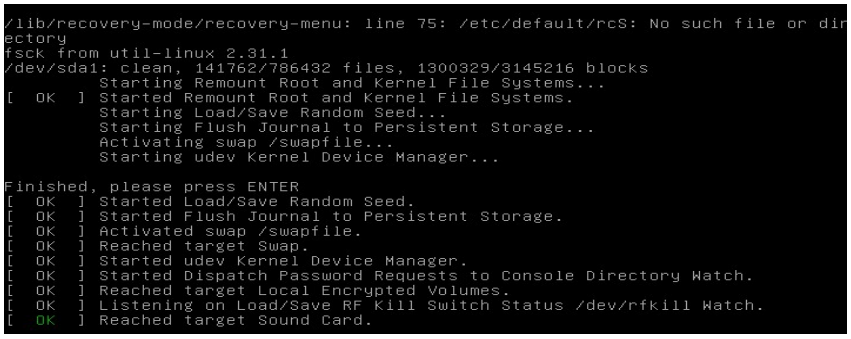
Select fsck Utility

fsck&df contd.

You will be asked if you wish to have your / filesystem remounted. Select "**yes**".

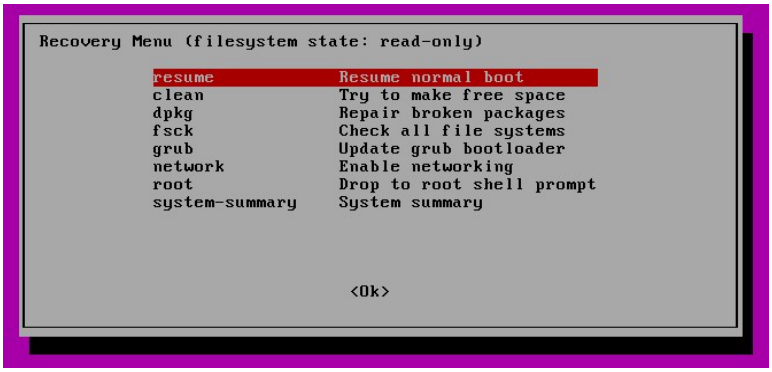


You should see something similar to this.



fsck&df contd

- You can then resume to normal boot, by selecting **"Resume"**.



Linux/Unix Process Management:
nohup, ps, kill, top, df, free, nice Commands

What is a Process?

- An instance of a program is called a Process. In simple terms, any command that you give to your Linux machine starts a new process.
- Having multiple processes for the same program is possible.
- Types of Processes:
- **Foreground Processes**: They run on the screen and need input from the user. For example Office Programs
- **Background Processes**: They run in the background and usually do not need user input. For example Antivirus.

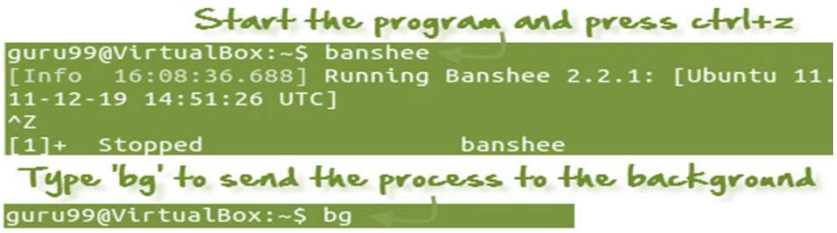
Process mngmt. Contd.

- Running a Foreground Process
- To start a foreground process, you can either run it from the dashboard, or you can run it from the terminal.
- When using the Terminal, you will have to wait, until the foreground process runs.

Process mngmt. Contd

- Running a Background process
- If you start a foreground program/process from the terminal, then you cannot work on the terminal, till the program is up and running.

- Particular, data-intensive tasks take lots of processing power and may even take hours to complete. You do not want your terminal to be held up for such a long time.
- To avoid such a situation, you can run the program and send it to the background so that terminal remains available to you.
  1. Stop the process by typing Ctrl+Z.
  2. Move the stopped process to the background by typing **bg**.



Process mngmt. Contd

- **Keep Background Processes Running After a Shell Exits**
- If your connection drops or you log out of the shell session, the background processes are terminated. There are several ways to keep the process running after the interactive shell session ends.
- to keep a process running after the shell exit is to use nohup.
- The nohup command executes another program specified as its argument and ignores all SIGHUP (hangup) signals. SIGHUP is a signal that is sent to a process when its controlling terminal is closed. To run a command in the background using the nohup command, type:
- **# nohup command &**
- The command output is redirected to the nohup.out file.
- nohup: ignoring input and appending output to 'nohup.out'
- If you log out or close the terminal, the process is not terminated.

Process mngmt. Contd

- Alternatives #
- There are a number of programs that allow you to have multiple interactive sessions at the same time.
- **Screen**
- Screen or GNU Screen is a terminal multiplexer program that allows you to start a screen session and open any number of windows (virtual terminals) inside that session. Processes running in Screen will continue to run when their window is not visible even if you get disconnected.
- **Tmux**
- Tmux is a modern alternative to GNU screen. With Tmux, you can also create a session and open multiple windows inside that session. Tmux sessions are persistent, which means that programs running in Tmux continue to run even if you close the terminal.
- **Conclusion**
- To run a command in the background, include & at the end of the command.
- When you run a command in the background, you don't have to wait until it finishes before you can execute another one.

Process mngmt. Contd

fg

- You can use the command "fg" to continue a program which was stopped and bring it to the foreground.

- The simple syntax for this utility is:
- **fg jobname**
- Example
- Launch 'banshee' music player
- Stop it with the '**ctrl +z**' command
- Continue it with the '**fg**' utility.

```
home@VirtualBox:~$ banshee
^Z
[1]+  Stopped                 banshee
home@VirtualBox:~$ fg banshee
banshee
[Info  00:36:19.400] Running Banshee 2.2.0: [Ubuntu oneiric
 (linux-gnu, i686) @ 2011-09-23 04:51:00 UTC]
```

process mngmt. Contd

Top

This utility tells the user about all the running processes on the Linux machine.

```
home@VirtualBox:~$ top

top - 23:57:43 up  2:54,  1 user,  load average: 0.00, 0.01, 0.05
Tasks: 189 total,   2 running, 187 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.7%us,  3.0%sy,  0.0%ni, 96.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1026080k total,   924508k used,   101572k free,    37000k buffers
Swap:  1046524k total,    21472k used,  1025052k free,   367996k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1525 home      20   0 1775m 100m  28m S  1.7 10.0   5:05.34 Photoshop.exe
  961 root      20   0 75972  51m 7952 R  1.0  5.1   2:23.42 Xorg
 1507 home      20   0  7644 4652  696 S  1.0  0.5   2:42.66 wineserver
 1564 home      20   0 75144  29m 9840 S  0.3  3.0   0:25.96 ubuntuone-syncd
 2999 home      20   0  127m  13m  10m S  0.3  1.4   0:01.36 gnome-terminal
 3077 home      20   0  2820 1188  864 R  0.3  0.1   0:00.76 top
    1 root      20   0  3200 1704 1260 S  0.0  0.2   0:00.98 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:00.95 ksoftirqd/0
```

process mngmt. Contd

Field	Description	Example 1	Example 2
PID	The process ID of each task	1525	961
User	The username of task owner	Home	Root
PR	Priority Can be 20(highest) or -20(lowest)	20	20
NI	The nice value of a task	0	0
VIRT	Virtual memory used (kb)	1775	75972
RES	Physical memory used (kb)	100	51
SHR	Shared memory used (kb)	28	7952

process mngmt. Contd

Field	Description	Example 1	Example 2
PID	The process ID of each task	1525	961
S	Status There are five types: 'D' = uninterruptible sleep 'R' = running 'S' = sleeping 'T' = traced or stopped 'Z' = zombie	S	R
%CPU	% of CPU time	1.7	1.0
%MEM	Physical memory used	10	5.1
TIME+	Total CPU time	5:05.34	2:23.42
Command	Command name	Photoshop.exe	Xorg

process mngmt. Contd

- **PS**
- This command stands for 'Process Status'. It is similar to the "Task Manager" that pop-ups in a Windows Machine when we use Cntrl+Alt+Del. This command is similar to 'top' command but the information displayed is different.

- To check all the processes running under a user, use the command –
- **# ps ux**

```
home@VirtualBox:~$ ps ux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
home      1114  0.0  0.8  46548  8512 ?        Ssl  Sep03   0:00 gnome-sess
home      1151  0.0  0.0   3856   140 ?        Ss   Sep03   0:00 /usr/bin/s
home      1154  0.0  0.0   3748   484 ?        S    Sep03   0:00 /usr/bin/d
home      1155  0.1  0.2   6656  3036 ?        Ss   Sep03   0:18 //bin/dbus
home      1157  0.0  0.2   9148  2368 ?        S    Sep03   0:00 /usr/lib/g
home      1162  0.0  0.2  31588  2296 ?        Ssl  Sep03   0:00 /usr/lib/g
home      1174  0.0  1.4 132472 14884 ?        Sl   Sep03   0:03 /usr/lib/g
```

process mngmt. Contd

- You can also check the process status of a single process, use the syntax -
- **# ps PID**

```
guru99@VirtualBox:~$ ps 1268
  PID TTY      STAT   TIME COMMAND
 1268 ?        S<l    0:02 /usr/bin/pulseaudio --start --log-target=syslog
```

process mngmt. contd

- **Kill**
- This command terminates running processes on a Linux machine.
- To use these utilities you need to know the PID (process id) of the process you want to kill
- Syntax -
- **kill PID**
- To find the PID of a process simply type
- **pidof *Process name***
- Let us try it with an example.

```
home@VirtualBox:~$ pidof Photoshop.exe
1525
home@VirtualBox:~$ kill 1525
```

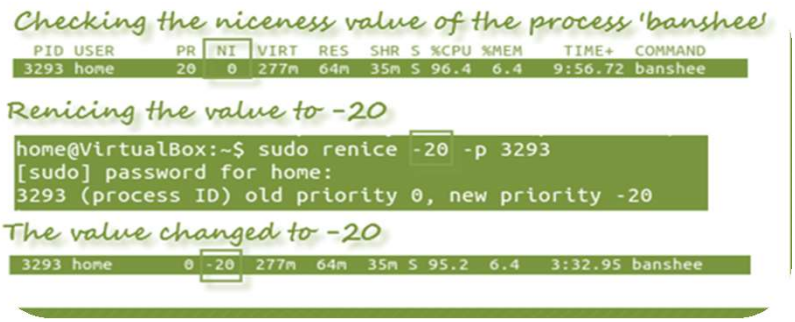
process mngmt. contd

- NICE
- Linux can run a lot of processes at a time, which can slow down the speed of some high priority processes and result in poor performance.
- To avoid this, you can tell your machine to prioritize processes as per your requirements.
- This priority is called Niceness in Linux, and it has a value between -20 to 19. The lower the Niceness index, the higher would be a priority given to that task.
- The default value of all the processes is 0.
- To start a process with a niceness value other than the default value use the following syntax
- nice -n 'Nice value' process name

```
home@VirtualBox:~$ nice -n 19 banshee
```

process mngmt. contd

- If there is some process already running on the system, then you can 'Renice' its value using syntax.
- **renice 'nice value' -p 'PID'**
- To change Niceness, you can use the 'top' command to determine the PID (process id) and its Nice value. Later use the renice command to change the value.
- Let us understand this by an example.
-



process mngmt. contd

- DF
- This utility reports the free disk space(Hard Disk) on all the file systems.

```
guru99@guru99-VirtualBox:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda1        7837756 2921376   4523216  40% /
udev              246488       4    246484   1% /dev
tmpfs             101512     752    100760   1% /run
none                5120       0      5120   0% /run/lock
none              253776      76    253700   1% /run/shm
```

- If you want the above information in a readable format, then use the command

- **# df -h**

```
guru99@guru99-VirtualBox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       7.5G  2.8G  4.4G  40% /
udev            241M  4.0K  241M   1% /dev
tmpfs           100M  752K   99M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            248M   76K  248M   1% /run/shm
```

process mngmt. contd

- **Free**
- This command shows the free and used memory (RAM) on the Linux system.

```
home@VirtualBox:~$ free
             total       used       free     shared    buffers     cached
Mem:       1026080     803604     222476          0      36312     343376
-/+ buffers/cache:     423916     602164
Swap:      1046524      35832    1010692
```

process mngmt. contd

- **Summary:**
- Any running program or a command given to a Linux system is called a process
- A process could run in foreground or background
- The priority index of a process is called Nice in Linux. Its default value is 0, and it can vary between 20 to -19
- The lower the Niceness index, the higher would be priority given to that task.

Command	Description
bg	To send a process to the background
fg	To run a stopped process in the foreground
top	Details on all Active Processes
ps	Give the status of processes running for a user
ps PID	Gives the status of a particular process
pidof	Gives the Process ID (PID) of a process
kill PID	Kills a process
nice	Starts a process with a given priority
renice	Changes priority of an already running process
df	Gives free hard disk space on your system
free	Gives free RAM on your system

Check the System Load on Linux

- If the demands being placed on a running program cause it to request excessive resources from your server this can lead to poor performance and system instability. First let's check the load on your server using the uptime command.

$ uptime

- 15:16:45 up 41 days, 2:35, 2 users, load average: 0.01, 3.01, 2.70

The example shows the output from uptime. When the command was run at 15:16:45, the server had been up for 41 days 2 hours and 35 minutes, there were two users logged on and the load averages were 0.01, 3.01 and 2.70.

