**Real Time Linux Training**:

**Creating, Removing, Copying, Moving files & Directories**

**ID: fs-0c2ae8f9**

**Creating file**:



**Appending a file**:



**Checking the newfile**:



**Creating the files**:

**Checking created files**:



Creating Directory:



Checking the Directory:

In the below o/p you can can see the new directory created:



Creating directory in a directory:

newdir1 newdir2 newdir3



Checking dir are created:



Copying the files:

Present file are:



Now copying the file newfile to newfile1

In the below screenshot we have copy the <newfile> <newfile1>, also verify the data is same in both files.



Copying the file into a directory:

The below are file and directory listed:





Checking the file got created in newdir:

Moving the into newdir:

The file got copied into the directory & also check the content information is same:



The below are list of files and dir



Copying the Dir to dir:



Moving the files:

Below are listed files:



Moving files:

**mv newfile1 newfile2**



Moving Dir:

Below are listed files:



Mv newdir newdir2

After this their will be no newdir and changed to newdir2



removing a file:



Removing the dir:

For removing the dir we need to empty the dir:



If the dir have some data still need to remove:



**VIM EDITOR**

**VI Visual display editor**

**VIM Visual display editor improved**

**This is command mode editor for files. Other editors in Linux are emacs, gedit**

**vi editor is most popular**

**It has 3 modes:**

**1 Command Mode**

**2 Insert mode (edit mode)**

**3 extended command mode**

**Note:** When you open the vim editor, it will be in the command mode by default.

In the command mode the cursor’s can be used as

h/l/k/j to move cursor left/right/up/down

|  |  |
| --- | --- |
| **Insert Mode: i** | **To begin insert mode at the cursor position** |
| **I** | **To insert at the beginning of line** |
| **a** | **To append to the next word’s letter** |
| **A** | **To Append at the end of the line** |
| **o** | **To insert a new line below the cursor position** |
| **O** | **To insert a new line above the cursor position** |
| **Command Mode: gg** | **To go to the beginning of the page** |
| **G** | **To go to end of the page** |
| **w** | **To move the cursor forward, word by word** |
| **b** | **To move the cursor backward, word by word** |
| **nw** | **To move the cursor forward to n words (5W)** |
| **nb** | **To move the cursor backward to n words (5B)** |
| **u** | **To undo last change (word)** |
| **U** | **To undo the previous changes (entire line)** |
| **Ctrl+R** | **To redo the changes** |
| **yy** | **To copy a line** |
| **nyy** | **To copy n lines (5yy or 4yy)** |
| **p** | **To paste line below the cursor position** |
| **P** | **To paste line above the cursor position** |
| **dw** | **To delete the word letter by letter (like Backspace)** |
| **x** | **To delete the world letter by letter (like DEL Key)** |
| **dd** | **To delete entire line** |
| **ndd** | **To delete n no. of lines from cursor position(5dd)** |
| **/** | **To search a word in the file** |

**Extended Mode: ( Colon Mode)**

|  |  |
| --- | --- |
| **Extended Mode is used for save and quit or save without quit using “Esc” Key with “:” Esc+:w** | **To Save the changes** |
| **Esc+:q** | **To quit (Without saving)** |
| **Esc+:wq** | **To save and quit** |
| **Esc+:w!** | **To save forcefully** |
| **Esc+wq!** | **To save and quit forcefully** |
| **Esc+:x** | **To save and quit** |
| **Esc+:X** | **To give password to the file and remove password** |
| **Esc+:20(n)** | **To go to line no 20 or n** |
| **Esc+: se nu** | **To set the line numbers to the file** |
| **Esc+:se nonu** | **To Remove the set line numbers** |

**To open multiple files in vim editor**

#vim –o file1 file2

To switch between files use **Ctrl +w**

**Listing files and directories:**

#ls list the file names

#ls -l long listing of the file

#ls –l filename to see the permissions of a particular file

#ls -al shows the files in ascending order of modification.

#ls p\* All the files start with p.

#ls ?ample Files with any first character and has ample

#ls -ld l\* Directory listing only

#ls –ld directory name to see the permissions of a particular directory

#ls [ae]\* First character of the filename must be a or e.

# ls [!ae]\* ! Symbol complements the condition that follows. The characters must not be a or e.

#ls [a-m][c-z][4-9] list all the files in specific range

|  |  |
| --- | --- |
| **Types of Files: Symbol** | **Type of File** |
| **-** | **Normal file** |
| **d** | **Directory** |
| **l** | **Link file (shortcut)** |
| **b** | **Block file (Harddisk, Floppy disk)** |
| **c** | **Character file (Keyboard, Mouse)** |

**Creating a soft link:**

**# ln –s <source file> <destination>**



**Creating a Hard link:**

**#ln <source file> <Destination>**



**In Linux there are three types of users.**

**1. Super user or root user** Super user or the root user is the most powerful user. He is the administrator user.

**2. System user** System users are the users created by the softwares or applications. For example if we install Apache it will create a user apache. These kinds of users are known as system users.

**3. Normal user** Normal users are the users created by root user. They are normal users like Rahul, Musab etc. Only the root user has the permission to create or remove a user.

**Whenever a user is created in Linux things created by default:-**

· A home directory is created(/home/username)

· A mail box is created(/var/spool/mail)

· unique UID & GID are given to user

**Linux uses UPG (User Private Group) scheme**

· It means that whenever a user is created is has its own private group

· For Example if a user is created with the name **Rahul,** then a primary group for that user will be **Rahul** only

**There are two important files a user administrator should be aware of.**

**1. "/etc/passwd"**

**2. "/etc/shadow"**

**Each of the above mentioned files have specific formats.**

**1. /etc/passwd**



**The above fields are**

· **root** =name

· **x**= link to password file i.e. /etc/shadow

· **0** or **1**= UID (user id)

· **0** or **1**=GID (group id)

· **root** or **bin** = comment (brief information about the user)

· **/root** or **/bin** = home directory of the user

· **/bin/bash** or /**sbin/nologin** = shell

**2. /etc/shadow**

**root:$1fdsfsgsdfsdkffefje:14757:0:99999:7:::**

**The fields are as follows,**

1. **root** = User name

2. **:$1fdsfsgsdfsdkffefje** = Encrypted password

3. **14757 =** Days since that password was last changed.

4. **0 =** Days after which password must be changed.

5. **99999 =** Days before password is to expire that user is warned.

6. **7 =** Days after the password is expires that the user is disabled.

7. A reserved field.

**Password Complexity Requirements:**

· A root user can change password of self and of any user in the system, there are no rules for root to assign a password. Root can assign any length of password either long or short, it can be alphabet or numeric or both. On the whole there is no limitation for root for assigning a password.

· A normal user can change only its password. Valid password for a normal user should adhere to the following rules

· It should be at least **7** characters but not more than **255** characters.

· At least one character should be **Upper case**

· At least one character should be **Lower case**

· At least one character should be a **symbol,** and one character should be a **number**.

· It should not match the previous password.

· It cannot have a sequence (ex: **123456** or **abcdef** )

· The **login name** and the **password** cannot be same.

**Note:** For security reasons don’t keep the password based on **date of birth** because it can easily be hacked.

***LAB WORK:-***

**Creating a user**

· The syntax for creating a user in Linux is

· **# useradd <option> <username>**

· **options are**

· **-u user id**

· **-G Secondary group id**

· **-g primary group id**

· **-d home directory**

· **-c comment**

· **-s shell**

**Let’s create a user with default attributes.**

· When no option is used with **useradd** command the options like **UID, GID, home dir** and **shell** will be assigned default.

