CS 33

Files Part 3

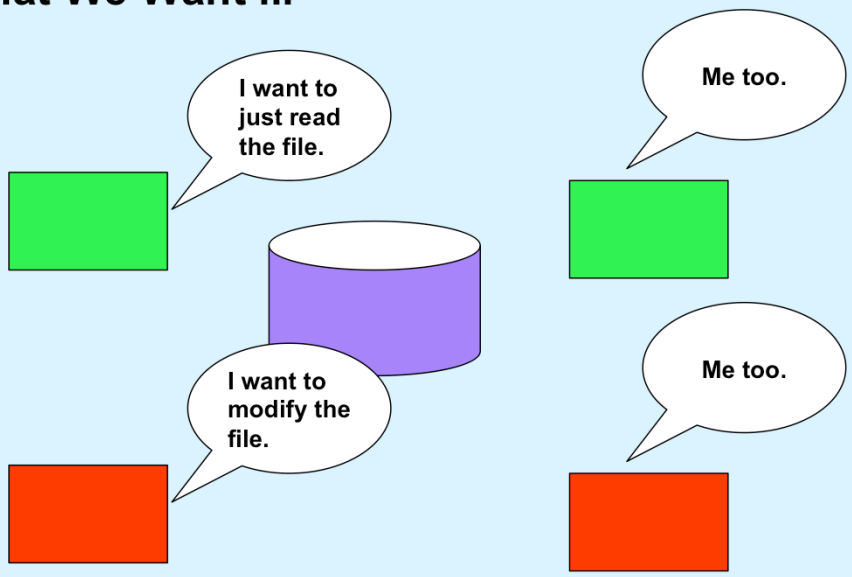
Sharing Files

- You're doing a project with a partner
- You code it as one 15,000-line file
  - the first 7,500 lines are yours
  - the second 7,500 lines are your partner's
- You edit the file, changing 6,000 lines
  - it's now 5am
- Your partner completes her changes at 5:01am
- At 5:02am you look at the file
  - your partner's changes are there
  - yours are not

Lessons

- Never work with a partner
- Use more than one file
- Read up on git
- Use an editor and file system that support file locking

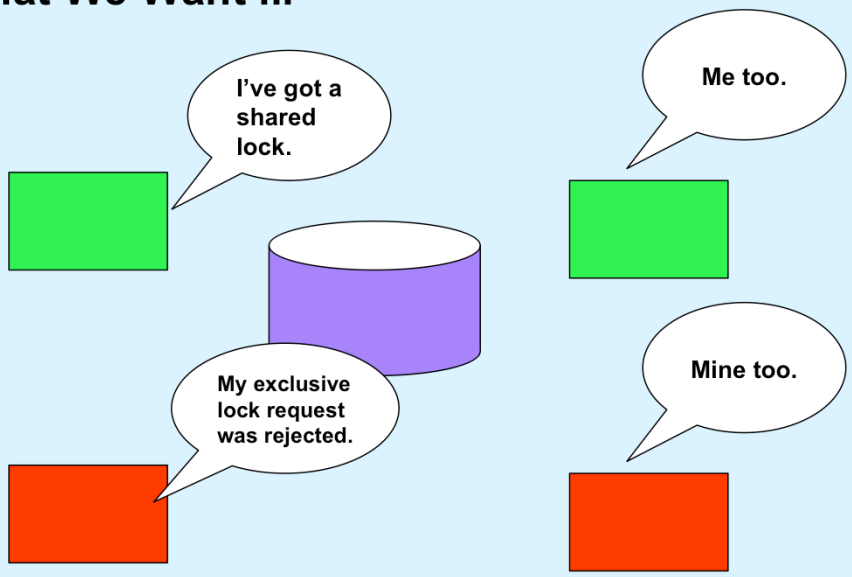
What We Want ...



