CS 134:
Operating Systems
File System Design Choices

2012-12-06

CS34

CS 134:
Operating Systems
File System Design Choices

Overview

Allocation

Overall Organization

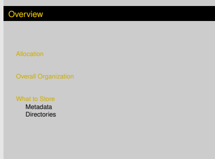
What to Store

- Metadata
- Directories

2012-12-06

CS34

└─Overview



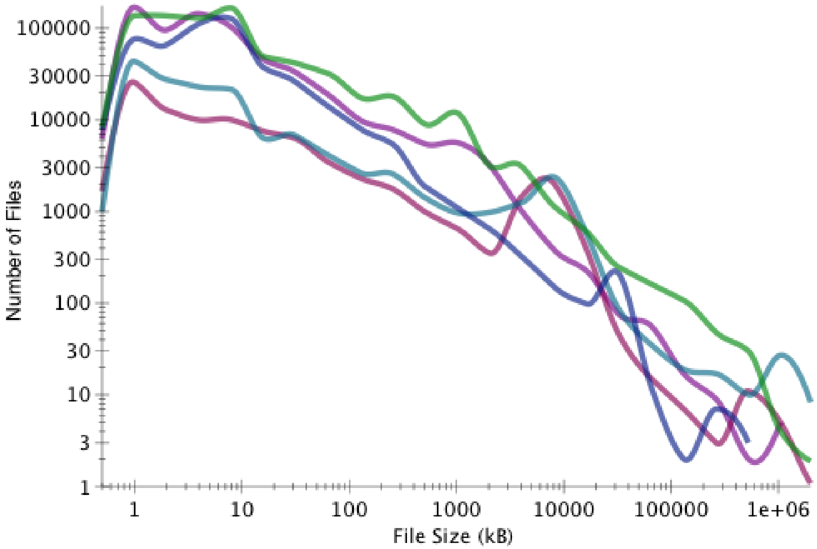
Overview

Allocation

Overall Organization

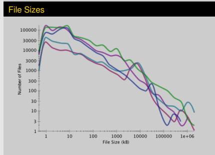
What to Store
Metadata
Directories

File Sizes



2012-12-06

CS34
└Allocation
└File Sizes



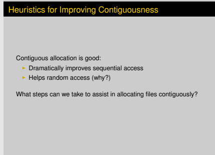
These are file-size distributions on several machines. Note similarities and differences. Also note it's a log-log plot!

Heuristics for Improving Contiguousness

Contiguous allocation is good:

- Dramatically improves sequential access
- Helps random access (why?)

What steps can we take to assist in allocating files contiguously?



Heuristics for Improving Contiguousness

Contiguous allocation is good:

- Dramatically improves sequential access
- Helps random access (why?)

What steps can we take to assist in allocating files contiguously?

Region-Based Approaches

Divide disk into regions (sometimes called *cylinder groups*), each with own free list

- Unless a file is very large, try to keep all of it in same region
- Try to put all the files in a directory in same region
- Put different directories in different regions

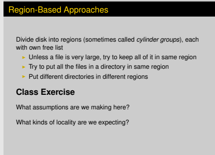
Class Exercise

What assumptions are we making here?

What kinds of locality are we expecting?

2012-12-06

CS34
└Allocation
└Region-Based Approaches



Region-Based Approaches

Divide disk into regions (sometimes called *cylinder groups*), each with own free list

- Unless a file is very large, try to keep all of it in same region
- Try to put all the files in a directory in same region
- Put different directories in different regions

Class Exercise

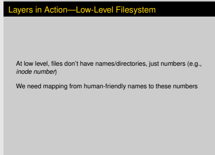
What assumptions are we making here?

What kinds of locality are we expecting?

Layers in Action—Low-Level Filesystem

At low level, files don’t have names/directories, just numbers (e.g., *inode number*)

We need mapping from human-friendly names to these numbers



Layers in Action—Low-Level Filesystem

At low level, files don’t have names/directories, just numbers (e.g., *inode number*)

We need mapping from human-friendly names to these numbers

Layers in Action—High-level Filesystem

Build on lower-level layer

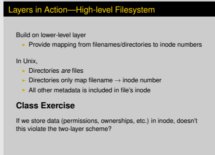
- Provide mapping from filenames/directories to inode numbers

In Unix,

- Directories *are* files
- Directories only map filename → inode number
- All other metadata is included in file's inode

Class Exercise

If we store data (permissions, ownerships, etc.) in inode, doesn't this violate the two-layer scheme?



