

File Systems Contd.

Ownership

Permissions

Symlinks

Quotas

Mounting

Two more quick commands

- `cat` (*concatenate*)
 - Opens a file and appends it to the screen
- `echo "text"`
 - Prints out "text"

Linux Permissions

- How would YOU do it?
- What are the important things to think about?
- How does Windows do it? OS X? BSD?
BeOS?

Users and groups

In most Unix systems every file has a user and a group. (And as you know by now, EVERYTHING in Unix is a file).

A group is just a list of users. This list can be modified by anybody with permissions to edit the group file (/etc/group).

ls -l

```
/home1/z/ # ls -l
```

```
drwx--x--x 27 zetsche  zetsche  4096  2008-04-30 15:26 zetsche
drwx--x--x 45 zfriess  zfriess  28672 2008-04-28 18:39 zfriess
drwx--x--x 50 zgs      zgs      20480 2008-07-09 19:44 zgs
drwxr-x--x 8  zhang6   zhang6   4096  2007-06-19 04:14 zhang6
drwxr-x--x 8  zhangbin zhangbin 4096  2007-07-31 14:18 zhangbin
drwx--x--x 8  zhangch  zhangch  4096  2007-09-05 12:12 zhangch
```

```
perms count user group size modified  name
```

Changing user / group of a file

“chown” command CHange OWNership

```
/home/zgold # chown <user>:<group> 123.txt
```

Ownership will affect permissions. The user of a file will have certain permissions, and the group will also have certain permissions.

Permissions

There are three types of permissions for three different classes. (Data is either a file or directory.)

3 Permissions:

Read

Ability to ask for the data.

Write

Ability to change the data

Execute

Ability to tell the OS to execute data.

3 Recipients of Permission

3 Classes

User

Group

Other

Each class gets its own set of RWX permissions. Hence there are 9 total “flags” of permission for every file

There are tools and options available to drastically extend this, but for now we'll just work with the basics.

Reading permissions

Three sets of three. --- --- ---, u g o

User's permissions, then group's, then others'.

Each set consists of **R**ead, **W**rite, **eX**ecute in that order.

So if user can read, group read and write and other none its

r-- rW- ---

Changing permissions

Change mode – chmod

chmod <permission change> files

<permission change> is one of 'u', 'g', 'o' then '+' or '-' then 'r', 'w' or 'x'.

e.g. chmod u+rwx 123.txt

e.g. chmod g-w 123.txt

Can also use a numbering scheme to do it all at once, see man chmod.

Directory permissions

To be able to 'read' a directory is to list its contents

To write to a directory is to modify its contents
(add or remove files)

To execute a directory is to be able to traverse into
it.

Traversing directories

Suppose we have a file in `/a/b/cde/123.txt`

Say `b`'s permissions were: `r--r--r--`.

We then ran the command `“cat /a/b/cde/123.txt”`.

Would this work?

What about `“ls /a/b”` ?

What about `“ls /a/b/cde”` ?

What about `“cd /a/b/cde”` ?

File Access

In summary, you need the ability to access all of the locations up to where the file is your concerned with. (Mostly just +x)

Special note: User is *not* Other. a.k.a.:

```
# zgs zgs --- --- r-x 123.txt
```

Other can read this file, zgs cannot! When might this be useful?

Symlinks

A more powerful version of a 'shortcut'.

A symlink appears to programs as a normal file.

They can read, write and execute the file as if it were actually there.

In reality programs are only accessing the 'linked' file which exists elsewhere.

Creating symlinks:

```
# ln -s /path/to/realfile /path/to/link
```

'rm'ing a symlinks removes the link, not the real file

Quotas

Linux allows you to set quotas on the number of files and total file size any user has in their possession.

Can view your current quota with the '*quota*' command.

What are the implications of *chown* on quotas?

a.k.a. Why can't you *chown* on eniac?

Mounting

- Remember the discussion of the “blue drive” and the “red drive” in the Linux file system tree?
- How did we tell Linux where to put each drive?
- A process called “mounting”.
- But how do you know what to tell Linux?

mount

```
mount /dev/sda1 /mnt/extra_drive/
```

- First argument is the “dev” node. Can be found by typing “cat /proc/partitions”
- A dev node represents a special file that the kernel can do different things with, and redirect.
- In this case */dev/sda1* represents a partition of data on the disk.
 - BINARY data! The file system is not yet used here, so the kernel only gives you access to the raw chunk of 1s and 0s.
 - Use mount to overlay a filesystem and use it normally

Appendix

- cat
- echo
- mount
- quota
- ls -l
- ln
- chmod
- chown