CMPSCI 377 Operating Systems **Fall 2005**

Lecture 16: November 15

Lecturer: Emery Berger *Scribe: Dennis Gove, Rob Silva*

Today: File Systems

- Who are they?
- What are they?
- Where do they come from?

16.1 File Systems and I/O

16.1.1 Overview

Files: another OS-provided abstraction over hardware resources

- applications operate on files in a file system
  - device-independent interface
  - open(), close(), link(), read(), write(), rename()
- device level interface
  - manage disk in terms of sectors
  - OS converts calls to hardware calls

USER EXPECTATIONS ON DATA

- Persistence - data lives over crashes, etc.
- Speed - quick data access
- Size - lots of data
- Sharing / Protection - share and restrict access
- Ease of use - find, examine, and modify data

HARDWARE / OS support for data

Hardware

- Persistence - disks (non-volatile memory)
- Speed - random access devices
- Size - disk capacity grows fast

OS

- Persistence - redundancy, fault tolerance
- Share / Protect - UNIX privileges
- Ease of use - names associated with data (files), hierarchical directories, transparent mapping of devices

FILES

Files are a named collection of related information recorded on secondary storage. This can include source code, binary code, relational databases, etc. They can be structured or unstructured.

structured example: IBM mainframe OS–series of records

unstructured example: UNIX file–stream of bytes

Files have attributes: name, type, location, size, protection, creation time, modified, accessed, etc...

16.1.2 User Interface to File System: Data Operations

- Open file table - shared by all
  - open count, file attributes, location of file on disk, pointers to locations of files in memory
- Per process file table - one for each process
  - pointers to entry in open file table, current position in file (offset), mode in which process accesses file (r, w, r/w), pointers to file buffers

FILE OPERATIONS: creating a file

create(name)

- allocate disk space (check disk quotas, permission, etc.)
- create file descriptor for file (name, location on disk, attributes) (adds file descriptor to directory that contains file)

may mark file with "type" attribute (especially Mac)

- advantages: error detection, launch appropriate application

- disadvantages: not supported everywhere, complicated

FILE OPERATIONS: deleting a file

unlink(fileDescription)

- find directory containing file
- free disk blocks used by file
- remove file descriptor from directory

*fileDescriptor is a pointer to some object (file name, location, attribute)

FILE OPERATIONS: open files

open(name, mode)

- check if file open by another process
- if not: find file, copy file into system-wide open file table

FILE OPERATIONS: close files

close(fd)

- remove entry for file in process's file table
- decrement open count in system-wide file table

FILE OPERATIONS: others

-reading files (random access vs. sequential access)

-writing to files (point to where you want to write, copy from buffers to file)

-seek

-memory mapping files

16.1.3 File Access Methods

common file access patterns from programmer's perspective

- sequential: data processed in order
  - most programs use this method
  - EX: compiler reading source file
- keyed:
  - address block based on key table

16.1.4 Naming of Directories

need method of retrieving files from disk (OS uses numbers, but we like names)

- Flat File Systems
  - one level directory
    * one namespace for entire disk, every name unique
    * directory contains (name, index) pairs
  - two level directory
    * separate directory for each user
- Hierarchical File Systems
  - tree-structured name space
  - $d\underbrace{---}_{user}\underbrace{---}_{group}\underbrace{---}_{other}$
  - used by all modern OS
  - directory becomes special file on disk

Referential Naming

Hard Links (UNIX: ln command) – allow mult links to single file
example: "ln A B"
-init $A \rightarrow file\sharp 100$
-after $A, B \rightarrow file\sharp 100$

Soft Links (UNIX: ln -s command) – symbolic pointer from one file to another
example: "ln -s A B"
-init $A \rightarrow file\sharp 100$
-after $A \rightarrow file\sharp 100, B \rightarrow A$

Protection

- OS must allow users to control access to files
- grant / deny access to file operations depending on protection info

access lists / groups (Windows)

- for each file with a user name and access type

access control bits (UNIX)

- 3 categories of users (owner, group, world)