Layers in Action—High-level Filesystem

Build on lower-level layer

- Provide mapping from filenames/directories to inode numbers

In Unix,

- Directories *are* files
- Directories only map filename → inode number
- All other metadata is included in file's inode

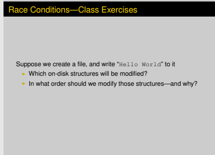
Class Exercise

If we store data (permissions, ownerships, etc.) in inode, doesn't this violate the two-layer scheme?

Race Conditions—Class Exercises

Suppose we create a file, and write “`Hello World`” to it

- Which on-disk structures will be modified?
- In what order should we modify those structures—and why?



Race Conditions—Class Exercises

Suppose we create a file, and write “`Hello World`” to it

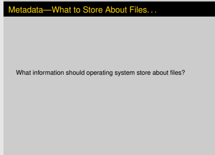
- Which on-disk structures will be modified?
- In what order should we modify those structures—and why?

Metadata—What to Store About Files. . .

What information should operating system store about files?

2012-12-06

CS34
└What to Store
  └Metadata
    └Metadata—What to Store About Files. . .



Metadata—What to Store About Files. . .

What information should operating system store about files?

Creator—Who Made the File?

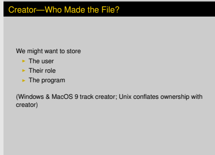
We might want to store

- The user
- Their role
- The program

(Windows & MacOS 9 track creator; Unix conflates ownership with creator)

2012-12-06

CS34
└What to Store
  └Metadata
    └Creator—Who Made the File?



Creator—Who Made the File?

We might want to store

- The user
- Their role
- The program

(Windows & MacOS 9 track creator; Unix conflates ownership with creator)

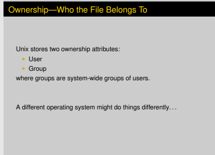
Ownership—Who the File Belongs To

Unix stores two ownership attributes:

- User
- Group

where groups are system-wide groups of users.

A different operating system might do things differently. . .



Ownership—Who the File Belongs To

Unix stores two ownership attributes:

- User
- Group

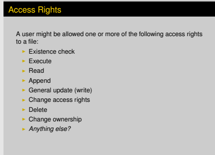
where groups are system-wide groups of users.

A different operating system might do things differently. . .

Access Rights

A user might be allowed one or more of the following access rights to a file:

- Existence check
- Execute
- Read
- Append
- General update (write)
- Change access rights
- Delete
- Change ownership
- *Anything else?*



Access Rights

A user might be allowed one or more of the following access rights to a file:

- Existence check
- Execute
- Read
- Append
- General update (write)
- Change access rights
- Delete
- Change ownership
- *Anything else?*

Windows NT provides “Take ownership.” Why do they do that?

Access—Who Can Access the File

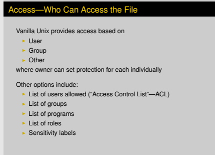
Vanilla Unix provides access based on

- User
- Group
- Other

where owner can set protection for each individually

Other options include:

- List of users allowed (“Access Control List”—ACL)
- List of groups
- List of programs
- List of roles
- Sensitivity labels



Access—Who Can Access the File

Vanilla Unix provides access based on

- User
- Group
- Other

where owner can set protection for each individually

Other options include:

- List of users allowed (“Access Control List”—ACL)
- List of groups
- List of programs
- List of roles
- Sensitivity labels