· **#useradd <username>**

· **#useradd ktusr**



**Observe that the uid, gid, home dir, and shell is assigned automatically.**

**Let’s create a user with our own attributes**

· Create a user with following attributes

· **Name** = **ktuser2**

· **uid=505**

· **home dir = /home/kernel**

· **comment =salesman**

· **#useradd ktuser2 –u 505 -g 505 –d /home/kernel –c salesman**



**Assigning password to the user:**

· As a root user we can assign any password to any user

· The syntax for assigning a password is

· **#passwd** to assign password to current user ( the one with which you have logged in, if it is root then root’s password will be changed)

· **#passwd <user name>** to assign a password to a specific user, only root can assign password to other user.



**Modifying the user’s attribute**

· After creating a user if we need to modify the attributes of user like changing uid, changing secondary group id or adding a comment, locking or unlocking the user account, can be done by following command

· **Syntax. # usermod <options> <username>**

· **options are:**

· **all the options which are used with useradd command can be used and also the following,**

· **-l to change login name**

· **-L to LOCK account**

· **-U to UNLOCK account**

· **ex. # usermod -l newname oldname** (changing the name of the user)

· **ex. # usermod -L newname** to lock the user account

· **ex. # usermod -U newname** to unlock the user account

· **Note: - when an account is locked it will show! (Exclamation mark) in /etc/shadow file.**

**Locking and unlocking a user account:**

· To lock a user a/c use the following

· **#usermod –L < user name>**

· **#usermod –L ktuser2**

· Verify it in **/etc/shadow file**, it shows exclamation mark before user a/c or try login as ktuser2



**Unlocking a user a/c:**

· Unlock the above a/c

· **#usermod –U < user name >**

· **#usermod –U ktuser2**

· Verify it in **/etc/shadow file**, it shows exclamation mark before user a/c or try login as ktuser2



**The password parameters.**

· For any user we can set the parameters for the password, like **min** and **max password age, password expiration warnings** and **a/c expiration date** etc.

· To view the advanced parameters of the user, use

· **#chage -l < user name>**

· **#chage -l ktusr**



**Last password change:** When the password was change last time.

· **Password expires:** Password expiry date

· **Password inactive:** After password expiry grace period before the account gets locked.

· **Account expires:** Date on which the account expires.

· **Minimum number of days b/w password change:** once the password is changed, it cannot be changed until a min period of specified date. **[0]** means never.

· **Max number of days b/w password change:** After changing the password how long it will be valid for.

· **Number of days of warning before password expires:** start of warnings to change the password, no. of days before the password expires.

**#chage ktusr**



The second method is for, if you want to change a particular field of password aging policy

· **#chage <option> <value> <username>**

· **The options which can be used are as follows**

· **-m** for Min password age

· **-M** for Max password age

· **-d** for last time the password is changed.

· **-W** Password expiration warnings

· **-I Password inactive** [-1 means inactive]**.**

· **-E A/C expiration date**

· Let’s see how to change only the account expiration date



**Deleting a User:**

· To delete a user the syntax used is

· **#userdel <username>** it will only delete the user but home directory will be there. To delete the user with its home directory use the following command.

· **#userdel –r < user name >**

· **#userdel –r ktuser2**



**Creating a Group with default options :**

· To create a group the syntax is

· **#groupadd <name for the group>**

· **#groupadd ktgroup**



**Creating a group with user specified group id (GID)**

· **#groupadd <option> <name for the group>**

· **#groupadd -g 595 ktgroup**

· Verify it in **/etc/group**



**Modifying the properties of the group**

· To modify the group properties the syntax is

· **#groupmod <option> <arguments> <group name>**

· The options are

· -g to change the group id

· -o to override the previous assigned id, if it matches with the new one.

· -n to change the group name

**Changing the GID of the group**

· **#groupmod –g 600 ktgroup**

· Verify it in /**etc/group**



**Changing the name of the group**

· The syntax for changing the group name is

· **#groupmod –n <new name > < existing name >**

· **#groupmod –n kernelgrp ktgroup**



**Adding and Removing Members to a Group**

· Adding the members to the group is to add users to the group. To add the members to the group the syntaxes are

· **To add single user to the group**

· **#usermod –G <group name > < user name>**

· **#usermod –G ktgroup ktuser**



**Adding multiple single or multiple users to the group with various attributes**

· **#gpasswd < option> <arguments> <group name>**

· Options:

· **-M For Adding Multiple users to a group**

· **-A for Adding a group Administrator**

· **-a for Adding a single user to a group**

· **-d removing a user from a group**

· **#gpasswd –M <user>,<user>,<user> <group>**

· **#gpasswd –M ktuser2,ktuser3,ktuser4 ktgroup**



**Adding a single user using gpasswd**

· **#gpasswd –a ktuser ktgroup (**verify it in **/etc/group)**



**Making a user as a administrator**

· **#gpasswd –A ktuser ktgroup (verify it in /etc/gshadow)**



**Removing a user from the group**

· **#gpasswd –d ktuser2 ktgroup**



**Regular Expressions, Pipelines & I/O Redirections**

**Grep:**

Grep stands for **Global Regular Expression Print.** It is used to pick out the required expression from the file and print the output. If grep is combined with another command it can be used to pick out the selected word, phrase from the output of first command and print it.

**Examples of Grep:**

Let us pick the information about **root** from the file **/etc/passwd** (/etc/passwd contains information about all the users present in the system)

**#grep root /etc/passwd**



**To avoid case sensitivity of the word (i.e. the word may be uppercase of lowercase) use -i**

**#grep –i kernel ktfile** (lets grep the word **kernel** whether upper of lower case in the file **ktfile**)



**To display a word and 2 lines after the word:**

**#grep –nA2 wheel /etc/group**



**To display a word and 2 lines after the word:**

**#grep -nB2 wheel /etc/group**



**To display the things except the given word:**

**grep –v kernel ktfile**



**To display the searched word in color**

**#grep --color root /etc/passwd**

**Combining grep with other commands**

**# cat ktfile | grep –I kernel (pipe | is used to combine to commands)**

**#ls –l |grep –I ktfile**

**# ifconfi g |grep –I eth0**

**Like this we can combine grep with many commands which we will see in later chapters**

**Filter Commands:**

· Filter commands are used to filter the output so that the required things can easily be picked up. The commands which are used to filter the output are

**#less**

**#more**

**#head**

**#tail**

**#sort**

**#cut**

**#sed**

**less:-**

The **less** command is used to see the output line wise or page wise.

Ex: less /etc/passwd



**Note: -**press **Enter** key to scroll down line by line (or)

Use **d** to go to next page

Use **b** to go to previous page

Use **/** to search for a word in the file

Use **v** to go vi mode where you can edit the file and once you save it you will back to less command

**more:-**

**more** is exactly same like **less**

**Ex:** #more /etc/passwd

**Note: -**press **Enter** key to scroll down line by line (or)

Use **d** to go to next page

Use **/** to search for a word in the file

Use **v** to go vi mode where you can edit the file and once you save it you will back to more command

**head:**

It is used to display the top **10 lines** of the file.

