CSSE2310/7231 — C.1

File Systems

Terms

- File — Bytes recorded in secondary storage
  - For this lecture, not talking about other uses of the filedescriptor interface
  - eg not, network socket, pipe, keyboard, ...
- Disk — Spinning magnetic storage
  - Most discussion also applies to SSD/flash storage (we aren't going down to the storage level).

File System

A file system is a datastructure which manages:

- Contents of files — "data"
- Information about files — "meta-data"
- Free space
  - File systems have a size
  - Files can be added?
  - Files can change in size

Data structures need to exist somewhere:

- on a disk [usually]
- as a file on another system (`.iso` files)?
- Mac `.dmg` files

FS vs OS

How can manipulate files may depend more on the file system than the operating sytem.
But, if

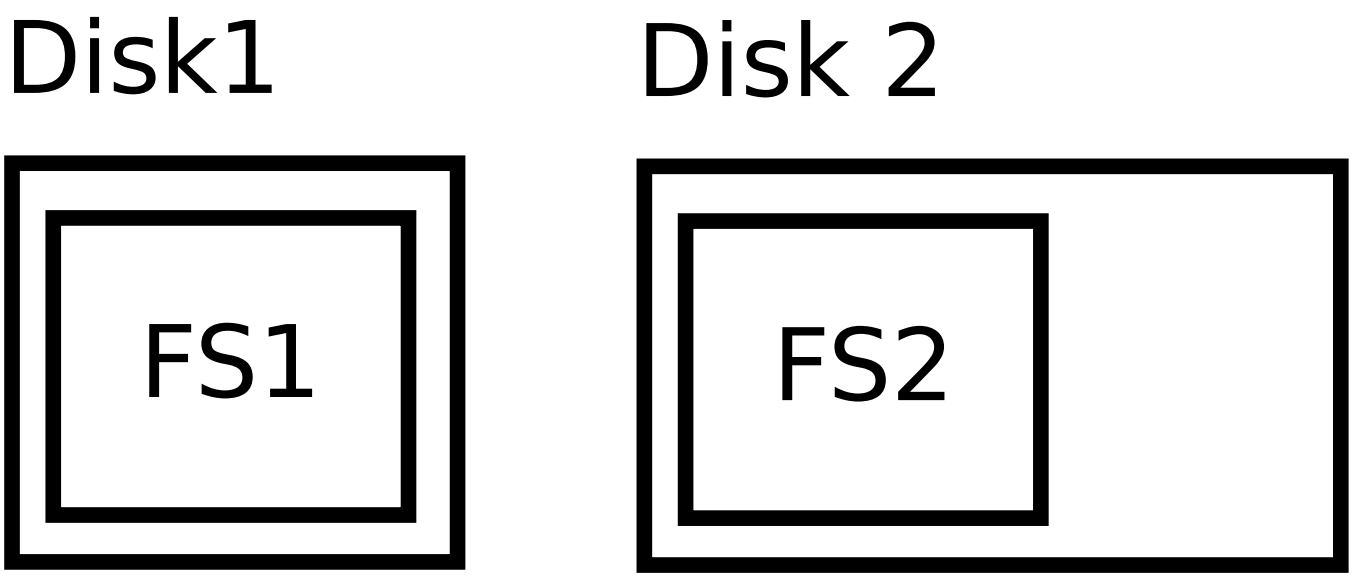
- the OS only provides one main FS
- that FS is hard to use on other OS

then the difference might not be obvious.
Eg:

