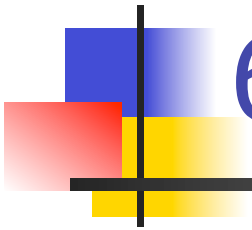

6.033 Spring 2009

Robert Morris

Lecture 7

Threads

Remember: Send

```
send(p, m):
    while true:
        acquire(p.lock)
        if p.in - p.out < N:
            p.buffer[p.in mod N] ← m
            p.in ← p.in + 1
            release(p.lock)
            return
        release(p.lock)
```

Send / Receive with Yield

```
send(p, m):
    while true:
        if something to do: do it
        else: yield()

receive(p):
    while true:
        if something to do: do it
        else: yield()
```

version 1

```
yield():
    acquire(t_lock)
    id = cpus[CPU()].thread
    threads[id].state = RUNNABLE
    threads[id].sp = SP

    do:
        id = (id + 1) mod N
    while threads[id].state != RUNNABLE

    threads[id].state = RUNNING
    SP = threads[id].sp
    cpus[CPU()].thread = id
    release(t_lock)
```

suspend

resume

Send with Yield

```
send(p, m):
   while true:
        acquire(p.lock)
        if p.in - p.out < N:
             p.buffer[p.in mod N] ← m
             p.in ← p.in + 1
             release(p.lock)
             return
        release(p.lock)
        yield()
```

Send with Wait/Notify

```
send(p, m):
   acquire(p.lock)
   while p.in - p.out == N:
      wait(p.notfull, p.lock)
   p.buffer[p.in mod N] ← m
   p.in ← p.in + 1
   notify(p.notempty)
   release(p.lock)
```

```
wait(cvar, lock):
    acquire(t_lock)
    release(lock)
    threads[id].cvar = cvar
    threads[id].state = WAITING
    yield()
    release(t_lock)
    acquire(lock)
```

```
wait(cvar, lock):
    acquire(t_lock)
    release(lock)
    threads[id].cvar = cvar
    threads[id].state = WAITING
    yield()
   release(t_lock)
    acquire(lock)

notify(cvar):
    acquire(t_lock)
    for i = 0 to N-1:
       if threads[i].cvar == cvar and
              threads[i].state == WAITING:
          threads[i].state = RUNNABLE
   release(t_lock)
```

version 1

```
yield():
    acquire(t_lock)
    id = cpus[CPU()].thread
    threads[id].state = RUNNABLE
    threads[id].sp = SP

    do:
        id = (id + 1) mod N
    while threads[id].state != RUNNABLE

    threads[id].state = RUNNING
    SP = threads[id].sp
    cpus[CPU()].thread = id
    release(t_lock)
```

idle loop

version 2

```
yield():
    id = cpus[CPU()].thread
    threads[id].sp = SP
    SP = cpus[CPU()].stack

    do:
        id = (id + 1) mod N
        release(t_lock)
        acquire(t_lock)
    while threads[id].state != RUNNABLE

    threads[id].state = RUNNING
    SP = threads[id].sp
    cpus[CPU()].thread = id
```

idle loop