

LPIC-1 Study Group

4 Managing Files

R. Scott Granneman
scott@granneman.com
www.granneman.com

© 2012 R. Scott Granneman
Last updated 20120906
You are free to use this work, with certain restrictions.
For full licensing information, please see the last slide/page.

1

This presentation
is based on
Roderick W. Smith's
*LPIC-1: Linux Professional Institute
Certification Study Guide,*
2nd edition
That said,
there are many
additions, subtractions, & changes

2

Introduction

3

Managing Files

4

Everything is a file
Everything
Gotta know how to
create,
delete,
move,
rename,
archive,
& manipulate files

5

File Naming & Wildcard Expansion Rules

6

Safest to stick to
letters,
numbers,
& these symbols:

. - _ ~

Avoid spaces

Never use
* ? / \ "

7

255 character filenames

dot files are hidden

.

current directory

..

parent directory

~

home directory

8

Case sensitivity

Foo.txt
is not
foo.txt
is not
FOO.txt

9

Wildcards
stand for other characters

?

*

[]

File globbing
Wildcard expansion in commands

10

?
Single character

f??k
matches
flak, folk, fork, funk

11

*
Any character or characters,
including none

f*k
matches
folk, flack, flank, firetruck

12

[]
Set of characters

f1[ao]ck
matches
flack & flock

f[a-z]ck
matches
fack, feck, fock, & that's it ☺

13

\$ ls f??k

is the same as

\$ ls flak folk fork funk

14

File Commands

15

```
ls
cp
mv
rm
touch
```

16

```
ls options files
List files
```

```
ls
List files
in current directory
```

17

```
ls -a
ls --all
Display all files,
including dot files
```

```
ls --color
Colorize listing
```

18

```
ls -d
ls --directory
List only directory names
```

```
ls -l
Long listing,
including
permissions, owner, group,
size, & creation date
```

19

```
ls -F
ls --file-type
Indicator code after file names
```

```
/ Directory
@ Symbolic (soft) link
= Socket
| Pipe
```

20

```
ls -R
ls --recursive
Display directory contents
recursively
```

21

```
cp options source destination
Copy files
```

22

```
cp foo bar
Change foo to bar
```

```
cp foo bar/
Keep foo as the name
```

```
cp foo bar/foobar
Change foo to foobar
```

```
cp ../bar .
Copy bar to current directory
```

23

```
cp -f
cp --force
Overwrite existing files
without prompting
```

```
cp -i
cp --interactive
Ask before overwriting
```

24

`cp -p`

`cp --preserve`

Preserve ownership & permissions

`cp -R (or -r)`

`cp --recursive`

Copy directory & all contents

`cp -a`

`cp --archive`

Recursive AND preserve
ownership & links

25

`cp -u`

`cp --update`

Copy only newer or non-existent files

26

`mv options source destination`

Move files

Acts to rename files too

27

Same options as cp,
except for
--preserve, --recursive, & --archive

28

rm options files
Remove (delete) files

No trash can, no restore

29

Same options as cp,
except for
--preserve, --update, & --archive

30

```
rm -rf
```

Only way to delete directories
with files in them

Very dangerous!

31

```
touch options files
```

Modify time stamps

32

3 time stamps for every file

Creation time

Last modification time

Last access time

33

```
touch foo
Set modification & access times
to current
```

If foo doesn't exist, create it

34

```
touch -a
touch --time=atime
Change access time
```

```
touch -m
touch --time=mtime
Change modification time
```

35

```
touch -t MMDDhhmm[[CC]YY][.ss]
```

```
MM month
DD day
hh hour (24-hour clock)
mm minute
YY year (12)
CCYY year (2012)
ss second
```

36

```
touch -r reffile
touch --reference=reffile
Replication reffiles's time stamp
```

37

File Archiving

38

Archiving
collects files
into a single file

Archiving \neq compression

39

```
tar
cpio
dd
```

40

```
tar
“tape archiver”
Don't need tape!
Archive files into a tarball
```

41

```
tar cvf foo.tar foo/
tar --create --verbose --file

tar zcvf foo.tar.gz foo/
tar --gzip --create --verbose --file
```