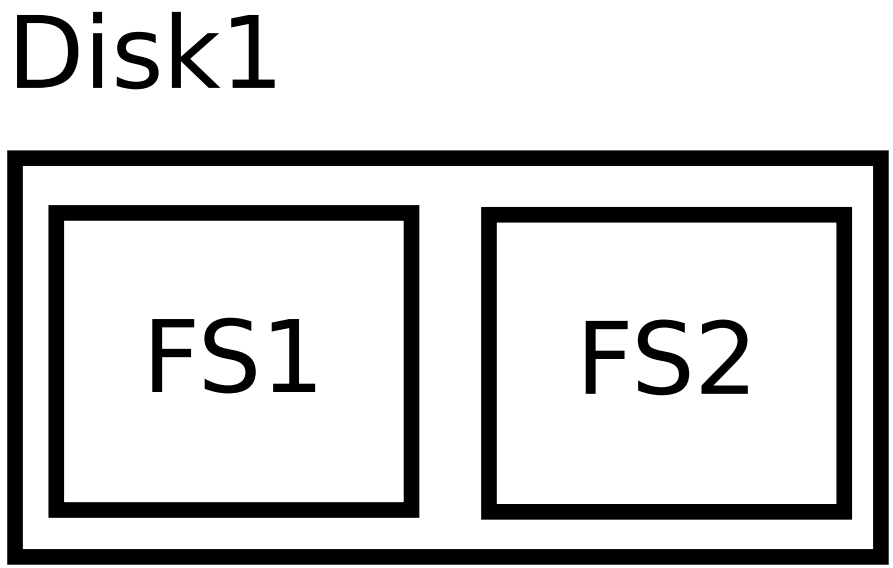
- NTFS — NT File System
- HFS+ — OSX
- ISO9660 — cdrom
- ext2, ext3, ext4, btrfs, xfs, . . . — unix

Files vs disks

Disks and File Systems are not (necessarily) 1-to-1



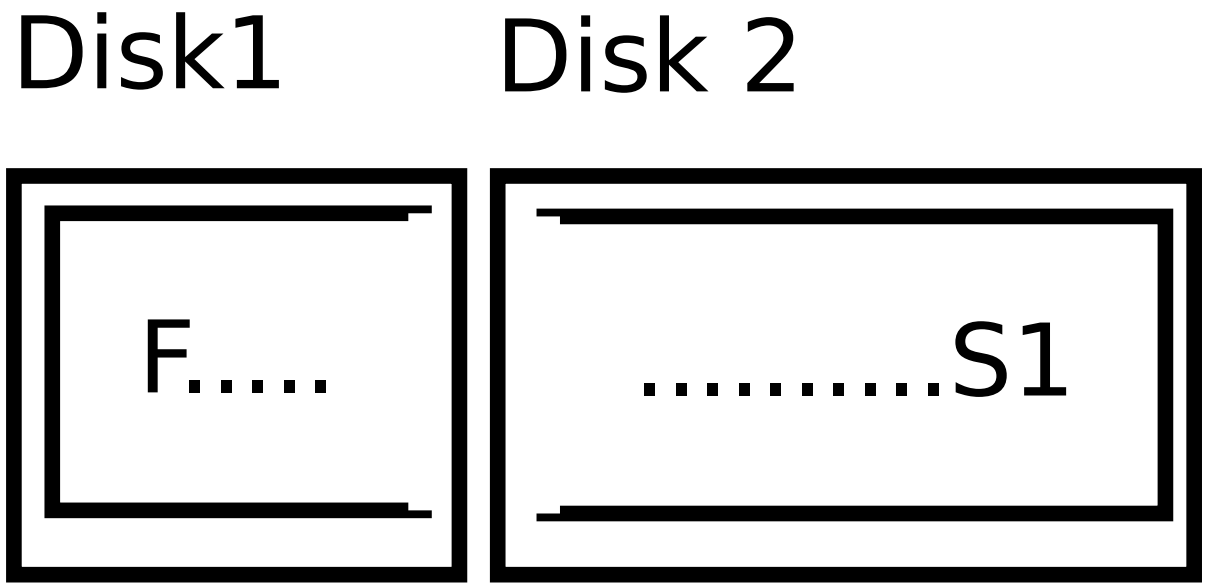
Files vs disks



Why?

- Dual boot
- (formerly) File-system limitations
- Separate data from OS
  - Some OS make this easier than others

Files vs disks



Why?

- Capacity increase
- RAID and other capacity + safety schemes

Files

- Sequence of bytes (start at the beginning and read till the end)?
- Storage for those bytes?

Variances

- Sequence of bytes?
  - "Record" based systems?
- Single sequence?
  - "old" Mac — resource fork vs data fork
  - NTFS — multiple streams
- Store bytes
  - sparse files?
  - "union" file systems

Metadata

Information about data which is not part of it. Not all properties are necessarily stored explicitly.

- Name — does the content change if the name does?
  - Is name unique?
  - Case sensitve?
  - Case preserving?
- Location/Path?
  - A derived property from a sequence of dirs?
  - What meaning can be inferred from structure?
  - Is it unique?

