Linux Introduction

Prepared by
Masoud Hemmatpour

September 2011

Outline

+ Exploring Linux Command Line Tools

+ Managing Software

+ Configuring Hardware

+ Managing Files

Exploring Linux Command-Line Tools

Starting shell

+ Type of shell

- bash(bourne again shell)
- bsh(bourne shell)
- tcsh
- csh
- Ksh
- ....
- What are the differences ??? ☺

Exploring Linux Command-Line Tools

Basic Shell Command

+ echo

  echo what ever you want to say!!

+ clear

  clear the screen

+ cd

  change directory

  - cd **specific path**
  - cd ..
  - cd ~

+ pwd

  print working directory

Environment Variable

+ **What is Environment Variable??**

-Environment Variable is one kind of variable that you can use it in just specific area.

+How set Environment variable??

- Variable-name=value , test=123

+How display Environment variable value??

- set
- printenv

+ **How export Environment variable ??**

-Export variable-name=value , export test=123

Type Of Streams

+ **standard Input** (stdin)
-Default standard input is Keyboard.

+ **standard output** (stdout)
- Default standard output is Screen.

+ **standard error** (stderr)
-   Default standard error is Screen

Redirectioning Input And Output

+ **Redirecting standard output**

- > , >>

Ex: echo > out.txt

+ Redirecting standard error

- 2> , 2>>

Ex: asdf 2>err.txt

+ Redirecting standard input

- < , <<

Ex : tr a A <test.txt

Piping Data Between Programs

+ **How can redirect output of one command to another command ??**

1- using redirection

- Command1 > tmp
- Command2<tmp

2- using piping

- Command1|command2|command3|......

File combining command

+ **Cat**

concatinate two files together , dispalying file content

FUO(Frequently Used Option):

-E : show end of line .

-n : numbering the line

-b : numbering theline without empty line .

Ex : cat filename, cat test.txt

File combining command

+ **Join**

  Join two file together

Ex : join -file-order column-number -file-order column-number file-1 file-2

```
join -1 1 -2 2 test1 test2
```

+ **Paste**

  Paste two file together

Ex : paste file-1 file-2

```
paste test1 test2
```

Test1

```
1  hi
2  this
3 is
4 just
5 test
```

Test2

```
Bye 1
That 2
Are 3
All 4
Ser 5
```

File Transforming Command

With file transforiming command dont change file content.

+ **sort**

Ex : sort –k 2 test

+ **uniq**

Ex: sort test | uniq

+ **split**

split file to many files according to specific option

- l (line)

- b (byte)

Ex : split -l number file-name prefix

Split –l 2 test , split –l 2 test sp

File Transforming Command

+ **tr**

change individual character of file

Ex : tr SET1 SET2 <file-name , tr a A <test , tr ab AB < test , tr abc AB <test

+ **sed**

change content of file

Ex : sed ‘s/group1/group1/’ file-name, sed ‘s/hi/bye/’ test , sed ‘s/hi/bye/g’ test

File viewing command

+ **cat**

Described before .....

+ **head**

Display as default 10 first line of file.

 FUO :

-n specify number of line to display.

Ex : head –n number , head –n 5

+ **tail**

Display as default 10 line of end of file.

Like head but display end of file !!

File Summarizing Command

+ **cut**

  cut the specified pieces of file and display.

  FUO :

  -d (delimiter)

  -f (filed)

  -b (byte)

