Bash: permissions

Glenn Bruns
CSUMB

Lecture Objectives

After this lecture, you should be able to:

- set permissions on files and directories
- use commands diff, sort, uniq, tar, gzip

File permissions

Users

- every user has a login name and numeric user ID

Groups

- a collection of users
- allows permissions to be set for a group at once

my user id

my default group id

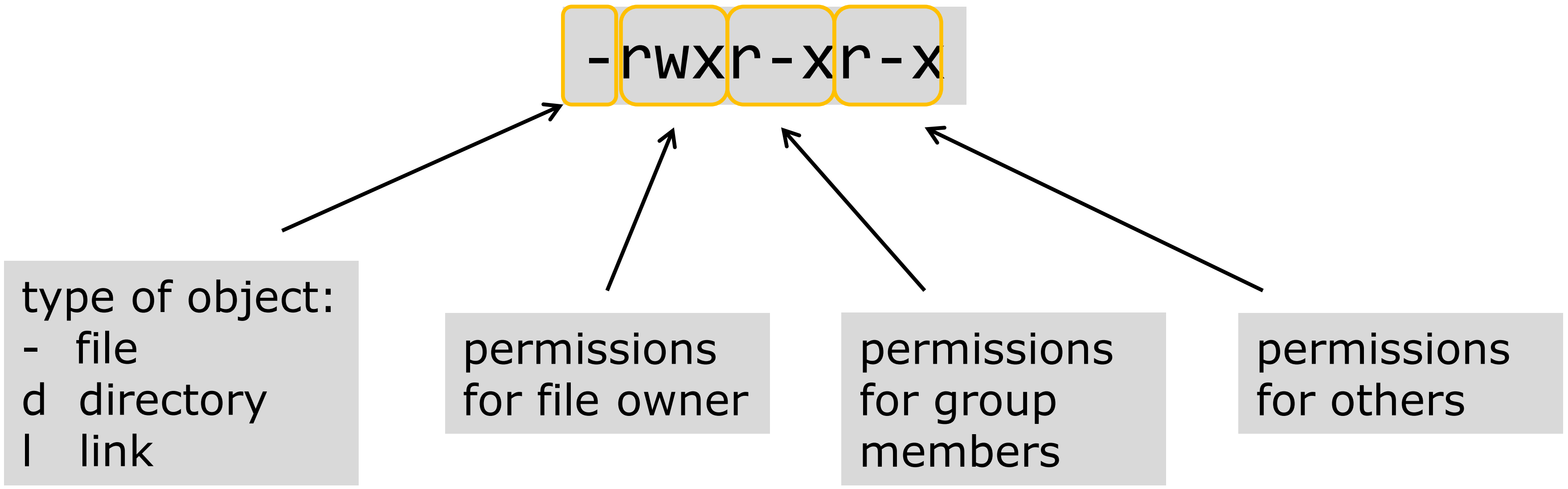
```
$ grep -A 2 brun1992 /etc/passwd
brun1992:x:11481:8041:Glenn Bruns:/home/CLASSES/brunsglenn:/bin/bash
agui2801:x:11482:120:Erin Margaret Aguilar:/home/CLASSES/aguilarerinm:/bin/bash
breu6125:x:11483:120:Chris Julian Breuner:/home/CLASSES/breunerchrisj:/bin/bash
```

```
$ groups
shell_faculty domain^users faculty power^users genetec^cardholder iptv csumb^-
^iptv csumb_faculty csumb_genetec csumb_powerusers sslvpnfaculty mlc104_faculty
csumb_templecturer walda_user csumb_allfaculty sophosuser
```

Permissions

owner group

```
$ ls -l
total 48
drwxr-xr-x. 2 brun1992 shell_faculty 4096 Sep 18 21:18 backup
-rw-r--r--. 1 brun1992 shell_faculty   53 Sep 13 08:20 Makefile
-rwxr-xr-x. 1 brun1992 shell_faculty 8262 Sep 25 12:34 msh3
-rw-r--r--. 1 brun1992 shell_faculty 4279 Oct  1 15:17 msh.c
-rw-r--r--. 1 brun1992 shell_faculty 1987 Sep 14 20:46 msh-hw2.c
-rw-r--r--. 1 brun1992 shell_faculty   58 Aug 27 11:52 README.txt
```



