

CSE 120

Principles of Operating Systems

Fall 2000

Lecture 13: File Systems

Geoffrey M. Voelker

File Systems

- Last time we talked about the physical characteristics of disks and their performance
- Today we'll talk about file systems
 - ◆ Files
 - ◆ Directories
 - ◆ Sharing
 - ◆ Protection
 - ◆ File System Layouts
 - ◆ File Buffer Cache
 - ◆ Read Ahead

File Systems

- File systems
 - ♦ Implement an abstraction (**files**) for secondary storage
 - ♦ Organize files logically (**directories**)
 - ♦ Permit sharing of data between processes, people, and machines
 - ♦ Protect data from unwanted access (security)

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

3

Files

- A file is data with some properties
 - ♦ Contents, size, owner, last read/write time, protection, etc.
- A file can also have a type
 - ♦ Understood by the file system
 - » Block, character, device, portal, link, etc.
 - ♦ Understood by other parts of the OS or runtime libraries
 - » Executable, dll, source, object, text, etc.
- A file's type can be encoded in its name or contents
 - ♦ Windows encodes type in name
 - » .com, .exe, .bat, .dll, .jpg, etc.
 - ♦ Unix encodes type in contents
 - » Magic numbers, initial characters (e.g., #! for shell scripts)

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

4

Basic File Operations

Unix

- creat(name)
- open(name, how)
- read(fd, buf, len)
- write(fd, buf, len)
- sync(fd)
- seek(fd, pos)
- close(fd)
- unlink(name)

NT

- CreateFile(name, CREATE)
- CreateFile(name, OPEN)
- ReadFile(handle, ...)
- WriteFile(handle, ...)
- FlushFileBuffers(handle, ...)
- SetFilePointer(handle, ...)
- CloseHandle(handle, ...)
- DeleteFile(name)
- CopyFile(name)
- MoveFile(name)

File Access Methods

- Some file systems provide different access methods that specify different ways for accessing data in a file
 - ◆ Sequential access – read bytes one at a time, in order
 - ◆ Direct access – random access given block/byte number
 - ◆ Record access – file is array of fixed- or variable-length records, read/written sequentially or randomly by record #
 - ◆ Indexed access – file system contains an index to a particular field of each record in a file, reads specify a value for that field and the system finds the record via the index (DBs)
- What file access method does Unix, NT provide?
- Older systems provide more complicated methods

Directories

- Directories serve two purposes
 - ◆ For users, they provide a structured way to organize files
 - ◆ For the file system, they provide a convenient naming interface that allows the implementation to separate logical file organization from physical file placement on the disk
- Most file systems support multi-level directories
 - ◆ Naming hierarchies (/, /usr, /usr/local, ...)
- Most file systems support the notion of a current directory
 - ◆ Relative names specified with respect to current directory
 - ◆ Absolute names start from the root of directory tree

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

7

Directory Internals

- A directory is a list of entries -- names and associated metadata
 - ◆ Metadata is not the data itself, but information that describes properties of the data (size, protection, location, etc.)
- List is usually unordered (effectively random)
 - ◆ Entries usually sorted by program that reads directory
- Directories typically stored in files
 - ◆ Only need to manage one kind of secondary storage unit

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

8

Basic Directory Operations

Unix

- Directories implemented in files
 - Use file ops to create dirs
- C runtime library provides a higher-level abstraction for reading directories
 - opendir(name)
 - readdir(DIR)
 - seekdir(DIR)
 - closedir(DIR)

NT

- Explicit dir operations
 - CreateDirectory(name)
 - RemoveDirectory(name)
- Very different method for reading directory entries
 - FindFirstFile(pattern)
 - FindNextFile()

Path Name Translation

- Let's say you want to open `"/one/two/three"`
- What does the file system do?
 - Open directory `"/` (well known, can always find)
 - Search for the entry `"one"`, get location of `"one"` (in dir entry)
 - Open directory `"one"`, search for `"two"`, get location of `"two"`
 - Open directory `"two"`, search for `"three"`, get location of `"three"`
 - Open file `"three"`
- Systems spend a lot of time walking directory paths
 - This is why open is separate from read/write
 - OS will cache prefix lookups for performance
 - » `/a/b`, `/a/bb`, `/a/bbb`, etc., all share `"/a"` prefix

File Sharing

- File sharing has been around since timesharing
 - ♦ Easy to do on a single machine
 - ♦ PCs, workstations, and networks get us there (mostly)
- File sharing is incredibly important for getting work done
 - ♦ Basis for communication and synchronization
- Two key issues when sharing files
 - ♦ Semantics of concurrent access
 - » What happens when one process reads while another writes?
 - » What happens when two processes open a file for writing?
 - ♦ Protection

Protection

- File systems implement some kind of protection system
 - ♦ Who can access a file
 - ♦ How they can access it
- More generally...
 - ♦ Objects are “what”, subjects are “who”, actions are “how”
- A protection system dictates whether a given action performed by a given subject on a given object should be allowed
 - ♦ You can read and/or write your files, but others cannot
 - ♦ You can read “/etc/motd”, but you cannot write it

Representing Protection

Access Control Lists (ACL)

- For each object, maintain a list of subjects and their permitted actions

Capabilities

- For each subject, maintain a list of objects and their permitted actions

	/one	/two	/three
Alice	rw	-	rw
Bob	w	-	r
Charlie	w	r	rw

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

13

ACLs and Capabilities

- The approaches differ only in how the table is represented
 - What approach does Unix use?
- Capabilities are easier to transfer
 - They are like keys, can handoff, does not depend on subject
- In practice, ACLs are easier to manage
 - Object-centric, easy to grant, revoke
 - To revoke capabilities, have to keep track of all subjects that have the capability – a challenging problem
- ACLs have a problem when objects are heavily shared
 - The ACLs become very large
 - Use groups (e.g., Unix)

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

14

File System Layout

How do file systems use the disk to store files?