  Ex : cat test | grep 1 | cut -d “ “ -f 2

+ **wc**

  word count

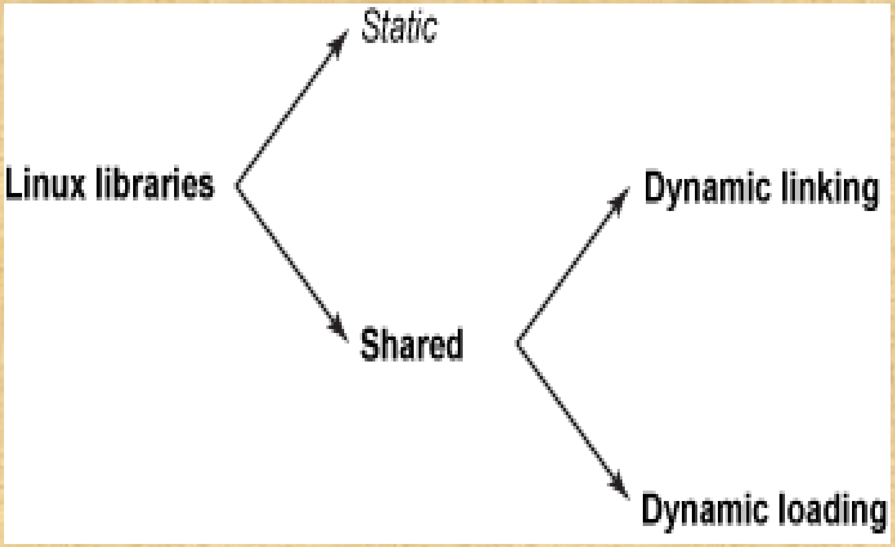
  Ex : wc file-name, wc test

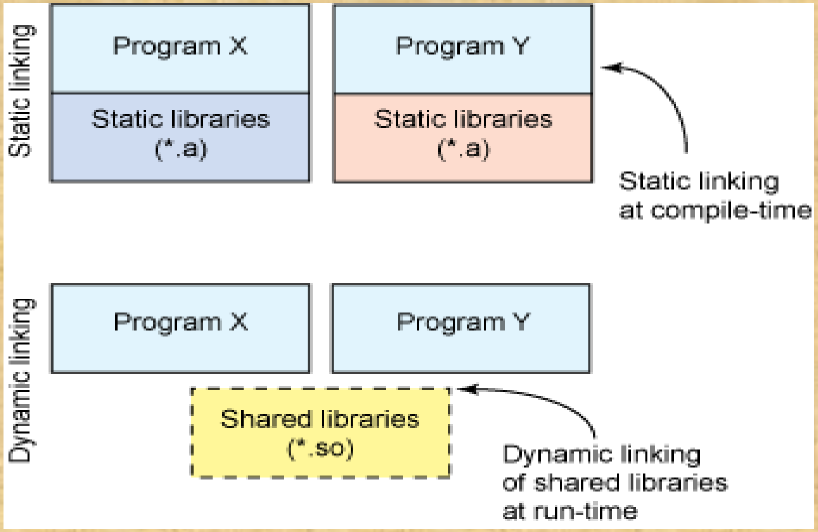
Managing shared libraries

Type of Library

The static library contains functionality that is bound to a program statically at compile time. This differs from dynamic libraries, which are loaded when an application is loaded and binding occurs at run time

Static library have .a filename extension .





Managing shared libraries

Locating library files

- **default library path**

/lib and /usr/lib are default library path for every linnux system .

- **LD_LIBRARY_PATH**

set the path for this environment variable this path add to library path .

- **/etc/ld.so.cache**

edit /etc/ld.so.conf this is just a ascii file and you can simply add specified path to that and then use **ldconfig** command for making **ld.so.cache**

Ld.so.cache is binary mirror of **ld.so.conf** file . System use this binary file because searching in binary file is more faster .

Managing process

+ **ps**

With ps command you can have snapshot of process in your system .

FUO :

-A or –e display all the processes in the system.

-x display all the processes owned by the user who give the command.

without any option list processes of termnal just.

-u list process owned by specifiied user

-f display hierarchy of relationship between process.

-l dispplay more information about process.

Ex : ps –Al , ps –u jhon , ps

Managing process

+ top

with top command you can interactively get processes informations .

FUO :

-n set the number of process to display
-s set the refresh time of process
-h display help
-k send the signal to the process
-q quit top

Ex: top → -n 5 , -s 5 , -h , -k PID and signal

Managing process

Exercise :

How send process to background and how back to the fronground ??

Use this command
+fg
+bg
+jobs
+&

Managing process

Managing process priorities

Sometimes you may want to prioritize your programs cpu use. You can assign a a priority with nice and renice command . Nice and priority are reverse more priority cause less nice and vice versa .

+ **nice**

FUO :

-n specify the nice number

--adjusment=specify the nice number

Ex : nice –n 12 test

nice –adjusment=12 test

nice 12 test

Managing process

+ renice

FUO:

-p specify PID

- u specify user

Ex : renice 7 1234

renice 7 –u jhon ,renice all process of jhon user

Managing process

+ kill

with kill command you can send a signal to a process .

You should use PID of process.

FUO :

-s specify signal

Ex : kill –s SIGKIL 1234,   kill -9 1234

+**killall**

With killall command you can send a signal to a process .

You should use program name. Option of this command is like kill.

Ex : killall -s SIGKIL vi , you just say the name of program.

Configuring hardware

+**What is IRQ ??**

An interrupt request or interrupt is a signal sent to cpu instructing it to suspend its current activity and to handle some external event such as keyboard input.for displaying interrupt request **: cat /proc/interrupts**.

+IRQ common uses

-0   system timer

- 1 keyboard
- 3 second RS-232 serial port (com2 inwindows system)
- 4 first RS-232 serial poer (com1 in windows system)

-5 second parallel port (LPT2 in windos)

-7 first parallel port (LPT1 in wndows)

Configuring hardware

+**What is I/O address?**

I/O addresses are unique locations in memory that are reserved for communications between the CPU and specific physical hardware devices.

For displaying I/O address of system : cat **/proc/ioports**

+I/O address common uses

Wname	**Lname**	**IRQ**	**I/O address**
-com1	**/dev/ttyS0**	**IRQ4**	**0x03f8**
- com2	**/dev/ttyS1**	**IRQ 3**	**0x02f8**
- com3	**/dev/ttyS2**	**IRQ4**	**0x03e8**
- com4	**/dev/ttyS3**	**IRQ3**	**0x02e8**

Configuring hardware

You may want to check how PCI devices are currently configured . The **lspci** command is used for this purpose .

+ lspci

FUO :

-v increase verbosity of output

-n display information in numeric codes rather than translating code to device name.

-nn display both numeric code and device name.

-t display the relathionship of device.

-i specified file use specified file to map vendor name and device id

Default is : /usr/share/misc/pci.ids

Configuring hardware

Loading kernel module and removing kernel module

Linux enables you to load kernel modules with two programs : insmod , modprobe.the insmod insert single modue to the kernel the modprobe program load automatically any depended-on module .you can leave modules loaded indefinitely .with rmmod command you can unload module.

+**insmode**

EX: insmod /lib/module/2.6.26/kernel/drivers/block/floppy.ko

+**modprobe**

Ex: modprobe floppy

+**rmmod**

Ex: rmmod floppy

Configuring hardware

Linux USB drivers
Several different usb controllers are available , with name such as UHCI,OHCI,EHCI and R8A66597 . Modern linux distribution ship with the drivers for the common USB controllers enabled .the UHCI and OHCI handle usb1.x device but most other handle usb2.x device. **lsusb** command can get information about your usb device.
+**lsusb**
FUO:
-v extend information about each product
-D specified file name display information about the device that accessible via filename wich should be a file in the **/proc/bus/usb** directory tree.