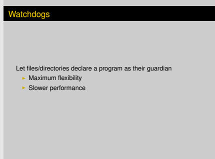
Watchdogs

Let files/directories declare a program as their guardian

- Maximum flexibility
- Slower performance

2012-12-06

CS34
└What to Store
 └Metadata
  └Watchdogs



Watchdogs

Let files/directories declare a program as their guardian

- Maximum flexibility
- Slower performance

Access Information

When the file was

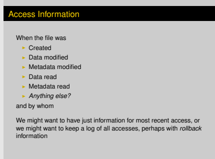
- Created
- Data modified
- Metadata modified
- Data read
- Metadata read
- *Anything else?*

and by whom

We might want to have just information for most recent access, or we might want to keep a log of all accesses, perhaps with *rollback* information

2012-12-06

CS34
└What to Store
 └Metadata
  └Access Information



Access Information

When the file was

- Created
- Data modified
- Metadata modified
- Data read
- Metadata read
- *Anything else?*

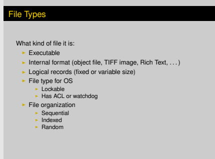
and by whom

We might want to have just information for most recent access, or we might want to keep a log of all accesses, perhaps with *rollback* information

File Types

What kind of file it is:

- Executable
- Internal format (object file, TIFF image, Rich Text, . . .)
- Logical records (fixed or variable size)
- File type for OS
  - Lockable
  - Has ACL or watchdog
- File organization
  - Sequential
  - Indexed
  - Random



File Types

What kind of file it is:

- Executable
- Internal format (object file, TIFF image, Rich Text, . . .)
- Logical records (fixed or variable size)
- File type for OS
  - Lockable
  - Has ACL or watchdog
- File organization
  - Sequential
  - Indexed
  - Random

See next slide for discussion of the Unix philosophy.

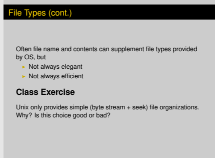
File Types (cont.)

Often file name and contents can supplement file types provided by OS, but

- Not always elegant
- Not always efficient

Class Exercise

Unix only provides simple (byte stream + seek) file organizations. Why? Is this choice good or bad?



File Types (cont.)

Often file name and contents can supplement file types provided by OS, but

- Not always elegant
- Not always efficient

Class Exercise

Unix only provides simple (byte stream + seek) file organizations. Why? Is this choice good or bad?

In the past, operating systems provided many different file types, and many different file organizations. But,

- Inflexible
- Complicated the operating system

Unix stores minimal file-type information. This follows the “worse is better” philosophy. It also has the serendipitous effect of allowing unexpected usages (e.g., `grep` through binaries or even a raw disk, or `dd` on a plain file).

Other…

Various other information

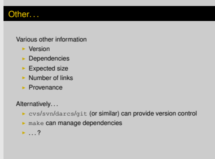
- Version
- Dependencies
- Expected size
- Number of links
- Provenance

Alternatively…

- `cvs/svn/darcs/git` (or similar) can provide version control
- `make` can manage dependencies
- …?

2012-12-06

CS34
└What to Store
  └Metadata
    └Other…



Other…

Various other information

- Version
- Dependencies
- Expected size
- Number of links
- Provenance

Alternatively…

- `cvs/svn/darcs/git` (or similar) can provide version control
- `make` can manage dependencies
- …?

Directories

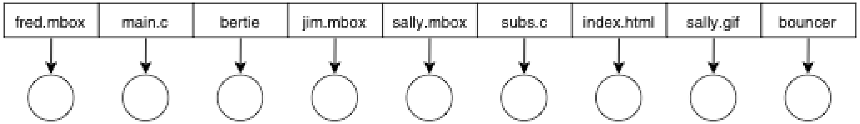
Why have ’em?

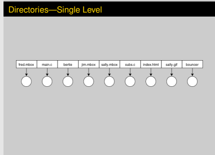
Directories

Why have ’em?

