

Unix File Structure

Benjamin Brewster

Adapted from slides by Jon Herlocker, OSU

One more cat demo

```
% echo "FISH"
```

```
FISH
```

```
% echo "FISH" > net.txt
```

```
% cat net.txt
```

```
FISH
```

```
% echo "CRAB" > net.txt
```

```
% cat net.txt
```

```
CRAB
```

```
% echo "BARNACLE" >> net.txt
```

```
% cat net.txt
```

```
CRAB
```

```
BARNACLE
```

Files!

- If everything is a file, how does the file structure work?

Files!

- Files are “hard links” to “inodes”.
- Inodes:
 - Contain all meta-information (size, permissions, etc)
 - Contain a "reference count":
 - How many hard links to the inode
 - Contain pointers to actual file data
 - Are identified by a unique number:
 - inode number

Hard Links

- An entry in a file system “directory” that points to an inode
- Connects a text file name to an inode on disk

Directories!

- You can create or remove directories
 - `mkdir()`, `rmdir()`
- Or read the contents of a directory
 - `opendir()`, `closedir()`, `readdir()`, `rewinddir()`
- `.`
- `..`

What's in a directory file?

```
% ls -pla
drwxr-xr-x  2 brewstbe upg22026   512 Jun 22 16:44 ./
drwxr-xr-x  8 brewstbe ftp       1024 Jun 22 15:46 ../
-rw-r--r--  1 brewstbe upg22026  1027 Jun 22 15:47 cursesDemo.c
-rw-r--r--  1 brewstbe upg22026 42558 Jun 22 15:55 Curses.pdf
-rw-r--r--  1 brewstbe upg22026  4208 Jun 22 16:24 index.html
-rw-r--r--  1 brewstbe upg22026 61554 Jun 22 15:46 IntroToUnixShell.html
-rw-r--r--  1 brewstbe upg22026    38 Jun 22 15:46 OnE1FAQ.txt
-rw-----  1 brewstbe upg22026   467 Jun 22 15:46 OnE1_sol.txt
-rw-r--r--  1 brewstbe upg22026   288 Jun 22 15:46 OnE1.txt
-rw-r--r--  1 brewstbe upg22026    38 Jun 22 15:55 Prog1FAQ.txt
-rw-r--r--  1 brewstbe upg22026  8098 Jun 22 15:46 Prog1.html
-rw-r--r--  1 brewstbe upg22026  7114 Jun 22 15:46 Prog1.test
-rw-r--r--  1 brewstbe upg22026    38 Jun 22 15:46 Prog2FAQ.txt
-rw-r--r--  1 brewstbe upg22026  4517 Jun 22 16:14 Prog2.html
```

Permissions	owner	group	size (bytes)	last modified	name
	hard link count				

What's in a directory file?

```
% vi .
```

```
" Press ? for keyboard shortcuts
" Sorted by name (.bak,~, .o, .h, .info, .swp, .obj at end of list)
"= /nfs/rack/u2/b/brewstbe/public_html/CS311/
../
Curses.pdf
IntroToUnixShell.html
OnE1.txt
OnE1FAQ.txt
OnE1_sol.txt
Prog1.html
Prog1.test
Prog1FAQ.txt
Prog2.html
Prog2FAQ.txt
cursesDemo.c
index.html
```



```
DIR *directory;  
struct dirent *dir_entry;  
  
directory = opendir(argv[1]);
```

```
if (directory == NULL)  
{  
    fprintf(stderr, "Could not open directory %s\n",  
argv[1]);  
    perror(argv[0]);  
    exit(1);  
}
```



Note the dangerous single equals

```
while (dir_entry = readdir(directory))  
{  
    if (dir_entry->d_ino != 0)  
        printf("%s\n", dir_entry->d_name);  
}
```

```
closedir(directory);
```

Creating Links

- When you create a file (using `open`), an inode is allocated and a hard link is automatically created
- However, you can create multiple hard links to the same inode
 - So a file can appear in multiple directories at the same time!
 - The same file can also appear under different names
 - even in the same directory
- To create a link, use the `ln` or `link` commands

Removing Files

- Removing is approximately unlinking everything
 - The inode is garbage collected when ref count == 0
- One way to "remove" a file:
 - `unlink(file_name)`
 - can't unlink directories
- Another way to "remove" a file
 - `remove(file_name)`
 - unlike `unlink`, `remove` will delete empty directories
 - if file, `remove` is identical to `unlink`
 - if directory, `remove` is identical to `rmdir`

Symbolic Link

- A symbolic link is like a Windows shortcut - It's not actually a file, it refers to the file.
- If you delete a file, any symbolic links to it become unusable, whereas a hard link to a file means that the file is not yet deleted.
- You can link to directories, or to files on another computer
 - Which you can't do with a hard link

Symbolic Link Example

```
flip% ssh babylon.eecs.oregonstate.edu -l brewsteb
```

- I'm on babylon, but I want access to my EECS files

```
babylon% ln -s /nfs/rack/u2/b/brewsteb eeecsLink  
babylon% cd eeecsLink
```

- I am now in my EECS filesystem using babylon!