Configuring hardware

Hard Disk Interfaces

+**PATA** Parallel Advanced Technology Attachment
Named like /dev/hdx x is a letter from “a” up
+**SATA** Serial Advanced Technology Attachment
Named like /dev/sdx x is a letter from “a” up
+**SCSI** Small Computer System Interface
SCSI DISK Named like /dev/sdx x is a letter from “a” up
SCSI TAPE Named like /dev/stx , /dev/nstx x is number from “0” up
SCSI CD and DVD ROM Named like /dev/scdx x is number from “0” up

Configuring hardware

Partition Name Convention
The convention changes depending on what you're looking at; hd0,0 looks similar to GRUB, while sd0 is similar to entries in /dev in /dev:
IDE drives start with hd, while SATA (and I believe any kind of serial device) start with sd drives are lettered starting with a cable order,
so /dev/sda is the first serial drive, and /dev/hdb is the second IDE drive
Partitions on a drive are numbered starting with 1, so /dev/sdb1 is the first partition on the second serial drive
GRUB 1 doesn't have the distinction between drive types, it's always of the form (hdX, Y):
X is the number of the drive, starting with 0, so sda is hd0, sdb is hd1, etc.
Y is the number of the partition, starting with **0** (not 1 like /dev),
so sda1 is (hd0, 0)

Configuring hardware

Creating Partitions

The traditional linux tool for disk partitioning is called **fdisk**. This name is short for **fixed disk** and the name is the name as DOS and Windows tool.

FUO :

-m help

-d delete partition

-p print the partition table

-n create a new partition

-q quit without saving changes

-w write the new partition table and exit

Ex : fdisk /dev/sda , q

Configuring hardware

Common file system Type
+Ex2fs Second Extended File System is the traditional linux native file system . Ex2fs is the best choice for small partitions (sub gigabyte).
+Ext3fs Third Extended File System is basically Ext2fs with journal added.
+Ext4fs Fourth Extended File System is the nex generation of file system .
It adds the ability to work very large disks over 32 terabytes and large files over 2 terabytes.
+Reiserfs This file system was designed from scratch as a journaling file system for linux .it handles large small file on partition.
+JFS IBM developed the journaled File System for its AIXS OS and later for OS/2.

Configuring hardware

Common file system Type

+XFS Silicon Graphics created it Extents File System for its IRIX OS.

+FAT File Allocation Table its an old and primitive file syystem. Its the only hard disk file system supported by DOS and windows. For this reason every major OS understand FAT and its a good file system for exchanging data between them.

+NTFS New Technology File System is the prefered file system for windows , unfortunately linux NTFS support is rather insufficient. Linux can read NTFS , and can overwrite existing file but it can’t write new file.

+HFS and HFS+ Apple has long used the Hierarchial File System with its mac os and linux support full read/write HFS.

Configuring hardware

+**ISO9660** the standard file system for CD-ROM has long been ISO-9660.It has Several levels . Level 1 is similar to the original FAT in that it supports only 8.3 filename . Level 2 and 3 add support for longer 32 character filenames.

The **Rock Ridge Interchange Protocol** is an extension to the ISO9660 volume format, commonly used on CDROM and DVD media, which addsPOSIXfile system semantics. The availability of these extension properties allows for better integration with Unix and Unix-like operating systems.

+**Joliet** This file system is used much like Rock Ridge but it's technically a seperate file system .

+**UDF** The Universal Disk Format is the next generation filesystem for optical disks.it's commonly used on DVD-ROM and recordable optical Disk.

Configuring hardware

Creating File system

Most filesystems have linux tools that can create the filesystem on a partition.

Typically these tools have filename of the form **mkfs.fstype where** fstype is the filsystem type code .these tools can use in this form **mkfs –t fstype**

Ex: mkfs.ext2 /dev/sdb1, mkfs.ext3 /dev/sdb1, mkfs –t ext3 /dev/sdb1

For ext2 and ext3 filesystems , the **mke2fs** is often used instead of **mkfs** .**mke2fs** is just nother name of **mkfs.ext2**.

Linux support some tools for non-linux filesystem the most important of these may be for **FAT** . **mkfs.msdos** and **mkfs.vfat ,** this tool can automatically adjust the fat size (data structure), you can override it by **–F** option and specify 12 ,16 ,32.

Configuring hardware

Creating swap space

Some partitions don’t hold files .Linux can use a swap partition which is a partition that linux treats as an extension of memory (linux can use a file instead !!!).

Instruction :

Mkswap /dev/sdb1
Swapon /dev/sdb1

Configuring hardware

Maintaining File system Health

File system can become “sick” in variety of way. They can become overloaded with too much data, they can tuned inppropriately for your system, they can become corrupted . Linux provides a variety of utilities that can help you.

Obtaining File System Information

+dumpe2fs Display file system information

Ex:Dumpe2fs /dev/sdb1

Adjust Tunable File System Parameter

+tune2fs tune file system parameter

Dont use tune utilities for mounted file system

Configuring hardware

FUO :

-c specify number of mount times without file system check .

-i set interval time between two file system check .

-j add journaling system to file system.

-m specify percentage of disk space for use by root. Default value is 5 .

-C specify number of mount time for file system (trick!!!!)

-r specify number of block reserved for root .

-L set file system label

-U set UUID

-l obtain file system label

-u obtain UUID

Ex: tune2fs -j /dev/sdb1

Configuring hardware

Checking file system

Errors, bugs, power failures, and mechanical problems can cause the data structure on a file system to become corrupt. If they're left unchecked , they can cause serve data loss. Linux has **fsck** utility to prevent problems.

+**fsck**

FUO:

-A check all filesystems marked to check in **/etc/fstab**

-C display a text-mode progress indicator of the check process.

-V verbose output of the check process.

Filesystem check programs for specific filesystems often have their own ptions.

EX: fsck –A /dev/sdb1

Configuring hardware

Monitoring disk use

One common problem with disks is that they can fill up. Linux has **df** and **du** utilities for avoiding this problesm by summarizing disk use on a partition-by-partition and directory-by-directory basis.

+df

FUO:

-a include pseudo filesystems with size of 0 in output.

-h channge re result to human readable ex:instead of 585974 blocks , 5.6GB

-i   summarize available and used inodes.