- The load average represents the work being done by the system. The three numbers show the load averages for the **last minute, 5 minutes and 15 minutes**, respectively. A load average of 1 reflects the full workload of a single processor on the system. A load of 2 on a system with two CPUs means that those CPUs were working at maximum. On a system with four CPUs, that 2 reflects a workload using about half of the available processing power.

- You can check the number of CPUs available to your instance by running the following command:
- **grep processor /proc/cpuinfo | wc -l :35**

System Load contd

- the **/proc** directory is **NOT** a real **File System**, in the sense of the term. It is a **Virtual File System**. Contained within the **procfs** are information about processes and other system information. It is mapped to **/proc** and mounted at **boot** time. the **/proc** directory is **NOT** a real **File System**, in the sense of the term. It is a **Virtual File System**. Contained within the **procfs** are information about processes and other system information. It is mapped to **/proc** and mounted at **boot** time.
- **# cd /proc**
- The first thing that you will notice is that there are some **familiar sounding files**, **numbered directories** etc. The **numbered directories** represent **processes**, better known as **PIDs**, and within them, a command that occupies them. The files contain system information such as **memory** (**meminfo**), **CPU information** (**cpuinfo**), and available **filesystems**.

System Load contd

- To view the memory information of your system,
- **# cat /proc/meminfo**
- MemTotal: 604340 kB
- MemFree: 54240 kB
- Buffers: 18700 kB
- Cached: 369020 kB
- SwapCached: 0 kB
- Active: 312556 kB
- Inactive: 164856 kB
- Active(anon): 89744 kB
- Inactive(anon): 360 kB
- Active(file): 222812 kB
- Inactive(file): 164496 kB
- Unevictable: 0 kB
- Mlocked: 0 kB
- As you can see, **/proc/meminfo** contains a bunch of information about your system's memory, including the total amount available (in **kb**) and the amount free on the top two lines.

System Load contd

- **Few proc files**

- **/proc/cmdline** – Kernel command line information.
- **/proc/console** – Information about current consoles including tty.
- **/proc/devices** – Device drivers currently configured for the running kernel.
- **/proc/dma** – Info about current DMA channels.
- **/proc/fb** – Framebuffer devices.
- **/proc/filesystem**s – Current filesystems supported by the kernel.
- **/proc/iomem** – Current system memory map for devices.
- **/proc/ioports** – Registered port regions for input output communication with device.
- **/proc/loadavg** – System load average.
- **/proc/locks** – Files currently locked by kernel.
- **/proc/meminfo** – Info about system memory (see above example).
- **/proc/misc** – Miscellaneous drivers registered for miscellaneous major device.
- **/proc/modules** – Currently loaded kernel modules.
- **/proc/mounts** – List of all mounts in use by system.
- **/proc/partitions** – Detailed info about partitions available to the system.
- **/proc/pci** – Information about every PCI device.
- **/proc/stat** – Record or various statistics kept from last reboot.
- **/proc/swap** – Information about swap space.
- **/proc/uptime** – Uptime information (in seconds).
- **/proc/version** – Kernel version, gcc version, and Linux distribution installed.

- Within **/proc's** numbered directories you will find a few **files** and **links**. Remember that these directories' numbers correlate to the **PID** of the command being run within them.

System Load contd

- In any numbered directory, you will have a similar file structure. The most important ones, and their descriptions, are as follows:

- cmdline – command line of the process
- environ – environmental variables
- fd – file descriptors
- limits – contains information about the limits of the process
- mounts – related information
- You will also notice a number of links in the numbered directory:

- cwd – a link to the current working directory of the process
- exe – link to the executable of the process
- root – link to the work directory of the process

System Load contd.

- **vmstat**
- The amount of memory a system has is one of the most common restraining factors. The swap is an area of the hard drive where data is moved to free up physical memory (RAM) for a process to use (not all servers have swap space configured). A system using its swap area does not necessarily mean it is low on memory, but if most of your system's swap is being consumed it could indicate that your server is trying to do more than its available memory permits.
- If swap space is configured and you suspect your server is running out of standard memory, you can use vmstat to show how much swapping is occurring.
- The following example displays a system's virtual memory statistics:

- **$ vmstat**

```
geek:~$ vmstat
-------memory---------- ---swap-- -----io---- -system-- ------cpu-----
pd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
88  92448  81296 692400    0    3   351   312  128  544  3  2 84 11  0
geek:~$
```

System Load contd.

```
geek:~$ vmstat
-------memory---------- ---swap-- -----io---- -system-- ------cpu-----
pd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
88  92448  81296 692400    0    3   351   312  128  544  3  2 84 11  0
geek:~$
```

- In the example the free column shows that the amount of free memory is only around 92MB.
- The **si** and **so** columns show the amount of data being transferred between the system's swap and its memory. **so** is the amount of data being moved to the swap to free up memory. **si** is the amount of data being pulled from the swap back in to memory. This happens when the data is required for a process to run.
- When a server is constantly swapping into and out of memory it is an indication that the load it is being placed under is too great for the resources it has available. **top** can be used to help identify the processes that are consuming the most resources.

System Load contd.

- Summary
- If top and vmstat indicate that the server is using all its resources you need to look at optimising your current set up;
- This can include running any processing jobs outside peak hours, killing any processes no longer required and reconfiguring processes so that they require less resources.
- You may also want to consider increasing the size of your server to better match your requirements.

Linux package management :YUM and RPM

- Package management is a method of installing, updating, removing, and keeping track of software updates from specific repositories (repos) in the Linux system. Linux distros often use different package management tools. Red Hat-based distros use RPM (RPM Package Manager) and YUM/DNF (Yellow Dog Updater, Modified/Dandified YUM).
- YUM is the primary package management tool for installing, updating, removing, and managing software packages in Red Hat Enterprise Linux.
- YUM performs dependency resolution when installing, updating, and removing software packages.
- YUM can manage packages from installed repositories in the system or from **.rpm** packages. The main configuration file for YUM is at **/etc/yum.conf**, and all the repos are at **/etc/yum.repos.d** .

package management

- **# yum -option command**
- There are many options and commands available to use with YUM. some commonly-used commands for YUM are listed below:

Command	Purpose
yum install	Installs the specified packages
remove	Removes the specified packages
search	Searches package metadata for keywords
info	Lists description
update	Updates each package to the latest version
repolist	Lists repositories
history	Displays what has happened in past transactions

Options	Purpose
-C	Runs from system cache
--security	Includes packages that provide a fix for a security issue
-y	Answers yes to all questions
--skip-broken	Skips packages causing problems
-v	Verbose

package management

- **Install a Package with YUM**
- To install a package called **Firefox 14**, just run the below command it will automatically find and install all required dependencies for Firefox
- **# yum install firefox**

```
Loaded plugins: fastestmirror
Dependencies Resolved

================================================================================
 Package              Arch       Version                Repository        Size
================================================================================
Updating:
firefox               i686       10.0.6-1.el6.centos    updates           20 M
Updating for dependencies:
 xulrunner            i686       10.0.6-1.el6.centos    updates           12 M

Transaction Summary
================================================================================
Install       0 Package(s)
Upgrade       2 Package(s)

Total download size: 32 M
Is this ok [y/N]: y
Downloading Packages:
(1/2): firefox-10.0.6-1.el6.centos.i686.rpm                | 20 MB   01:10
(2/2): xulrunner-10.0.6-1.el6.centos.i686.rpm              | 12 MB   00:52
--------------------------------------------------------------------------------
Total                                          63 kB/s | 32 MB   02:04

Updated:
  firefox.i686 0:10.0.6-1.el6.centos

Dependency Updated:
  xulrunner.i686 0:10.0.6-1.el6.centos

Complete!.
```

- The above command will ask confirmation before installing any package on your system. If you want to install packages automatically without asking any confirmation, use option **-y** as shown in below example.
- **# yum -y install firefox**

package management

- **Search for a Package using YUM**
- If you don't remember the exact name of the package, then use **search** function to search all the available packages to match the name of the package you specified. For example, to search all the packages that matches the word .
- **# yum search vsftpd**

```
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.neu.edu.cn
 * epel: mirror.neu.edu.cn
 * extras: mirror.neu.edu.cn
 * rpmforge: mirror.nl.leaseweb.net
 * updates: ftp.iitm.ac.in
============================== Matched: vsftpd
==============================
ccze.i386 : A robust log colorizer
pure-ftpd-selinux.i386 : SELinux support for Pure-FTPD
vsftpd.i386 : vsftpd - Very Secure Ftp Daemon
```

List all Installed Packages using YUM
To list all the installed packages on a system, just issue below command, it will display all the installed packages.
 # yum list installed | less

package management

- **RPM (RPM Package Manager)**
- RPM is a popular package management tool in Red Hat Enterprise Linux-based distros.
- Using RPM, you can install, uninstall, and query individual software packages. Still, it cannot manage dependency resolution like YUM. RPM does provide you useful output, including a list of required packages.
- An RPM package consists of an archive of files and metadata. Metadata includes helper scripts, file attributes(r, w, x), and information about packages.
- RPM has some basic modes: query, verify, install, upgrade, erase, show querytags, show configuration. At least one of these modes needs to be selected to perform package management tasks.
- Every mode has its own set of options. For example, install mode i has its own set of installation options. Options for the modes are found on the RPM man pages at man rpm.

package management

- RPM maintains a database of installed packages, which enables powerful and fast queries. The RPM database is inside /var/lib, and the file is named __db*.
- **# file /var/lib/rpm/***
- /var/lib/rpm/Basenames: Berkeley DB (Hash, version 7, native byte-order)
- /var/lib/rpm/Conflictname: Berkeley DB (Hash, version 7, native byte-order)
- /var/lib/rpm/__db.001: data
- /var/lib/rpm/__db.002: X11 SNF font data, LSB first
- /var/lib/rpm/__db.003: X11 SNF font data, LSB first
- /var/lib/rpm/Dirnames: Berkeley DB (Btree, version 8, native byte-order)
- /var/lib/rpm/Filemd5s: Berkeley DB (Btree, version 8, native byte-order)
- /var/lib/rpm/Group: Berkeley DB (Hash, version 7, native byte-order)
- /var/lib/rpm/Installtid: Berkeley DB (Btree, version 8, native byte-order)
- /var/lib/rpm/Name: Berkeley DB (Hash, version 7, native byte-order)

•To **list all the files** in an RPM package, combine the -q, -p, and -l options (example output truncated):

•$ rpm -qpl qt-4.6.2-17.fc12.x86_64.rpm

•/etc/Trolltech.conf
•/usr/bin/qdbus
•/usr/lib64/libQtCore.so.4
•/usr/lib64/libQtCore.so.4.6
•/usr/lib64/libQtCore.so.4.6.2
•/usr/lib64/libQtDBus.so.4
•/usr/lib64/libQtDBus.so.4.6

package management

Some commonly-used modes are listed below:

Mode	Description
-i	Installs a package
-U	Upgrades a package
-e	Erases a package
-V	Verifies a package
-q	Queries a package

Here are some commonly-used general options:

General options	Purpose
-? | --help	Prints help
--version	Prints version number
-v	Prints verbose output

Example:
#rpm -i package-file
#rpm -U package-file
#rpm -ivh package-file
The flag **-i** is for install, **U** is for upgrade, **v** for verbose, **h** for hash (this option displays the # as a progress bar for the operation). **v** and **h** are optional flags.

package management

- **# sudo rpm -ivh vim-enhanced-8.0.1763-13.el8.x86_64.rpm**
- The flag -i is for install, U is for upgrade, v for verbose, h for hash (this option displays the # as a progress bar for the operation). In this example, v and h are optional flags.

```
[kc@localhost ~]$ sudo rpm -ivh vim-enhanced-8.0.1763-13.el8.x86_64.r
pm
Verifying...
################################# [100%]
Preparing...
################################# [100%]
Updating / installing...
   1:vim-enhanced-2:8.0.1763-13.el8
################################# [100%]
[kc@localhost ~]$
```

To query for a package using RPM issue following command:

syntax : rpm -q query-options package

 # rpm -qa vim-enhanced

```
[kc@localhost ~]$ rpm -qa vim-enhanced
vim-enhanced-8.0.1763-13.el8.x86_64
[kc@localhost ~]$
```

package management

- To erase a package, use the following command:
- Syntax: rpm -e erase-options package-name
- **#rpm -evh vim-enhanced**

```
[kc@localhost ~]$ sudo rpm -evh vim-enhanced
Preparing...
################################# [100%]
Cleaning up / removing...
   1:vim-enhanced-2:8.0.1763-13.el8
################################# [100%]
[kc@localhost ~]$
```

Querying Package Files Remotely

Using RPM , you can access RPM package files over a network using FTP or HTTP connections.

syntax:

- ***rpm -qp ftp://username:password@hostname:port/path/to/rpm/file***
- ***rpm -qp http://hostname:port/path/to/rpm/file***

package management

If your system resides behind a firewall with a proxy server, use the options in the following table to name the proxy. Note that these proxy options only work with the TIS Firewall toolkit (The free firewalls toolkit (TIS FWTK), from **Trusted Information Systems**, includes a number of proxy servers of various types. TIS FWTK also provides a number of other tools for authentication and other purposes).

Network Proxy Option	Meaning
--ftpproxy proxy_hostname	Names the proxy system
--ftpport proxy_port_number	Network port number on the proxy system
--httpproxy proxy_hostname	Names the proxy system
--httpport proxy_port_number	Network port number on the proxy system

Network interface management

- A system administrator's routine tasks include configuring, maintaining, troubleshooting. Linux networking commands are used extensively to inspect, analyze, maintain, and troubleshoot the network/s connected to the system.

