A Hierarchy of Critical Sections

- We have two different critical sections…
- …that occur at different levels of abstraction
  - the first relies a R/W lock
    - protects access to some database (say)
    - allows multiple readers in the CS
  - the second relies on split binary semaphores
    - protects the shared variables ($nreaders$, $r\_gate.count$, etc) and implements the conditions we use to implement R/W locks
    - allows only one thread at a time in CS

Starvation

- Our R/W implementation can starve writers
- Change the waiting and release conditions:
  - when a reader tries to enter the CS, wait if

    there is a writer in the CS or

    writers at the write gate waiting to enter CS
  - exiting reader prioritizes releasing a waiting writer
  - exiting writer prioritizes releasing a waiting reader

See Chapter 17 in the Harmony book

Conditional Critical Sections

- We know of two ways to implement them:

Busy Waiting	Split Binary Semaphores
Wait for condition in loop, acquiring lock before testing for condition, and releasing it if condition does not hold	Use a collection of binary semaphores and keep track of state, including information about waiting threads
Easy to understand the code	State tracking is complicated
OK-ish for true multi-core, but bad for virtual threads	Good for both multicore and virtual threading

Language support?

- Can the programming language be more helpful here?
    - Helpful syntax
    - Or at least some library support

Enter Monitors

- Collect shared data into an object/module
- Define methods for accessing shared data
- Separate the concerns of mutual exclusion and condition synchronization
- Monitors are comprised of
  - one mutex lock, and
  - zero or more condition variables for managing concurrent access to shared data

Condition Variables

- An abstraction for conditional synchronization associated with a monitor
- Enable threads to wait for a given condition to hold while inside the monitor (after releasing the monitor lock) and be alerted when the condition holds
- A misnomer
  - can neither be read nor set to a value
  - think of a condition variable as a label associated with a condition and a queue
  - threads wait in the queue (inside the monitor) until notified that condition holds

Resource Variables

- Each condition variable should be associated with a resource variable (RV) tracking the state of that resource
    - e.g., the number of reader slots that have been claimed
    - It is your job to maintain the RV!
- Check its RV before calling wait() on a condition variable to ensure the resource is truly unavailable
- Once the resource is available, claim it (subtract the amount you are using!)
- Before notifying you are releasing a resource, indicate it has become available by increasing the corresponding RV

Two Types of Monitors

Hoare Monitors

Mesa Monitors

Tony Hoare

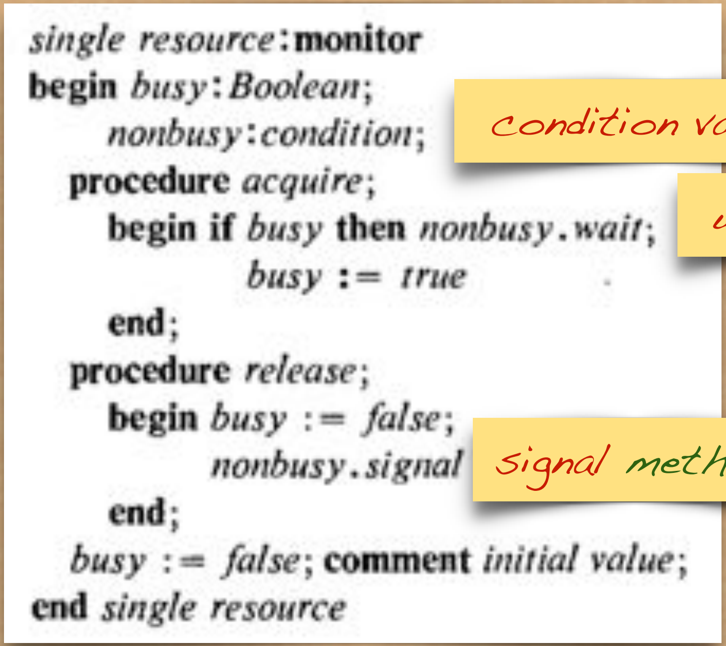
Butler Lampson

Different semantics as to what happens when a thread waiting on a condition is alerted that the condition holds