-l omit network filesystem .

-T display filesystem type to output

-t only display information about filesystem that specified

Configuring hardware

-x display all filesystem exclude specified filesystem
EX: df /dev/sdb1, df /home
+du
The **df** command is helpful for finding out which partitions are in danger of becoming overloaded . But once you’ve obtained this information, you may need to fine-tune the diagnosis and track down the directories and files that are chewing up. The tool for this task is **du**. By default **du** report space used by files in the directories not files individually.
FUO:
-a report files individually
-h human readable output

Configuring hardware

-l   count each hard link seperately

--max-depth   limit max depth

-x limit the output to current filesystem

-s summarize output

Ex: du –x

Configuring hardware

Mounting Filesystems

Linux provides the mount command to mount a filesystem to mount point.

The **umount** command reverse this process.

+mount mount options device mountpoint

FUO:

-a mount all filesystem specified at /etc/fstab

-r cause linux to mount the filesystem read-only

-v produce verbose output

-w attempt to mount filesystem for both read and write operations .

-t specify filesystem type

-L mount by label or **–U** by uuid

Configuring hardware

When you do need to use special parameter with **–o** option you can add special filesystem option during mounting.
-auto, noauto mount or does’nt mount the filesystem at boot time or when root issue the **mount –a** command.
-user, nouser allow or disallow to mount filesystem by ordinary users , just user who mount filesystem can **umount** it.
-users like above but any user can **umount** filesystem .
-remount if you like to change filesystem parameters and **remount** it again.
-umask set the umask for permissions on files and directories.
-dmask set permissions for directories.
-fmask   set permissions for files.

Configuring hardware

+Umount

This command is more simple and use for un mounting filesystem.You just need to specify device or mount point not both.

FUO:

-a un mount all filesystem .

-f   force to un mount filesystem .

-r if can’t un mount it try to remout partition read-only again.

-t un mount just specified filesystem .

Ex :

```
Umount  -f /dev/sdb1
```

Configuring hardware

Permanently Mounting Filesystem

The **/etc/fstab** file control how linux provides access to disk partitions an removable media devices.

Filesystem Table file structure:

```
#device     #mountpoint       #filesystem       #options       #dump
#fsck

/dev/sdb1     /                              ext3              defaults          1
1
LABEL=/home  /home                           reiserfs          defaults          0
0
UUID=361a288-673e-40f5-9e96-6539fec46839   /           reiserfs
umask=0     0        0
//winsrv/shr    /other/win        cifs      users,credentials=/etc/creds    0
0
```

Creds file keep user and password for authenticatinng .

Configuring hardware

Choosing filesystem depends on :

+CPU architecture
+Size of data
+Size of partition
+number of data(inodes)
+Type of data
+Platform
+utilities for that filesystem

Managing Files

File Naming

Linux filenames can contain uppercase or lowercase letters,numbers and even most punctuation and control characters To simplify your life and avoid confusion , I recommend restricting non-alphanumeric symbols to dot(.) , dash (-) , and the underscore(_). Some programs backup files that end in the tilde (~) . Linux filenames can contain spaces.A few characters have special meaning and should never be used in filenames.these include the asterisk (*) , question mark (?) , the forward slash(/) ,the back slash (\) and the quotation mark (“) .although you can create files that contain all of them except forward slash (/) .linux filename length depends on the filesystem in use , on ext2fs , ext3fs , reiserfs ,xfs and many others , the limit is 255characters .file name can begin with dot(.) that most of utility hidden these files.

Managing Files

File Commands

+ls Displays the names of files in directory

FUO:

-a display (.) dot files.

--color differenntiates directories , files ,symbolic links and so on by displaying them in different colors .

-l display longlist information

-F display file type

/ directory, @ Symbolic link, = Socket, | Pipe

Ex : ls –F

Managing Files

+cp copy the file and directory from source to destination
FUO:
-f force to overwrite existing file
-i ask you before overwriting
-R copy directory and subdirectory of source
-p preserve the file ownership and permssion if is possible
+mv move the file and directory from source to destination
Like cp command but –p ,-r dosnt apply to this
+rm remove file
FUO:
-r recursively remove file

Managing Files

+touch by default touch sets the modification and access times to the current time.

Three time stamp in linux

-Creation time

- last modification time

- last access time

-a change access time

-m channge modification time

- **c** dosnt create file if already dont exist.

Managing Files

File Archiving

A file archiving tool collects a group of files into single package file .

+tar stand for “tape archiver”

FUO:

-c create archive file

-x extract file

-f uses the specified file

-j processses an archive through bzip2

-z processes an archive through gzip

EX: tar –cf test.tar test1.txt test2.txt test3.txt

Managing Files

+cpio the cpio program is similar in parinciple to tar .

FUO:

-o create

-i extract

Ex : find . –name test.t* | cpio –o >test.cpio , cpio –i < test.cpio

+dd archive file at lowlevel system

FUO:

-if input file

-of output file

- **bs** block size (Bytes)

- **count** number of block

Ex: dd if=/dev/sdb1 of=/opt/sdb1.iso bs=1024 count=10

Managing Files

Managing Links

In linux, a link is a way to give a file multiple identities,similar to shortcut in windows. Two type of link exist : Hard Link , Soft Link.

+ln

FUO:

-s create softlink

Ex : ln -s /usr/source   /usr/softlinklink , ln   /usr/source   /usr/hardlink

Managing Files

Directory Commands

Most of the command that apply to files also apply to directory.

+mkdir Creating directory

FUO:

-p   create necessary parent directory

-m new permission after creating the file

Ex:mkdir –p /usr/test1/test2

+rmdir remove existing directory

FUO:

-p remove entire directory tree.