- Let us first know the list of the basic networking commands used in Linux followed by a detailed explanation of each.

1. ifconfig
2. ip
3. traceroute
4. tracepath
5. ping
6. netstat
7. ss
8. dig
9. nslookup
10. route
11. host
12. arp
13. iwconfig
14. hostname
15. curl or wget
16. mtr
17. whois
18. ifplugstatus
19. iftop
20. tcpdump

Network management

- 1**. ifconfig**
- Linux ifconfig stands for interface configurator. It is one of the most basic commands used in network inspection.

- ifconfig is used to initialize an interface, configure it with an IP address, and enable or disable it. It is also used to display the route and the network interface.

- Basic information displayed upon using ifconfig are:

  - ❖ IP address
  - ❖ MAC address
  - ❖ MTU(Maximum Transmission Unit)

To get all the details using ifconfig

- Syntax:
- **#Ifconfig**
- Output picture will show the IP address of 3 networks, Ethernet, local network, and WLAN.

Network management

- Using following command, you can get details of a specific interface as shown below:
- **# ifconfig eth0**
- **# ifconfig lo**
- **# ifconfig wlan0 (wlan0** is your wifi card. wlan is wireless lan and 0 is the number of your card. The count starts from 0 and goes up (0,1,2,3,etc..).
- **eth0** is the first Ethernet interface. (Additional Ethernet interfaces would be named eth1, eth2, etc.) This type of interface is usually a **N**etwork **I**nterface **C**ard(NIC) connected to the network by a category 5/6 cable. **lo** is the loopback interface and is a special network interface that the system uses to communicate with it
- **# ifconfig eth0**

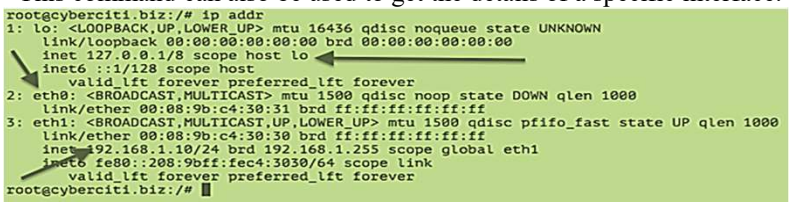
- eth0 Link encap:Ethernet HWaddr 00:0C:29:28:FD:4C
- inet addr:192.168.50.2 Bcast:192.168.50.255 Mask:255.255.255.0
- inet6 addr: fe80::20c:29ff:fe28:fd4c/64 Scope:Link
- UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
- RX packets:6119 errors:0 dropped:0 overruns:0 frame:0
- TX packets:4841 errors:0 dropped:0 overruns:0 carrier:0
- collisions:0 txqueuelen:1000
- RX bytes:6127464 (5.8 MiB) TX bytes:539648 (527.0 KiB)
- Interrupt:18 Base address:0x2000self.

Network management

- **To assign an IP address and Gateway to an interface**
- Following command can be used to assign an IP address and Gateway to an interface. However, these details will be reset after the system reboot.
- **ifconfig eth0 <address> netmask <address>**
- **To enable an interface**
- **Syntax: ifup eth0**
- **To disable an interface**
- **Syntax: ifdown eth0**
- **To set the size of MTU**
- Maximum transmission unit (MTU) determines the maximum payload size of a packet that is sent.By default, MTU has a size of 1500. This can be however set externally by using ifconfig.
- **Syntax: Ifconfig eth0 mtu xxxx**

Network management

- **2. ip**
- This is the latest and updated version of ifconfig command.
- **Syntax:**
- ip a
- ip addr
- This command gives the details of all networks like ifconfig.
- This command can also be used to get the details of a specific interface.



- **Commands to get details are:**
- **Syntax:**
- **ip a show eth0**
- **ip a show lo**
- **ip a show wlan0**

Network management

- **ARP** stands for Address Resolution Protocol, which is used to find the address of a network neighbor for a given IPv4 address.
- **Display neighbour/arp cache**
- The syntax is:
  ip n show
  ip neigh show

```
74.xx.yy.zz dev eth1 lladdr 00:30:48:yy:zz:ww REACHABLE
10.10.29.66 dev eth0 lladdr 00:30:48:c6:0a:d8 REACHABLE
74.ww.yyy.xxx dev eth1 lladdr 00:1a:30:yy:zz:ww REACHABLE
10.10.29.68 dev eth0 lladdr 00:30:48:33:bc:32 REACHABLE
74.fff.uu.cc dev eth1 lladdr 00:30:48:yy:zz:ww STALE
74.rr.ww.fff dev eth1 lladdr 00:30:48:yy:zz:ww DELAY
10.10.29.65 dev eth0 lladdr 00:1a:30:38:a8:00 REACHABLE
10.10.29.74 dev eth0 lladdr 00:30:48:8e:31:ac REACHABLE
```

- The last field show the the state of the "**neighbour unreachability detection**" machine for this entry:
- **STALE** – The neighbour is valid, but is probably already unreachable, so the kernel will try to check it at the first transmission.
- **DELAY** – A packet has been sent to the stale neighbour and the kernel is waiting for confirmation.
- **REACHABLE** – The neighbour is valid and apparently reachable.

Network management

- **Add a new ARP entry**
- **Syntax** :
  ip neigh add {IP-HERE} lladdr {MAC/LLADDRESS} dev {DEVICE} nud {STATE}
- Example: add a permanent ARP entry for the neighbour 192.168.1.5 on the device eth0:
  ip neigh add 192.168.1.5 lladdr 00:1a:30:38:a8:00 dev eth0 nud perm

Neighbour state (nud)	Meaning
permanent	The neighbour entry is valid forever and can be only be removed administratively
noarp	The neighbour entry is valid. No attempts to validate this entry will be made but it can be removed when its lifetime expires.
stale	The neighbour entry is valid but suspicious. This option to ip neigh does not change the neighbour state if it was valid and the address is not changed by this command.
reachable	The neighbour entry is valid until the reachability timeout expires.

Network management

- **PING Command**
- PING (Packet INternet Groper) command is the best way to test connectivity between two nodes. Whether it is Local Area Network (LAN) or Wide Area Network (WAN). Ping use ICMP (Internet Control Message Protocol) to communicate to other devices. You can ping host name of ip address using below command.
- **# ping 4.2.2.2**

- PING 4.2.2.2 (4.2.2.2) 56(84) bytes of data.
- 64 bytes from 4.2.2.2: icmp_seq=1 ttl=44 time=203 ms
- 64 bytes from 4.2.2.2: icmp_seq=2 ttl=44 time=201 ms
- 64 bytes from 4.2.2.2: icmp_seq=3 ttl=44 time=201 ms

- OR

- **# ping www.yahoo.com**

- PING yahoo.com (50.116.66.136) 56(84) bytes of data.
- 64 bytes from 50.116.66.136: icmp_seq=1 ttl=47 time=284 ms
- 64 bytes from 50.116.66.136: icmp_seq=2 ttl=47 time=287 ms
- 64 bytes from 50.116.66.136: icmp_seq=3 ttl=47 time=285 ms

Network management

- **# traceroute 4.2.2.2**
- **traceroute to 4.2.2.2 (4.2.2.2), 30 hops max, 60 byte packets**
- **1 192.168.50.1 (192.168.50.1) 0.217 ms 0.624 ms 0.133 ms**
- **2 227.18.106.27.mysipl.com (27.106.18.227) 2.343 ms 1.910 ms 1.799 ms**
- **3 221-231-119-111.mysipl.com (111.119.231.221) 4.334 ms 4.001 ms 5.619 ms**
- **4 10.0.0.5 (10.0.0.5) 5.386 ms 6.490 ms 6.224 ms**
- **5 gi0-0-0.dgw1.bom2.pacific.net.in (203.123.129.25) 7.798 ms 7.614 ms 7.378 ms**
- **6 115.113.165.49.static-mumbai.vsnl.net.in (115.113.165.49) 10.852 ms 5.389 ms 4.322 ms**
- **7 ix-0-100.tcore1.MLV-Mumbai.as6453.net (180.87.38.5) 5.836 ms 5.590 ms 5.503 ms**
- **8 if-9-5.tcore1.WYN-Marseille.as6453.net (80.231.217.17) 216.909 ms 198.864 ms 201.737 ms**
- **9 if-2-2.tcore2.WYN-Marseille.as6453.net (80.231.217.2) 203.305 ms 203.141 ms 202.888 ms**
- **10 if-5-2.tcore1.WV6-Madrid.as6453.net (80.231.200.6) 200.552 ms 202.463 ms 202.222 ms**
- **11 if-8-2.tcore2.SV8-Highbridge.as6453.net (80.231.91.26) 205.446 ms 215.885 ms 202.867 ms**
- **12 if-2-2.tcore1.SV8-Highbridge.as6453.net (80.231.139.2) 202.675 ms 201.540 ms 203.972 ms**
- **13 if-6-2.tcore1.NJY-Newark.as6453.net (80.231.138.18) 203.732 ms 203.496 ms 202.951 ms**
- **14 if-2-2.tcore2.NJY-Newark.as6453.net (66.198.70.2) 203.858 ms 203.373 ms 203.208 ms**
- **15 66.198.111.26 (66.198.111.26) 201.093 ms 63.243.128.25 (63.243.128.25) 206.597 ms 66.198.111.26 (66.198.111.26) 204.178 ms**
- **16 ae9.edge1.NewYork.Level3.net (4.68.62.185) 205.960 ms 205.740 ms 205.487 ms**
- **17 vlan51.ebr1.NewYork2.Level3.net (4.69.138.222) 203.867 ms vlan52.ebr2.NewYork2.Level3.net (4.69.138.254) 202.850 ms vlan51.ebr1.NewYork2.Level3.net (4.69.138.222) 202.351 ms**
- **18 ae-6-6.ebr2.NewYork1.Level3.net (4.69.141.21) 201.771 ms 201.185 ms 201.120 ms**
- **19 ae-81-81.csw3.NewYork1.Level3.net (4.69.134.74) 202.407 ms 201.479 ms ae-92-92.csw4.NewYork1.Level3.net (4.69.148.46) 208.145 ms**
- **20 ae-2-70.edge2.NewYork1.Level3.net (4.69.155.80) 200.572 ms ae-4-90.edge2.NewYork1.Level3.net (4.69.155.208) 200.402 ms ae-1-60.edge2.NewYork1.Level3.net (4.69.155.16) 203.573 ms**
- **21 b.resolvers.Level3.net (4.2.2.2) 199.725 ms 199.190 ms 202.488 ms**

Network management

- **tcpdump**

  tcpdump is a *packet sniffing tool* and can be of great help when resolving network issues. It listens to the network traffic and prints packet information based on the criteria you define.

  For example, you can examine all packets sent to or from a particular host, Ubuntu18 in this example:

- **$ sudo tcpdump host ubuntu18 -n -c 5**
- tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
- listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
- 14:12:11.509092 IP 10.0.0.4.22 > 183.83.208.234.9633: Flags [P.], seq 2991049004:2991049112, ack 2956233368, win 501, options [nop,nop,TS val 292041322 ecr 405604219], length 108
- 14:12:11.509146 IP 10.0.0.4.22 > 183.83.208.234.9633: Flags [P.], seq 108:252, ack 1, win 501, options [nop,nop,TS val 292041322 ecr 405604219], length 144
- 14:12:11.509218 IP 10.0.0.4.22 > 183.83.208.234.9633: Flags [P.], seq 252:288, ack 1, win 501, options [nop,nop,TS val 292041322 ecr 405604219], length 36
- 14:12:11.509259 IP 10.0.0.4.22 > 183.83.208.234.9633: Flags [P.], seq 288:500, ack 1, win 501, options [nop,nop,TS val 292041322 ecr 405604219], length 212
- 14:12:11.509331 IP 10.0.0.4.22 > 183.83.208.234.9633: Flags [P.], seq 500:768, ack 1, win 501, options [nop,nop,TS val 292041322 ecr 405604219], length 268
- 5 packets captured
- 6 packets received by filter
- 0 packets dropped by kernel

Network management

- **NMAP**
- The Nmap tool offers various methods to scan a system. In this example, lets perform a scan using hostname as server2 to find out all open ports, services and MAC address on the system.
- **[root@server1 ~]# nmap**
- Starting Nmap 4.11 (http://www.insecure.org/nmap/) at 2013-11-11 15:42 EST
- Interesting ports on server2 (192.168.0.101):
- Not shown: 1674 closed ports
- PORT STATE SERVICE
- 22/tcp open ssh
- 80/tcp open http
- 111/tcp open rpcbind
- 957/tcp open unknown
- 3306/tcp open mysql
- 8888/tcp open sun-answerbook
- MAC Address: 08:00:27:D9:8E:D7 (Cadmus Computer Systems)

- Nmap finished: 1 IP address (1 host up) scanned in 0.415 seconds
- You have new mail in /var/spool/mail/root

Network management

- **netstat** (network statistics) is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc.
- Update: The Linux netstat command is replaced by new **ss** command, which is capable of displaying more information about network connections and it is much faster than the older netstat command.
- **Using the Netstat Command to Display the Routing Table**
- Any device on a network needs to decide where to route the data packets. The routing table contains information to make these decisions. To acquire the contents of the routing table in numerics, we use the following command option:
- **# netstat –rn**

```
aprataksh@aprataksh-HP-Notebook:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.43.1    0.0.0.0         UG        0 0          0 wlp19s0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 wlp19s0
192.168.43.0    0.0.0.0         255.255.255.0   U         0 0          0 wlp19s0
```