Hoare Monitors

Tony Hoare, 1974

- Syntactic sugar above split binary semaphores
  - monitor: one thread can execute at a time
  - wait(cond. var.): thread waits for given condition
  - signal(cond. var.): transfer control to a thread waiting for the given condition, if any



```
single resource:monitor
begin busy:Boolean;
    nonbusy:condition;
  procedure acquire;
    begin if busy then nonbusy.wait;
               busy := true
    end;
  procedure release;
    begin busy := false;
          nonbusy.signal
    end;
  busy := false; comment initial value;
end single resource
```

condition variable

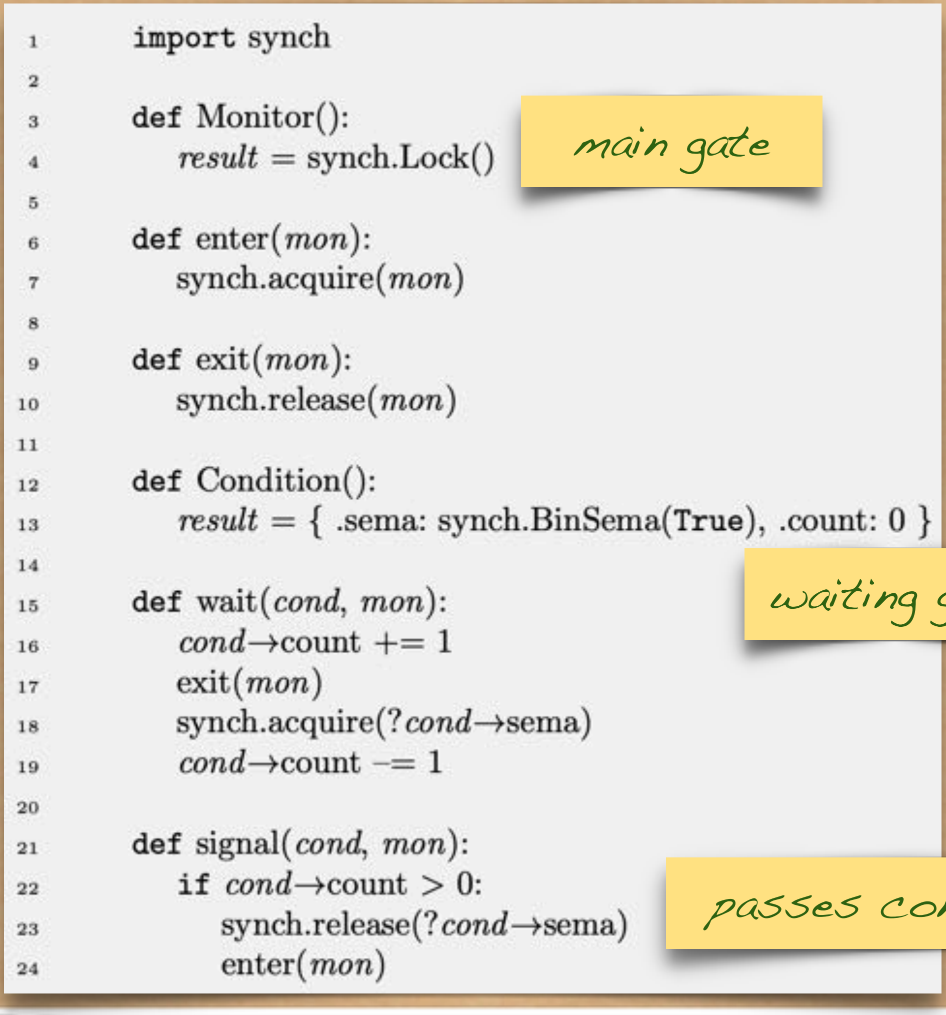
wait method

signal method

Similar construct proposed by Per Brinch Hansen

in 1973

Hoare Monitors in Harmony



```
import synch

def Monitor():
    result = synch.Lock()

def enter(mon):
    synch.acquire(mon)

def exit(mon):
    synch.release(mon)

def Condition():
    result = { .sema: synch.BinSema(True), .count: 0 }

def wait(cond, mon):
    cond→count += 1
    exit(mon)
    synch.acquire(?cond→sema)
    cond→count −= 1

def signal(cond, mon):
    if cond→count > 0:
        synch.release(?cond→sema)
        enter(mon)
```