Ex: rmdir -p /usr/test1/test2

Managing Files

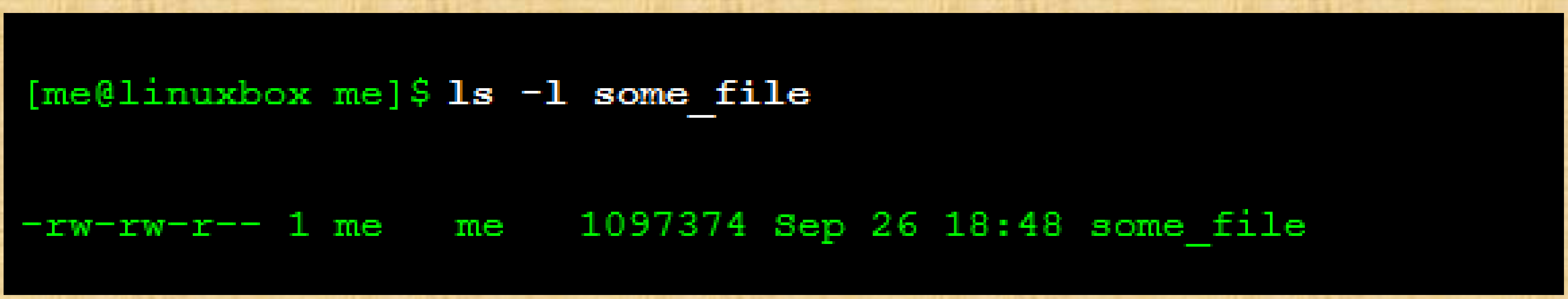
Controlling Access To Files

Linux permissions are fairly complex. In addition to providing access control for files, a few special bits exist, which provide some unusual features.file permission string in linux is 10 character long. The first character has special meaning, its the **file type code**. The file type code determines how linux will interpret the file-as ordinary data, a directory, or a special file type .we summarize these file type codes

-	Normal data file
d	Directory
l	symbolic link
p	Named pipe
s	Socket

Managing Files

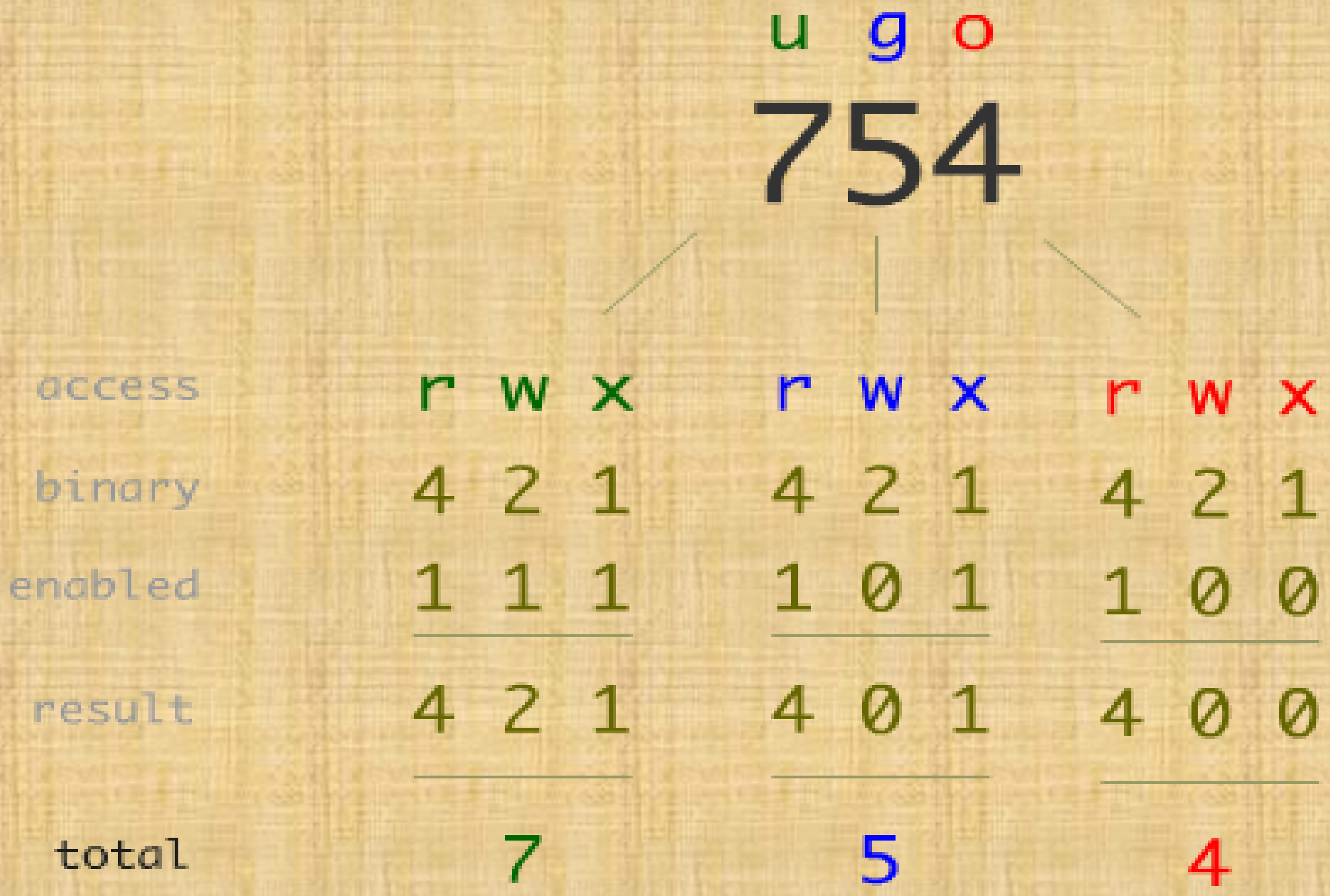
b block device
c character device
The remaining nine character of the permission are brokenup to three groups first group control the file owner , second group control group's acess to the file, third group control allother user. In each of these three cases, the permission string determines the persence or absence of each of three types of access :Read, Write, Execute. For displaying file permissiong use **ls –l** command.



```
[me@linuxbox me]$ ls -l some_file

-rw-rw-r-- 1 me    me    1097374 Sep 26 18:48 some_file
```