The Internet Assigned Numbers Authority (IANA) has assigned port numbers to commonly used services like SSH, FTP, HTTP, HTTPS, and others. Here are some of the most common ones:

Port Number	Usage
20	File Transfer Protocol (FTP) Data Transfer
21	File Transfer Protocol (FTP) Command Control
22	Secure Shell (SSH)
23	Telnet - Remote login service, unencrypted text messages
25	Simple Mail Transfer Protocol (SMTP) E-mail Routing
53	Domain Name System (DNS) service
80	Hypertext Transfer Protocol (HTTP) used in World Wide Web
110	Post Office Protocol (POP3) used by e-mail clients to retrieve e-mail from a server
119	Network News Transfer Protocol (NNTP)
123	Network Time Protocol (NTP)
143	Internet Message Access Protocol (IMAP) Management of Digital Mail
161	Simple Network Management Protocol (SNMP)
194	Internet Relay Chat (IRC)

FTP server installation &settings

- FTP stands for File Transfer Protocol. It was written by **Abhay Bhushan** and published in 1971. FTP is supported by all the operating systems and browsers.
- It is a client-server based protocol.
- **How FTP Works**
- Step a: Client connects to server on port 21.
- Step b: Server responds and ask for authentication.
- Step c: Client decides weather to connect passively or actively and authenticate with credentials(user name password).
- Step d: If it is an active connection, server opens port 20 for data transfer and gives ftp prompt after successful authentication.
- **As a linux Administrator you should know**
- FTP stand for File Transfer Protocol.
- FTP does not require to login directly into the remote host
- FTP transfer data without encryption
- vsftpd is the only stand-alone FTP distributed With RHEL
- vsftpd stand for Very Secure FTP Daemon and is secure, fast and stable version of FTP
- vsftpd efficiently handle large numbers of connection securely
- one should use SFTP instead of FTP while transferring data over public network like Internet

FTP server

Installing FTP Server In Centos

Step 1: We will use below host name and IP address for our test machine to setup FTP server

Server IP: 192.168.0.9

Host Name: ftp.linux

Just edit file /etc/hosts

#vi /etc/hosts

and add the line on bottom and save

192.168.0.9 ftp.linux

vsftpd package is required for FTP Server. Check whether package is installed or not. If package is missing install it first.

Step 2: Install vsftpd (very secure FTP daemon) package.

Either by **#yum install vsftpd ftp** or by **#rpm -qa vsftpd***

```
[root@server ~]# rpm -qa vsftpd*
vsftpd-2.2.2-6.el6_0.1.x86_64
[root@server ~]# _
```

Configure vsftpd service to start at boot

```
[root@server ~]# chkconfig vsftpd on
[root@server ~]# _
```

FTP server

Configuring FTP Server In Linux Centos

- Current status of vsftpd service must be running. Start if it is stopped. Restart vsftpd service whenever you made any change in configuration file.

```
[root@server ~]# service vsftpd status
vsftpd is stopped
[root@server ~]# service vsftpd start
Starting vsftpd for vsftpd:                                [  OK  ]
[root@server ~]# service vsftpd restart
Shutting down vsftpd:                                      [  OK  ]
Starting vsftpd for vsftpd:                                [  OK  ]
[root@server ~]# service vsftpd status
vsftpd (pid 3331) is running...
[root@server ~]# _
```

FTP Server is by default configured to listen on port 21. Port 21 must be opened if you have configured firewall. The configuration of a firewall for an FTP server is a relatively simple process.

#iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 21 -j ACCEPT

```
[root@server ~]# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 21
 -j ACCEPT
[root@server ~]# _
```

FTP server

CREATE 2 NORMAL USER ACCOUNTS FOR TESTING

- Create a normal user

```
[root@server ~]# useradd sanjay
[root@server ~]# passwd sanjay
Changing password for user sanjay.
New password:
BAD PASSWORD: it is too simplistic/systematic
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@server ~]# _
```

- create another normal user

```
[root@server ~]# useradd vikarm
[root@server ~]# passwd vikarm
Changing password for user vikarm.
New password:
BAD PASSWORD: it is too simplistic/systematic
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@server ~]# _
```

FTP server

- CONFIGURE FTP CLIENT
- To run **ftp** command **ftp** package is required. Install it if it is not installed.

```
[root@linuxclient ~]# rpm -qa ftp*
[root@linuxclient ~]# cd /media/RHEL_6.1\ x86_64\ Disc\ 1/Packages/
[root@linuxclient Packages]# rpm -ivh ftp*
warning: ftp-0.17-51.1.el6.x86_64.rpm: Header V3 RSA/SHA256 Signatur
431d51: NOKEY
Preparing...                ########################################
   1:ftp                    ########################################
[root@linuxclient Packages]# cd
[root@linuxclient ~]# rpm -qa ftp*
ftp-0.17-51.1.el6.x86_64
[root@linuxclient ~]# _
```

Check connectivity with FTP Server.

```
[root@linuxclient ~]# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.77 ms
^C
--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 932ms
rtt min/avg/max/mdev = 1.773/1.773/1.773/0.000 ms
[root@linuxclient ~]# _
```

Now try to run ftp command

```
[root@linuxclient ~]# ftp 192.168.1.1
Connected to 192.168.1.1 (192.168.1.1).
220 (vsFTPd 2.2.2)
Name (192.168.1.1:root): _
```

FTP server

- **Most commonly commands used on ftp prompt are:**
- **put** To upload files on server
- **get** To download files from server
- **mput** To upload multiple files
- **mget** To download all files
- **?** To see all available command on ftp prompts
- **cd** To change remote directory
- **lcd** To change local directory.

SSH server installation and settings

- As a Linux administrator you should know

- SSH stand for Secure Shell.
- SSH is a network protocol for secure data communication.
- SSH protocol allows remote command line login.
- SSH protocol enables remote command execution.
- To use SSH you need to deploy SSH Server and SSH Client program respectively.
- OpenSSH is a FREE version of the SSH.
- Telnet, rlogin, and ftp transmit unencrypted data over internet.
- OpenSSH encrypt data before sending it over insecure network like internet.
- OpenSSH effectively *eliminate* eavesdropping, connection hijacking, and other *attacks*.
- OpenSSH provides secure tunneling and several authentication methods.
- OpenSSH replace Telnet and rlogin with SSH, rcp with scp, ftp with sftp.

SSH

- **SSH Tools**

- sshd
- The daemon service that implements the ssh server. By default it must be listening on port 22 TCP/IP.
- ssh
- The ssh [Secure Shell command] is a secure way to log and execute commands in to SSH Server system.
- scp
- The Secure Copy command is a secure way to transfer files between computers using the private/public key encryption method.
- ssh-keygen
- This utility is used to create the public/private keys.
- ssh-agent
- This utility holds private keys used for RSA authentication.
- ssh-add
- Adds RSA identities to the authentication agent ssh-agent.

- **Exercises**

- Configure a SSH server and SSH client
- Create two user user1 and user2 and verify that both users can login in SSH server from SSH client.
- Do not allow root and user1 users to login to it and allow the rest of users. To confirm it login from user2.
- Re-configure SSH Server to allow login only using public / private keys. Generate keys for user2 and verify that user2 can login using keys.
- Change default ssh port to 2223

SSH

How to configure SSH Server in RHEL

Two RPM are required to configure and run OpenSSH server.

- openssh-server
- openssh

Before you start configuration make sure that you have necessary RPM packages installed. Install if any RPM is missing.

```
[root@server ~]# rpm -qa openssh*
openssh-5.3p1-52.el6.x86_64
openssh-clients-5.3p1-52.el6.x86_64
openssh-server-5.3p1-52.el6.x86_64
openssh-askpass-5.3p1-52.el6.x86_64
[root@server ~]# _
```

Check the current status of **sshd** service, it must be running. If service is stopped start i

Options you need with service command are **start** | **stop** | **restart** | **status**

```
[root@server ~]# service sshd status
openssh-daemon (pid  5788) is running...
[root@server ~]# service sshd stop
Stopping sshd:                                             [  OK  ]
[root@server ~]# service sshd start
Starting sshd:                                             [  OK  ]
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# service sshd status
openssh-daemon (pid  5978) is running...
[root@server ~]# _
```

SSH

Configure it to start when the system is booted

```
[root@server ~]# chkconfig sshd on
[root@server ~]# _
```

IP address of OpenSSH server is required, note it down

```
[root@server ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:6F:D9:13
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe6f:d913/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1667 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1721 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:227903 (222.5 KiB)  TX bytes:254089 (248.1 KiB)

[root@server ~]# _
```

We use to configure a firewall to either block or allow network communication through one or more ports. So if you have configured firewall then you have to allow SSH.

```
[root@server ~]# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
[root@server ~]# _
```

SSH

- How to configure SSH client

Check necessary RPM, install if any missing

```
[root@linuxclient ~]# rpm -qa openssh*
openssh-askpass-5.3p1-52.el6.x86_64
openssh-5.3p1-52.el6.x86_64
openssh-clients-5.3p1-52.el6.x86_64
openssh-server-5.3p1-52.el6.x86_64
[root@linuxclient ~]# _
```

Check **sshd** service status it must be running. Start it if it is off

```
[root@linuxclient ~]# service sshd status
openssh-daemon (pid  30293) is running...
[root@linuxclient ~]# _
```

Configure **sshd** service to start to at boot time

```
[root@linuxclient ~]# service sshd status
openssh-daemon (pid  30293) is running...
[root@linuxclient ~]# _
```

SSH

Check connectivity from SSH server

```
[root@linuxclient ~]# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.974 ms
^C
--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 830ms
rtt min/avg/max/mdev = 0.974/0.974/0.974/0.000 ms
[root@linuxclient ~]# _
```

That's all setting which we need on client system.

Create two user user1 and user2 and verify that both users can login in SSH server from SSH client.

Go on server and create two users **user1** and **user2**

```
[root@server ~]# useradd user1
[root@server ~]# useradd user2
[root@server ~]# passwd user1
Changing password for user user1.
New password:
BAD PASSWORD: it is too simplistic/systematic
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@server ~]# passwd user2
Changing password for user user2.
New password:
BAD PASSWORD: it is too simplistic/systematic
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
```

SSH

- Open main configuration file sshd_config

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

Check the value of Password Authentication directive. In order to accept local user password base authentication it must be set to yes. Set it to yes if it is set to no and save the file.

```
# To disable tunneled clear text passwords,
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication yes  _
```

Restart the service if you have made any change in sshd_config

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# _
```

SSH

Go on linuxclient system and verify that both users can login in SSH server. Also verify from root user.

```
[root@linuxclient ~]# ssh user1@192.168.1.1
The authenticity of host '192.168.1.1 (192.168.1.1)' can't be established.
RSA key fingerprint is d5:fc:d0:88:c0:9a:b4:b1:50:9d:85:01:49:4b:42:02.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.1' (RSA) to the list of known hosts.
user1@192.168.1.1's password:
[user1@server ~]$ who am i
user1    pts/0        2013-09-17 21:57 (192.168.1.10)
[user1@server ~]$ exit                                          user1
logout
Connection to 192.168.1.1 closed.
[root@linuxclient ~]# ssh user2@192.168.1.1
user2@192.168.1.1's password:
[user2@server ~]$ who am i                                      user2
user2    pts/0        2013-09-17 22:00 (192.168.1.10)
[user2@server ~]$ exit
logout
Connection to 192.168.1.1 closed.
[root@linuxclient ~]# ssh root@192.168.1.1
root@192.168.1.1's password:                                    root
Last login: Tue Sep 17 21:41:38 2013
[root@server ~]# who am i
root     pts/0        2013-09-17 22:00 (192.168.1.10)
[root@server ~]# exit
```

Do not allow root and user1 users to login to it and allow the rest of users. Also to confirm it login from user2.

User and Host Based Security

SSH

- **User and Host Based Security**
- Following additional directives can be added to /etc/sshd/sshd_config file in order to make the ssh server more restrictive.
- **Block empty passwords**
- PermitEmptyPasswords no
- **Block root user to log on the system using ssh.**
- PermitRootLogin no
- **Limit the users allowed to access a system via SSH. In this case only users 'laxmi' and 'vinita' are allowed to login on the system using SSH**
- AllowUsers laxmi vinita
- **Make it more restrictive and add node address with user name. In following case only allow login through SSH users 'laxmi' and 'vinita' from 192.168.1.10 node.**
- AllowUsers laxmi@192.168.1.10 vinita@192.168.1.10
- **In addition you can restrict the access to users. In this case all users except 'user1' are allowed to connect to the SSH server.**
- DenyUsers user1
- **Go back on server and open main configuration file again**

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

SSH

- **In the end of file add following directives and save the file**
- PermitRootLogin no
- DenyUsers user1

```
# Example of overriding settings on a per-user basis
#Match User anoncvs
#        X11Forwarding no
#        AllowTcpForwarding no
#        ForceCommand cvs server

PermitRootLogin no ———This will block root login
DenyUsers user1 ———This will block user1
```

Restart the sshd service

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# _
```

SSH

- Go back on linuxclient system and verify that we have blocked user1 and root. Also verify that user2 able to login in SSH server.

```
[root@linuxclient ~]# ssh user1@192.168.1.1
user1@192.168.1.1's password:
Permission denied, please try again.
user1@192.168.1.1's password:

[root@linuxclient ~]# ssh root@192.168.1.1
root@192.168.1.1's password:
Permission denied, please try again.
root@192.168.1.1's password:

[root@linuxclient ~]# ssh user2@192.168.1.1
user2@192.168.1.1's password:
Last login: Tue Sep 17 22:00:06 2013 from 192.168.1.10
[user2@server ~]$ who am i
user2    pts/0        2013-09-17 22:35 (192.168.1.10)
[user2@server ~]$ exit
logout
Connection to 192.168.1.1 closed.
[root@linuxclient ~]# _
```

user1 and root is blocked

user2 is allowed

Re-configure SSH Server to allow login only using public / private keys. Generate keys for user2 and verify that user2 can login using keys.