Permissions: read, write, execute

```
$ ls -l
total 48
drwxr-xr-x. 2 brun1992 shell_faculty 4096 Sep 18 21:18 backup
-rw-r--r--. 1 brun1992 shell_faculty   53 Sep 13 08:20 Makefile
-rwxr-xr-x. 1 brun1992 shell_faculty 8262 Sep 25 12:34 msh3
-rw-r--r--. 1 brun1992 shell_faculty 4279 Oct  1 15:17 msh.c
-rw-r--r--. 1 brun1992 shell_faculty 1987 Sep 14 20:46 msh-hw2.c
-rw-r--r--. 1 brun1992 shell_faculty   58 Aug 27 11:52 README.txt
```

```
-rwxr-xr-x
```

↑

r	read
w	write
x	execute

Questions:

- Who is allowed to read Makefile?
- Who is allowed to modify msh.c?
- Who is allowed to run msh3?

Examples

```
$ ls -l
total 48
drwxr-xr-x. 2 brun1992 shell_faculty 4096 Sep 18 21:18 backup
-rw-r--r--. 1 brun1992 shell_faculty   53 Sep 13 08:20 Makefile
-rwxr-xr-x. 1 brun1992 shell_faculty 8262 Sep 25 12:34 msh3
-rw-r--r--. 1 brun1992 shell_faculty 4279 Oct  1 15:17 msh.c
-rw-r--r--. 1 brun1992 shell_faculty 1987 Sep 14 20:46 msh-hw2.c
-rw-r--r--. 1 brun1992 shell_faculty   58 Aug 27 11:52 README.txt
```

	type	owner permissions	group permissions	others permission
`-rw-r--r--`	file	read/write	read	read
`-rwxr-xr-x`	file	read/write/exec	read/exec	read/exec
`-rw-------`	file	read/write	none	none
`drwxr-xr-x`	dir	read/write/exec	read/exec	read/exec

Directory permissions

```
$ ls -l
total 48
drwxr-xr-x. 2 brun1992 shell_faculty 4096 Sep 18 21:18 backup
-rw-r--r--. 1 brun1992 shell_faculty   53 Sep 13 08:20 Makefile
-rwxr-xr-x. 1 brun1992 shell_faculty 8262 Sep 25 12:34 msh3
-rw-r--r--. 1 brun1992 shell_faculty 4279 Oct  1 15:17 msh.c
-rw-r--r--. 1 brun1992 shell_faculty 1987 Sep 14 20:46 msh-hw2.c
-rw-r--r--. 1 brun1992 shell_faculty   58 Aug 27 11:52 README.txt
```

In a directory:

- read: user can list files in the directory
- write: user can create, rename, delete files in the directory, and can modify directory attributes
- execute: user can enter the directory and access files and directories within

Directory permissions can get tricky.

Setting permissions

```
$ ls -l
total 28
drwxr-xr-x. 2 brun1992 shell_faculty 4096 Oct 20 14:48 backup
-rw-r--r--. 1 brun1992 shell_faculty   41 Oct 20 14:48 Makefile
-rwxr-xr-x. 1 brun1992 shell_faculty 4924 Oct 20 14:48 ssort
-rw-r--r--. 1 brun1992 shell_faculty  332 Oct 20 14:48 ssort.c
-rwxr-xr-x. 1 brun1992 shell_faculty 5786 Oct 20 14:48 tests1
$
$ chmod 600 Makefile
$ ls -l Makefile
-rw-------. 1 brun1992 shell_faculty 41 Oct 20 14:48 Makefile
$
$ chmod +x Makefile
$ ls -l Makefile
-rwx--x--x. 1 brun1992 shell_faculty 41 Oct 20 14:48 Makefile
$ chmod -x Makefile
$ ls -l Makefile
-rw-------. 1 brun1992 shell_faculty 41 Oct 20 14:48 Makefile
```

octal mode

symbolic mode

Example

```
$ cat > temp
#!/bin/bash
ls -l | sort
$ ls -l temp
-rw-r--r--. 1 brun1992 shell_faculty 25 Mar 27 11:47 temp
$ chmod u+x temp
$ ls -l temp
-rwxr--r--. 1 brun1992 shell_faculty 25 Mar 27 11:47 temp
$ chmod a+x temp
$ ls -l temp
-rwxr-xr-x. 1 brun1992 shell_faculty 25 Mar 27 11:47 temp
```

u	user
g	group
o	others
a	all

chmod +x is shorthand for chmod a+x

Octal mode

An octal (like "eight") number is a number from 0 to 7.
In binary it is three bits.