42

```
tar W
tar --verify
Verify archive after writing it
```

43

```
tar xvf foo.tar
tar --extract --verbose --file
```

```
tar zxvf foo.tar.gz
tar --gunzip --extract --verbose --file
```

44

```
tar t
tar --list
List archive's contents
```

45

tar A

tar --concatenate

Append tar files to an archive

tar r

tar --append

Appends non-tar files to an archive

tar u

tar --update

Append files that are newer
than those in an archive

46

tar d

tar --diff

tar --compare

Compare archive to files on disk

tar p

tar --same-permissions

Preserves permissions

47

tar --exclude

Exclude file from archive

tar X file

tar --exclude-from *file*

Exclude files
listed in *file* from archive

48

cpio
"Copy In, Copy Out"

Originally for backup to tape

49

3 modes

Copy-out

`cpio -o` or `--create`

Create archive & copy files into it

Copy-in

`cpio -i` or `--extract`

Extract data from existing archive

Copy-pass

`cpio -p` or `--pass-through`

Combines copy-out & copy-in
to copy directory tree
from one place to another

50

Copy-out
creates an archive

Uncompressed

```
find ./stuff | cpio -o > stuff.cpio
```

Compressed

```
find ./stuff | cpio -o | gzip >  
stuff.cpio
```

51

Copy-in
extracts data from an archive

From uncompressed
`cpio -i < stuff.cpio`

From compressed
`gunzip -c stuff.cpio.gz | cpio -i`

52

`dd`
Low-level copying & archiving

(Think “disk duplication”)

53

`dd if=source of=target`
`dd if=/dev/sda3 of=/tmp/data.iso`

54

Good way to create exact backup
of an entire partition

Not so good
as a general backup tool

- ✓ Backs up entire partition
including empty space
- ✓ Cannot restore individual files
unless you can mount target

55

Create empty file
of a particular size

```
dd if=/dev/zero of=empty.img  
bs=1024 count=720
```

bs = block size

count = number of blocks

56

Managing Links

57

In options source *Link*
Create a link

58

Link
Gives a file multiple identities,
like *shortcuts* in Windows
& *aliases* in Mac OS X

2 kinds of links

- ✓ *hard* links
- ✓ *soft (symbolic)* links

59

Hard links

- ✓ 2 files that point to the same inode
 - ✓ Both are valid
 - ✓ To delete the file,
you must delete all hard links
- ✓ Cannot point across filesystems

Soft links

- ✓ Soft link points to original file
 - ✓ If you delete source,
link target is broken;
if you delete link target,
original source still exists
- ✓ Can point across filesystems

60

```
ln foo bar  
Create hard link
```

```
ln -s foo bar  
ln --symbolic foo bar  
Create soft link
```

61

```
ln -f  
ln --force  
Remove existing links or files  
that have the target link name
```

```
ln -i  
ln --interactive  
Remove existing links or files  
that have the target link name,  
but ask first
```

62

```
ln -d  
ln -F  
ln --directory  
Attempts to create hard links  
to directories  
Often doesn't work
```

63

To see what a link points to,
use `ls -l`

```
$ ls -l link  
link -> original
```

64

Directory Commands

65

`mkdir`

`rmdir`

66

```
mkdir options directory  
Create directory
```

67

```
mkdir -m mode  
mkdir --mode=mode  
New directory  
has specified permissions mode  
(Octal number)
```

68

```
mkdir -p /path/to/directory  
mkdir --parents /path/to/directory  
Creates necessary parent directories
```

```
$ mkdir /tmp/foo/bar  
No such file or directory  
$ mkdir -p /tmp/foo/bar  
$ ls /tmp  
foo  
$ ls /tmp/foo  
bar
```

69

```
rmdir options directory  
Deletes empty directory
```

70

```
rmdir --ignore-fail-on-non-empty  
If directory is not empty,  
don't show error message
```

```
rmdir -p foo/bar  
rmdir --parents foo/bar  
Delete entire directory tree  
(if all are empty)
```