**Ex:# head /etc/passwd**



**To display the custom lines**

**#head -n /etc/passwd (where n can be any number)**



**tail:**

It is used to display the **last 10** lines of the file

#tail /etc/passwd



**To display the custom lines**

#tail -n /etc/passwd (where n can be any number)



**Sort:**

**It is used to sort the output in numeric or alphabetic order**

**#sort filename**



**To sort the file according to numbers**

**#sort –d ktfile or #sort –h ktfile**



**To remove the duplicate entries from the output**

**#sort –u ktfile**



**cut command:**

**The cut command is used to pick the given expression (in columns) and display the output.**

**# cut -d -f filename** (where d stands for delimiter ex. : , “ “ etc and f stands for field)



**To delimit spaces and print the field**

**#cut –d “ “ –f1 filename**

**To delimit commas and print the field**

**#cut –d, -f1 filename**



**sed command:**

**sed** stands for **stream editor**, which is used to search a word in the file and replace it with the word required to be in the output

**Note**: it will only modify the output, but there will be no change in the original file.

**#sed ‘s/searchfor/replacewith/g’ filename**



**I/O Redirection:**

Redirection is a process where we can copy the output of any command(s), file(s) into a new file. There are two ways of redirecting the output into a file.

Using **>** or **>> filename** after the command, and

Using **tee** command

**Let’s see the > and >> option first**

**Syn: command > new file**

**Note: if the given name of the file is not available a new file will be created automatically. If the file already exists then it will overwrite contents of that file.**



Page **48** of **274 www.kerrneltech.com sed command:**

**sed** stands for **stream editor**, which is used to search a word in the file and replace it with the word required to be in the output

**Note**: it will only modify the output, but there will be no change in the original file.

**#sed ‘s/searchfor/replacewith/g’ filename**

**I/O Redirection:**

Redirection is a process where we can copy the output of any command(s), file(s) into a new file. There are two ways of redirecting the output into a file.

Using **>** or **>> filename** after the command, and

Using **tee** command

**Let’s see the > and >> option first**

**Syn: command > new file**

**Note: if the given name of the file is not available a new file will be created automatically. If the file already exists then it will overwrite contents of that file.**

**Appending another output in same the same file**



Likewise there are many options where we can use redirections

**Ex:**

**Copying contents of two files in a new file**

**#cat file1 file2 > file3**

**Using tee command:**

**The above options of redirections will not display any output, but directly save the output in a file. Using tee command will not only redirect the output to new file but it will also display the output.**

**Syn: cat <filename> | tee <new file name>**

**Note: if the given name of the file (newfile) is not available a new file will be created automatically. If the file already exists then it will overwrite contents of the file.**

**#cat ktfile |tee ktf1**



**Appending data in the same file using tee command**

**Syn: cat filename |tee –a filename2**

**#cat ktfile1 | tee –a ktf1**



**Find command:**

**find** command is used to find the files or directory’s path, it is exactly like the find option in windows where you can search for a file.

**Syntax: find / (under root) –option filename**

**Options that can be used with find command:**

|  |  |
| --- | --- |
| Option | **Usage** |
| **-name** | For searching a file with its name |
| **-inum** | For searching a file with particular inode number |
| **-type** | For searching a particular type of file |
| **-user** | For files whose owner is a particular user |
| **-group** | For files belonging to particular group |

**Finding a File with name**

**#find / -name Kernel Tech**



**Finding a file with its inode number**

**#find / -inum 5934**



**Finding the files, whose owner is a user called “ktuser”**

**#find / -user ktuser**



**Finding the files whose group is “ktgroup”**

**#find / -group ktgroup**



**File Permissions:**

**Permissions are applied on three levels:-**

· Owner or User level

· Group level

· Others level

**Access modes are of three types:-**

· r read only

· w write/edit/delete/append

· x execute/run a command

|  |  |  |
| --- | --- | --- |
| **Access modes are different on file and directory: Permissions** | **Files** | **Directory** |
| **r** | **Open the file** | **'ls' the contents of dir** |
| **w** | **Write, edit, append, delete file** | **Add/Del/Rename contents of dir** |
| **x** | **To run a command/shell script** | **To enter into dir using 'cd'** |



**Filetype+permission, links, owner, group name of owner, size in bytes, date of modification, file name**

**Permission can be set on any file/dir by two methods:-**

**1 Symbolic method (ugo)**

**2 Absolute methods (numbers)**

**1 Symbolic method (ugo):**

· Symbolic mode: General form of symbolic mode is:

**# chmod [who] [+/-/=] [permissions] file**

who 🡪 To whom the permissions to be assigned

**User/owner (u); group (g); others (o)**

**Example: -**

**Assigning different permissions to the file (user=rwx, group=rw and others=r)**

#chmod u=rwx,g=rw,o=r ktfile (where ktfile is the name of the file)



**Assigning full permission to the file i.e. rwx to all**

#chmod ugo=rwx <file name>



**#chmod u+x ktfile (Adding execute permission to user only)**

· **#chmod go-wx ktfile (Removing write and execute permissions from group and other)**

· **#chmod go+wx ktfile (Adding write and execute permissions from group and other)**

· **#chmod go=r ktfile (Giving only read permission to group and other)**

**2 Absolute Method (numbers)**

In Absolute method we use numbers instead of using symbols i.e.

· **Read=4**

· **Write=2**

· **Execute=1**

**Assigning different permissions to the file (user=rwx, group=rw and others=r)**

**#chmod 764 ktfile (where 7 means rwx i.e. 4+2+1, rw=6 i.e. 4+2 and 1 indicates x)**



**Assigning full permission to the file i.e. rwx to all**

**#chmod 777 ktfile**



**Likewise you can give different permissions according to your requirement**

**Removing all permissions from others**

#chmod 770 ktfile (where **0** indicates **no** permissions)

**Note: All the above permissions and procedure is same for files and directories.**

**Umask:**

When we create any file using touch, cat or vi commands they get created with default file permissions as stored in umask **(User file creation mask)**.umask is a 4 digit octal number which tells Unix which of the three permissions are to be denied rather than granted. Umask will decide that what should be the default permissions for a file and directory when it is created.

**The default umask value is 0022**

**#umask**



**Calculation of default permissions for file and directory, basing upon the umask value**

**Note:** For a file by default it cannot have the execute permission, so the maximum full permission for a file at the time of creation can be **666** (i.e. 777 -111 = 666), whereas a directory can have full permissions i.e. **777**

· **The full permission for the file 666**

· **Minus the umask value -022**

· **The default permission for file is 644 (rw-,r--,r--)**



**The full permission for the directory 777**

· **Minus the umask value - 022**

· **The default permission for file is 755 (rwx, r-x, r-x)**



Sticky BIT:

Command:

Chmod o+t <dirname>

ACL permissions:

Create one dir:

Mkdir /acltest

Setting the permission for a user:

setfacl -m u:test5: rwx /acltest

getfacl /acltest

**Modifying the umask value:**

**#umask 002**

The Modified default Permission for a **file** will be **666-002=664** i.e. **rw,rw,r,** and for the **directory** it will be **777-002=775** i.e. **rwx,rwx,r-x.**



**MANAGING PARTITIONS & FILE SYSTEMS**

**What is a partition?**

Partitioning is a means to divide a single hard drive into many logical drives. A partition is a contiguous set of blocks on a drive that are treated as an independent disk. A partition table is an index that relates sections of the hard drive to partitions.

**Why have multiple partitions?**

· Encapsulate your data. Since file system corruption is local to a partition, you stand to lose only some of your data if an accident occurs.

· Increase disk space efficiency. You can format partitions with varying block sizes, depending on your usage. If your data is in a large number of small files (less than 1k) and your partition uses 4k sized blocks, you are wasting 3k for every file. In general, you waste on average one half of a block for every file, so matching block size to the average size of your files is important if you have many files.

· Limit data growth. Runaway processes or maniacal users can consume so much disk space that the operating system no longer has room on the hard drive for its bookkeeping operations. This will lead to disaster. By segregating space, you ensure that things other than the operating system die when allocated disk space is exhausted.