To make Linux server more secure linux administrator usually disable password authentication on the SSH server and allow only public/private keys authentication.

SSH

- **Private Keys**
- Private keys are stored on server and must be secured. Anything encrypted with public key can only be decrypted with paired private key. So it must be accessible only to the user owner of that key, in the .ssh subdirectory of that user's home directory.
- **Public Keys**
- Public keys are publicly available. Public keys are required to connect with server. The public keys for SSH servers belong on administrative workstations.
- **Go back on server and open main configuration file again**

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

Uncomment following directives and save the file

RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys
#AuthorizedKeysCommand none
#AuthorizedKeysCommandRunAs nobody
```

Uncomment these

SSH

Restart the sshd service

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# _
```

Login form user2 and create a ssh directory with permission 755

```
Red Hat Enterprise Linux Server release 6.1 (Santiago)
Kernel 2.6.32-131.0.15.el6.x86_64 on an x86_64

server login: user2
Password:
Last login: Tue Sep 17 22:35:07 from 192.168.1.10
[user2@server ~]$ mkdir ~/.ssh
[user2@server ~]$ chmod 755 ~/.ssh
[user2@server ~]$ _
```

Come back on linuxclient system and create a normal user account user2.

```
[root@linuxclient ~]# useradd user2
[root@linuxclient ~]# passwd user2
Changing password for user user2.
New password:
BAD PASSWORD: it is too simplistic/systematic
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@linuxclient ~]# _
```

SSH

- Login form user2 and create a ssh directory with permission 755

```
Red Hat Enterprise Linux Server release 6.1 (Santiago)
Kernel 2.6.32-131.0.15.el6.x86_64 on an x86_64

linuxclient login: user2
Password:
[user2@linuxclient ~]$ mkdir ~/.ssh
[user2@linuxclient ~]$ chmod 755 ~/.ssh
[user2@linuxclient ~]$ _
```

Generate the public/private key pair. Accept default location for key file.

```
linuxclient login: user2
Password:
[user2@linuxclient ~]$ mkdir ~/.ssh
[user2@linuxclient ~]$ chmod 755 ~/.ssh
[user2@linuxclient ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user2/.ssh/id_rsa):
```

Press Enter to accept default location

Enter passphrase 'I love linux' and confirm

```
[user2@linuxclient ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user2/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user2/.ssh/id_rsa.
Your public key has been saved in /home/user2/.ssh/id_rsa.pub.
The key fingerprint is:
5b:3d:9a:10:77:cf:22:22:dd:47:dc:b7:b7:8a:02:32 user2@linuxclient
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|            . .  |
|         . . + . .|
|        . + + o ..|
|       . S + = o..|
|      E o  = = o  o|
|       o o o     . |
|          .  .  .  |
|                   |
```

I love linux

SSH

- Public key is stored in /home/user2/.ssh/id_rsa.pub. Create a copy of public key

```
[user2@linuxclient ~]$ cat ~/.ssh/id_rsa.pub >> authorized_keys
[user2@linuxclient ~]$ _
```

Copy the **authorized_keys** file on server to **/home/user2/.ssh/authorized_keys**. Enter user2 [user account on server] password when asked

```
[user2@linuxclient ~]$ scp authorized_keys user2@192.168.1.1:/home/user2/.ssh/
```

On **server** verify that we have successfully copied public key on server. Also set permission to 644 for **authorized_keys**

```
[user2@server ~]$ ll ~/.ssh/
total 4
-rw-rw-r--. 1 user2 user2 399 Sep 17 23:12 authorized_keys
[user2@server ~]$ chmod 644 ~/.ssh/authorized_keys
[user2@server ~]$ _
```

Login from **root** on server and open **sshd_config** file

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

Set **PasswordAuthentication** directive to **no** and save the file. This will block login using password.

```
# To disable tunneled clear text passwords,
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no_
```

SSH

- Restart the **sshd** service

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# _
```

Come back on linuxclient system.
Logout from user2 and login back.
Now try to login from user2 on linuxclient. Enter passphrase 'I love linux'

```
[user2@linuxclient ~]$ ssh -l user2 192.168.1.1
Enter passphrase for key '/home/user2/.ssh/id_rsa':
Last login: Tue Sep 17 23:31:31 2013 from 192.168.1.10
[user2@server ~]$ _
```

Change default ssh port to 2223
Come on server and open **sshd_config** file again

```
[root@server ~]# vi /etc/ssh/sshd_config_
```

Uncomment following directive and change value to **2223**
#port 22

```
# The strategy used for options in the default sshd_config
# OpenSSH is to specify options with their default value w
# possible, but leave them commented.  Uncommented options
# default value.

Port 2223 ———— Uncomment and change it to 2223
```

SSH

- restart the **sshd** service

```
[root@server ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@server ~]# _
```

Go back on **linuxclient** system and try to connect with default port

```
[user2@linuxclient ~]$ ssh -l user2 192.168.1.1
ssh: connect to host 192.168.1.1 port 22: Connection refused
[user2@linuxclient ~]$ _
```

Now specify the new port

