File Systems Interface

Today

- Files and access methods
- Directory structures
- Sharing and protection

Next

- File system implementation

Files and file systems

Most computer applications need to:

- Store large amounts of data (larger than their address space)
- that must survive process termination and
- can be access concurrently by multiple processes

→ Usual answer: Files – form user's perspective, the smallest allotment of logical secondary storage

File system – part of the OS dealing with files

- Supports the file abstraction of storage
- Naming – how do users select files?
- Protection – users are not all equal
- Reliability – information must be safe for long periods of time
- Storage mgmt. – efficient use of storage and fast access to files

File attributes

- Names – different for each OS
  - Upper and/or lower case
- Type, when supported
- Location (in a device) and size
- A few other useful attributes

Protection	Who can access the file & in what way
Creator	ID of creator
System flag	0 for normal files; 1 for system ones
Creation time	Date & time of creation
Time of last access	Date & time of last access
Current size	In bytes

File operations …

- File is an ADT (Abstract Data Type) – what operations?
  - Create, delete, write, read
  - Reposition within file – file seek
  - Truncate
  - Other operations can be built on this basic set (e.g. cp)

- Most operation involve searching the directory for file
  - Instead, use open first
  - open (F_i) - search directory for entry F_i, move content to memory (open-file table)
  - close () – remove entry from open file table

File operations

- Open/Close in multiuser systems
  - Per-process and system-wide tables
    - Entry in the per-process table points to system-wide table
  - System-wide table keeps process-independent information (e.g. file size)
  - Open counts to see if entry is needed

- File locks – restricting access to a file
  - Shared (read) and exclusive (write) locks
  - Mandatory (OS enforced) and advisory locks (cooperative model, UNIX)
  - Like with any other lock – be careful w/ deadlocks
  - Lock files
    - Used to indicate that a given resource is locked (e.g. if the resource to lock is *not* a file)
    - Content is normally irrelevant, commonly the PID of the lock holder

File types

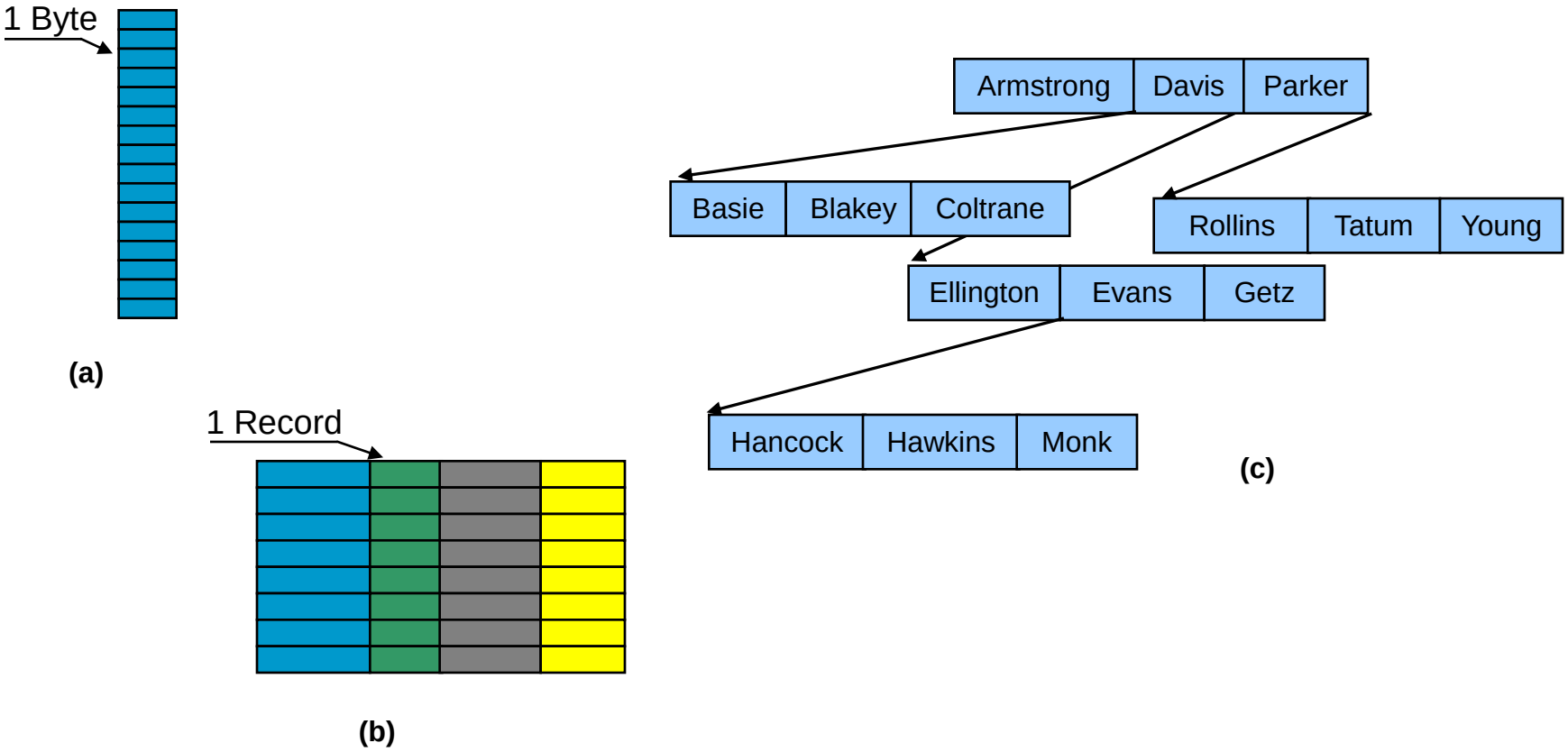
- Different OSs support different file types
  - Regular, binary, directories, …
  - Character special (model terminals [/dev/tty], printers, etc) and block special files (model disks [/dev/hd1])
  - Extensions as hints & the use of magic numbers
    - Some typical file extensions