main gate

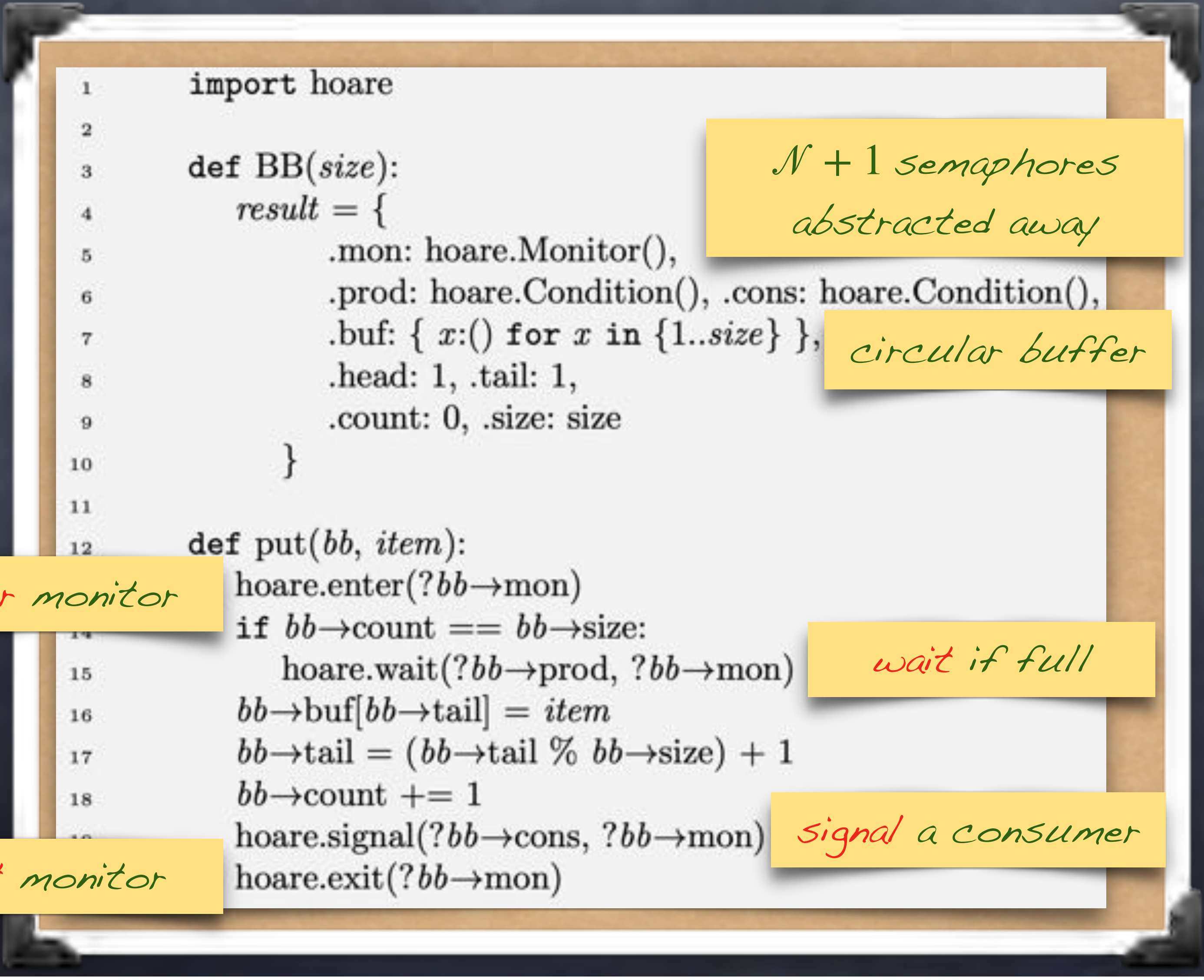
waiting gate

passes control immediately

What happens when a thread signals?

- Hoare semantics:
  - signaling thread is suspended and, atomically, ownership of the lock is passed to one of the waiting threads, whose execution is immediately resumed.
  - signaling thread is resumed if former waiter exits monitor, or if it waits again

Producer/Consumer with Bounded Buffer



```
import hoare

def BB(size):
    result = {
            .mon: hoare.Monitor(),
            .prod: hoare.Condition(), .cons: hoare.Condition(),
            .buf: { x:() for x in {1..size} },
            .head: 1, .tail: 1,
            .count: 0, .size: size
        }

def put(bb, item):
    hoare.enter(?bb→mon)
    if bb→count == bb→size:
        hoare.wait(?bb→prod, ?bb→mon)
    bb→buf[bb→tail] = item
    bb→tail = (bb→tail % bb→size) + 1
    bb→count += 1
    hoare.signal(?bb→cons, ?bb→mon)
    hoare.exit(?bb→mon)
```