Managing Files



u g o

754

access	r	w	x	r	w	x	r	w	x
binary	4	2	1	4	2	1	4	2	1
enabled	1	1	1	1	0	1	1	0	0
result	4	2	1	4	0	1	4	0	0
total		7			5			4	

Managing Files

Special Bits

+Sticky Bit

The sticky bit was introduced in the fifth edition of Unix in 1974 for use with pure executable **files**. When set, it instructed the operating system to retain the text segment of the program in swap space after the process exited. This speeds up subsequent executions by allowing the kernel to make a single operation of moving the program from swap to real memory. The most common use of the **sticky bit** today is on ***directories***. When the sticky bit is set, only the item's owner, the directory's owner, or the superuser can rename or delete files.

Managing Files

+SUID

When a binary executable file has been given the **setuid** attribute, normal users on the system who have permission to execute this file gain the privileges of the user who owns the file (commonly root) within the created process.

+SGID

The **sgid** attribute will allow for changing the group based privileges within a process, like the **suid** flag does for user based privileges.

Managing Files

SUID

If set, then replaces "x" in the owner permissions to "s", if owner has execute permissions, or to "S" otherwise.

Examples:

-rws------ both owner execute and SUID are set

-r-S------ SUID is set, but owner execute is not set

SGID

If set, then replaces "x" in the group permissions to "s", if group has execute permissions, or to "S" otherwise.

Examples:

-rwxrws--- both group execute and SGID are set

-rwxr-S--- SGID is set, but group execute is not set

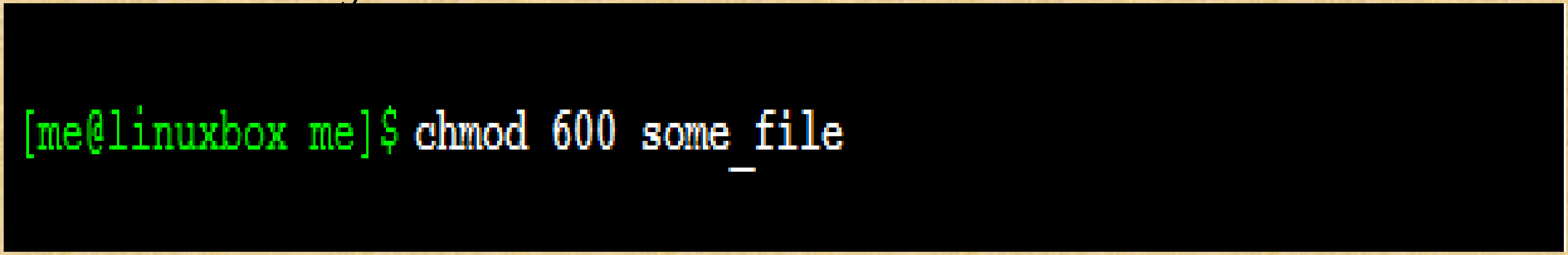
Managing Files

Sticky
If set, then replaces "x" in the others permissions to "t", if others have execute permissions, or to "T" otherwise.
Examples:
-rwxrwxrwt both others execute and sticky bit are set
-rwxrwxr-T sticky bit is set, but others execute is not set

Managing Files

Changing a File’s Mode

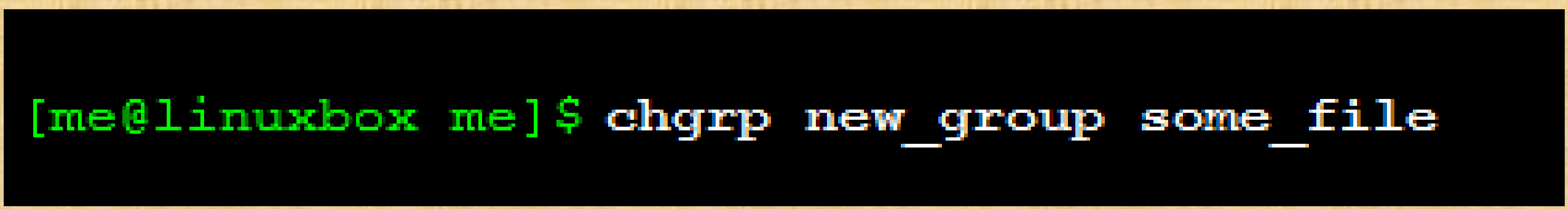
You can modify a file’s permission using the chmod command. This command may be issued in many different ways to achieve the same effect.



```
[me@linuxbox me]$ chmod 600 some_file
```