file.gif	Graphical Interchange Format Image
file.mpg	Movie encoded with MPEG standard
file.o	Object file
file.txt	General text file

  - Pros and cons of strongly typed files

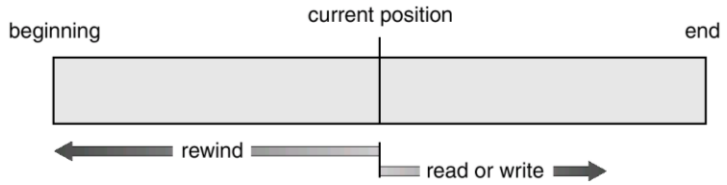
File structures

- Several file structures, three common ways
  - Byte sequence - Unix & Windows; user imposes meaning (a)
  - Record sequence – think about 80-column punch cards (b)
  - Tree – records have keys, tree is sorted by it (d)



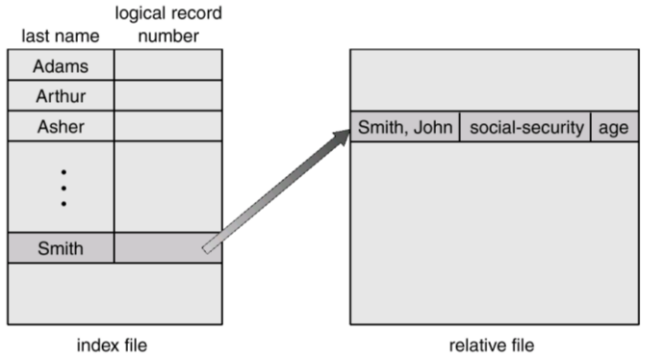
File access methods

- Sequential Access – tape model
  - Simplest and most common
  - read next/write next



- Random/direct access – disk model
  - Two approaches
    - Read *n*/write *n*,
    - Position to *n* and read next/write next
  - Retain sequential access – read/write + update last position

- Other access methods
  - On top of direct access
  - Normally using indexing
  - Multi-level indexing for big files
    - E.g. IBM ISAM (Indexed Sequential Access Method)