71

File Ownership

72


```
ls -l  
chown  
chgrp
```

73

Each file has an owner & group
Each group contains users

3 tiers of permissions

✓ Owner

✓ Group

✓ All other users

74

Assessing File Ownership

75

```
ls -l
Show ownership
```

76

```
$ ls -l
-rw-r--r-- 1 scott staff 426 Nov 12 2009 foo.txt
drwxr-xr-x 7 scott staff 238 Apr 1 16:52 Music
```

Shows owner, group, & permissions

If you delete a user account,
you'll see a number
instead of a name

77

Changing a File's Owner

78

`chown options newowner:newgroup file`
Change owner (& group)

Can only be used by root!

79

`chown scott foo`
Change owner

`chown scott:websanity bar`
Change owner & group

`chown :websanity baz`
Change group

80

`chown -R`
`chown --recursive`
Recursively changes ownership
through an entire directory tree

81

Changing a File's Group

82

```
chgrp options newgroup file  
Change group for file
```

Can be used by non-root users!

83

```
chgrp -R  
chgrp --recursive  
Recursively changes  
group ownership  
through an entire directory tree
```

84

File type code

- File
- d Directory
- l Soft link
- p Named pipe (lets 2 Linux programs communicate with each other)
- s Socket (Like named pipe, but allows network & bidirectional links)
- b Block device
- c Character device

88

Permissions

- r Read
- w Write (edit, delete, manipulate)
- x Execute files
Search/list directories
- Not applicable

r = 4

w = 2

x = 1

89

```
777 rwxrwxrwx
755 rwxr-xr-x
750 rwxr-x---
700 rwx-----
666 rw-rw-rw-
664 rw-rw-r--
660 rw-rw----
644 rw-r--r--
640 rw-r-----
600 rw-----
400 r-----
```

90

Soft links always have 777
(just the link, not the file)

Root can read or write to,
& can change permissions on,
every file

91

Special permission bits

SUID

SGID

Sticky bit

92

SUID (Set user ID)

Run program with permissions
of file owner
not the user running the program

Indicated by `s`
in owner's execute bit position

`rwsr-xr-x`

93

SGID (Set group ID)

Run program with permissions
of file's group owner

On a directory,
new files & subdirectories
created in that directory
will inherit group's ownership
not the user's current group

Indicated by `s`
in group's execute bit position

`rwxr-sr-x`

94

Sticky bit

Protects files from being deleted
by those who don't own the files

On a directory,
files inside can only be deleted
by their owners,
the directory's owner, or root

Indicated by `t`
in world's execute bit position

`rwxr-xr-t`

95

Changing a File's Mode

96

`chmod`
Change file's permissions (mode)

97

Specify mode 2 ways

Octal
Symbolic

98

Octal

```
chmod 755 file  
rwxr-xr-x  
chmod 644 file  
rw-r--r--
```

99

If 4 digits,
1st interpreted as special permissions

4 SUID
2 SGID
1 Sticky bit

6 = SUID + SGID
3 = SGID + Sticky bit

100

Symbolic

u Owner
g Group
o World
a All

+ Add
- Remove
= Equal to

101

r Read
w Write
x Execute
X Execute if directory or already
executable
s SUID or SGID
t Sticky bit
u Existing owner's permissions
g Existing group's permissions
o Existing world permissions

102

```
chmod a+x foo
rw-r--r-- → rwxr-xr-x

chmod ug=rw bar
r----- → rw-rw----

chmod o-rwx baz
rwxrwxr-x → rwxrwx---

chmod g=u qux
rw-r--r-- → rw-rw-r--

chmod g-w,o-rw conge
rw-rw-rw- → rw-r-----
```

103

```
chmod -R
chmod --recursive
Change permissions on all files
in a directory tree
```

104

Setting the Default Mode & Group

105

New files
have default ownership
& permissions

Default owner is
user who created file

Default group is
user's current group

Default permissions
set by umask

106

`umask`

Shows current umask in octal

`umask -S`

Shows current umask symbolically

```
$ umask
```

```
0022
```

```
$ umask -S
```

```
u=rwx,g=rx,o=rx
```

