Chapter 9: File-System Interface



Operating System Concepts essentials – 8th Edition

Silberschatz, Galvin and Gagne ©2011



Chapter 9: File-System Interface

- File Concept
- Access Methods
- Disk and Directory Structure
- File-System Mounting
- File Sharing
- Protection







- To explain the function of file systems
- To describe the interfaces to file systems
- To discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures
- To explore file-system protection





File Concept

- Uniform logical view of information storage (no matter the medium)
- OS abstracts from physical properties into a logical storage unit, the file
- Files mapped onto physical devices, usually nonvolatile
- File is a collection of related information
 - Smallest allotment of nameable storage
- Contiguous logical address space
- Types:
 - Data
 - numeric
 - character
 - binary
 - Program
 - May be free form or rigidly formed (structured)





File Structure

- None sequence of words, bytes
- Simple record structure
 - Lines
 - Fixed length
 - Variable length
- Complex Structures
 - Formatted document
 - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides:
 - Operating system
 - Program / programmer





File Attributes

- **Name** only information kept in human-readable form
- Identifier unique tag (number) identifies file within file system
- **Type** needed for systems that support different types
- Location pointer to file location on device
- **Size** current file size
- Protection controls who can do reading, writing, executing
- **Time, date, and user identification** data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk
 - Typically file's name and identifier
 - Identifier locates other file attributes
- Attributes may be > 1KB
- Directory structures may be > 1MB



Operating System Concepts Essentials – 8th Edition



File Operations

- File is an abstract data type
- Operations include the following (and usually more)
- **Create** find space, add entry to directory
- Write write data at current file position pointer location and update pointer
- **Read** read file contents at pointer location, update pointer
- **Reposition within file (seek)** change pointer location
- **Delete** free space and remove entry from directory
- **Truncate** delete data starting at pointer





Open Files

- Open(F_i) allow process to access a file
 - Returns a file handle for system call reference to the file
 - Search the directory structure on disk for entry F_i , and move the content or cache some of entry to memory
- Close(file handle) end processes' access to the file
 - Move the content of entry F_i in memory to directory structure on disk





Open File Data Structures

- Usually a global table containing process-independent open file information
 - Size
 - Access dates
 - Disk location of the file: cache of data access information
 - File-open count: counter of number of times a file is open
 - To allow removal of data from open-file table when last processes closes it
- Per-process open file table contains pertinent info, plus pointer to entry in global open file table
 - Current file position pointer: pointer to next read/write location
 - Access rights: per-process access mode information
 - read, write, append





Open File Locking

- Provided by some operating systems and file systems
- Mediates access to a file
 - shared
 - exclusive
- Mandatory or advisory:
 - Mandatory access is denied depending on locks held and requested
 - Advisory processes can find status of locks and decide what to do





File Locking Example – Java API

```
import java.io.*;
import java.nio.channels.*;
public class LockingExample {
    public static final boolean EXCLUSIVE = false;
    public static final boolean SHARED = true;
    public static void main(String arsg[]) throws IOException {
        FileLock sharedLock = null;
        FileLock exclusiveLock = null;
        try {
                RandomAccessFile raf = new RandomAccessFile("file.txt", "rw");
                // get the channel for the file
                FileChannel ch = raf.getChannel();
                // this locks the first half of the file - exclusive
                exclusiveLock = ch.lock(0, raf.length()/2, EXCLUSIVE);
                /** Now modify the data . . . */
                // release the lock
                exclusiveLock.release();
```





File Locking Example – Java API (Cont.)

```
// this locks the second half of the file - shared
       sharedLock = ch.lock(raf.length()/2+1, raf.length(),
       SHARED);
       /** Now read the data . . . */
       // release the lock
       sharedLock.release();
} catch (java.io.IOException ioe) {
       System.err.println(ioe);
}finally {
       if (exclusiveLock != null)
       exclusiveLock.release();
       if (sharedLock != null)
       sharedLock.release();
```