```
[user2@linuxclient ~]$ ssh -l user2 192.168.1.1 ——— Default port [22]
ssh: connect to host 192.168.1.1 port 22: Connection refused
[user2@linuxclient ~]$ ssh -l user2 192.168.1.1 -p 2223
Enter passphrase for key '/home/user2/.ssh/id_rsa':
Last login: Tue Sep 17 23:25:45 2013 from 192.168.1.10
[user2@server ~]$ _                    Connection accepted
```

SSH Configuration files

There are two different sets of configuration files

System-wide SSH configuration :- stored in the **/etc/ssh/** directory
User-specific SSH configuration :- stored in **~/.ssh/** within the user's home directory

SSH

- **System-wide configuration files**

File	Description
/etc/ssh/ssh_config	The default SSH client configuration file.
/etc/ssh/sshd_config	The configuration file for the sshd daemon.
/etc/ssh/ssh_host_dsa_key	The DSA private key used by the sshd daemon.
/etc/ssh/ssh_host_dsa_key.pub	The DSA public key used by the sshd daemon.
/etc/ssh/ssh_host_key	The RSA private key used by the sshd daemon for version 1 of the SSH protocol.
/etc/ssh/ssh_host_key.pub	The RSA public key used by the sshd daemon for version 1 of the SSH protocol.
/etc/ssh/ssh_host_rsa_key	The RSA private key used by the sshd daemon for version 2 of the SSH protocol.
/etc/ssh/ssh_host_rsa_key.pub	The RSA public key used by the sshd daemon for version 2 of the SSH protocol.

- **User-specific configuration files**

File	Description
~/.ssh/authorized_keys	Holds a list of authorized public keys for servers.
~/.ssh/id_dsa	Contains the DSA private key of the user.
~/.ssh/id_dsa.pub	The DSA public key of the user.
~/.ssh/id_rsa	The RSA private key used by ssh for version 2 of the SSH protocol.
~/.ssh/id_rsa.pub	The RSA public key used by ssh for version 2 of the SSH protocol.
~/.ssh/identity	The RSA private key used by ssh for version 1 of the SSH protocol.
~/.ssh/identity.pub	The RSA public key used by ssh for version 1 of the SSH protocol.
~/.ssh/known_hosts	Contains DSA host keys of SSH servers accessed by the user.

Installation and configuration of APACHE

Web servers are a remote computers or computer programs that delivers web content (like web pages, etc.) to the end user over the internet upon request through a web browser. It comprises a computer and a server program. Every web server will have an internet protocol(IP) address and domain name through which it is identified over the internet.

Many web servers run on high speed internet connection. The basic function of a web server is to *host websites and to deliver web content* from its hosted websites over the internet. During the delivery of web pages, web servers follow a network protocol known as *hyper text transfer protocol (HTTP).* Web hosting service providers use web servers to host multiple websites. Web servers need a *continuous power supply and necessary cooling* for them to function efficiently.

Role of web servers in web hosting

Hosting websites refers to placing websites on web servers to bring them into access by people over internet. Web servers play a significant role in web hosting services as they form the key elements. Following are few functions performed by web servers in hosting:

APACHE

Stores and secures website data: In web hosting services, a web server stores all website data and secures it from unauthorized users when it is properly configured.

Provides web database access: A web server's responsibility is to provide access to websites that are hosted. Web hosting service providers own some web servers that are used in variable ways to provide different web hosting services, such as backend database servers.

Serve the end user requests: Web servers accept requests from different users connected over the internet and serve them accordingly.

Common Tasks by Administrators

Starting, stopping, and restarting/reloading are the most common tasks when working with an Apache webserver.

The commands for managing the Apache service are different across Linux distributions.

APACHE

- Most of the recent Linux distributions are using SystemD as the default init system and service manager. Older distributions are based on SysVinit and using init scripts to manage services. Another difference is the name of the service. In Ubuntu and Debian, the Apache service is named **apache2**, while in Red Hat based system such as CentOS, the name of the service is **httpd**.
- **######Before You Begin#####**
- The instructions assume that you are logged in as root or user with sudo privileges.

Both SystemD service units and SysVinit script takes the following arguments to manage the Apache service:

- **start**: Starts the Apache service.
- **stop**: Terminates the Apache service.
- **restart**: Stops and then starts the Apache service.
- **reload**: Gracefully restarts the Apache service. On reload, the main Apache process shuts down the child processes, loads the new configuration, and starts new child processes.
- **status**: Shows the service status

APACHE

Install Apache Server

To install Apache web server, use your default distribution package manager as shown.

$ sudo apt install apache2 **[On Debian/Ubuntu]**
$ sudo yum install httpd **[On RHEL/CentOS]**
$ sudo dnf install httpd **[On Fedora 22+]**
$ sudo zypper install apache2 **[On openSUSE]**

Check Apache Version

$ sudo httpd -v
OR
$ sudo apache2 –v
Server version: Apache/2.4.6 (CentOS)