**Disk Partitioning Criteria:**

**M EXTENDED**

**B P P P**

**R L L FREE**

**MBR = MASTER BOOT RECORD**

**P= PRIMARY PARTITION**

**EXTENDED= EXTENDED PARTITION**

**L= LOGICAL PARTITION**

**FREE= FREE SPACE**

**The Structure of Disk Partition**

· On the disk where O/S is installed, will have the first partition as **MBR.**

· **MBR** is a Master Boot Record, which contains two important utilities, **IPL** (Initial Program Loader) and **PTI** (Partition Table information)

· **IPL is** responsible for booting the operating the system, because it contains the **boot loader.**

· In earlier versions of Linux i.e. up to **RHEL 4,** the default boot loader was **LILO** (Linux Loader). But, since **RHEL5** onwards it has been changed to **GRub** (Grand Unified Boot loader), which is far more superior to **LILO.**

· The **PTI** (Partition Table information) is the information about the number of partitions on the disk, sizes of the partition and types of partitions.

**THE CRITERIA OF DISK PARTITIONING:**

· Every disk can have only **3** Primary Partitions.

· **Primary Partition** is a partition which usually holds the **operating system**. Only **one** amongst the 3 primary partitions can be active which will be booted by **MBR** to load the operating system.

· **Extended Partition** is a special type of primary partition which can be subdivided into multiple logical partitions. As there can be only 3 primary partitions per disk, and if the user is required to make further partitions then all the space remaining on the disk should be allocated to extended partition, which can be used to create the logical partitions later. There can be only **one extended partition** per disk.

· **Logical partitions** are the partitions which are created under extended partition, all the space in the extended partition can be used to create any number of logical partitions.

**Disk Identification:**

Different type of disks will be having different initials in Linux

∙ **IDE** drive will be shown as **/dev/hda**

· **SCSI** drive will be shown as **/dev/sda**

· **Virtual** drive will be shown as **/dev/vda**

**FILE SYSTEM:**

· It is method of storing the data in an organized fashion on the disk. Every partition on the disk except **MBR** and **Extended partition** should be assigned with some file system in order to make them store the data. File system is applied on the partition by formatting it with a particular type of file system.

**Types of file systems used in RHEL 6:**

· The file systems supported in Linux are **ext2, ext3 and in RHEL 6 ext4, vfat, etc.**

· **Ext** file system is the widely used file system in Linux, whereas vfat is the file system to maintain a common storage between **Linux and windows** ( in case of multiple o/s ‘

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | **EXT2** | **EXT3** | **EXT4** |
| 1. | Stands for Second Extended File System | Stands for Third Extended File System | Stands for Fouth Extended File System |
| 2. | It was introduced in 1993 | It was introduced in 2001 | It was introduced in 2008. |
| 3. | Does not have journaling feature. | Supports Journaling  Feature. | Supports Journaling  Feature. |
| 4. | Maximum File size can be from **16 GB to 2 TB** | Maximum File Size can be from **16 GB to 2 TB** | Maximum File Size can be from **16 GB to 16 TB** |
| 5. | Maximum ext2 file system size can be from **2 TB to 32 TB** | Maximum ext3 file system size can be from **2 TB to 32 TB** | Maximum ext4 file system size is **1 EB (Exabyte). 1 EB = 1024 PB (Petabyte). 1 PB = 1024 TB (Terabyte).** |
| 6. | Cannot convert ext file system to ext2. | You can convert an ext2 file system to ext3 file system directly (without backup/restore). | All previous ext file systems can easily be converted into ext4 file system. You can also mount an existing ext3 f/s as ext4 f/s (without having to upgrade it). |

**MOUNTING:-**

· Attaching a directory to the file system in order to access the partition and its file system is known as mounting.

· The mount point is the directory (usually an empty one) in the currently accessible file system to which a additional file system is mounted.

· The /mnt directory exists by default on all Unix-like systems. It, or usually its subdirectories (such as /mnt/floppy and /mnt/usb), are intended specifically for use as mount points for removable media such as CDROMs, USB key drives and floppy disks.

**Files which is related to mounting in Linux:**

· **/etc/mtab** is a file which stores the information of all the currently mounted file systems; it is dynamic and keeps changing.

· **/etc/fstab** is the file which is keeps information about the permanent mount point. If you want to make your mount point permanent, so that it will be mounted even after reboot, then you need to make an appropriate entry in this file.

***LAB WORK:-***

**To view the existing partitions**

#fdisk –l or parted –l





**Partition Administration using fdisk**

To enter into disk utility, the syntax is

**#fdisk <disk name>**

**#fdisk /dev/sda**



Use **m** to list out various options that can be used in fdisk

**Creating a new partition**

**#fdisk /dev/sda**

· Use **p** to list out the partition information first and

· Use **n** to create a new partition.



**Now use n to create a new partition and verify it again with p**



**Deleting a partition**

Let’s delete the partition we’ve created above i.e. /dev/sda7

· Use **d** to delete a partition and specify the device name, in our case it is **7.**



**Note:** Never delete the system partitions i.e. **1-7**

**Saving the partition changes**

Every time you make a partition or delete a partition, the changes made has to be saved using **w,** otherwise the creation and deletion will not be considered to be happen. For practice purpose you can make any no. of partition and delete it and just quit using **q** so that it will not be saved.



**Updating the partition table without restarting the system**

After creating or deleting a partition the changes will be effected in the partition table only after the restart of the system. But there is a way to avoid this circumstance. We can use **partprobe or partx** command to update the partition information without restarting the system

**#partprobe /dev/sda**

Or

**#partx –a /dev/sda**

Or

**#kpartx /dev/sda**

**Formatting a partition with ext4 filesystem**

After creating a partition we need to assign some file system to it so that we can start storing the data into it. To format a partition the following syntax is used.

**# mkfs.<file system type> <partition name>**

**#mkfs.ext4 /dev/sda7** (where sda7 is our newly created partition)



Likewise you can format the different partitions with different file systems like

· #mkfs.ext3 /dev/sda8

· #mkfs.vfat /dev/sda9

**Note: Even after formatting the partition we cannot add the data into the partition. In order to add the data in the partition it is required to be mounted.**

**Mounting a partition**

Mounting is a procedure where we attach a directory to the file system. There are two types of mounting which will be used in Linux or any UNIX.

· **Temporary Mounting**

· **Permanent Mounting**

**Temporary Mounting**

In a temporary mount point we will create a directory and mount it, but this mount point will last only till the system is up, once it is rebooted the mounting will be lost.

Syntax:

**#mount <device name> <directory name (mount point)>**

**#mount /dev/sda7 /kernel**



**To View all the mounted partitions**

**#mount**



Now we have successfully mounted the partition we can access it and can store the data

· To add the data access the mount point

· #cd /kernel

· Add the data and exit the directory

**Unmounting a partition**

**#umount <mount point directory>**

**#umount /kernel**

verify it with **mount** command.

**Permanent Mounting**

Permanent mounting procedure is exactly same like temp mounting, but here we will update the **/etc/fstab** file with the mounting details, so that it will be mounted even after the system is reboot.

**Steps To make a permanent mount point:**

· Make a directory or use an existing directory

· Add entry in **/etc/fstab** file

· Use **mount –a** command to check it is mounting. (mount –a will mount all the entry placed in **/etc/fstab)**

Here we will be using our existing **/kernel** directory as mount point which is created previously.

**#vim /etc/fstab**



**mount –a**



You can now access the directory and add, delete or modify the contents and can also unmount the file system at any point

Sometimes a directory reflects error while unmouting, the possible causes for it are

· You are in the same directory and trying to unmount it. Check with **pwd** command

· Some users are present in the directory and using the contents in it.

