Filing

Services for Non-Volatile Storage

1

What does a File System Do?

- Create/Modify/Delete units of secondary storage (the file)
- Organize files logically
- Permit sharing of data between processes, people, and machines
  - may even provide for synchronization via locking
- Protect data from prying eyes

2

What's in a file?

- A file is an array of bytes with some properties
  - contents, size, owner, last read time, last write time, protection (who, what), parent
  - type
    - understood by file system
      - block, character, device, portal, link
    - understood by other parts of the OS or supporting services
      - executable, dll, source, object, text, jpg
    - type may be encoded in name, or contents
      - NT encodes type in name
        - » .com, .exe, .jpg, .dll, .msc
      - UNIX encodes type in contents
        - » Magic numbers

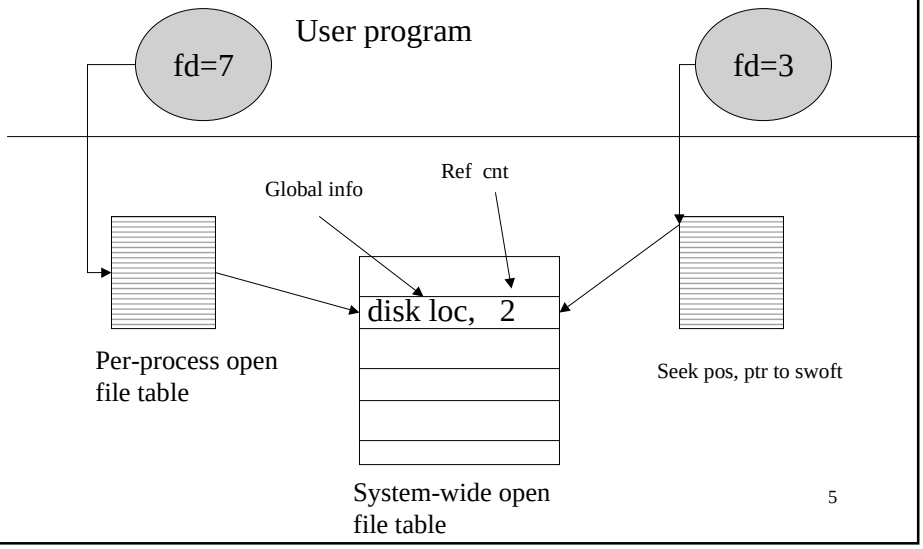
3

Basic File Operations

- fd = create(filename)
  - create the named file
- fd = open(filename, how)
  - returns handle to named file
  - generally involves lots of work to get the file open
- cc = read(fd, buf, len)
  - read next len bytes into buf from opened file
- cc = write(fd, buf, len)
  - write len bytes from buf into opened file at current position
- sync(fd)
  - commit any uncommitted writes
- seek(fd, pos)
  - adjust "current position" to affect next read or write operation
- close(fd)
  - release resources for opened file
- delete(filename)
  - remove the named file from the file system

4

Key OS Data Structures



5

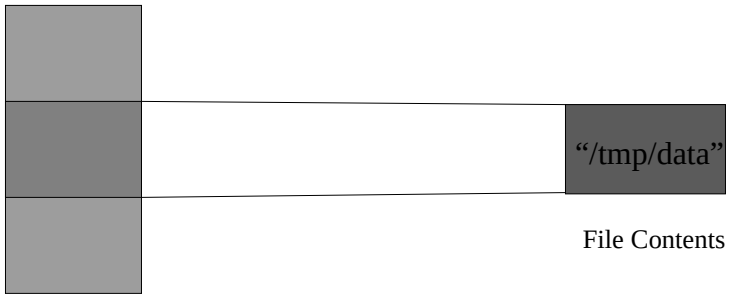
Memory Mapped Files

- Problem: programs use read and write to transfer data from/to files to/from memory.
- Once in memory they use load/store to manipulate the data
- Introduces two serious ineffeciencies
  - programmers have to think about data in two ways
    - is it a file or is it memory
  - data must be copied from the fs into vm and then back
- Insight: Recall that VM is a client of the FS (demand paging, for example)…
  - generalize to memory mapped files

6

A Memory Mapped File

```
char *base;
int len;
err = map_file("/tmp/data", &base, &len);
for (i = 0; i < len;i++)
        putchar(base[i]);
```