Metadata

- Size
- Type of content?
  - Infer from name?
    - Windows stage 1
    - Some linux GUIs but check contents
  - Encode type with file?
    - Old Mac
  - Guess from contents
    - `file` using `/etc/magic`

Metadata

Permissions?

- By role
  - Unix systems — files have one "owner" and one "group"
  - `rwxr-x---`
  - Owner can...
  - Group can...
  - Everyone else can...
- ACL[1] — by user
  - Windows, Unix also support it as an option
  - Permissions don't change if role does
  - Not as clear[2]

[1] "Access Control List"

[2] My opinion

Permissions reminder

- Change permissions on the commandline with `chmod`
  - u+r : change the owner(user) permissions to add read
  - g+r : change the group permissions to add read
  - o+r : change the other permissions to add read
- `x` permissions on (normal) files
  - Needed to exec
  - For interpretted scripts (eg shell scripts) also need `r`
- Directory permissions:
  - `x` : needed to interact with anything in the directory.
  - `r` : needed to see what is in the directory
  - `x` with no `r`, can access things in the directory if you already know what they are called.
- To follow a path, you need `x` on all directories in the path.

Spinning disks

Why are spinning disks "slow"?
Consider polar coordinates (r, θ).

- Changing r — moving the heads towards (or away from) the centre.
- Changing θ — waiting for disk to rotate (rotational latency)

The more widely scattered operations are, the most cost incurred.
Why spinning disks? Still need them for:

- Lower cost
- Large capacity[3]

[3] Also tape

Fragmentation

Files

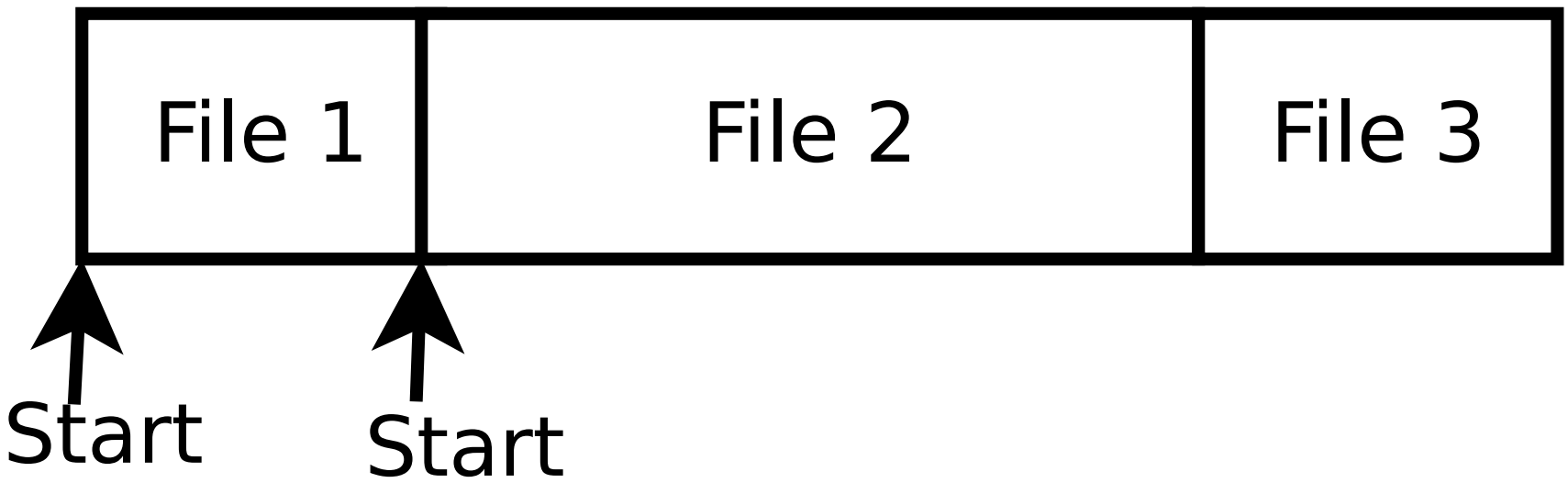
- Reading through file leads to jumping around the device