· Check with **fuser –cu /dev/sda7**



Check for the files which are open with **lsof /dev/sda7**

· Kill the open connections using **fuser –ck /kernel/hello** where hello is the file which is open.

· Now you can use **umount** command to unmount the file system.

**To view the usage information of mounted partition:**

To view the usage information of mounted partition use the command **df –h**

**#df -h**



**To view the size of a file or directory**

To view the size of the file or directory uses the command **du –h file or directory name.**

**Mounting a partition permanently with its block id (UUID)**

· To check the uuid of a partition use **blkid /dev/sda7** command.

· Copy the uuid

· Make an entry in **/etc/fstab** using UUID

· Verify it with **mount –a** option



**vim /etc/fstab**



Now mount it with **mount –a** command and verify it with **mount** command

**Creating a Swap Partition:**

Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM. Swap space is located on hard drives, which have a slower access time than physical memory.

|  |  |
| --- | --- |
| **Recommended System Swap Space Amount of RAM in the System** | **Recommended Amount of Swap Space** |
| **4GB of RAM or less** | **a minimum of 2GB of swap space** |
| **4GB to 16GB of RAM** | **a minimum of 4GB of swap space** |
| **16GB to 64GB of RAM** | **a minimum of 8GB of swap space** |
| **64GB to 256GB of RAM** | **a minimum of 16GB of swap space** |
| **256GB to 512GB of RAM** | **a minimum of 32GB of swap space** |

**Commands to be used in maintaining Swap spaces**

· To see the memory size and the swap space size

**#free –m**

· To see the swap usage use

**#swapon –s**

· To format the partition with swap file system use

**#mkswap <parititon name>**

· To activate the swap space use

**#swapon <partition name>**

· To deactivate the swap space use

**#swapoff <partition name>**

**Creating a Swap partion**

· Create a normal partition using fdisk and change hex code to make it swap partition.

· The hex code for SWAP is **82.** (To change the use **t** in fdisk and list all the hex code use **l**)

· Update the partition table using **partx –a** command



**Format the partition with swap file system**

**#mkswap /dev/sda6**



**Turn on the newly created swap space and verify it.**

· To turn on the swap space the syntax is

**#swapon /dev/sda6**



**Making the Newly Created SWAP Partition to mount after reboot.**

· In order to make the swap partition mount automatic after reboot, we need to make an entry in **/etc/fstab** file.

**#vim /etc/fstab**



**Removing the SWAP Partition**

· Deactivate the swap partition

**#swapoff <device name>**

· Remove the entry from **/etc/fstab.**

· Delete the partition through **fdisk**

**Logical Volume Management**

The Linux Logical Volume Manager (LVM) is a mechanism to virtualize the disks. It can create "virtual" disk partitions out of one or more physical hard drives, allowing you to grow, shrink, or move those partitions from drive to drive as your needs change. It also allows you to create larger partitions than you could achieve with a single drive. Traditional uses of LVM have included databases and company file servers, but even home users may want large partitions for music or video collections, or for storing online backups. LVM can also be convenient ways to gain redundancy without sacrificing flexibility.

A typical example for the need of LVM can be, assuming that we are having a disk of size 2GB and we start adding the data in the form of a single file, eventually it grows to the size of 2GB. In this case the possibility is, you go for another disk which is larger than 2GB, let’s say 4GB. But what if the file again grows more than 4GB? How far you will be migrating file from one disk to another so on and so forth? It requires a down time as well which is not possible in real time, so to avoid these circumstances we implement LVM and store data in LV’s whose size can be easily increased whenever required without a downtime.



Above picture shows the structure of LVM. LVM consists of **Physical Volumes, Volume Group, Logical Volumes** and finally **file systems.** The Physical partitions are known as **Physical Extents (PE),** and the logical partitions are known as **logical Extents (LE)**

**Components of LVM in Linux:**

· Physical Volumes (PV)

· Physical Extent (PE)

· Volume Group (VG)

· Logical Volume (LV)

· Logical Extent (LE)

**Physical Volume (PV)**

It is the standard partition that you add to the LVM. Normally, a physical volume is a standard primary or logical partition with the hex code **8e**.

**Physical Extent (PE)**

It is a chunk of disk space. Every PV is divided into a number of equal sized PEs.

**Volume Group (VG)**

It **is** composed of a group of PV’s and LV’s. It is the organizational group for LVM.

**Logical Volume (LV)** is composed of a group of LEs. You can format and mount any file system on an LV. The size of these LV’s can easily be increased or decreased as per the requirement.

**Logical Extent (LE)**

It is also a chunk of disk space. Every LE is mapped to a specific PE.

|  |  |
| --- | --- |
| **LVM Command** | **Function** |
| **pvs** | **Displays all the physical volumes** |
| **vgs** | **Displays all volume groups in the system** |
| **lvs** | **Displays all the logical volumes in the system** |
| **pvdisplay** | **Displays detailed information on physical volumes** |
| **vgdisplay** | **Displays detailed information on volume groups** |
| **lvdisplay** | **Displays detailed information on logical volumes** |
| **pvcreate** | **Create a new physical volume** |
| **vgcreate** | **Create a new volume group.** |
| **lvcreate** | **Creates a new logical volume** |
| **vgextend** | **Add a new physical disk to a volume group.** |
| **lvextend** | **Extends a logical volume** |
| **lvresize** | **Resizes a logical volume** |
| **lvreduce** | **Reduces a logical volume** |
| **pvmove** | **Moves/migrates data from one physical volume to another** |
| **vgreduce** | **Reduces a volume group by removing a PV from it.** |
| **pvremove** | **Deletes a physical volume** |
| **vgremove** | **Removes /Deletes a volume group** |
| **lvremove** | **Removes /Deletes a logical volume** |
|  |  |

Commands:

Creating physical volume:



Creating volume group:





Creating logical volume:



Checking logical volume creation







Issue blkid command

Where you will see the UUID can be mounted respectively.

Extending the mount point with LVM:



First creating the lvm for /dev/xvdb:





blkid



Mounting to lvm dir:



Now extending the LVM:

Umount the /lvm



Checking the new disk added:

Parted -l



Adding the /dev/xvdc to the lvm









Need to format: /dev/vg1/lv1

And remount:



For deleting the lvm:

Lvremove

Vgremove

Pvremove

**Increasing the disk size**:

First need to modify the volume & increase disk space:

Command:

growpart /dev/DEVICE\_NAME    PARTITION\_NUMBER

check fdisk -l

will increase the disk space accordingly.

**Network Configuration & Trouble Shooting:**

**Important SELinux configuration Files**

**/etc/selinux/config** is the main configuration file of **SELinux.**

**/etc/sysconfig/selinux** contains a symbolic link to the actual configuration file, /etc/selinux/config.



**Modes of SELinux**

· There are three modes in which SELinux can be at a time, they are

· **Enforcing**, **Permissive** and **Disabled**

· **Enforcing**

Enable and enforce the SELinux security policy on the system, denying access and logging actions

· **Permissive**

Permissive mode is similar to Debugging Mode. In Permissive Mode, SELinux policies and rules are applied to subjects and objects, but actions ( for example, Access Control denials) are not affected. The biggest advantage of Permissive Mode is that log files and error messages are generated based on the SELinux policy implemented.

· **Disabled**

SELinux is turned off and no warn and log messages will be generated and stored.