- File systems define a block size (e.g., 4KB)
 - ♦ Disk space is allocated in granularity of blocks
- A “Master Block” determines location of root directory
 - ♦ Always at a well-known disk location
 - ♦ Often replicated across disk for reliability
- A free map determines which blocks are free, allocated
 - ♦ Usually a bitmap, one bit per block on the disk
 - ♦ Also stored on disk, cached in memory for performance
- Remaining disk blocks used to store files (and dirs)
 - ♦ There are many ways to do this

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

15

Disk Layout Strategies

- Files span multiple disk blocks
- How do you find all of the blocks for a file?
 1. Contiguous allocation
 - » Like memory
 - » Fast, simplifies directory access
 - » Inflexible, causes fragmentation, needs compaction
 2. Linked structure
 - » Each block points to the next, directory points to the first
 - » Good for sequential access, bad for all others
 3. Indexed structure (indirection, hierarchy)
 - » An “index block” contains pointers to many other blocks
 - » Handles random better
 - » May need multiple index blocks (linked together)

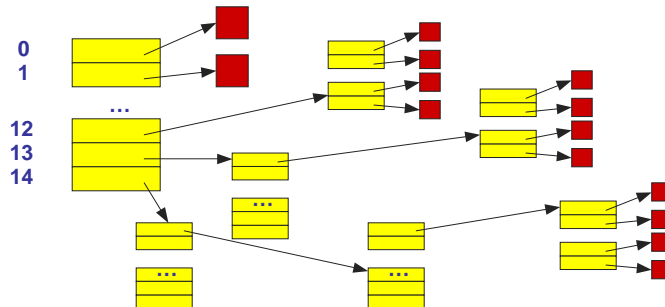
November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

16

Unix Inodes

- Unix inodes implement an indexed structure for files
- Each inode contains 15 block pointers
 - First 12 are direct blocks (e.g., 4 KB blocks)
 - Then single, double, and triple indirect



November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

17

Unix Inodes and Path Search

- Unix Inodes are **not** directories
- They describe where on the disk the blocks for a file are placed
 - Directories are files, so inodes also describe where the blocks for directories are placed on the disk
- Directory entries map file names to inodes
 - To open `"/one"`, use Master Block to find inode for `"/"` on disk
 - Open `"/"`, look for entry for `"one"`
 - This entry gives the disk block number for the inode for `"one"`
 - Read the inode for `"one"` into memory
 - The inode says where first data block is on disk
 - Read that block into memory to access the data in the file

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

18

File Buffer Cache

- Applications exhibit significant locality for reading and writing files
- Idea: Cache file blocks in memory to capture locality
 - ♦ This is called the **file buffer cache**
 - ♦ Cache is system wide, used and shared by all processes
 - ♦ Reading from the cache makes a disk perform like memory
 - ♦ Even a 4 MB cache can be very effective
- Issues
 - ♦ The file buffer cache competes with VM (tradeoff here)
 - ♦ Like VM, it has limited size
 - ♦ Need replacement algorithms again (LRU usually used)

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

19

Caching Writes

- On a write, some applications assume that data makes it through the buffer cache and onto the disk
 - ♦ As a result, writes are often slow even with caching
- Several ways to compensate for this
 - ♦ "write-behind"
 - » Maintain a queue of uncommitted blocks
 - » Periodically flush the queue to disk
 - » Unreliable
 - ♦ Battery backed-up RAM (NVRAM)
 - » As with write-behind, but maintain queue in NVRAM
 - » Expensive
 - ♦ Log-structured file system
 - » Always write next block after last block written
 - » Complicated

November 15, 2000

CSE 120 -- Lecture 13 -- File Systems

20

Read Ahead

- Many file systems implement “read ahead”
 - FS predicts that the process will request next block
 - FS goes ahead and requests it from the disk
 - This can happen while the process is computing on previous block – overlap I/O with execution
 - When the process requests block, it will be in cache
 - Compliments the disk cache, which also is doing read ahead
- For sequentially accessed files, can be a big win
 - Unless blocks for the file are scattered across the disk
 - File systems try to prevent that, though (during allocation)

Summary

- Files
 - Operations, access methods
- Directories
 - Operations, using directories to do path searches
- Sharing
- Protection
 - ACLs vs. capabilities
- File System Layouts
 - Unix inodes
- File Buffer Cache
 - Strategies for handling writes
- Read Ahead

Next time...

- Read Sections 17.11.1, 17.8.3, 14.4.3