Space

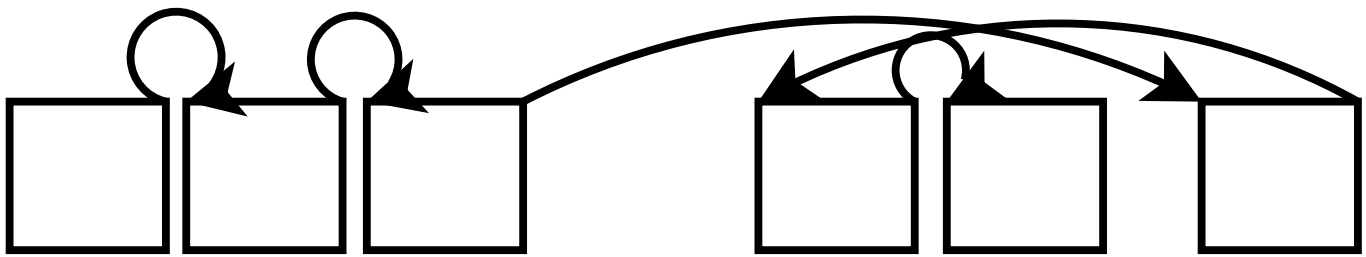
- External fragmentation
  - Free space is spread out over the device
  - Could lead to fragmented files
- Internal fragmentation
  - Unused space inside allocated blocks

Storage structures

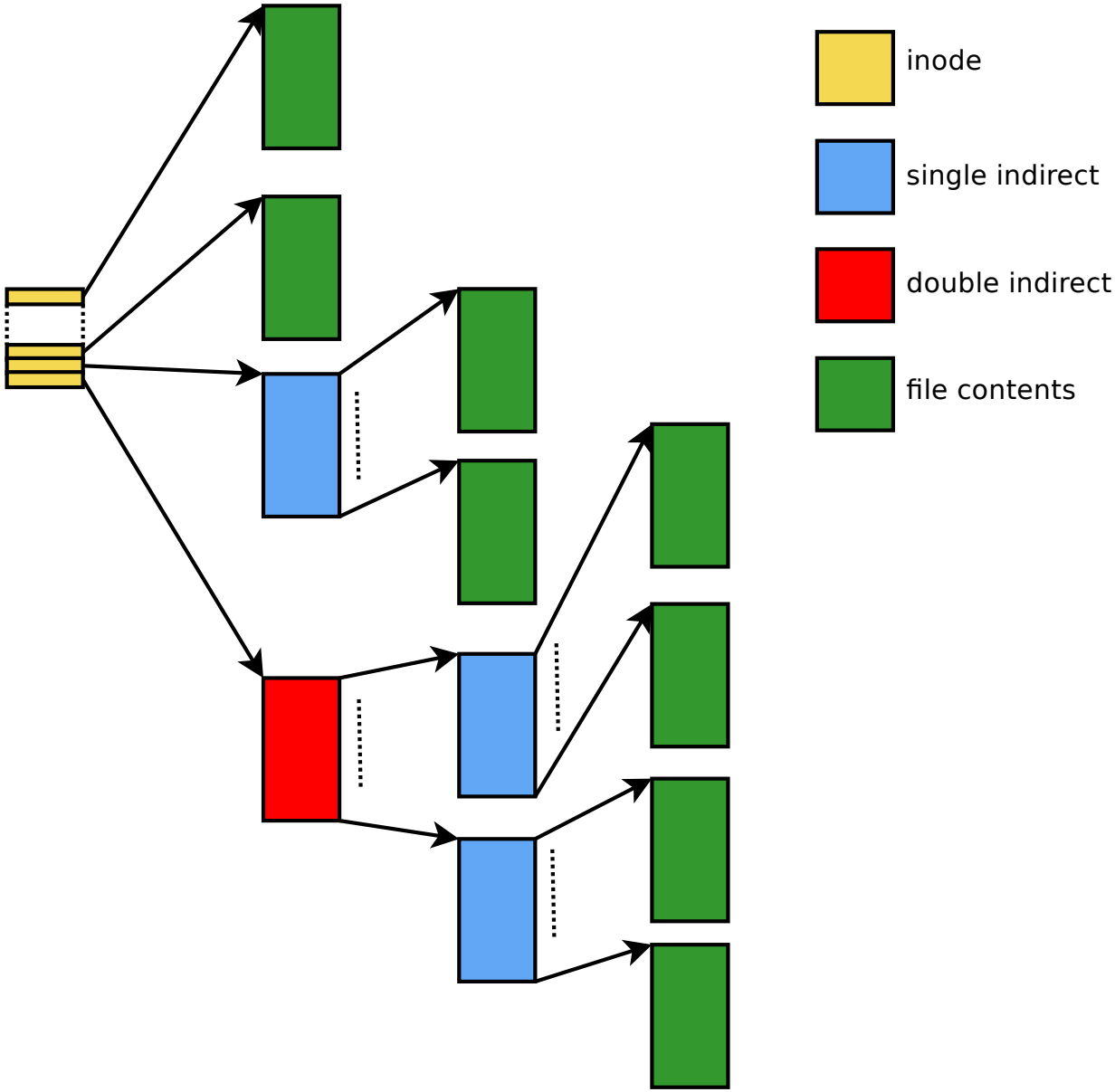
(We're dealing with all of these in abstract.)
Array-like