**Booleans**

· Booleans are variables that can either be set as true or false. Booleans enhance the effect of SELinux policies by letting the system administrator fine tune a policy. A policy may protect a certain daemon or service by applying various access control rules. In real world scenarios, a system administrator would not like to implement all the access controls specified in the policy.

**SELinux Policy**

· The SELinux Policy is the set of rules that guide the SELinux security engine. It defines *types* for file objects and *domains* for processes. It uses roles to limit the domains that can be entered, and has user identities to specify the roles that can be attained. In essence, types and domains are equivalent, the difference being that types apply to objects while domains apply to processes.

**SELinux Context**

· Processes and files are labeled with a SELinux context that contains additional information, such as a SELinux user, role, type, and, optionally, a level.

***LAB WORK:-***

**To check the SELinux Mode**

**#getenforce**



**Sestatus**



**BOOTING PROCEDURE AND KERNEL PARAMETER**

Press the power button on your system, and after few moments you see the Linux login prompt.

Have you ever wondered what happens behind the scenes from the time you press the power button until the Linux login prompt appears?



**1. BIOS**

· BIOS stands for Basic Input/Output System

· Performs some system integrity checks

· Searches, loads, and executes the boot loader program.

· It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.

· Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.

· So, in simple terms BIOS loads and executes the MBR boot loader.

**2. MBR**

· MBR stands for Master Boot Record.

· It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda

· MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.

· It contains information about GRUB (or LILO in old systems).

· So, in simple terms MBR loads and executes the GRUB boot loader.

**3. GRUB**

· GRUB stands for Grand Unified Bootloader.

· If you have multiple kernel images installed on your system, you can choose which one to be executed.

· GRUB displays a splash screen, waits for few seconds, if you don’t enter anything, it loads the default kernel image as specified in the grub configuration file.

· GRUB has the knowledge of the filesystem (the older Linux loader LILO didn’t understand filesystem).

· Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf



**4. Kernel**

· Mounts the root file system as specified in the “root=” in grub.conf

· Kernel executes the /sbin/init program

Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.

· initrd stands for Initial RAM Disk.

· initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

**5. Init**

1. Looks at the /etc/inittab file to decide the Linux run level.

2. Following are the available run levels

· 0 – halt

· 1 – Single user mode

· 2 – Multiuser, without NFS

· 3 – Full multiuser mode

· 4 – unused

· 5 – X11

· 6 – reboot

3. Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.

4. Execute ‘grep initdefault /etc/inittab’ on your system to identify the default run level

5. If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.

6. Typically you would set the default run level to either 3 or 5.

**6. Runlevel programs**

· When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail …. OK”. Those are the run level programs, executed from the run level directory as defined by your run level.

· Depending on your default init level setting, the system will execute the programs from one of the following directories.

· Run level 0 – /etc/rc.d/rc0.d/

· Run level 1 – /etc/rc.d/rc1.d/

· Run level 2 – /etc/rc.d/rc2.d/

· Run level 3 – /etc/rc.d/rc3.d/

· Run level 4 – /etc/rc.d/rc4.d/

· Run level 5 – /etc/rc.d/rc5.d/

· Run level 6 – /etc/rc.d/rc6.d/

Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.

***LAB WORK:-***

**To check the default run level in linux**

· To see the default run level in linux the command is

**#who –r**



**Changing the default run level to some other like 3**

· To change the run level edit the **/etc/inittab** and make the following changes

**#vim /etc/inittab**



Now reboot the system and check in which runlevel it is.

**#init 6**



To start the graphical interface when you are in runlevel **3**, use the following command

**#startx**

· Change it back to runlevel **5** and reboot the system.

**To see the details regarding the kernel installed**

· To see the version of the kernel use

**#uname –r**



To see the same thing with more details use

**#uname –a**



**Note: The same information can be seen in /boot/grub/grub.conf**



**To check the architecture of the O/S**

· To check the architecture of the O/S the command is

**#arch**

**#uname –m**



**To check the version of the O/S in the system**

· To check the O/S version you have to navigate to the following file

**# cat /etc/redhat-release**



**JOB AUTOMATION**

**Automation with cron and at**

· In any operating system, it is possible to create jobs that you want to reoccur. This process, known as ***job scheduling***, is usually done based on user-defined jobs. For Red Hat or any other Linux, this process is handled by the cron service or a daemon called **crond**, which can be used to schedule tasks (also called *jobs*). By default, Red Hat comes with a set of predefined jobs that occur on the system (hourly, daily, weekly, monthly, and with arbitrary periodicity). As an administrator, however, you can define your own jobs and allow your users to create them as well.

· The importance of the job scheduling is that the critical tasks like taking backups, which the clients usually wants to be taken in nights, can easily be performed without the intervention of the administrator by scheduling a cron job. If the cron job is scheduled carefully than the backup will be taken at any given time of the client and there will be no need for the administrator to remain back at nights to take the backup.

**Important Files related to cron and at**

· **/etc/crontab** is the file which stores all scheduled jobs

· **/etc/cron.deny** is the file used to restrict the users from using cron jobs.

· **/etc/cron.allow** is used to allow only users whose names are mentioned in this file to use cron jobs. (this file does not exist by default)

· **/etc/at.deny** same as cron.deny for restricting at jobs

· **/etc/at.allow** same as cron.allow for allowing user to use at jobs.

**Crontab format**

· To assign a job in the Crontab file the format used is the following



|  |  |
| --- | --- |
| **Options** | **Explanation** |
| **\*** | **Is treated as a wild card. Meaning any possible value.** |
| **\*/5** | **Is treated as ever 5 minutes, hours, days, or months. Replacing the 5 with another numerical value will change this option.** |
| **2,4,6** | **Treated as an OR, so if placed in the hours, this could mean at 2, 4, or 6 o-clock.** |
| **9-17** | **Treats for any value between 9 and 17. So if placed in day of month this would be days 9 through 17. Or if put in hours it would be between 9 and 5.** |

|  |  |
| --- | --- |
| **Crontab Commands Command** | **Explanation** |
| **crontab –e** | Edit your crontab file, or create one if it doesn’t already exist. |
| **crontab –l** | Display your crontab file. |
| **crontab –r** | Remove your crontab file. |
| **crontab -u** | If combined with **–e,** edit a particular user’s Crontab file and if combined with **–l,** display a particular user’s crontab file. If combined with **–r,** deletes a particular user’s Crontab file |

***LAB WORK:-***

**CRON JOBS:**

**To check the assigned cron jobs of currently logged in user**

· To check the cron jobs the command is

**#crontab –l**



**To check the cron jobs of a particular user**

· To check a user’s cron jobs, the syntax is

**#crontab –l –u <user name>**

**#crontab –l –u ktuser**



**Setting a job to display the current date for every minute on present console**

· To set the above job the steps are

· Check the console on which you are working by following command

**#tty**



**Note: /dev/pts/1** is the console address

· Schedule the task as shown below

**#crontab –e** and enter the field as shown below and save it as in **VI editor**



**Note:** where **\*** means every possible value.

Restart the cron services

**#service crond restart**



Wait for a minute and check whether time is displaying or not. Every min time will be displayed as below.



**Schedule a cron job to create a directory “ktdir” under “/root” on “Sunday 22 October at 1:30 AM”**

· To schedule above job edit the crontab file as shown below and restart the service

**#crontab –e**







**Note:** you can use **0** or **7** for **Sunday.**

Check whether it got created or not on scheduled day, if it created you can see the directory otherwise a error mail will be generated to your mail.

**Schedule a job to run the backup script “bkpscript.sh” on every “Saturday 12:30 PM”**

· In order to schedule above job the steps are.