File Types

- Most operating systems recognize file types
 - Filename *extension*
 - I.e. resume.doc, server.java, readerthread.c
- Most support them
 - Automatically open a type of file via a specific application (.doc)
 - Only execute files of a given extension (.exe, .com)
 - Run files of a given type via a scripting language (.bat)
- Can get more advanced
 - If source code modified since executable compiled, if attempt made to execute, recompile and then execute (TOPS-20)
 - Mac OS encodes creating program's name in file attributes
 - Double clicking on file passes the file name to appropriate application
 - Unix has magic number stored in file at first byte indicating file type





File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine- language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes com- pressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information



Operating System Concepts Essentials – 8th Edition

Silberschatz, Galvin and Gagne ©2011



File Structure

- Types can indicate internal file structure
 - Some Oses enforce, some use as hints, some ignore
- But some most conform to OS-required format
 - I.e. executable file
 - Some support more formats
 - DEC VMS supported 3
 - The more that are supported, the more kernel code, etc
 - Some enforce access methods
 - Others allow arbitrary access
 - Unix supports directory files, executable files
 - But all files are strings of bytes
 - Can open a directory file via a text editor
- Files stored in fixed-size disk blocks
 - Can have internal fragmentation



Operating System Concepts Essentials – 8th Edition



Access Methods

Sequential Access – tape model of a file read next write next reset no read after last write (rewrite) **Direct Access** – random access, relative access read n write *n* position to *n* read next write next rewrite *n*

n = relative block number

- Can accommodate structured data in file by mapping record number to block number
- Oses usually support both kinds, sometimes require access method declaration during create()



Sequential-access File







sequential access	implementation for direct access	
reset	cp=0;	
read next	<i>read cp</i> ; <i>cp</i> = <i>cp</i> + 1;	
write next	write cp ; cp = cp + 1;	



Example of Index and Relative Files





Operating System Concepts Essentials – 8th Edition



Disk Structure

- Disk can be subdivided into partitions
- Disks or partitions can be RAID protected against failure
- Disk or partition can be used raw without a file system, or formatted with a file system
- Partitions also known as minidisks, slices
- Entity containing file system known as a volume
- Each volume containing file system also tracks that file system's info in device directory or volume table of contents or directory)
 - Records information for all files on the volume
- As well as general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer











File System Types

- Operating systems have multiple file system types
 - One or more general-purpose (for storing user files)
 - One or more special-purpose, i.e.
 - tmpfs—"temporary" file system in volatile main memory, contents erased if the system reboots or crashes
 - objfs—a "virtual" file system (essentially an interface to the kernel that looks like a file system) that gives debuggers access to kernel symbols
 - ctfs— a virtual file system that maintains "contract" information to manage which processes start when the system boots and must continue to run during operation
 - lofs—a "loop back" file system that allows one file system to be accessed in place of another one
 - **procfs**—a virtual file system that presents information on all processes as a file system





Directory Overview

- Directory similar to symbol table translating file names to their directory entries
 - Can be organized in many ways
- Organization needs to support operations including:
 - Search for a file or multiple files
 - Create a file
 - Delete a file
 - List a directory
 - Rename a file
 - Traverse the file system





Directory Organization

- Should have the features
 - Efficiency locating a file quickly
 - Naming convenient to users
 - Two users can have same name for different files
 - The same file can have several different names
 - Grouping logical grouping of files by properties, (e.g., all Java programs, all games, ...) or arbitrarily





Single-Level Directory

A single directory for all users



Naming problem

Grouping problem



Operating System Concepts Essentials – 8th Edition



Two-Level Directory

Separate directory for each user



- Path name
- Can have the same file name for different users
- Efficient searching
- No grouping capability





Added Directory Concepts

- Many variations, but some components essential
- Idea of current directory default location for activities
- Now need a **path** specification
 - If file is in current directory, just name it
 - If in another directory, must specify by more detailed name
 - Also need way to specify different filesystems
 - MS-DOS gives letter to each volume, "\" separates directory name from file name C:\userb\test
 - VMS uses letter for volume and "[]" for directory specification u:[sst.jdeck]login.com;1
 - Note the support for versions via the trailing number
 - Unix treats volume name as part of directory name /u/pbg/test
- Many Oses search a set of paths for command names
 - "Is" might search in current directory then in system directories





Tree-Structured Directories





Tree-Structured Directories (Cont.)