rw- is 110 in binary, or 6 in octal
r-x is 101 in binary, or 5 in octal

Question: what number if I want all permission, want everyone else to have no permissions?

```
$ ls -l *.c
-rw-rw-rw- 1 brun1992 shell_faculty 0 Oct  3  2016 baz.c
-rw-r--r-- 1 brun1992 shell_faculty 0 Oct  3  2016 foo.c
$ chmod 644 baz.c
$ ls -l *.c
-rw-r--r-- 1 brun1992 shell_faculty 0 Oct  3  2016 baz.c
-rw-r--r-- 1 brun1992 shell_faculty 0 Oct  3  2016 foo.c
$ chmod 666 baz.c
$ ls -l *.c
-rw-rw-rw- 1 brun1992 shell_faculty 0 Oct  3  2016 baz.c
-rw-r--r-- 1 brun1992 shell_faculty 0 Oct  3  2016 foo.c
```

sort – sorting data

```
$ who > temp.txt
$
$ wc -l temp.txt
52 temp.txt
$
$ head -3 temp.txt
grab9610 pts/0        2015-09-10 11:42 (10.11.157.14)
anto1513 pts/1        2015-09-10 09:59 (10.12.171.160)
shaw9409 pts/2        2015-09-10 10:02 (10.11.178.106)
$
$ sort temp.txt | head -3
alex4124 pts/21       2015-09-10 10:37 (10.12.171.184)
alex4124 pts/50       2015-09-10 11:17 (10.12.171.184)
amar6699 pts/18       2015-09-10 10:37 (10.12.171.95)
```

The 'who' command shows who is logged in

sort has lots of options:

- select the field to use for sorting
- specify how to sort (numerically, alphabetically, etc.)
- specify field delimiter
- …

sort – common use cases

```
$ sort -r temp.txt | head -3
yoo9408  pts/27       2015-09-10 10:37 (10.12.171.133)
snyd4924 pts/23       2015-09-10 10:37 (10.12.171.187)
smit9960 pts/29       2015-09-10 10:38 (10.12.171.146)
$
$ sort -k 4,4 temp.txt | head -3
ibar1694 pts/5        2015-09-10 10:10 (10.11.132.71)
brun1992 pts/6        2015-09-10 10:30 (10.11.84.204)
aria3918 pts/12       2015-09-10 10:36 (10.11.128.211)
$
$ sort -k 1.5,1.9 temp.txt | head -3
grec1046 pts/28       2015-09-10 10:38 (10.11.160.176)
dhar1102 pts/10       2015-09-10 10:47 (10.11.129.215)
sixt1161 pts/46       2015-09-10 11:23 (10.11.116.163)
$
$ sort -t. -k3n,3 temp.txt | head -3
brun1992 pts/25       2015-09-10 11:06 (10.11.84.204)
brun1992 pts/6        2015-09-10 10:30 (10.11.84.204)
rich7002 pts/15       2015-09-10 10:37 (10.11.113.218)
```

reverse sort

sort key is field 4

sort key is characters 5-9 of field 1

'.' is field delimiter; sort key is field 3; sorting is numeric

Example: sort and du

```
$ du -h
4.0K    ./public_html
44K     ./fall15/os/homework
60K     ./fall15/os
64K     ./fall15
188K    ./data1
66M     ./data
4.0K    ./Mail
24K     ./.emacs.d/auto-save-list
28K     ./.emacs.d
16K     ./ctests/addresses
44K     ./ctests/ptrs
36K     ./ctests/assem
20K     ./ctests/gsh/backup
44K     ./ctests/gsh
188K    ./ctests/pro
344K    ./ctests
12K     ./mail
16K     ./bin
66M     .
```

This shows space used by current directory and all subdirectories

```
$ du -h | sort -nr
344K    ./ctests
188K    ./data1
188K    ./ctests/proc_api
66M     ./data
66M     .
64K     ./fall15
60K     ./fall15/os
44K     ./fall15/os/homework
44K     ./ctests/ptrs
44K     ./ctests/gsh
36K     ./ctests/assem
28K     ./.emacs.d
```

A reverse, numeric sort on ‘du’ output

sort and uniq

```
$ cat nums.txt
2
4
2
4
0
4
$ sort -n nums.txt
0
2
2
4
4
4
```

```
$ sort -n nums.txt | uniq
0
2
4
$ sort -n nums.txt | uniq -c
      1 0
      2 2
      3 4
$ uniq nums.txt
2
4
2
4
0
4
```

diff – comparing files

```
$ cat hello.c
#include <stdio.h>

int main() {
   printf("hello\n");
}
$ cat hello-new.c
#include <stdio.h>

int main() {
   printf("hello\n");
   exit(0);
}
$ diff hello.c hello-new.c
4a5
>    exit(0);
$
```