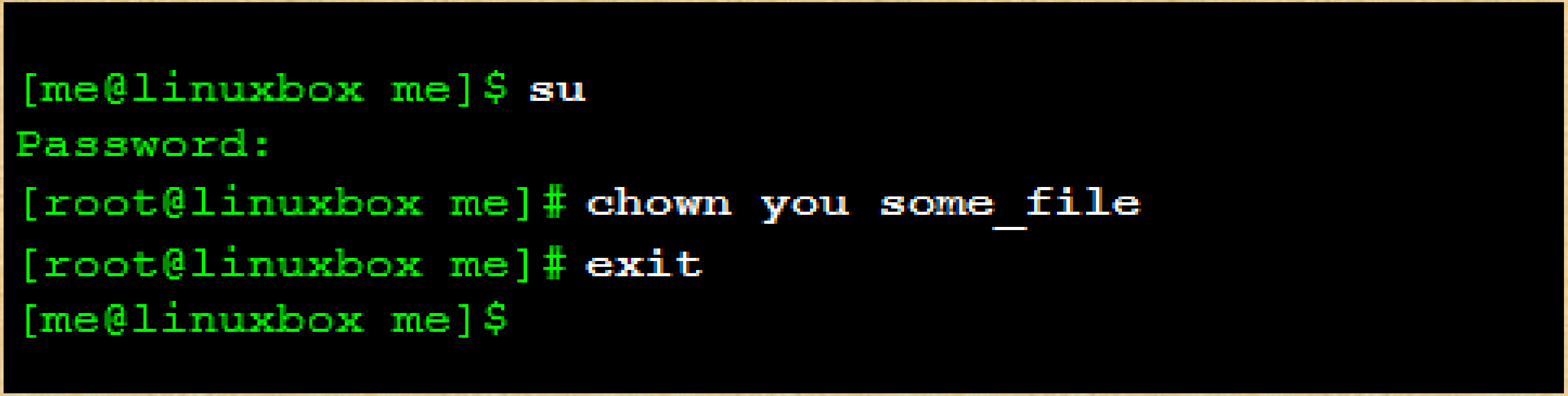
Managing Files

Changing File’s Group



```
[me@linuxbox me]$ chgrp new_group some_file
```

Changing File’s Owner



```
[me@linuxbox me]$ su
Password:
[root@linuxbox me]# chown you some_file
[root@linuxbox me]# exit
[me@linuxbox me]$
```

Managing Files

Setting the default mode and group

When a user create a file , that file has default ownership and permission.

The default owner is understandably the user who created the file. User mask (umask) which is set by **umask** command you can set default permission.

Directory   (777) – (umask-value)

Files   (666) – (umask-value)

Ex: umask 022

Result : directory 777-022=755 , files 666-022=644

Managing Files

Changing file attribute

Some filesystem support attributes in addition to those described.
chattr command you can change these attributes.
FUO:
-a disable write access to the file except for appending data(log file)
-c cause the kernel to automatically compress data and uncompress it when it's read back.
-j tell the kernel to journal all data written to the file. This improves recoverability of data written to file after a system crash.
-i make a file immutable which goes a step beyond simply disabling write access to the file .the file can't be deleted, links to it can't be created and the file can't be renamed.

Managing Files

Ex:

```
Chattr  +i   test.txt
Chattr   -i   test.txt
```

Disk Quotas

A **disk quota** is a limit set by a system administrator that restricts certain aspects of file system usage on modern operating systems. The function of using disk quotas is to allocate limited disk space in a reasonable way.

Managing Files

Locating Files

Maintaining your filesystem in perfect health, setting permission , and so on is pointless if you can’t find your files. Linux provides several tools to help you locate the files you need to use. The first of these tools is actually a standard for where files are located.

FSSTND filesystem standard

FSSTND standardized several specific features , such as the following

-Standardized the program that reside in /bin , /usr/bin.

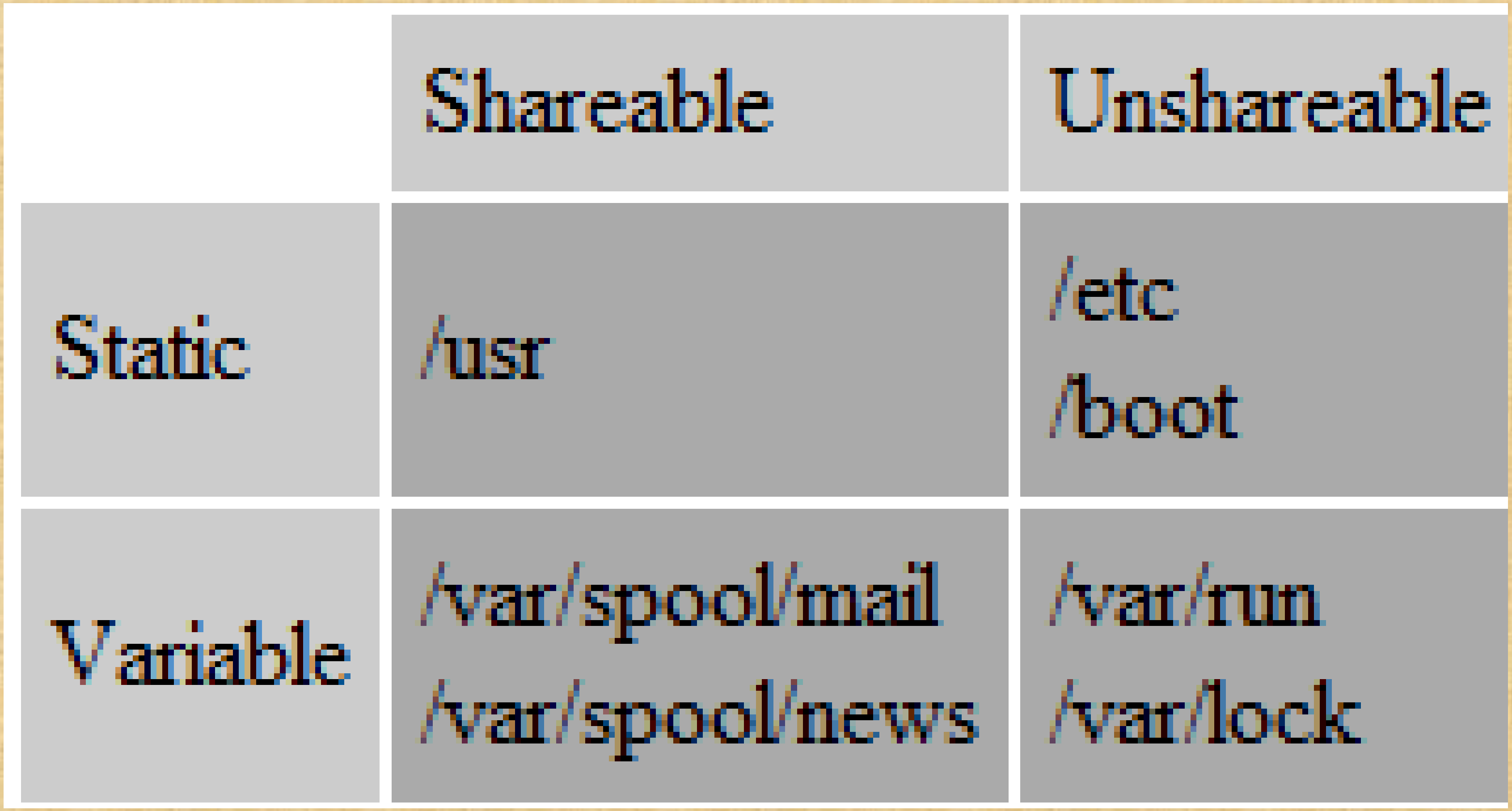
- Specified that executable files shouldn’t reside in /etc