- Most common
- For example, allows users to can create directories within their directory
- Directory can then contain files or other directories
- Directory can be another file with defined formatting and attribute indicating its type
- Separate system calls to manage directory actions
- Absolute path is full specification of file local /foo/bar/baz
- Relative path is location relative to current directory ../baz
- Efficient searching
 - Search path
- Grouping Capability
- Current directory (working directory)
 - cd /spell/mail/prog
 - type list





Tree-Structured Directories (Cont)

- Creating a new file is done in current directory
- Delete a file

rm <file-name>

Creating a new subdirectory is done in current directory
 mkdir <dir-name>

Example: if in current directory /mail mkdir count



Deleting "mail" \Rightarrow deleting the entire subtree rooted by "mail"?

- Make users manually delete contents (and subcontents) first (MS-DOS)
- Provide an option to delete all contents (Unix)





Acyclic-Graph Directories

Have shared subdirectories and files





Acyclic-Graph Directories (Cont.)

- Adds ability to directly share directories between users
 - But can now have multiple absolute paths to the same file
- Two different names (aliasing)
- If *dict* deletes $list \Rightarrow$ dangling pointer Solutions:
 - Backpointers, so we can delete all pointers Variable size records a problem
 - Entry-hold-count solution
- New directory entry type
 - Link another name (pointer) to an existing file
 - Indirect pointer
 - Delete link separate from the files
 - Hard and symbolic
 - **Resolve the link** follow pointer to locate the file





General Graph Directory



Operating System Concepts Essentials – 8th Edition

Silberschatz, Galvin and Gagne ©2011



General Graph Directory (Cont.)

- How do we guarantee no cycles?
 - Allow only links to file not subdirectories
 - Garbage collection
 - Every time a new link is added use a cycle detection algorithm to determine whether it is OK
 - Or just bypass links during directory traversal





File System Mounting

- A file system must be mounted before it can be accessed
 - Privileged operation
 - First check for valid file system on volume
 - Kernel data structure to track mount points
- Some systems have separate designation for mount point (i.e. "c:")
- Others integrate mounted file systems into existing directory naming system
 - In separate space (i.e. /volumes) or within current name space
- A unmounted file system on /device/dsk (i.e., Fig. 11-11(b)) is mounted at a mount point
- What if the mount point already has contents?
- Configuration file or data structure to track default mounts
 - Used at reboot or to reset mounts
- What if files are open on a device that is being **unmounted**?



Operating System Concepts Essentials – 8th Edition

Silberschatz, Galvin and Gagne ©2011

(a) Existing (b) Unmounted Partition





(b)



Operating System Concepts Essentials – 8th Edition



Mount Point





Operating System Concepts Essentials – 8th Edition



File Sharing

- Sharing of files on multi-user systems is desirable
- Sharing may be done through a **protection** scheme
- On distributed systems, files may be shared across a network
- Network File System (NFS) is a common distributed file-sharing method





File Sharing – Multiple Users

- **User IDs** identify users, allowing permissions and protections to be per-user
- **Group IDs** allow users to be in groups, permitting group access rights





