L6: Client/server in one computer; atomicity
Bounded buffer send

send(bb, m):
    while True:
        if bb.in – bb.out < N:
            bb.buf[bb.in mod N] ← m
            bb.in ← bb.in + 1
    return
send(bb, m):
    while True:
        if bb.in – bb.out < N:
            bb.buf[bb.in mod N] ← m
            bb.in ← bb.in + 1
        return

receive(bb):
    while True:
        if bb.in > bb.out:
            m ← bb.buf[bb.out mod N]
            bb.out ← bb.out + 1
        return m
send(bb, m):
    while True:
        if bb.in – bb.out < N:
            bb.in ← bb.in + 1
            bb.buf[bb.in-1 mod N] ← m
        return

receive(bb):
    while True:
        if bb.in > bb.out:
            m ← bb.buf[bb.out mod N]
            bb.out ← bb.out + 1
        return m
send(bb, m):
    while True:
        if bb.in – bb.out < N:
            bb.buf[bb.in mod N] ← m
            bb.in ← bb.in + 1
        return

receive(bb):
    while True:
        if bb.in > bb.out:
            m ← bb.buf[bb.out mod N]
            bb.out ← bb.out + 1
        return m
Send with locking

send(bb, m):
    acquire(bb.send_lock)
    while True:
        if bb.in – bb.out < N:
            bb.buf[bb.in mod N] ← m
            bb.in ← bb.in + 1
    release(bb.send_lock)
    return
Does this send work?

```
send(bb, m):
    acquire(bb.send_lock)
    while True:
        if bb.in – bb.out < N:
            acquire(bb.send_lock)
            bb.buf[bb.in mod N] ← m
            bb.in ← bb.in + 1
            release(bb.send_lock)
    return
```
File system: no concurrency

move(dir1, dir2, name):
    unlink(dir1, name)
    link(dir2, name)
Coarse-grained locking

move(dir1, dir2, name):
    acquire(fs_lock)
    unlink(dir1, name)
    link(dir2, name)
    release(fs_lock)
Fine-grained locking

move(dir1, dir2, name):
acquire(dir1.lock)
unlink(dir1, name)
release(dir1.lock)

acquire(dir2.lock)
link(dir2, name)
release(dir2.lock)
Fine-grained locking

move(dir1, dir2, name):
    acquire(dir1.lock)
    unlink(dir1, name)
    release(dir1.lock)
    acquire(dir2.lock)
    link(dir2, name)
    release(dir2.lock)
Holding multiple locks

move(dir1, dir2, name):
    acquire(dir1.lock)
    acquire(dir2.lock)
    unlink(dir1, name)
    link(dir2, name)
    release(dir1.lock)
    release(dir2.lock)
Deadlock

move(dir1, dir2, na):
acquire(dir1.lock)
acquire(dir2.lock)
unlink(dir1, na)
link(dir2, na)
release(dir1.lock)
release(dir2.lock)

move(dir2, dir1, nb):
acquire(dir2.lock)
acquire(dir1.lock)
unlink(dir2, nb)
link(dir1, nb)
release(dir2.lock)
release(dir1.lock)
Avoiding deadlock

move(dir1, dir2, name):
    if dir1.inum < dir2.inum:
        acquire(dir1.lock)
        acquire(dir2.lock)
    else:
        acquire(dir2.lock)
        acquire(dir1.lock)
    unlink(dir1, name)
    link(dir2, name)
    release(dir1.lock)
    release(dir2.lock)
Summary

- Client/server in one computer: bounded buffers
- Concurrent programming is tricky!
- Locks help make several actions look atomic
  - Before-or-after atomicity