Server built: Nov 5 2018 01:47:09

Start Apache Service

---------- On CentOS/RHEL ----------
$ sudo systemctl start httpd **[On Systemd]**
$ sudo service httpd start **[On SysVInit]**
---------- On Ubunt/Debian ----------
$ sudo systemctl start apache2 **[On Systemd]**
$ sudo service apache2 start **[On SysVInit]**

APACHE admin

- **Check Apache Configuration Syntax Errors**
- To check your Apache configuration files for any syntax errors run the following command, which will check the validity of the config files, prior to restart the service.

- **$ sudo httpd -t**
- **OR**
- **$ sudo apache2ctl –t**
- Sample output

AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using tesla.com.
Set the 'ServerName' directive globally to suppress this message
Syntax OK

Enable Apache Service

The previous command only starts the Apache service for the meantime, to enable it auto-start at system boot, run the following command.

---------- On CentOS/RHEL ----------
$ sudo systemctl enable httpd [On Systemd]
$ sudo chkconfig httpd on [On SysVInit]

---------- On Ubunt/Debian ----------
$ sudo systemctl enable apache2 [On Systemd]
$ sudo chkconfig apache2 on [On SysVInit]

APACHE admin

- **Restart Apache Service**
- To restart Apache (stop and then start the service), run the following command.

- ---------- On CentOS/RHEL ----------
- **$ sudo systemctl restart httpd [On Systemd]**
- **$ sudo service httpd restart [On SysVInit]**

- ---------- On Ubunt/Debian ----------
- **$ sudo systemctl restart apache2 [On Systemd]**
- **$ sudo service**
- **View Apache Service Status**
- To check the Apache service run time status information, run the following command.

- ---------- On CentOS/RHEL ----------
- **$ sudo systemctl status httpd [On Systemd]**
- **$ sudo service httpd status [On SysVInit]**

- ---------- On Ubunt/Debian ----------
- **$ sudo systemctl status apache2 [On Systemd]**
- **$ sudo service apache2 status [On SysVInit] apache2 restart [On SysVInit]**

APACHE

Monitoring the server: Log Files

- **Using linux Command-Line Tools**
- One approach to monitor Apache logs from the server is by using linux command-line tools. And although Linux commands won't provide graphical representation of the logs, but possible to get the desired results.
- You can access Apache logs from **var/log/log_type** . For example, you can access Apache logs from the Apache Linux server by looking in the following directories:
- **/var/log/apache/access.log**
- **/var/log/apache2/access.log**
- **/etc/httpd/log/ access_log (on MacOS)**
- **/var/log/apache2/error.log**

On a Linux server, you can access Apache **error** logs from var/log/apache2/error.log. You can then log out the errors from the error log file by writing the following command: **sudo tail -f /var/log/apache2/error.log**. When you run this command, you'll be able to view the errors in the terminal as they occur in real time. The tail command tells the machine to read the file and display the results on the terminal. You can use these errors to monitor the operations on your website and better troubleshoot issues occurring on the web server.

APACHE log

- **Specifying an IP Address**
- To get log messages from a certain IP address, simply run this single line of command and get information from a specific IP address, lets take it 192.168.206.1 :
- tail -f /var/log/apache2/access.log | grep 192.168.206.1
- The combination of these commands will match the wanted partner

ip packages filtering: netfilter &iptables

- There are three generations of kernel-based IP filtering in Linux, and each has had its own configuration mechanism.
- The **first generation** was called ipfw (for "IP firewall"), and provided basic filtering capability but was somewhat inflexible and inefficient for complex configurations. ipfw is rarely used now.
- The **second generation** of IP filtering, called IP chains, improved greatly on ipfw, and is still in common use.
- The **latest generation** of filtering is called netfilter/iptables. netfilter is the kernel component and iptables is the user-space configuration tool; these terms are often used interchangeably. netfilter is not only much more *flexible* to configure, but is *extensible* as well.
- The primary tool for manipulating and displaying the filtering tables is called iptables and is included in all current Linux distributions.
- The iptables command allows configuration of a rich and complex set of firewall rules and hence has a large number of command-line options.

netfilter

- An important concept in **netfilter** is the notion of a chain, which consists of a list of rules that are applied to packets as they enter, leave, or traverse through the system. The kernel defines three chains by default, but the administrator can specify new chains of rule and link them to the predefined chains. The three predefined chains are:
  - INPUT: This chain applies to packets that are received and are destined for the local system.
  - OUTPUT: This chain applies to packets that are transmitted by the local system.
  - FORWARD: This chain applies whenever a packet will be routed from one network interface to another through this system. It is used whenever the system is acting as a packet router or gateway, and applies to packets that are neither originating from nor destined to this system.