-Remove changable files from /usrdirectory tree

Managing Files

FHS filesystem hierarchy standard
This standard is based on FSSTND but extends it .two important concept in FHS is **Static and variable files** variable files may be changed by users automated scripts,but static files is fixed.executable program is the best example for static files. **Sharable and unsharable files** Sharable files may be resonably shared between computers, such as user data files and unsharable can’t do this such as configuratuion file.

Managing Files



	Shareable	Unshareable
Static	/usr	/etc /boot
Variable	/var/spool/mail /var/spool/news	/var/run /var/lock

Managing Files

Important Directories And their Content

The FHS defines some directories very precisely, but details for others are left un resoled .the most common directories defined by the FHS or used by convention are the following :

/ Every linux file system traces its roots to a single directory known as /

Linux doesn't use drive letters, instead every partition disk is mounted at a mount point .certain critical subdirectories such as /etc , /sbin must be on root partition. Don't confuse root directory with /root directory.

/boot The /boot/ directory contains static and unsharable files required to boot the system, such as the Linux kernel. These files are essential for the system to boot properly.for older X86 BIOS /boot must reside below the 1024th cylander of hard disk.

Managing Files

+/bin The directory contain certain critical executable files such as **ls, cp** and **mount**. These commands are accessible to all users and constitute the most important commands that ordinary users might issue. You wont normally find commands for big application programs in **/bin**.

+/sbin The directory issimilar to **/bin** but it contains programs that are normally run only by the system administrator.

+/lib The directory is similar to /bin /sbin, but it contains program libraries,which are made up of code that’s are shared across many programs. **/lib/module** subdirectory contain kernel modules that can be loaded and unloaded as required.

/usr The directory hosts the bulk of linux computers’programs .many administrator s split **/usr** off into a seprate partition.

Managing Files

+/usr/local The directory contains subdirectory that miror the organization of **/usr**.The /usr/local contains files that a system administrator installs locally, for instance, packages that are compiled on the target computer.
+/usr/X11R6 The directory houses files related to the X Window System.
+/opt The directory is similar to /usr/local in many ways , but it’s intended for ready made package that don’t ship with the OS like commercial word processor or games .
+/home This directory contains user’s data , althought the /home directory is considered optional in FHS, in practice it’s a matter of the name being optional.

Managing Files

+/root This is the home directory for the root user . Bcause root is critical account.
+/var This directory contains transient files of various type, system log files,print spool files , mail files , and so on. The directory content is variable. Many system administrator s put /var in it's own partition.
+/tmp Many programs need to create temprorary files, and usual place to do this is /tmp .most distributions include routines that clean at bootup.many administrators create a seprate partition to prevent runaway process from causing problems on root filesystem.
+/mnt Linux mount removable-media devices within its normal directory structure , and **/mnt** is provided for this purpose. Some distributions create
Subdirectories within **/mnt**, such as /mnt/floppy, /mnt/cdrom others use /mnt

Managing Files

directly or even use separate mount points such as /floppy and /cdrom

+/media This directory is an optional part of the FHS . It’s like /mnt but it should contain subdirectories for specific media type , such as /media/floppy and /media/cdrom. Many modern distribution use /media subdirectories as default mount point.

+/dev because Linux treats most hardware devices as files,the OS must have a location in its filesystem where thesedevice files reside .

+/proc This is an unusual directory because it doesn’t correspond to a regular directory or partition ,Instead it’s a virtual filesystem that’s created

dynamically by linux to provide access to certain type of hardware information that arn’t accessible via /dev,for instance if you type

cat /proc/cpuinfo it respons by displayinng information about your CPU.

Managing Files

Tools for Locating Files
You use file location command to locatea file on your computers.
These commands help you locate a file by name.
+find
The find utility implements a brute-force approach to finding files.this program find files by searching through the specified directory tree.
FUO:
-name search by filename you can use wildcard for searching
-perm mode searching file by permission specified in octal as mode .
-size n search by size of file that **n** is specified by 512-bytes block but you can change it by adding c,k,.. At the end of value such as 12k=12kbytes

Managing Files

-maxdepth n limit level of search

Ex: find . -name “tes*”

+locate The locate utility work like find but it differes in two important ways

-The locate tool is far less sophisticated in it search options. You search by filename just.

- the **locate** program works from a database .most distributions include a cron job that call **locate** with options that update database periodically ,you can use **updatedb** for updating database.

Ex: locate test.rpm

+whereis search for files in restricted location such as standard binary file directories,libraryfiles , man page directory.

Ex: whereis ls

Managing Files

+Which which is very weak ,It merely searches your PATH for the command that you type and list the complete path to the first match.
Ex:which xterm
+Type it is not really a search command ,instead it tells you how a command you type will be interpreted –as built in command external command and alias and so on.
Ex:type ls