· Check the location of script and also check whether it is having execute permission or not. If not then add the execute permissions to all user on it.



Apply the job in **crontab** and restart the service

**#crontab –l**







**Note: !ser** is the command to restart the last restarted service

**Schedule a job so that a user “ktuser” should get a mail regarding meeting on 24th, 29th and 31st October at 2:25 PM.**

· To set above task edit the crontab in following passion, and restart the service

**#crontab –e –u <user name>**

**#crontab –e –u ktuser**







**Schedule a job so that a user “ktuser” should get the mail from 15th to 20th and 25th to 30st November as a reminder of some session at 2:25 PM**

· This task is very much similar to the previous one but there is only a small change in format.

**#crontab –e –u ktuser**







There are still various method you can schedule the cron jobs, Do some **R&D** on it to find out more.

**Restrict users “ktuser” “amit” “vivek” from using cron jobs**

· To restrict any user from using cron job facility, enter their names in **/etc/cron.deny** and save it

**#vim /etc/cron.deny**





Now login as one of those users and try to use crontab.



If again want to allow them to use cron job facilities just remove their names from **/etc/cron.deny** file.

If again want to allow them to use cron job facilities just remove their names from **/etc/cron.deny** file.

**Allow only two users “musab” and “rahul” to use cron jobs out of all the users in the system**

· Assuming that we have 100 users in our system, putting all 98 names in **/etc/cron.deny** file is a time consuming process. Instead of that, we can create one more file **/etc/cron.allow,** in which we can assign names of those users who are allowed to use cron jobs.

· Remove the **/etc/cron.deny** file and create **/etc/cron.allow,** still if both files are existing **cron.allow** file will be having precedence over **cron.deny** file. Just to avoid confusion it is good to remove **cron.deny** file

**Note: /etc/cron.deny** file exists by default, but we need to create **/cron.allow** file. If your name is not there in **cron.allow** file then you will not be allowed to use cron jobs, and as mentioned above, if both files are existing **cron.allow** file will be having precedence over **cron.deny** file. If neither **cron.deny** nor **cron.allow** files exists then only **root** can use cron jobs.

· Now, let’s put those two users “musab” and “rahul” name in **/etc/cron.allow** file and check the results.

**#vim /etc/cron.allow**







**ADMINISTRATING REMOTE SYSTEM**

· **Remote shell Access using SSH**

**What Is SSH?**

There are a couple of ways that you can access a shell (command line) remotely on most Linux/Unix systems. One of the older ways is to use the telnet program, which is available on most network capable operating systems. Accessing a shell account through the telnet method though poses a danger in that everything that you send or receive over that telnet session is visible in plain text on your local network, and the local network of the machine you are connecting to. So anyone who can "sniff" the connection in-between can see your username, password, email that you read, and command that you run. For these reasons you need a more sophisticated program than telnet to connect to a remote host.

SSH, which is an acronym for Secure SHell, was designed and created to provide the best security when accessing another computer remotely. Not only does it encrypt the session, it also provides better authentication facilities.

SSH configuration file is **/etc/ssh/sshd\_config**

· SSH demon or service is **sshd**

***LAB WORK:-***

**Accessing the remote machine using ssh**

· To access the remote machine using ssh, the syntax is

**#ssh <ip address/ host name of remote machine>**

**Note:** hostname can only be used when the hostname is saved in **/etc/hosts** file or, if **DNS** is configured.

**#ssh 192.168.10.98**



The first time around it will ask you if you wish to add the remote host to a list of known\_hosts, go ahead and say **yes.**

· Enter the password of the remote system correctly, once logged in check hostname and ip address to confirm login.





To leave the session, just type exit or logout command and you will be back to your own machine through which you are logged in.



**Password less login using SSH keys**

· As a system administrator, one person will be assigned to manage many systems, for example one person has to manage more than 10 systems at a time. In this situation admin has to transfer some files from one system to another 9 systems or vice versa, for every login on remote system it will prompt for password. Even for transferring files for every transfer we need to enter the password.

· Above situation will be very annoying for system admin to type password for every step. Therefore SSH provides a best way to escape password prompting every now and then.

· By generating SSH keys, a public key and a private key, an admin can copy the public key into other system and done, it will work as authorized access from the admin’s system. Now whenever we are logging from admin’s system to other system in which we have stored the public key of admin’s system, it will not prompt us for password and we can login to that system as many time as we want without being prompt for the password.

· SSH keys are an implementation of public-key cryptography. They solve the problem of brute-force password attacks by making them computationally impractical.

· Public key cryptography uses a *public key* to **encrypt data** and a *private key* to **decrypt it.**

***LAB WORK***

**Generating SSH key pair**

· To generate the SSH key pair, the syntax is

**#** ssh-keygen



It will prompt above to mention the file where these keys shoud be stored, to keep its default directory just press “**Enter”.** The default location will be **/root/.ssh/** directory



Now it will ask for passphrase, which will be used instead of password. The passphrase will only be asked once per session. Enter your desired passphrase twice as shown on next page, and press enter.



Okay now our keys are successfully generated, go to **/root/.ssh/** directory and check for the keys.

**#cd /root/.ssh**



The **id\_rsa** is a private key and **id \_rsa.pub** is the public key which will be used later to make password less login.

**Copying the public key on Client system**

· To copy the server’s public key in client system, the command is

**#ssh-copy-id –i <public key location> <clients IP address>** (or user @ client IP)

**#ssh-copy-id -i /root/.ssh/id\_rsa.pub 192.168.10.95**



Enter the password of the client to proceed, check it on client side whether it is copied or not

**Move to client system and check whether the key is copied properly or not**

· To check the key navigate to **/root/.ssh/** directory and check for **authorzed\_keys** file which will hold all the system which are authorized and will not be asked for password..

**#cd /root/.ssh/**

**#cat authorized\_keys**



**Try login to the client machine using SSH, check whether it is asking for password**

· For logging into client machine the procedure is same as shown earlier. But as we have assigned a passphrase it will ask us for it. Once you enter a passphrase it will last until you logged out of server’s session. Let’s see it practically.

**Remote file transfer with SCP and RSYNC**

**SCP (SECURE COPY)**

· scp stands for secure cp (copy), which means that you can copy files across an ssh connection that will be encrypted, and therefore secured. As scp will be using ssh protocol to transfer the data, hence it is termed as the safest method of transferring data from one location to another.

***LAB WORK:***

**To copy a file using SCP to remote machine from source location**

· We are having a file **ktfile** in “**/”** directory, in the server **ktlinux.kt.com** who’s IP is **192.168.10.98,** and we need to copy the same in **other** server’s i.e. **ktcl5.kt.com** with an IP **192.168.10.95, /root directory. Then,**

· The syntax for SCP a file from source location.

**#scp <file name > <remote hosts IP >:/<location to copy the file >**

**#scp /ktfile 192.168.10.95:/root/**



Now log in to destination system and check whether if the file is there.



**To copy a file using SCP from a remote machine being in destination’s location**

· Let’s reverse the previous task, login to **ktcl5** machine whose **IP** is 192.168.10.95, and transfer a file from **ktlinux** machine whose **IP** is **192.168.10.98**

· Let’s first remove the earlier copied file **ktfile,** then copy it again from destination’s location.

· The syntax for SCP a file from destination location.

**#scp <source system’s IP>:/<location of file to be copied> <destination location to copy>**

**ENHANCED USER SECURITY WITH SUDO**

**SUDO**

· Sudo stands for either "substitute user do" or "super user do" (depending upon how you want to look at it). What sudo does is incredibly important and crucial to many Linux distributions. Effectively, sudo allows a user to run a program as another user (most often the root user). There are many that think sudo is the best way to achieve "best practice security" on Linux

· Users can login using their username and password and can issue administrative commands placing sudo in front of the commands, e.g. sudo rpm -Uvh \*.rpm , to run the command which installs and updates programs in Linux (rpm).