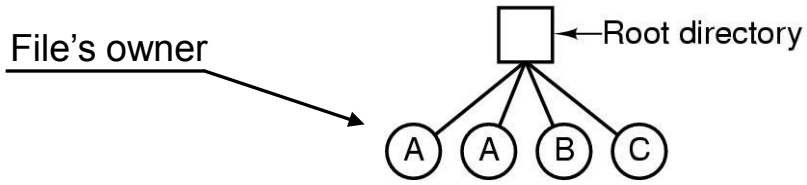


Directory structure

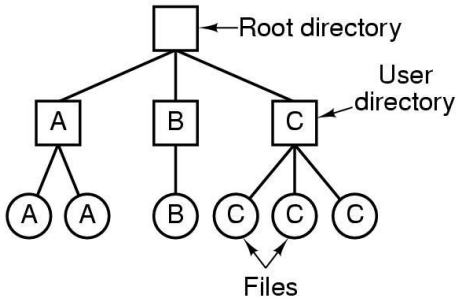
- To manage volume of info.: partitions & directories
- Directory: set of nodes with information about all files
  - Name, type, address, current & max. length, date last accessed
- Operations on directories
  - Open/close directories, create/delete/rename files from a directory, readdir, link/unlink, traverse the file system
- Directory organizations - goals
  - Efficiency – locating a file quickly.
  - Naming – convenient to users.
  - Grouping – logical grouping of files by properties (e.g. all Java progs., all games, …)

Single and two-level directory systems

- A single level directory system
  - Early PCs, early supercomputers (CDC 6600), embedded systems?
  - Pros and cons
    - Fast file searches
    - Name clashing
  - Contains 4 files owned by 3 != people

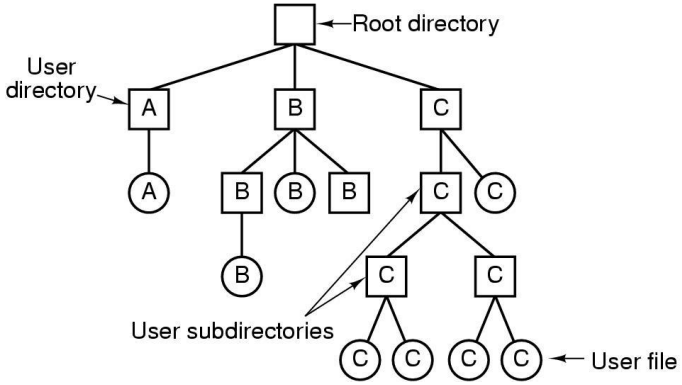


- Two-level directory system
  - Avoid name conflicts bet/ users
  - You may need a system's directory
  - Problems if you have too many files

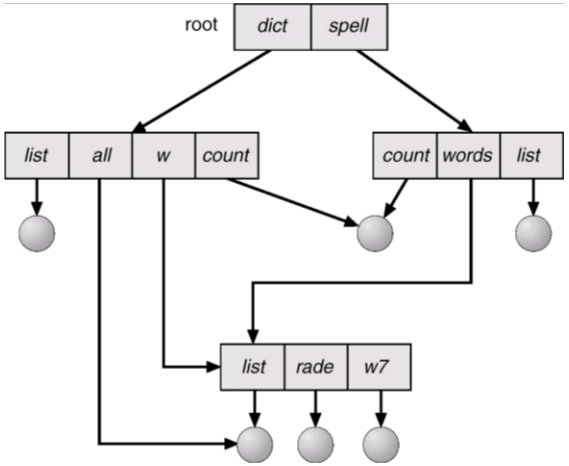


Hierarchical & general directory systems

- Hierarchical
  - Avoid name clashing for users (MULTICS)
  - Powerful structuring tool for organization (decentralization)



- Acyclic graphs – sharing
  - Two different names (aliasing)
  - If *dict* del. *list* → dangling pointer
    - Backpointers & counter
  - Unix links – pointers to files
    - Soft & hard links – (in)direct pointer