107

Any bit set in the umask
is *removed* from the final permission

It's not just simple subtraction
(as you'll see)

If a bit isn't set & is 0,
the umask bit doesn't affect it

108

A umask of 7 sets
1 bit for user (4)
1 bit for group (2)
1 bit for world (1)

Ordinary file has permissions set to
rw- (110)

111 3rd column is 0
- 110 ← because umask
000 doesn't touch 0s

109

umask	Created Files	Created Directories
000	666 rw-rw-rw-	777 rwxrwxrwx
002	664 rw-rw-r--	775 rwxrwxr-x
022	644 rw-r--r--	755 rwxr-xr-x
027	640 rw-r-----	750 rwxr-x---
077	600 rw-----	700 rwx-----
277	400 r-----	500 r-x-----

110

Admins set umask default
at /etc/profile

Usually set to 002 or 022

However, users can override

111

Changing File Attributes

112

`chattr`
Change file attributes

113

`chattr +attribute file`
Add attribute

`chattr -attribute file`
Remove attribute

114

- a Disable write except for append
- c Automatically compress data written & uncompress data when read
- i Immutable: can't be deleted, renamed, or linked to
- j Journal all data written to file
- s Secure deletion by zeroing data blocks
- t Disable tail-merging, so small pieces of files aren't merged with other files to save disk space
- A Don't update access time stamp

115

Disk Quotas

116

Disk quotas

Limits enforced by the OS
on how many files
or how much disk space
a user may consume

117

Enabling Quota Support

118

For quotas,
need kernel support
& user-space utilities

1-2.4.x kernels
have *quota v1* support

2.6.x-now kernels
use *quota v2* system

119

Modify `/etc/fstab` for quotas
by adding mount options

`usrquota`
User quotas

`grpquota`
Group quotas

```
/dev/hdc5 /home ext3 usrquota,grpquota 1 1
```

120

May need to configure
SysV startup scripts
to run when OS boots

Typically something like
`chkconfig quota on;`

121

Once installed & configured,
reboot
or use `modprobe`
to load the kernel module
& then remount with
`mount -o remount /mountpoint`

122

Setting Quotas for Users

123

edquota
Sets quotas
using vi to edit /etc/quotatab

124

```
$ edquota alice
Quotas for user alice:
/dev/hda2: blocks in use: 3209, limits (soft = 5000, hard = 6500
inodes in use: 403, limits (soft = 1000, hard = 1500)
```

Hard limit

Maximum number allowed

Soft limit

Can be temporarily exceeded,
with warnings;
if exceeded past grace period,
treated like a hard limit

125

edquota -t
Set grace period for soft limits

Grace periods set
on a per-filesystem basis
instead of per-user

126

`quotacheck`

Verifies & updates quota info
Usually run as a startup script
or via cron job

127

`repquota /dev/hda2`

Summarizes quota info
for filesystem

`requota -a`

Summarize quota info
on *all* filesystems

128

Locating Files

129

The FHS

130

40 years of UNIX history
means there are historical reasons
things are where they are

Even if they don't always
make sense!

131

FSSTND
Filesystem Standard
1st released in 1994

Standardized contents
of /bin & /usr/bin

Specified no executables in /etc

Removed changeable files from /usr
so it could be mounted read-only

132

FSSTND unfortunately was limited

FHS

Filesystem Hierarchy Standard

Initial release in 1994

Latest release in 2004

133

Distinctions

- ✓ Shareable & unshareable files
- ✓ Static & variable files

FHS tries to isolate directories
between these distinctions,
but some are mixed (/var)

134

Shareable files

May be shared between computers,
like user data & programs,
often via NFS

Unshareable files

System-specific config files
that are not shared between computers

135

Static files

Don't normally change
except through direct intervention
by sysadmin;
e.g., programs

Variable files

May be changed
by users, scripts, servers, etc.

136

	Shareable	Unshareable
Static	/usr /opt	/etc /boot
Variable	/home /var/mail	/var/run /var/lock

137

Common directories

138

/
root
All other directories branch off

/bin
Critical executable files
available in single user mode
for all users
(ls, cp, mount)

139

/boot
Boot files
(kernels, initrd, etc.)