Linked storage



Indexed storage



Trees

"Typically" the directories on a file system form a tree.

- Avoiding cycles means recursive traversals will eventually terminate.
- Removing a subdirectory shouldn't remove the directory you are in because something is its own grandparent.

While directories are stored as files, operations on them are restricted to system calls rather than allowing arbitrary writes as with files.
Directories don't actually contain files.

Hard Links

Consider a file `A.c`:

```
ln A.c B.c
```

Adds `B.c` to the directory.

```
diff A.c B.c
```

Shows no difference.
But, after modifying `A.c`, diff still shows no difference.

ln

- In the filesystems we use, directory entries are (hard links) ie a `name:i-number` mapping.
- All the explicit properties of a file (apart from a name) are stored in an "i-node" (see indexed file earlier).
- The i-number lets the system find the i-node:
  - i-nodes could be in a table
  - or i-number could indicate where on disk the node is
- The internals of inodes can vary with FS
  - For our purposes we'll assume a structure like that shown ealier.

ln

- `ls -i` will show the i-numbers for each directory entry.
- `ls -l` will show how many links there are to a file.
- All hard links are equal, as long as there is at least one link to a file, it will remain on disk.
  - The system call to get rid of a directory entry is `unlink`[4]
  - Files will be kept with a link count of zero if a process has them open.
- Hard links can't cross into other filesystems.

[4] As opposed to something like delete

Directories?

The link count for a directory is 2 plus the number of direct subdirectories it has.
Eg: `/tmp/bob` will have one link in `/tmp`

For the other, `ls -a /tmp/bob` shows directories:

- `.` ← second link
- `..`

If we `mkdir /tmp/bob/sub`, then there will be another link added from

- `/tmp/bob/..`

Hardlinking directories is not allowed (or restricted).

Symbolic Links

- Sym-links point from one name to another (as opposed to name to file contents).
- You've seen these used in the testing framework:
  - `tests/` is a symbolic link to *`joel`*`/public/20XX/ptests`
- create with `ln -s target newname`

Symbolic links

- Can cross file systems (uses paths not low level ids)
- Have '`l`' in the type field
- Permission of '`rwxrwxrwx`'
  - Which is a "lie"
  - actual permissions are those of the target
- A symlink will not prevent a target being deleted
- If target moves or is deleted, the symlink won't work
- Symlinks can target directories.