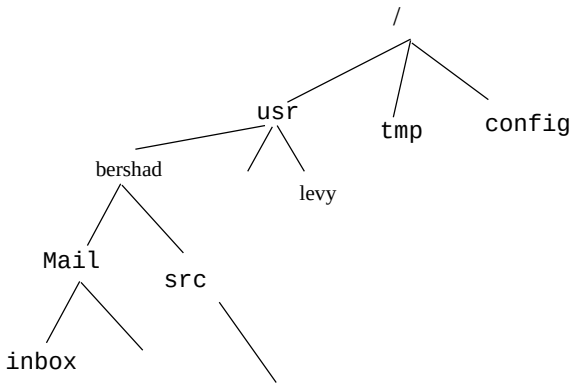
7

Directories

- Directories serve two purposes
  - for users, they provide a structured way to organize related files.
  - for the file system, they provide a convenient naming interface which allows the implementation to hide details about where a file's particular data item .
- Most systems support multi-level directories, where a file name describes its path from a root through the directories to the file at the leaf.
- Most systems have a *current* directory, from which names can be specified relatively, as opposed to absolutely from the root of the directory tree.
- Directories are an example of a naming hierarchy.
  - naming hierarchies appear all over in the design of systems.
  - //usr/bershad/Mail

8

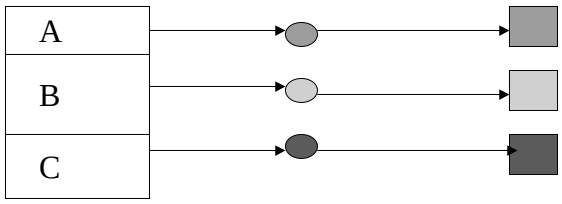
A directory hierarchy



9

What's in a Directory?

- A directory is simply a list of entries -- names and references to metadata
  - metadata is not the data itself, but information that says how to get the data
  - of course, all of this is stored on disk



Three files' names Three files' metadata Three files' data

10

Getting to There

- A file system has a root.
  - In UNIX, this is /
  - In NT, there may be many roots, one per logical disk (c:\)
- The root directory is at a well known place on the disk (never moves)
- The root directory contains the names of other directories and files.
- Systems tend to spend a lot of time walking the directory.
  - this is why open is separate from read/write.

11

File Sharing

- File sharing has been around since the days of timesharing
  - it was one of the first things missed when we went from timesharing to personal computers.
- Whether we're on a network or not, file sharing is incredibly important for getting work done.
- Two key issues arise when sharing files
  - semantics of concurrent access
  - protection

12

Protection

- A protection system allows us to specify
  - WHO can access a file and
  - HOW they can access it
- More generally, we have
  - Objects,
  - Subjects,
  - Actions
- A protection system tells us if a given action performed by a given subject on a given object should be allowed.
  - "Johnny can read his email but not delete it."
  - "Suzy can not read or write Johnny's email"

objects

subjects	**/a**	**/b**	**/c**	**/d**
Johnny	**rw**	**-**	**r**	**w**
Suzie	**w**	**-**	**-**	**rw**
Fred	**w**	**rw**	**rw**	**-**
Brian	**rw**	**rw**	**rw**	**rw**

13

Two Key Ways to Represent Protection

- Access Control List
  - For each object, maintain a list of principals and their permitted actions.
- Capability
  - For each subject, maintain a list of objects and their permitted actions.

objects

subjects	/a	/b	/c	/d
Johnny	rw	-	r	w
Suzie	w	-	-	rw
Fred	w	rw	rw	-
Brian	rw	rw	rw	rw

The approaches differ only in how we represent the table (and the implications)

14

In Practice...

- ACLs are easier to manage
  - "object"-centric
    - easy to grant, revoke
- ACLs have a problem when objects are heavily shared
  - the ACL gets quite big
  - Groups are pretty effective here.
    - Note UNIX
- READ-capabilities can be simulated with encryption

15

Reliability & Availability

- Reliability
  - if you put something there, you can get it back later.
  - backup/restore is a popular mechanism
  - consistency semantics can be quite painful however
- Availability
  - if you put something there, you can get it back sooner rather than later
  - disk mirroring

16

Summary

- OS exports a File System API for use by clients.
- Key issues yet to be discussed are how this actually gets implemented.

17