File Sharing – Remote File Systems

- Uses networking to allow file system access between systems
 - Manually via programs like FTP
 - Automatically, seamlessly using distributed file systems
 - Semi automatically via the world wide web
 - Using FTP under the covers
- **Client-server** model allows clients to mount remote file systems from servers
 - Server can serve multiple clients
 - Client and user-on-client identification is insecure or complicated
 - **NFS** is standard UNIX client-server file sharing protocol
 - **CIFS** is standard Windows protocol
 - Standard operating system file calls are translated into remote calls
- Distributed Information Systems (distributed naming services) such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing
 - LDAP / Active Directory becoming industry standard -> Secure Single Sign-on
 - IP addresses can be **spoofed**
 - Protect remote access via firewalls
- Open file request to remote server first checked for client-to-server permissions, then user-id checked for access permissions, then file handle returned
 - Client process then uses file handle as it would for a local file

Silberschatz, Galvin and Gagne ©2011



File Sharing – Failure Modes

- Remote file systems add new failure modes, due to network failure, server failure
 - Data or metadata loss or corruption
- Recovery from failure can involve state information about status of each remote request
- Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security
 - But stateless protocols can lack features, so NFS V4 and CIFS are both state-ful



File Sharing – Consistency Semantics

- Consistency semantics specify how multiple users are to access a shared file simultaneously
 - Similar to Ch 7 process synchronization algorithms
 - ▶ Tend to be less complex due to disk I/O and network latency (for remote file systems)
 - Andrew File System (AFS) implemented complex remote file sharing semantics
 - Unix file system (UFS) implements:
 - Writes to an open file visible immediately to other users of the same open file
 - Sharing file pointer to allow multiple users to read and write concurrently
 - AFS has session semantics
 - Writes only visible to sessions starting after the file is closed
 - Easier to implement is immutable shared files
 - Once file is declared "shared", can't be renamed or modified





Protection

- File owner/creator should be able to manage **controlled access**:
 - What can be done
 - By whom
 - But never forget physical security
- Types of access
 - Read
 - Write
 - Execute
 - Append
 - Delete
 - List
 - Others can include renaming, copying, editing, etc
 - System calls then check for valid rights before allowing operations
 - Another reason for open()
 - Many solutions proposed and implemented





Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users

			RWX
a) owner access	7	\Rightarrow	111
,			RWX
b) group access	6	\Rightarrow	110
			RWX
c) public access	1	\Rightarrow	001

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.







Access Control

- More generally solved via access control lists
 - For a given entity, keep list of user-ids allowed to access and what access methods
 - Constructing such as list can be tedious and unrewarding
 - Data structure must be stored somewhere
 - Variable size





Windows XP Access-Control List Management

tex Properties		? ×
General Security Summary		
Group or user names:		
Administrators (PBG-LAPTOP\A	dministrators)	
Guest (PBG-LAPTOP\Guest)		
SYSTEM		
🕵 Users (PBG-LAPTOP\Users)		
	Add	Bemove
		- Remove
Permissions for Guest	Allow	Deny
Full Control		
Modify		
Read & Execute		
Read		
Write		
Special Permissions		
 For special permissions or for advan click Advanced. 	cea seπings,	Advanced
OK	Cancel	Apply

Operating System Concepts Essentials – 8th Edition

Silberschatz, Galvin and Gagne ©2011



A Sample UNIX Directory Listing

-rw-rw-r-drwx----drwxrwxr-x drwxrwx----rw-r--r---rwxr-xr-x drwx--x--x drwx----drwxrwxrwx

staff 1 pbg 5 pbg staff 2 pbg staff 2 pbg student 1 pbg staff 1 pbg staff 4 pbg faculty 3 pbg staff 3 pbg staff

Sep 3 08:30 Jul 8 09.33 Jul 8 09:35 Aug 3 14:13 Feb 24 2003 Feb 24 2003 Jul 31 10:31 Aug 29 06:52 Jul 8 09:35

intro.ps private/ doc/ student-proj/ program.c program lib/ mail/



test/

End of Chapter 9



Operating System Concepts essentials – 8th Edition

Silberschatz, Galvin and Gagne ©2011