Another Link Example

```
% touch temp
% ls -pla temp*
-rw----- 1 brewsteb upg22026      13 Jun 29 14:49 temp
% ln temp temp2
% ls -pla temp*
-rw----- 2 brewsteb upg22026      13 Jun 29 14:49 temp
-rw----- 2 brewsteb upg22026      13 Jun 29 14:49 temp2
% echo "Hello World!" >> temp
% cat temp
Hello World!
% cat temp2
Hello World!
% rm -f temp
% ls -pla temp*
-rw----- 1 brewstbe upg22026      13 Jun 29 14:49 temp2
```

Permissions

- Files in UNIX have access permissions for three designations of people:
 - owner
 - group
 - all others
- Three kinds of access permissions for each:
 - read
 - write
 - execute

Read Permissions

- File
 - The file's contents can be read
- Directory
 - The directory's contents can be read (ie, a listing of the files in the directory can be returned)

Write Permissions

- File
 - The file can be written to (ie, the contents of the file can be changed)
- Directory
 - Files can be added/removed/renamed/etc. to/in the directory

Execute Permissions

- File
 - The file appears in directories
 - The file can be executed (program, shell script)
- Directory
 - The directory can be cd'd to
 - The contents can be listed
 - The directory appears in its parents directory listing

chmod

- You can change the permissions on a file by using the `chmod` (**change mode**) command

Man page deficiency – no `chmod` examples!

- Here is a sample file listing (generated by `ls -pla`) of a file and dir:

```
-rw-r--r--  1 brewstbe upg22026  4517 Jun 22 16:14 Prog2.html  
drwx--x--x  2 brewstbe upg22026   512 Jun 22 17:48 tempDir/
```

Owner

Group

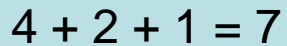
All others

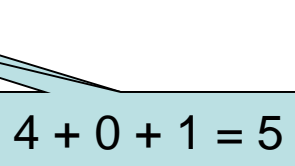
chmod

- Why does everything in UNIX have to be so hard?

- $r = 4$
- $w = 2$
- $x = 1$

```
-rw-r--r-- 1 brewstbe upg22026 4517 Jun 22 16:14 Prog2.html  
drwxr-xr-x 2 brewstbe upg22026 512 Jun 22 17:48 tempDir/
```

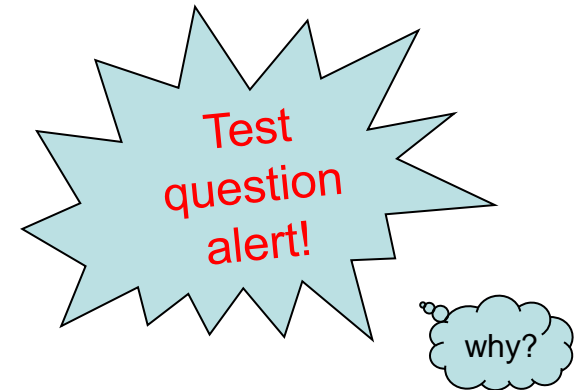

$$4 + 2 + 1 = 7$$


$$4 + 0 + 1 = 5$$

```
% chmod 644 Prog2.html    <- Standard rights for a publicly viewable file  
% chmod 755 tempDir       <- Standard rights for a publicly viewable directory
```

chmod

- There IS an easier way:



```
----- 1 brewstbe upg22026 4517 Jun 22 16:14 Prog2.html
```

```
% chmod o+rx Prog2.html
```

```
-rwx----- 1 brewstbe upg22026 4517 Jun 22 16:14 Prog2.html
```

```
% chmod g+rx,a+rx Prog2.html
```

```
-rwxr-xrwx 1 brewstbe upg22026 4517 Jun 22 16:14 Prog2.html
```

```
% chmod a-w Prog2.html
```

```
-rwxr-xr-x 1 brewstbe upg22026 4517 Jun 22 16:14 Prog2.html
```

umask

- The *creation mask* setting defines the default **attributes** for new files.
- You can set this mask with `umask`
- If no argument is included, *umask* displays the current setting

<http://unix.t-a-y-l-o-r.com/UAumask.html>

umask

- Since this is a mask...

```
umask 022
```

- ...would give the owner full privileges, while the group and all others would not have write privileges
- This is the complement of what we saw with `chmod`

<http://unix.t-a-y-l-o-r.com/UAumask.html>

umask

```
% umask
```

```
77
```

```
% umask 077
```

```
% umask
```

```
77
```

```
% umask 707
```

```
% umask
```

```
707
```


Which groups?

```
flip 134 CS311% id  
uid=22026(brewstbe) gid=6009(upg22026)  
groups=6009(upg22026),12028(transfer)
```

- These groups are the same groups referred to when using the `chmod` command

du

- Returns the total in kilobytes of the specified directory

```
% du
```

```
1479 .
```

df

- Displays unused space:

```
%df
Filesystem                1K-blocks      Used Available Use% Mounted on
/dev/sda8                  1035660    462276    520776   48% /
/dev/sda1                   101086     16797     79070   18% /boot
none                       2045244         0    2045244    0% /dev/shm
/dev/sda7                  1035660     34112     948940    4% /private
/dev/sda9                  17346660    585992   15879508    4% /scratch
/dev/sda5                   4127076     41108    3876324    2% /tmp
/dev/sda2                   5162828   3934176     966392   81% /usr
/dev/sda3                   4127108    337768    3579692    9% /var
guille.eecs.oregonstate.edu:/a2
                           154893632   71398656   81946048   47% /nfs/guille/a2
stak.ENGR.ORST.EDU:/a1
                           252994240   84486624  165977664   34% /nfs/stak/a1
stak.ENGR.ORST.EDU:/u14
                           206524224   67558688  136900288   34% /nfs/stak/u14
rack.engr.oregonstate.edu:/u2
                           139490368   48546656   89548800   36% /nfs/rack/u2
```

Standard Directories

- Root dir:

- /

- Home dir:

- ~

- Bad idea:

- `rm -rf /*`