/dev
Since hardware devices are files,
you need a place for device files
Hardware interfaces
Actually a virtual filesystem
created on the fly

140

/etc
System-wide config files

/etc/opt
Config files for /opt

/etc/X11
Config files for X Window System

141

/home

Users' data & personal settings

/lib

Program libraries for /bin & /sbin

/lib/modules

Kernel modules

142

/media

Optional part of FHS

Like /mnt

Often default mount points
for common removable disks

/mnt

Mount removable-media devices
(/mnt/cdrom & /mnt/floppy)

143

/opt

Optional software
& ready-made packages,
like commercial apps or games
(/opt/foo & /opt/bar)

/proc

Virtual filesystem created dynamically
to provide access to hardware info,
kernel & process statuses

144

`/root`
Home for root

`/sbin`
Programs run only by root
(e.g., `fdisk` & `e2fsck`)

`/srv`
Site-specific data served
by the system

145

`/tmp`
Temporary files
Cleaned out at boot

`/usr`
Most Linux multi-user programs

`/usr/bin`
Non-essential programs
not needed in single-user mode

146

`/usr/lib`
Libraries for programs
in `/usr/bin` & `/usr/sbin`

`/usr/local`
Subdirectories mirroring organization
of `/usr`

(`/usr/local/bin` & `/usr/local/lib`)

Programs installed by sysadmin
Safe from automatic system upgrades

147

`/usr/sbin`

Non-essential system programs

`/usr/src`

Source code;
e.g., kernel source code

`/usr/X11R6`

X Window System files

Subdirectories similar to `/usr`
(`/usr/X11R6/bin` & `/usr/X11R6/lib`)

148

`/var`

Transient, variable files
(logs, print spools, mail, etc.)

`/var/cache`

Application cache data

`/var/lib`

State information
modified by programs as they run

149

`/var/lock`

Lock files
keeping track of resources
currently in use

`/var/log`

Log files

`/var/mail`

Mailboxes

150

`/var/run`

Info about running system
since last boot
(currently logged-in users
& running daemons)

`/var/spool`

Spool for tasks waiting to be processed
(print queues & unread mail)

`/var/tmp`

Temp files preserved between reboots

151

Tools for Locating Files

152

`find`
`locate`
`whereis`
`which`
`type`

153

`find`

154

`locate`
Find files
based on database
usually created by cron job

May not find recent files
or find deleted files

Very fast results, though

155

`whereis`
Search for files
in restricted set of locations

Quick way to find programs
& related files
(documentation & configs)

156

```
which
Search your path
for command
& lists complete path
to first match
```

```
which -a
Return all matches,
not just first
```

157

```
type
Tells you how command
will be interpreted
(as built-in, external, alias, etc.)
```

158

```
$ type ls
ls is aliased to `/_bin/ls -FG'
$ type cat
cat is /bin/cat
$ type cd
cd is a shell builtin
```

159

Review

160

Thank you!

Email: scott@granneman.com

Web: www.granneman.com

Publications: www.granneman.com/pubs

Blog: ChainSawOnATireSwing.com

Twitter: [scottgranneman](https://twitter.com/scottgranneman)

161

LPIC-1 Study Group 1 Command Line Tools

R. Scott Granneman

scott@granneman.com

www.granneman.com

© 2012 R. Scott Granneman
Last updated 20120906
You are free to use this work, with certain restrictions.
For full licensing information, please see the last slide/page.

162

Licensing of this work

This work is licensed under the Creative Commons Attribution-ShareAlike License.

To view a copy of this license, visit

<http://creativecommons.org/licenses/by-sa/1.0>
or send a letter to Creative Commons, 559 Nathan Abbott Way,
Stanford, California 94305, USA.

In addition to the rights and restrictions common to all Creative Commons licenses, the Attribution-ShareAlike License features the following key conditions:

Attribution. The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original author credit.

Share Alike. The licensor permits others to distribute derivative works under a license identical to the one that governs the licensor's work.

Questions? Email scott@granneman.com