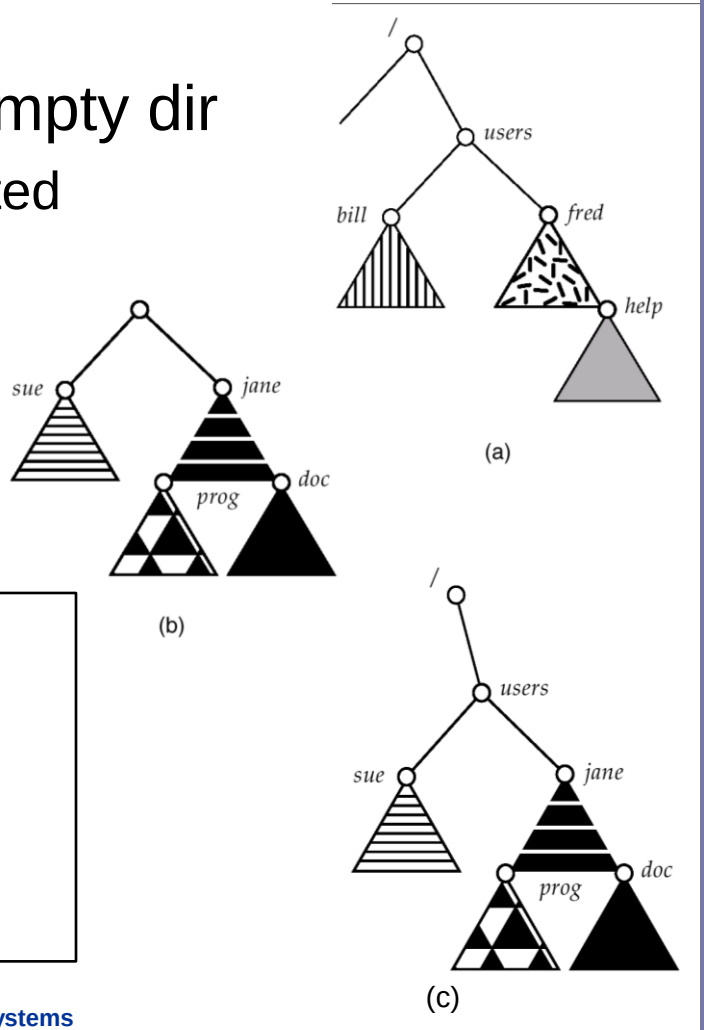
- Path names
  - Absolute & relative path names
  - "." & ".."

File system mounting

- A FS must be mounted to be available
  - What do you do if you have more than one disk? Put a self contained FS on each (*C:…)* or…
- Typically, a mount point is an empty dir
  - Existing file system (a) & unmounted partition (b)
  - After it was mounted (c)

```
# mount /dev/sda1 /users
```

- `fstab` file in Unix




```
(10:41am) ~ % more /etc/fstab
# This file is edited by fstab-sync - see 'man fstab-sync' for details
LABEL=/           /           ext3    defaults        1 1
none              /dev/pts    devpts  gid=5,mode=620  0 0
/dev/sdb1         /export     ext3    defaults        1 2
none              /proc       proc    defaults        0 0
none              /dev/shm    tmpfs   defaults        0 0
LABEL=/usr        /usr        ext3    defaults        1 2
…
```

Protection …

- File owner/creator should be able to control
  - what can be done & by whom
- Types of access
  - Read, Write, Execute, Append, Delete, List, …
- A general & common approach – access control list (ACL)
  - Per resources – user names & types of access allowed
  - Long!
- Unix: short version access lists & groups
  - Access modes: read, write, execute
  - Classes of users: owner, group, public
  - 3 bits per for each access mode
  - Mask provides a default (mine '022' - octal)
  - File created with 777 and mask 022 → 755

Rights	Code
rwx	7 (111)
rw-	6 (110)
r-x	5 (101)
r--	4 (100)
-wx	3 (011)
-w-	2 (010)
--x	1 (001)
---	0 (000)

Protection

- Combining both approaches - Solaris (2.6+) access lists - setfacl & getfacl

```
% getfacl -a exam.tex
 # file: exam.tex
 # owner: fabianb
 # group: other
 user::rw-
 group::r--     #effective:r--
 mask:r--
 other:r
```

Intersection of specified permissions and mask field.

```
% setfacl -r -m u:sbirrer:rw-  exam.tex
% getfacl -a exam.tex
 # file: exam.tex
 # owner: fabianb
 # group: other
 user::rw-
 user:sbirrer:rw-      #effective:rw-
 group::r--            #effective:r--
 mask:rw-
 other:r--
```

- Problems with this?
- Other schemes: passwords per file/directory, …(TOPS-20, IBM VM/CMS, …)

Next Time

- Details on file system implementations and some examples …