- Convenience for users:
  - Names allow user control, rather than machine control, of file identifiers
  - Logical grouping of files
- More efficient
- Many-to-one mapping (one file, many names)

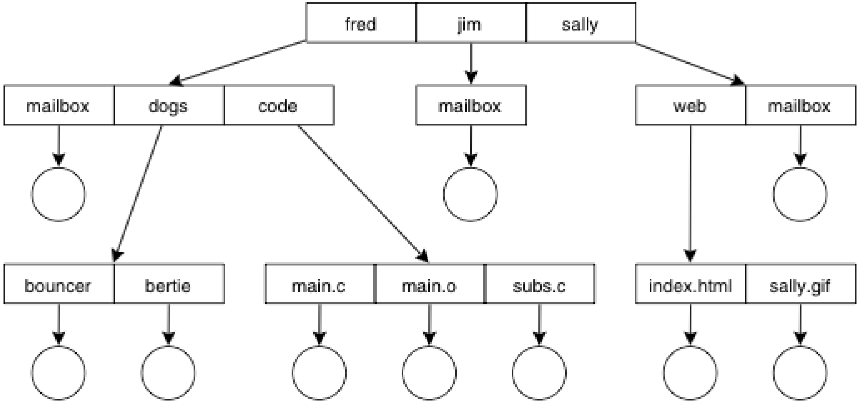
Directories—Single Level





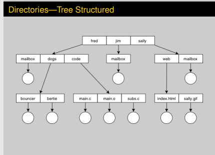
- Non-hierarchical
- Simple
- Inflexible:
  - Naming problems
  - Grouping problems
- Inefficient search

Directories—Tree Structured

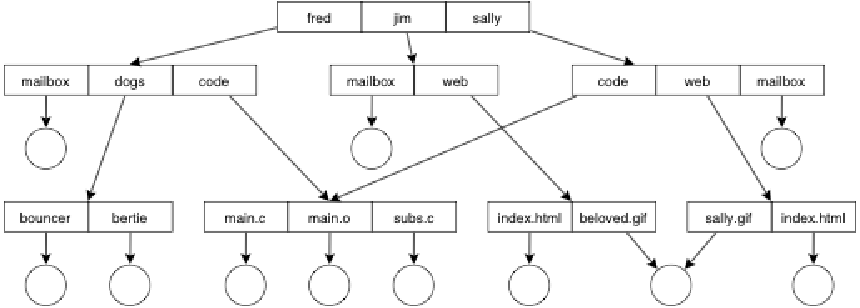


2012-12-06

CS34
└What to Store
└Directories
└Directories—Tree Structured

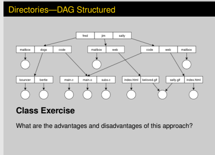

Directories—Tree Structured

Directories—DAG Structured



Class Exercise

What are the advantages and disadvantages of this approach?


Directories—DAG Structured
Class Exercise
What are the advantages and disadvantages of this approach?

Advantages:

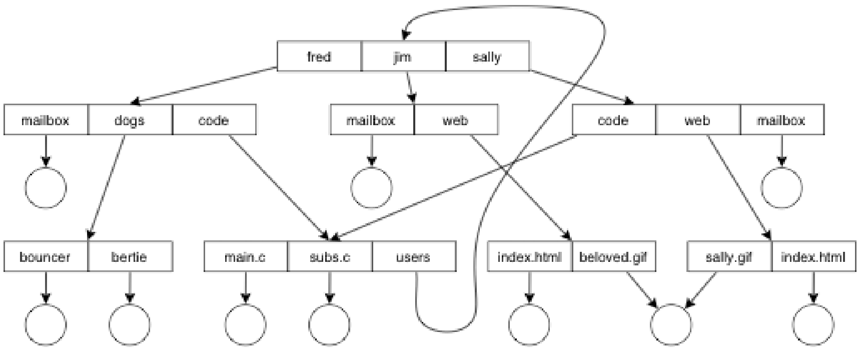
- Lets user set up convenient paths to things
- Eases sharing
- Why not?

Disadvantages:

- What does “. .” mean?
- Users can manage to confuse themselves

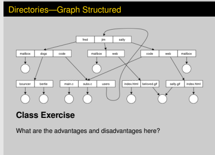
Does Unix allow this? Did original Unix?

Directories—Graph Structured



Class Exercise

What are the advantages and disadvantages here?


Directories—Graph Structured
Class Exercise
What are the advantages and disadvantages here?

Advantages:

- Complete generality and flexibility

Disadvantages:

- Users can confuse themselves
- . . becomes almost meaningless
- Possiblity of disconnected subgraphs (if reference counting is used) or accidental wiping out of complete subtrees (if proper garbage collection)