Files and Directories

Glenn Bruns
CSUMB

Lecture Objectives

After this lecture, you should be able to:

- list some design goals for file systems
- explain the file system abstractions for persistent storage (files, directories)
- explain the concepts of hard link, symbolic link, volume, mount

Persistent storage

Why do we need more than the memory within a process' address space?

- it's not very big
- data is lost when process terminates
- data can't be shared across processes

We need permanent, or persistent data storage.

Exercise: name some kinds of persistent storage.

Virtualizing persistent storage

A key OS goal is to provide convenient abstractions.

What abstraction for persistent storage?

One idea: view a hard drive as a "linearly addressable" sequence of blocks.

Why not?

- need to make it easy to find information
- need to protect data access
- need to know which blocks are in use

Design goals for persistant storage

Can you name some design goals?

- High performance
- Meaningful naming scheme
- Controlled sharing/protection
- Reliability

File system concept

It's easy to forget how great the file concept is.

Two key abstractions:

- file
  - a linear array of bytes
  - has some kind of id (inode number)
  - OS knows little about file structure
- directory
  - has some kind of id (inode number)
  - contains a list of (user-readable-name, inode) pairs
  - the inode in a pair can be a directory inode

Example

files

3
bytes ….

5
bytes ….

10
bytes ….

12
bytes ….

directories

2
(foo, 3)
(bar, 5)
(baz, 15)

15
(thud, 12)

The idea that “directories contain files” is just a metaphor.

inodes and hard links

```
$ cat foo
crazzle
$ ls -i
1064831 foo
$ ln foo bar
$ ls -i
1064831 bar  1064831 foo
$ ls
bar  foo
$ ls -l
total 8
-rw-r--r--. 2 brun1992 shell_faculty 8 Nov  5 14:26 bar
-rw-r--r--. 2 brun1992 shell_faculty 8 Nov  5 14:26 foo
$ rm foo
$ cat bar
crazzle
$ ls -l
total 4
-rw-r--r--. 1 brun1992 shell_faculty 8 Nov  5 14:26 bar
$
```

create “hard link” bar

foo and bar are equals

Recall: pathnames

absolute

- start with '/'
- examples: / /foo /home/CLASSES

relative

- don't start with ‘/’
- interpreted relative to current working directory

File API

- creating a file: open()
- reading/writing: read(), write()
- writing immediately: fsync()
- renaming: rename()
- getting status: stat(), fstat()
- deleting: unlink() ?!

man page:

unlink() deletes a name from the filesystem. If that name was the last link to a file and no processes have the file open, the file is deleted and the space it was using is made available for reuse.

Directory API

- creating a directory: mkdir()
- reading: opendir(), readdir(), closedir()
- deleting: rmdir()
- adding entry: link(old_path, new_path)
- removing entry: unlink()

The file system keeps track of how many file names have been linked to an inode.

When the count becomes zero, the inode and related disk blocks are freed.

Symbolic links

- looks similar to a hard link
- a symbolic link is actually a file
- contents of the symbolic link is the pathname of the linked-to file

Symbolic links example

```
$ echo foo > file
$ ln -s file file2
$ cat file
foo
$
$ ls -l
total 4
-rw-r--r--. 1 brun1992 shell_faculty 4 Nov  5 15:36 file
lrwxrwxrwx. 1 brun1992 shell_faculty 4 Nov  5 15:36 file2 -> file
$
$ rm file
$ ls -l
total 0
lrwxrwxrwx. 1 brun1992 shell_faculty 4 Nov  5 15:36 file2 -> file
$
$ cat file2
cat: file2: No such file or directory
$
```

create “symbolic link” file2

file and file2 are **not** equals

Disk partitions and volumes

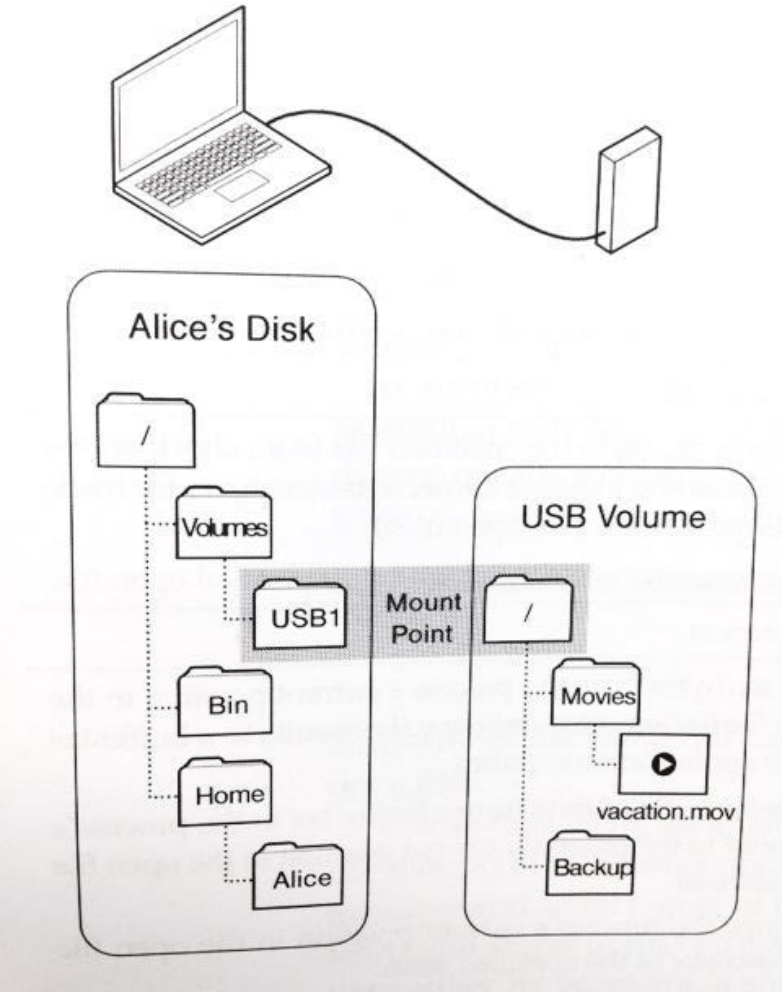
- A physical disk can be subdivided into partitions
- sometimes volume is used for a partition on which a file system has been created
- A related idea is logical volume, which could be part of a drive, or span multiple drives
- The concept of logical volume is now supported on Linux through the Logical Volume Manager

File systems and mounts

- You can create a file system on a partition
- A file system is basically an on-disk data structure
- you can then mount this file system at a directory in the system’s file-system tree

If you are interested, look into the 'mkfs' and 'mount' commands.

Mounting a USB drive



The file system on the flash drive is mounted at /Volumes/USB1.

Then, Alice can access vacation.mov using the path /Volumes/USB1/Movies/vacation.mov

Mounted file systems on hosting.csumb

```
$ mount
/dev/mapper/vg_mlc104-lv_root on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs
       (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda1 on /boot type ext4 (rw)
/dev/mapper/vg_mlc104-lv_home on /home type ext4 (rw)
/dev/mapper/vg_mlc104-lv_var on /var type ext4 (rw)
/dev/mapper/vg0-lv--users on /home/CLASSES type ext4
       (rw,usrquota,grpquota)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

Summary

We learned:

- some design criteria for file systems
- about file and directory abstractions for persistent storage
- how directories work; hard and soft links
- creating a file system and mounting it

Storage capacity of modern drives

If you printed 1 TB of text on paper, how high would the stack of paper be:

a) 100 ft.

b) 500 ft.

c) 2000 ft.

answer: 20 miles