netfilter

- It is possible to develop **extensions** that enhance the way **netfilter** operates. Some examples of more sophisticated packet handling actions are:
- **Packet logging :** possible to create rules that do nothing more than log a description of the matching packet so that it can be captured for analysis later. This is very useful for detecting attacks and for testing a filtering configuration.
- **Stateful inspection:** netfilter includes a set of helper modules that can perform stateful connection inspection, such as management of FTP connections, as described earlier.
- **Network Address Translation:** Network Address Translation (NAT), also called IP masquerading, provides a means of rewriting the IP addresses and port numbers of packets as they pass through a chain. NAT is most commonly used to allow systems on a private network to use a connection to the Internet with a single IP address.
- **Packet and byte accounting** netfilter provides counters that allow you to measure how the network traffic handled each rule, and several IP accounting systems are based on these statistics. These counters are visible when you use iptables to list rulesets in verbose mode;

Installing MondoRescue on RHEL / CentOS / Scientific Linux

- The latest **Mondo Rescue** packages (current version of **Mondo** is **3.0.3-1**) can be obtained from the "**MondoRescue Repository**". Use "**wget**" command to download and add repository under your system. The Mondo repository will install suitable binary software packages such as **afio**, **buffer**, **mindi**, **mindi-busybox**, **mondo** and **mondo-doc** for your distribution, if they are available.
- **For RHEL/CentOS/SL 6 – 64-Bit**

  # cd /etc/yum.repos.d/

  ## On RHEL/CentOS/SL 6 - 64-Bit ##

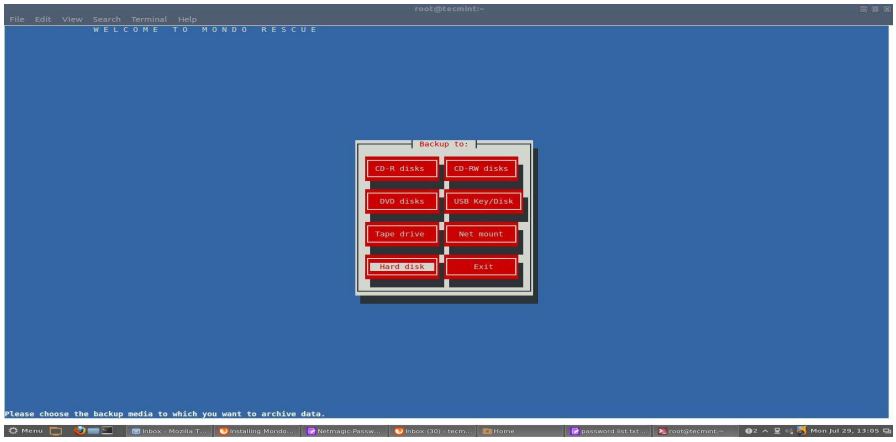
  # wget ftp://ftp.mondorescue.org/rhel/6/x86_64/mondorescue.repo

  Once you successfully added repository, do "yum" to install latest Mondo tool.
- # yum install mondo

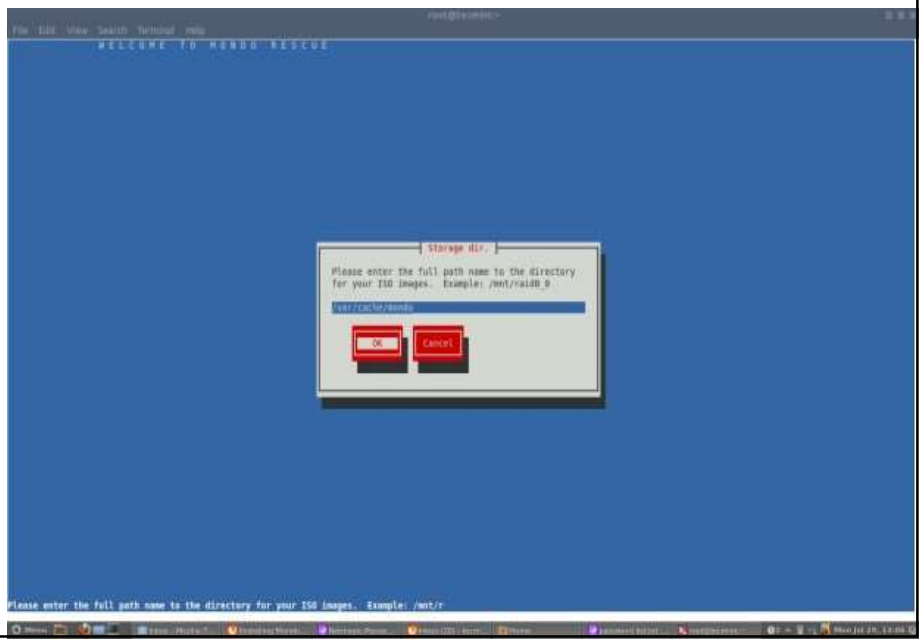
Creating Cloning or Backup ISO Image of System/Server

- After installing Mondo, Run “mondoarchive” command as “root” user. Then follow screenshots that shows how to create an ISO based backup media of your full system.
- **# mondoarchive**

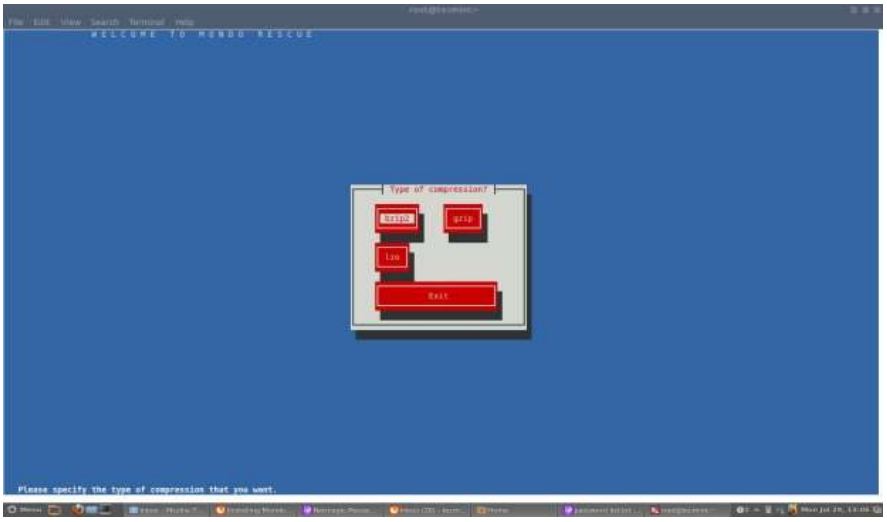
  Welcome to Mondo Rescue



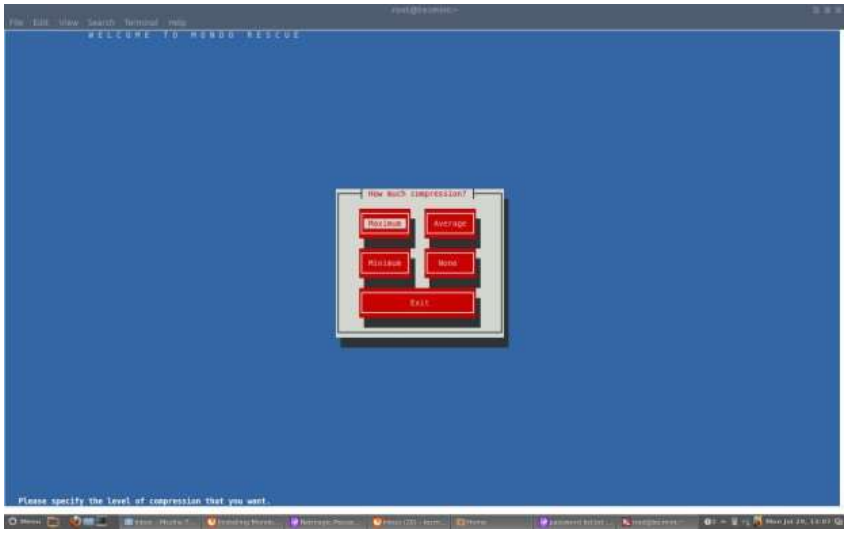
- Please enter the full path name to the directory for your **ISO Images**. For example: **/mnt/backup/**



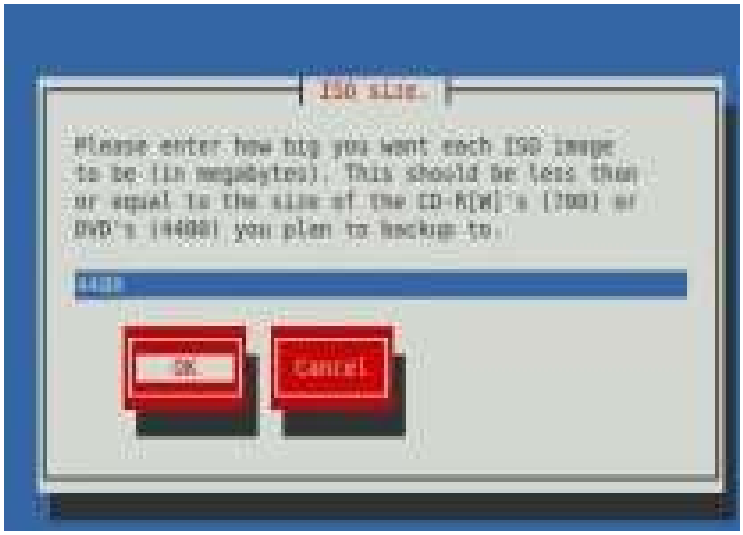
- Select Type of compression. For example: bzip, gzip or lzo.



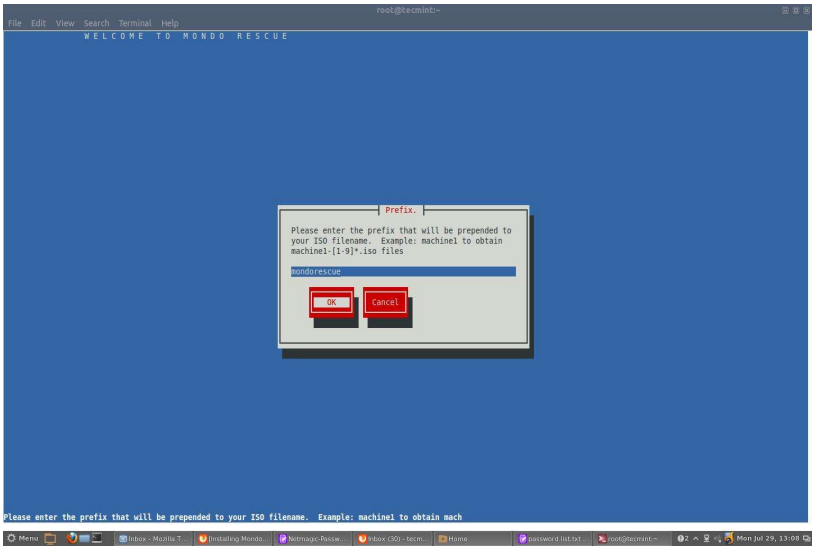
- Select the maximum compression option.



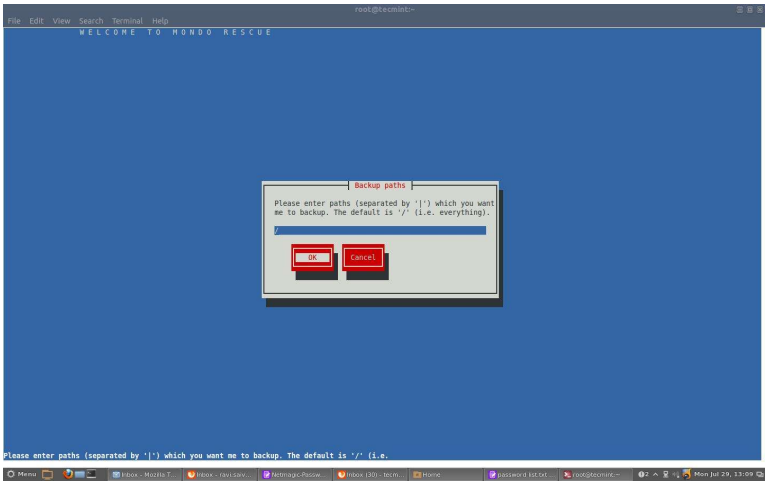
- Please enter how large you want each ISO image in MB (Megabytes). This should be less than or equal to the size of the CD-R(W)'s (i.e. 700) and for DVD's (i.e. 4480).



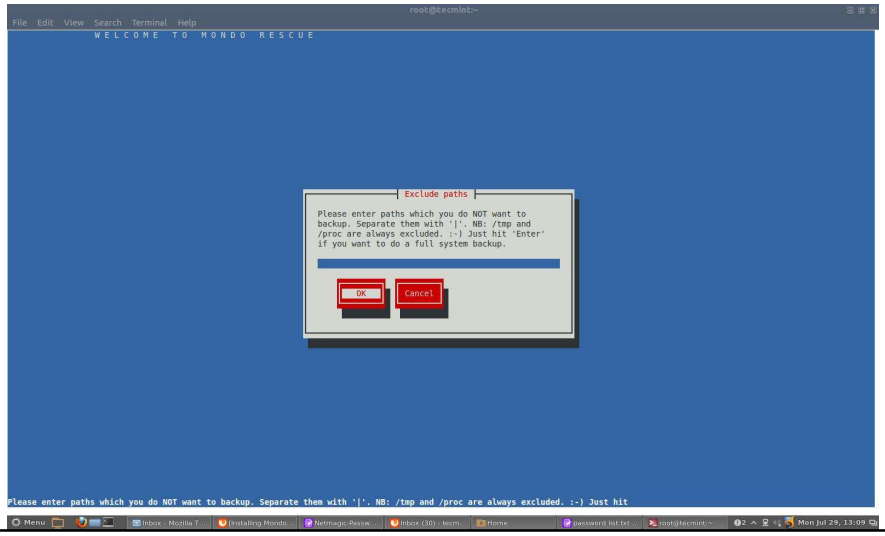
- Please give a name of your ISO image filename. For example: machine1 to obtain machine-[1-9]*.iso files.



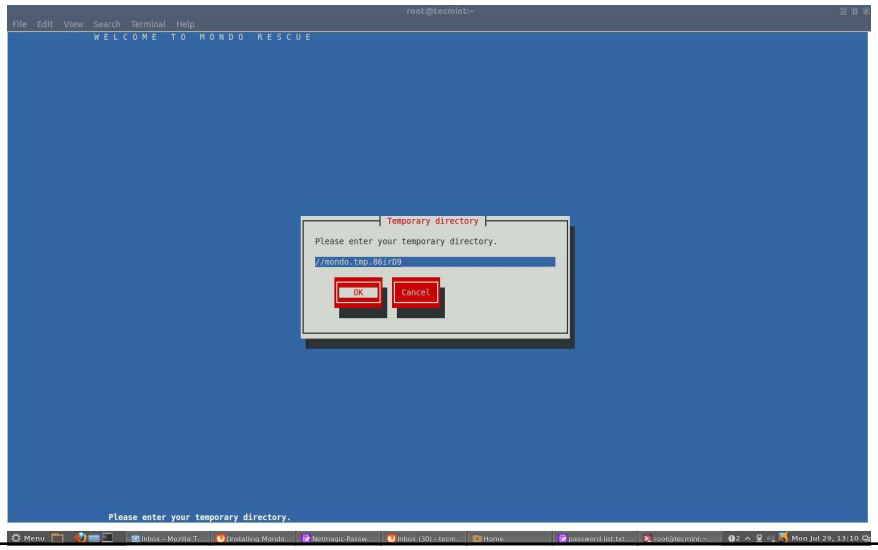
- Please add the filesystems to backup (separated by “|“). The default filesystem is “/” means full backup.



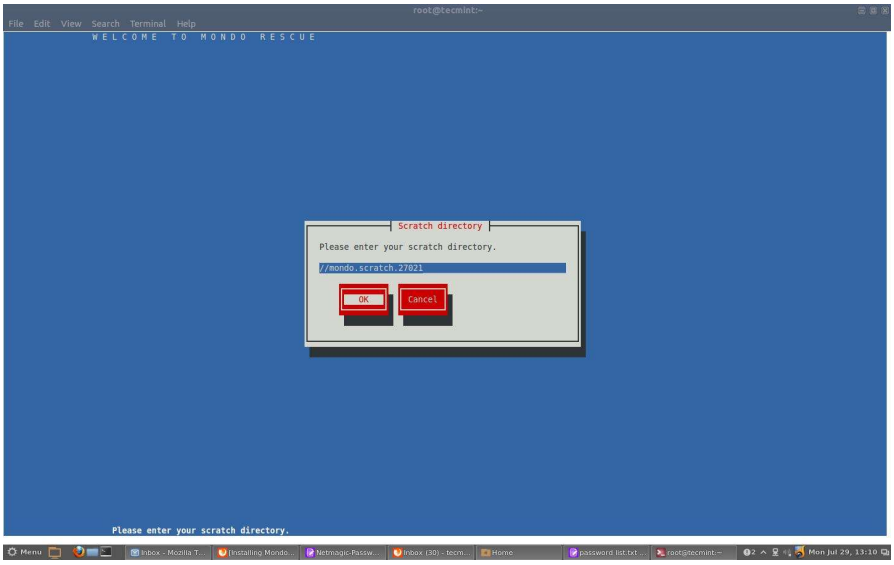
- Please exclude the filesystem that you don’t want to backup (separated by “|“). For example: “/tmp” and “/proc” are always excluded or if you want full backup of your system, just hit enter.



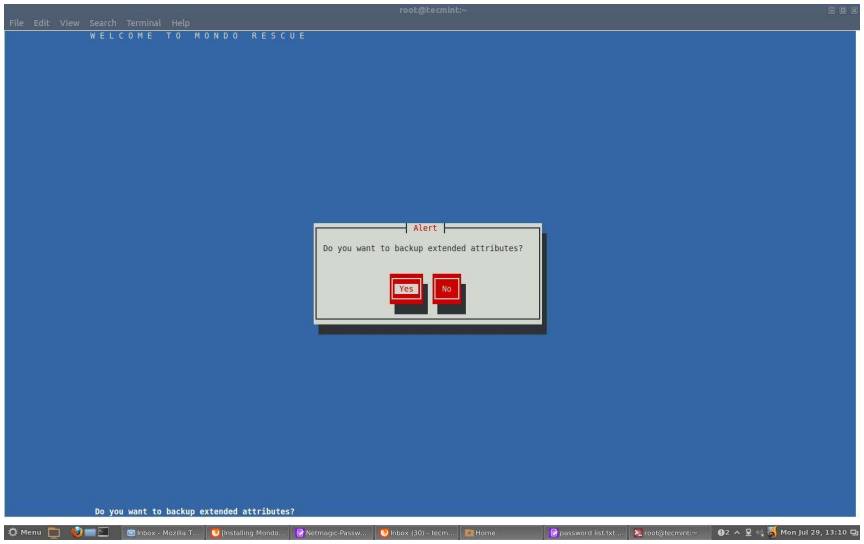
- Please enter your temporary directory path or select default one.



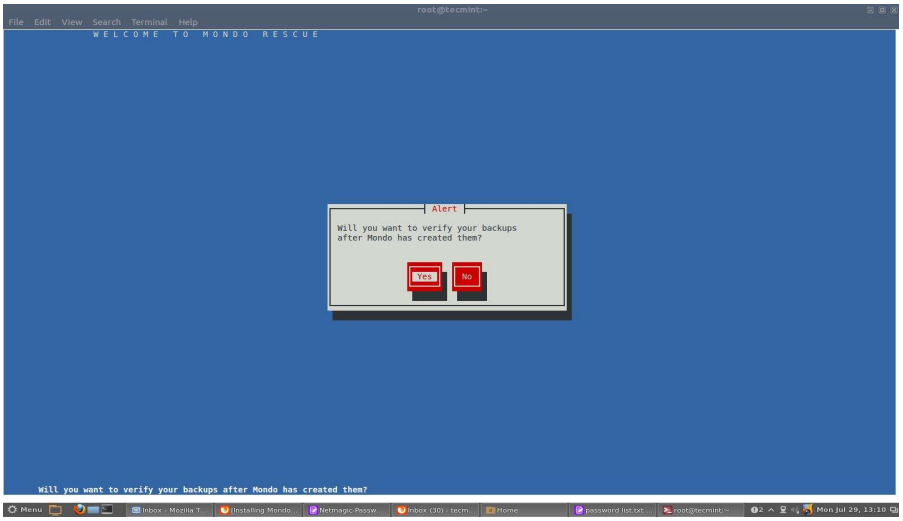
- Please enter your scratch directory path or select default one.



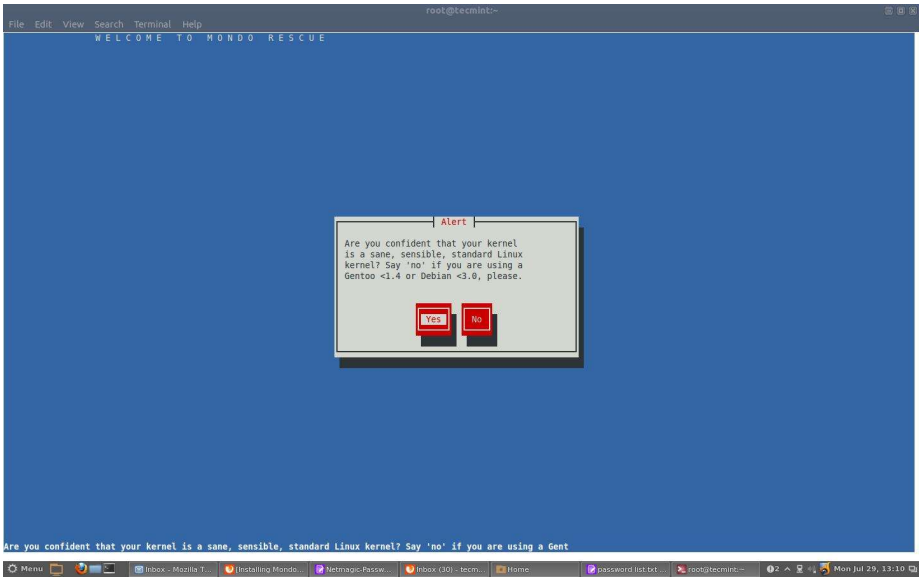
- If you would like to backup extended attributes. Just hit “enter“.



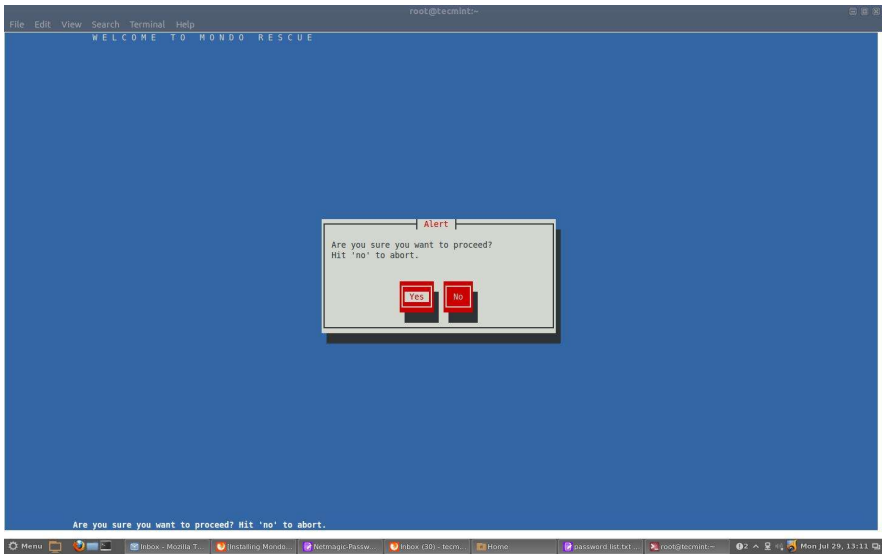
- If you want to Verify your backup, after mondo has created them. Click “Yes“.



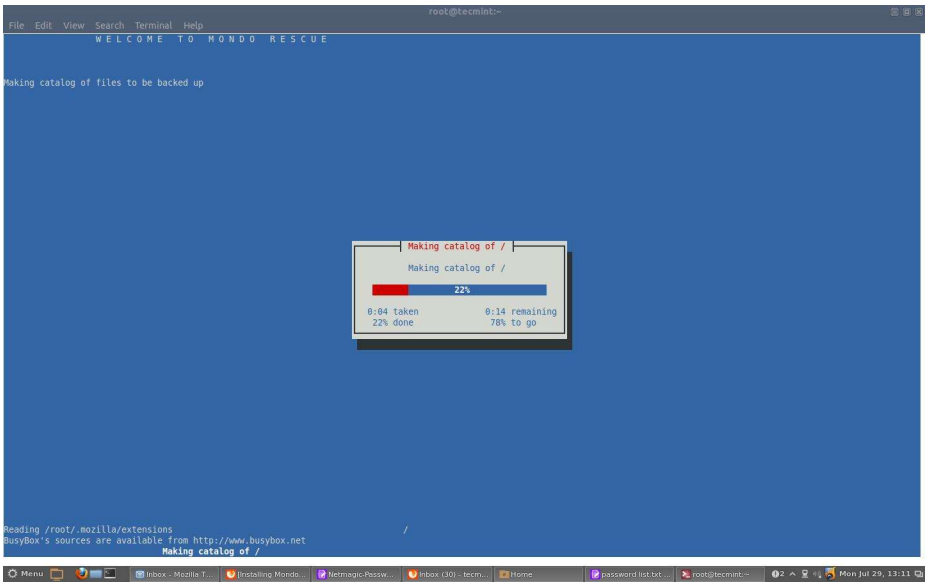
- If you’re using stable standalone Linux Kernel, click “Yes” or if you using other Kernel say “Gentoo” or “Debain” hit “No“.



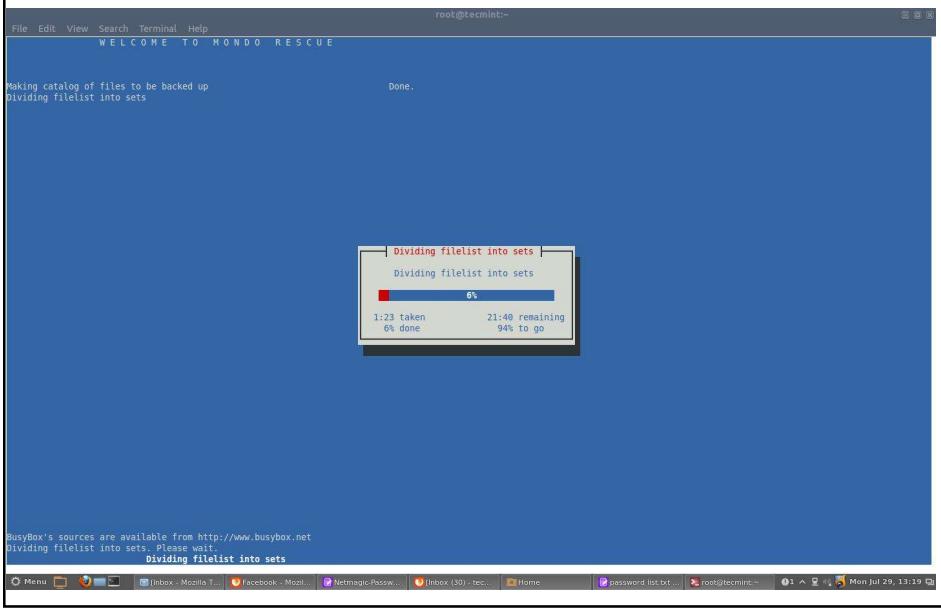
- Click “Yes” to proceed further.



- Creating a catalog of “/” filesystem.

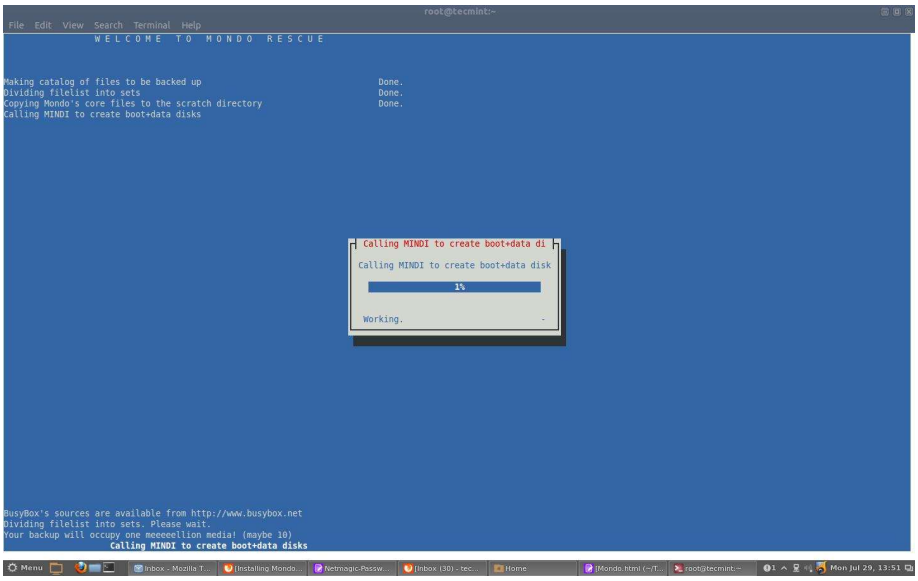


- Dividing filelist into sets.

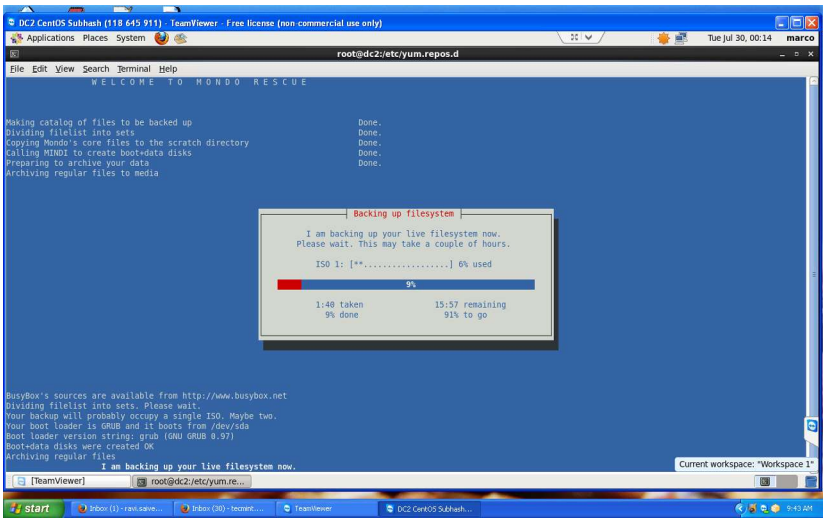


- Calling MINDI to create boot+data disk.

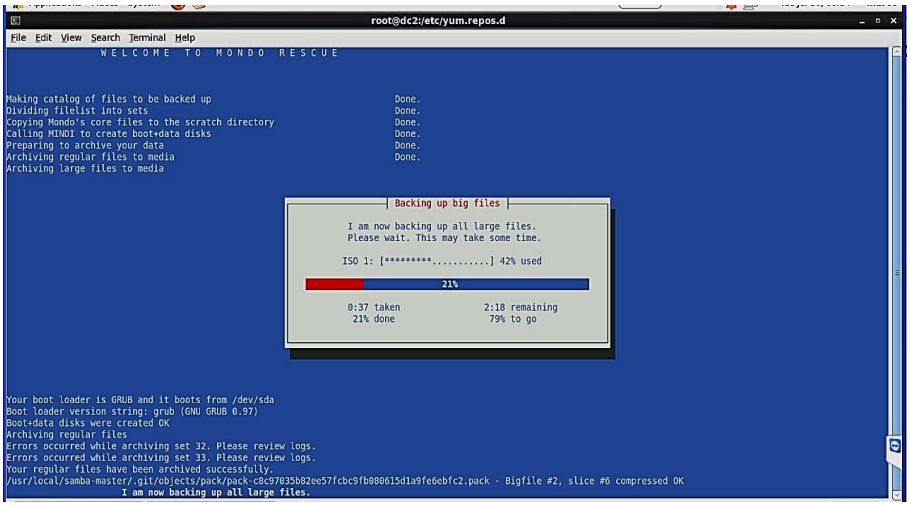
Mondo Rescue Boot Data Disk



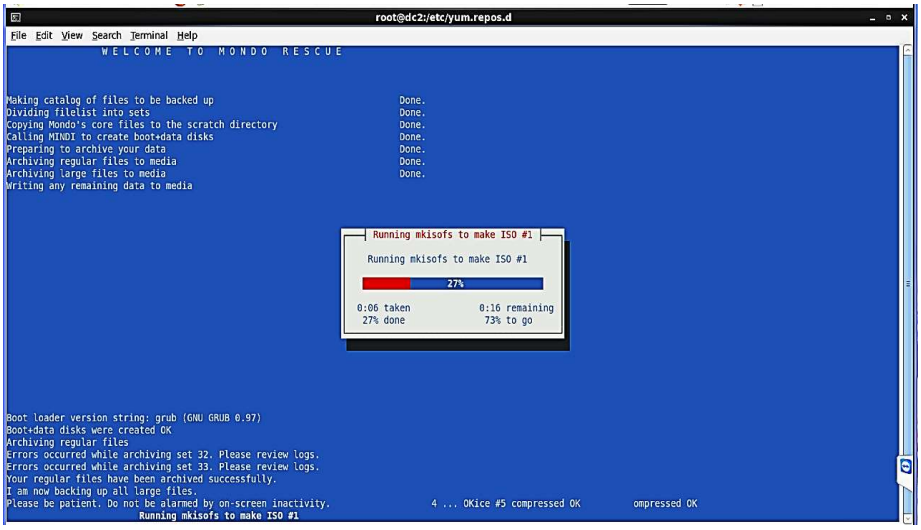
- Backing up filesytem. It may take a couple of hours, please be patient.



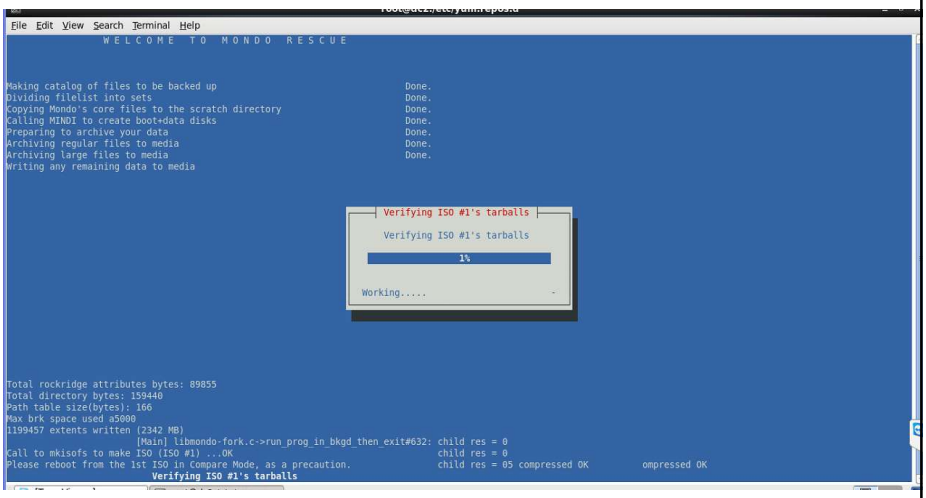
- Backing up big files.



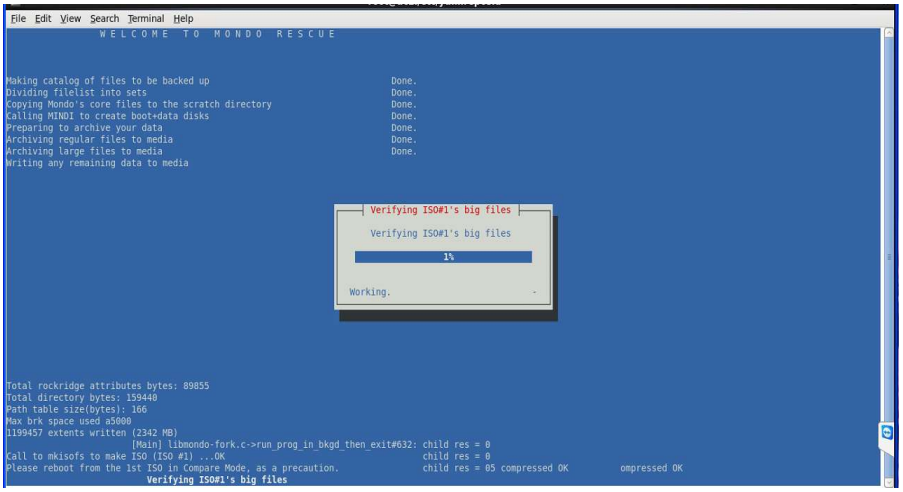
- Running “mkisofs” to make ISO Image.



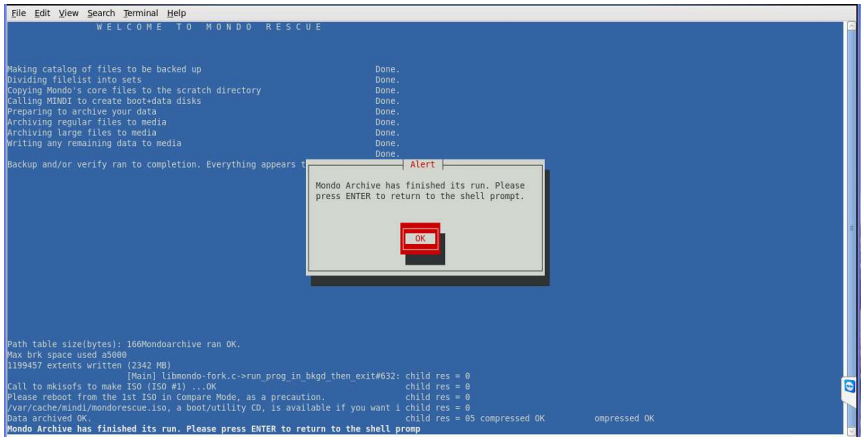
- Verifying ISO Image tarballs.



- Verifying ISO Image Big files.



- Finally, Mondo Archive has completed. Please hit “Enter” to back to the shell prompt.



- If you’ve selected default backup path, you will see an ISO image under “/var/cache/mondo/“, that you can burn into a CD/DVD for later restore.

- To restore all files automatically, boot the system with Mondo ISO Image and at boot prompt type “nuke” to restore files. Here is the detailed video that demonstrates how to restore files automatically from CD/DVD media.

Watch mondo rescue installation video

- https://www.youtube.com/watch?v=wFK8sgyTy08&t=160s

****

Thank you

****