Mounting

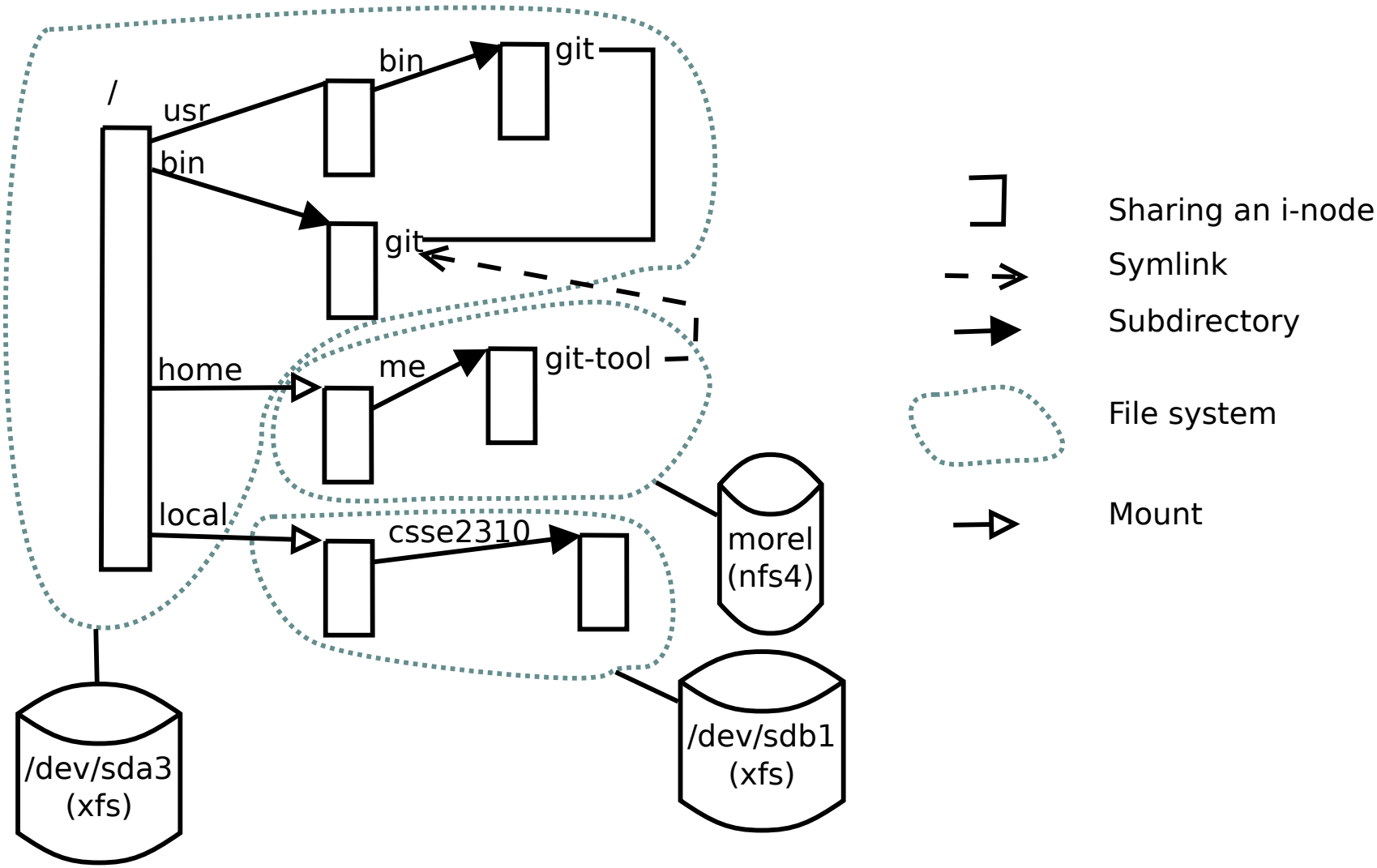
- To allow the system to interact with the contents of a file system, the FS must be "mounted".
- Normally, file systems should be "unmounted" before being removed.
  - Unmount, eject, "safely remove"
  - Why?
    - FS may have auditing info or background tasks.
    - Buffering may mean changes haven't been written yet.
    - `fflush` means it is out of your process
    - `sync` "should" mean written to disk[5]
- Unix `mount` command will list mounted file systems.

[5] See doco for FS

Mount points

- Windows
  - Forms a forest of "trees"
  - `A:\`, `C:\` ...
  - UNC paths
- Unix
  - All the directories of all mounted FS form a unified tree
  - Can mount into any directory the admin chooses
  - Temporary mounts eg `/media`
- OSX
  - See unix
  - Tends to be under `/Volumes`

Combined example



Summary

```
ls -ali
```

```
1441875 drwxr-xr-x 30 joel grp  4096 Jun 12  2018 .
1441800 drwxr-xr-x 40 joel grp  4096 Oct 15 18:31 ..
1446196 drwxr-xr-x  2 joel grp  4096 Jun 12  2018 bin
1446457 drwxr-xr-x  5 joel grp  4096 Jun 12  2018 build
1446461 -rw-r--r--  1 joel grp 13364 Jun 12  2018 config.log
1446433 -rw-r--r--  1 joel grp   203 Jun 12  2018 CREDITS
1444671 drwxr-xr-x  8 joel grp  4096 Jun 12  2018 cusplibrary
...
A       BC          D   E   F    G   H             I
```

A	i-number	F	File group
B	Type	G	Size
C	Permissions	H	Modification time
D	Link count	I	Name
E	File owner		

Reminder:
The linux tute covers some of this material.

Calculations / Exercises