Diff does a lot more than simple file comparison:

- you can have 'diff' ignore case differences

  option -i

- you can have 'diff' ignore all whitespace

  option -w

- you can compare sets of files at once
- it can even recursively compare directories!

  option -r

tar – creating an archive

```
$ ls
bin      data    fall15   Mail          README.TXT   temp        temp.txt
ctests   data1   mail     public_html   sent         temp1.txt   test.txt
$ ls data
complaints.csv   ps-out.txt   salaries.csv   songs2.csv   temp.txt
employees.txt    README.txt   songs1.csv     songs.csv
$ tar cvf data.tar data
data/
data/complaints.csv
data/README.txt
data/employees.txt
data/songs1.csv
data/songs.csv
data/songs2.csv
data/salaries.csv
data/ps-out.txt
data/temp.txt
$ ls
bin      data    data.tar   mail   public_html   sent   temp1.txt   test.txt
ctests   data1   fall15     Mail   README.TXT    temp   temp.txt
```

c = create
v = verbose
f = use following name as output filename

tar – extracting files from an archive

```
$ mkdir tempdir
mv data.tar tempdir
$ cd tempdir
$ ls
data.tar
$ tar xvf data.tar
data/
data/complaints.csv
data/README.txt
data/employees.txt
data/songs1.csv
data/songs.csv
data/songs2.csv
data/salaries.csv
data/ps-out.txt
data/temp.txt
$ ls
data  data.tar
$ ls data
complaints.csv  ps-out.txt  salaries.csv  songs2.csv  temp.txt
employees.txt   README.txt  songs1.csv    songs.csv
```

x = extract
v = verbose
f = use following name as
input filename

tar saves and recreates the directory tree structure

essential tar commands

`tar cf tarfile files`	creates tar file from files
`tar xf tarfile`	extracts files from tar file

remember: tar file comes immediately after the 'f'

Examples:

```
tar cf foo.tar *.c
```

```
tar xf foo.tar
```

```
tar -xf foo.tar
```

dash is optional

gzip – file compression

gzip - compress   gunzip - uncompress

```
$ ls -l *.csv
-rw-r--r--. 1 brun1992 shell_faculty 67216929 Sep 10 13:07 complaints.csv
-rw-r--r--. 1 brun1992 shell_faculty   742652 Sep 10 13:07 salaries.csv
-rw-r--r--. 1 brun1992 shell_faculty    90587 Sep 10 13:07 songs1.csv
-rw-r--r--. 1 brun1992 shell_faculty    49344 Sep 10 13:07 songs2.csv
-rw-r--r--. 1 brun1992 shell_faculty    90707 Sep 10 13:07 songs.csv
$
$ gzip *.csv
$ ls -l *.gz
-rw-r--r--. 1 brun1992 shell_faculty 7700687 Sep 10 13:07 complaints.csv.gz
-rw-r--r--. 1 brun1992 shell_faculty  191971 Sep 10 13:07 salaries.csv.gz
-rw-r--r--. 1 brun1992 shell_faculty   34552 Sep 10 13:07 songs1.csv.gz
-rw-r--r--. 1 brun1992 shell_faculty   17914 Sep 10 13:07 songs2.csv.gz
-rw-r--r--. 1 brun1992 shell_faculty   34595 Sep 10 13:07 songs.csv.gz
$
$ gunzip *.gz
$ ls -l *.csv
-rw-r--r--. 1 brun1992 shell_faculty 67216929 Sep 10 13:07 complaints.csv
-rw-r--r--. 1 brun1992 shell_faculty   742652 Sep 10 13:07 salaries.csv
-rw-r--r--. 1 brun1992 shell_faculty    90587 Sep 10 13:07 songs1.csv
-rw-r--r--. 1 brun1992 shell_faculty    49344 Sep 10 13:07 songs2.csv
-rw-r--r--. 1 brun1992 shell_faculty    90707 Sep 10 13:07 songs.csv
```

Summary

- file permissions
  - how they work
  - modifying them with chmod
- Commands introduced in this lecture:
  - chmod
  - who, sort, uniq, diff, gzip, gunzip, tar

Bonus: Software tools philosophy

Unix culture includes a philosophy on how to build powerful, flexible software tools.

The philosophy includes these ideas:

- do one thing well
- process lines of text, not binary format
- use regular expressions
- default to standard I/O
- don't be chatty

source: Robbins and Beebe, Classic Shell Scripting, section 1.2