$N + 1$ semaphores abstracted away

circular buffer

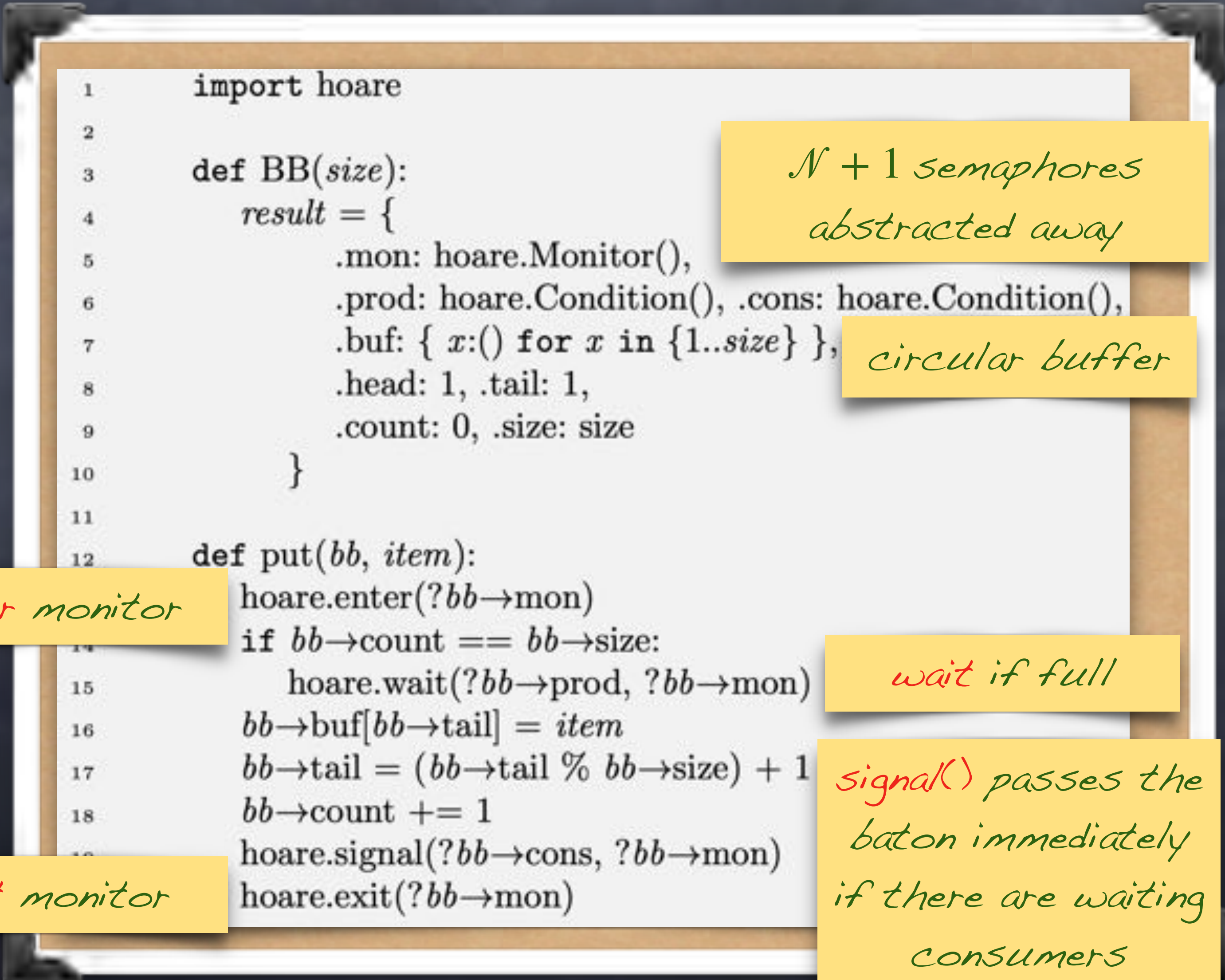
enter monitor

wait if full

signal a consumer

exit monitor

Producer/Consumer with Bounded Buffer



```
import hoare

def BB(size):
    result = {
            .mon: hoare.Monitor(),
            .prod: hoare.Condition(), .cons: hoare.Condition(),
            .buf: { x:() for x in {1..size} },
            .head: 1, .tail: 1,
            .count: 0, .size: size
        }

def put(bb, item):
    hoare.enter(?bb→mon)
    if bb→count == bb→size:
        hoare.wait(?bb→prod, ?bb→mon)
    bb→buf[bb→tail] = item
    bb→tail = (bb→tail % bb→size) + 1
    bb→count += 1
    hoare.signal(?bb→cons, ?bb→mon)
    hoare.exit(?bb→mon)
```