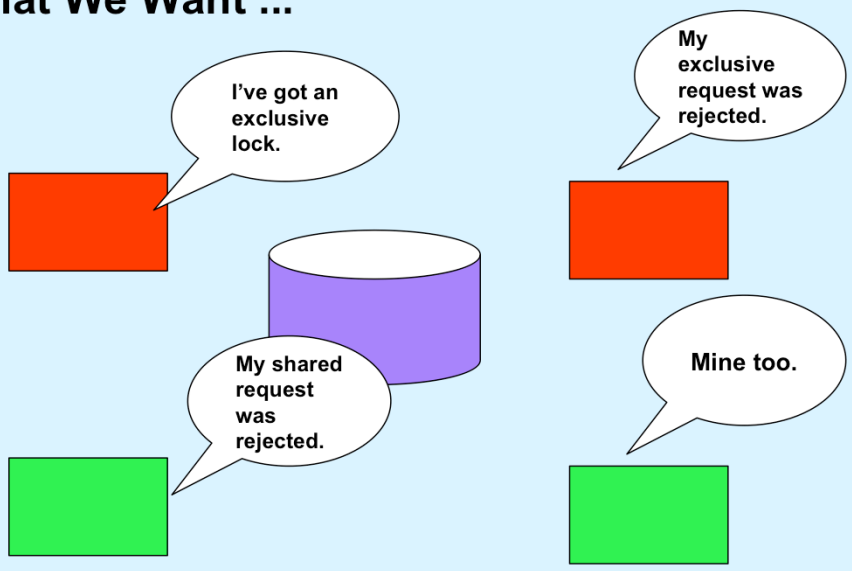
Types of Locks

- Shared (readers) locks
  - any number may have them at same time
  - may not be held when an exclusive lock is held
- Exclusive (writers) locks
  - only one at a time
  - may not be held when a shared lock is held

What We Want ...



What We Want ...



Locking Files

- Early Unix didn't support file locking
- How did people survive?
  - `open("file.lck", O_RDWR|O_CREAT|O_EXCL, 0666);`
    - » operation fails if *file.lck* exists, succeeds (and creates file.lck) otherwise
    - » requires cooperative programs

Locking Files (continued)

- How it's done in "modern" Unix
  - "advisory locks" may be placed on files
  - don't ask: no problem
    - » may request shared (readers) or exclusive (writers) lock
      - *fcntl* system call
    - » either succeeds or fails
    - » *open*, *read*, *write* always work, regardless of locks
    - » a lock applies to a specified range of bytes, not necessarily the whole file
    - » requires cooperative programs

Locking Files (still continued)

- **How to:**

```
struct flock fl;
fl.l_type = F_RDLCK;       // read lock
// fl.l_type = F_WRLCK;    // write lock
// fl.l_type = F_UNLCK;    // unlock
fl.l_whence = SEEK_SET;    // starting where
fl.l_start = 0;            // offset
fl.l_len = 0;              // how much? (0 = whole file)
fd = open("file", O_RDWR);
if (fcntl(fd, F_SETLK, &fl) == -1)
  if ((errno == EACCES) || (errno == EAGAIN))
    // didn't get lock
  else
    // something else is wrong
else
  // got the lock!
```


Alternatively, one may use l_type values of F_RDLCKW and F_WRLCKW to wait until the lock may be obtained, rather than to return an error if it can't be obtained.

Locking Files (yet still continued)

- Making locks mandatory:
  - if the file's permissions have group execute permission off and set-group-ID on, then locking is enforced
    - *read*, *write* fail if file is locked by someone other than the caller
  - however …
    - doesn't work on NFSv3 or earlier
      - (we run NFSv3 at Brown CS)

Quiz 1

- Your program currently has a shared lock on a portion of a file. It would like to "upgrade" the lock to be an exclusive lock. Would there be any problems with adding an option to *fcntl* that would allow the holder of a shared lock to wait until it's possible to upgrade to an exclusive lock, then do the upgrade?

  1) either no problems whatsoever or some easy-to-deal-with problems
  2) at least one major problem