The file **/etc/sudoers** file has the rules that users have to follow when using sudo command. That means that whatever commands access is provided to any user in **/etc/sudoers** file, that user can only run those commands.

· Do not edit the **/etc/sudoers** directly; instead use **"visudo"** command to edit the sudoers file. There are two reasons for that- it prevents two users from editing the file at the same time, and it also provides limited syntax checking. Even if you are the only root user, you need the syntax checking, so use "visudo".

**Advantages of using SUDO**

**Two of the best advantages about using sudo are:**

· **Limited user privileges**

As we have studied above that we can restrict users to use certain commands as a privileged user as per the role of the user.

**E.g.:** Networking commands for Network user and Admin commands for Admin users etc.

∙ **Logs of the actions done by users**

All commands executed by sudo users will be stored in **/var/log/secure** file, but still if you want you can make your own log file by passing an entry in **/etc/sudoers** file at the bottom as **“Defaults logfile=/var/log/sudo.log”** or whatever name you want, to save the logs of what commands is executed by which sudo user.

**The /etc/sudoers file**

· As we learnt above that it is the configuration file for sudo users, which is used to assign specific commands to the specific users or groups.

· Always use **visudo** command to edit this file. it prevents two users from editing the file at the same time, and it also provides limited syntax checking .

· When you run **visudo** command the output will be as follows



As you can see there is basically one line

· **root ALL=(ALL) ALL**

· This lines means that the user root can execute from ALL terminals, acting as ALL (any) users, and run ALL (any) command.

· So the first part is the **user**, the second is the **terminal** from where the user can use sudo, the third is **as which user he may act**, and the last one, is which **commands** he may run.

· The advantage of **visudo** command , while editing if there are any syntax error it will be reflected as follows



***LAB WORK:-***

**Allow a user “ktuser” all privileges like root**

· To assign root privileges to user add a line by using sudoers file as shown below.

**#visudo** (save the sudoers file as we save a vim file using **“wq!”)**



Otherway way adding the users in sudo file:

**Allow a group called ktgroup, all root previleges.**

· Let’s first check the members of ktgroup and then apply root previleges.

**#tail /etc/gshadow**



Okay as we know the users in ktgroup, let’s assign it root previleges.

**#visudo** and look for the below line.



Now, login as one of the user of ktgroup try root commands



**Allow a user “ktuser2” to run all commands without prompting for his password any time.**

· To allow run all commands, the syntax we have already seen, but allow him run command’s without prompting password a small change is to be made,



**SOFTWARE MANAGEMENT**

**To manage the software in Linux, two utilities are used**,

**1. RPM – REDHAT PACKAGE MANAGER**

**2. YUM – YELLOWDOG UPDATER MODIFIED**

**RPM –REDHAT PACKAGE MANAGER**

RPM is a package managing system (collection of tools to manage software packages). RPM is a powerful software management tool for installing, uninstalling, verifying, querying and updating software packages. RPM is a straight forward program to perform the above software management tasks.

**Features:**

· RPM can verify software packages.

· RPM can be served as a powerful search engine to search for software’s.

· Components, software’s etc can be upgraded using RPM without having to reinstall them

· Installing, reinstalling can be done with ease using RPM

· During updates RPM handles configuration files carefully, so that the customization is not lost.

***LAB WORK:-***

**To check all the installed packages in the system**

· To check all the installed packages in the system, the syntax is

· **#rpm –qa** (where **q** stands for query, and **a** stands for all)





**To check whether a particular package is installed or not**

· To check whether a package is installed or not out of the list of installed package, the syntax is

**#rpm –qa <package name> or**

**#rpm –q < package name>**

**#rpm –qa vsftpd** or **#rpm –q vsftpd**



One more method of checking the installed package, when you are not sure about the package name, like whether it starts with capital letter and full name etc.

**#rpm –qa | grep –i < package name>**

**#rpm –qa |grep –i vsft\***



**To check whether a package is consistent or not, before installing it. (Testing the installation)**

· To check the package’s consistency,

· Move to the directory where you have kept the rpm package which you wish to install



The command used to check the package’s consistency is

**#rpm –ivh - -test <package name>**

Where i = install, v= verbose view, and h = hash progress.

**#rpm –ivh - - test finger-0.17-39.el6.i686.rpm**



**To install a package using rpm command and check whether it is installed properly or not.**

· To install the package first we need to be in the directory of the package



To install the package the syntax is

**#rpm –ivh <package name>**

**#rpm –ivh finger-0.17-39.el6.i686.rpm**



To check whether it is installed or not

**#rpm –qa finger**



Check the installed package by using it command, finger is used to check user’s details.

**#finger <user name>**

**#finger ktuser**



**To remove a package or uninstall the package**

· To remove a package the syntax is

**#rpm –e < package name>**

**#rpm –e finger**

Verify it by **#rpm –q or rpm –qa command**



**To see the information about the package before installing**

· To see the info about a particular package which is not installed, move to the directory where you have kept the packages.



To see the info of a package, the syntax is

**#rpm –qip <package name> (**where **q** is for query, **i** is for install and **p** is for package**)**

**#rpm –qip finger-0.17-39-el6.1686.rpm**



**To see the information about the installed package**

· To see the information or details about the installed package, the syntax is

**#rpm –qi < package name >**

**#rpm –qi vsftpd**



**To check the package of a particular command**

· To check the package of a particular command, first check the installed location of a command

**#which <command name>**

**#which cat**



Now, use the following command,

**#rpm –qf <path of the command>**

**#rpm –qf /bin/cat**



**To install a package forcefully**

· Before installing a package forcefully, first understand a situation where we need this force option.

· Let me corrupt one command and show you how to install its package forcefully.

· First check the package of the command we are going to corrupt. Let say **mount** command

**#which mount**

**#rpm –qf /bin/mount**



Packages installation with Yum:

**yum list**

**vim /etc/yum.repos.d/ktcl5.repo**

**/etc/yum.repos.d/**

Yum install <packagename>

Yum remove <packagename>

Yum update <packagename>

**BACKUP AND RESTORE**

***LAB WORK:-***

**tar –cvf <destination and name to be > < source file>**

**tar –cvf /opt/etc.tar /etc**

**Still want to zip a file:**

**In the below tab etc.tar – file got created**



**Now doing gzip for tarfile:**



**Ungizping the .gz file**



**Now untar the etc.tar**



**MANAGING INSTALLED SERVICES**

· Services are programs (called daemons) that once started run continuously in the background and are ready for input or monitor changes in your computer and respond to them. For example the Apache server has a daemon called **httpd** (the d is for daemon) that listens on port 80 on your computer and when it receives a request for a page it sends the appropriate data back to the client machine.

· Many services are required to run all the time however many can be safely turned of for both security reasons as running unnecessary services opens more doors into your computer, but also for performance reasons. It may not make much difference but your computer should boot slightly faster with less services it has to start on boot.

· One of the techniques in every Linux administrator's toolbox to improve security of a box is to turn off unneeded services.

**Commands for checking the services:**

**systemctl status sshd**



**Restarting the services:**



**Command to disable & enable the service:**



**Top for checking system usage**

**NFS(NETWORK FILE SYSTEM/SHARING)**

**rpm –q nfs-utils**

**exportfs –avr**

**service nfs start**

**DNS (Domain Name System) SERVER**