$N + 1$ semaphores abstracted away

circular buffer

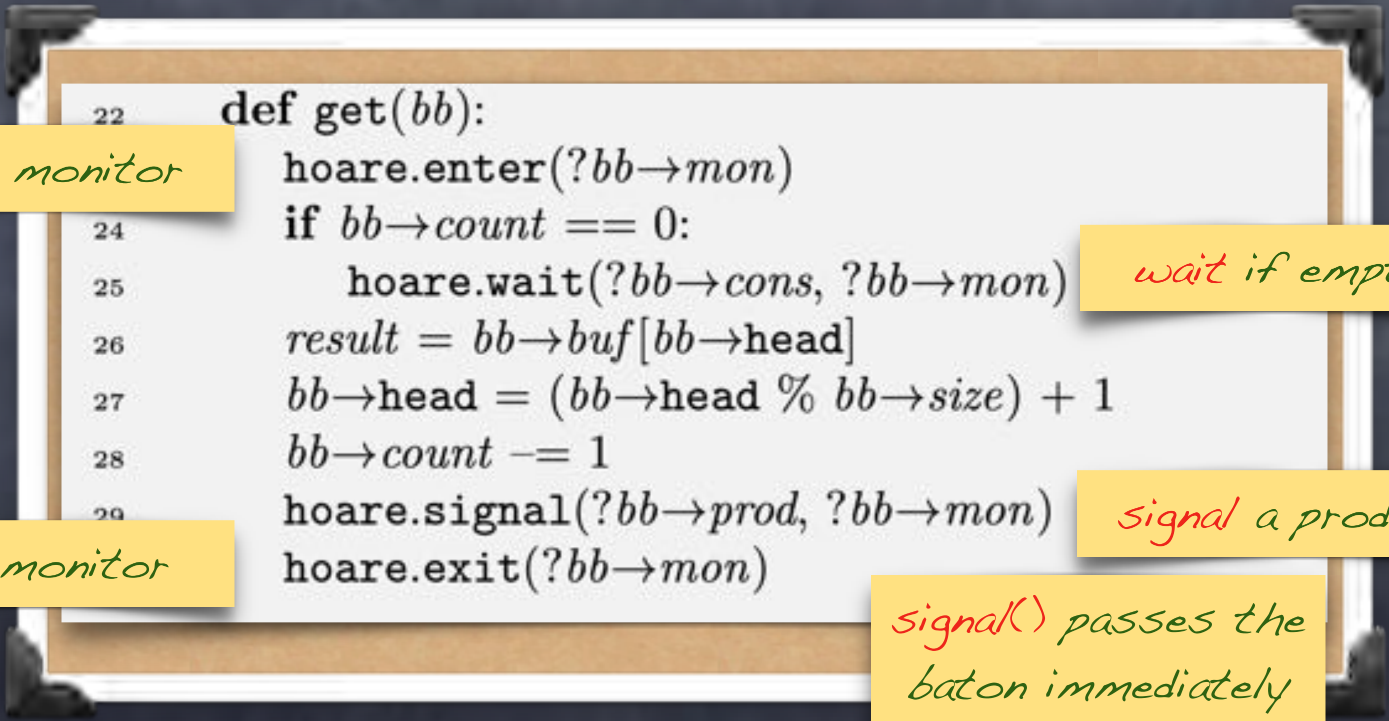
enter monitor

wait if full

signal() passes the baton immediately if there are waiting consumers

exit monitor

Producer/Consumer with Bounded Buffer



```
def get(bb):
    hoare.enter(?bb→mon)
    if bb→count == 0:
        hoare.wait(?bb→cons, ?bb→mon)
    result = bb→buf[bb→head]
    bb→head = (bb→head % bb→size) + 1
    bb→count −= 1
    hoare.signal(?bb→prod, ?bb→mon)
    hoare.exit(?bb→mon)
```

enter monitor

wait if empty

signal a producer

exit monitor

signal() passes the baton immediately if there are waiting producers

Mesa Monitors

Mesa Language, Xerox PAak 1980

- Syntactically similar to Hoare monitors
  - monitors and condition variables
- Semantically closer to busy waiting
  - wait(cond. var.): wait for condition, but may get back the CPU when condition is not satisfied (!)
  - notify(cond. var.): move to ready queue a thread waiting for the condition, if any, but don't transfer control to it
  - notifyAll(cond. var.): move to ready queue all threads waiting for the condition, but don't transfer control to any of them

Very different from Hoare's monitors

What are the implications?

Hoare

- Signaling is atomic with the resumption of waiting thread
  - shared state cannot change before waiting thread is resumed
  - safety requires to signal only when condition holds
- Shared state can be checked using an if statement
- Makes it easier to prove liveness
- Tricky to implement

Mesa

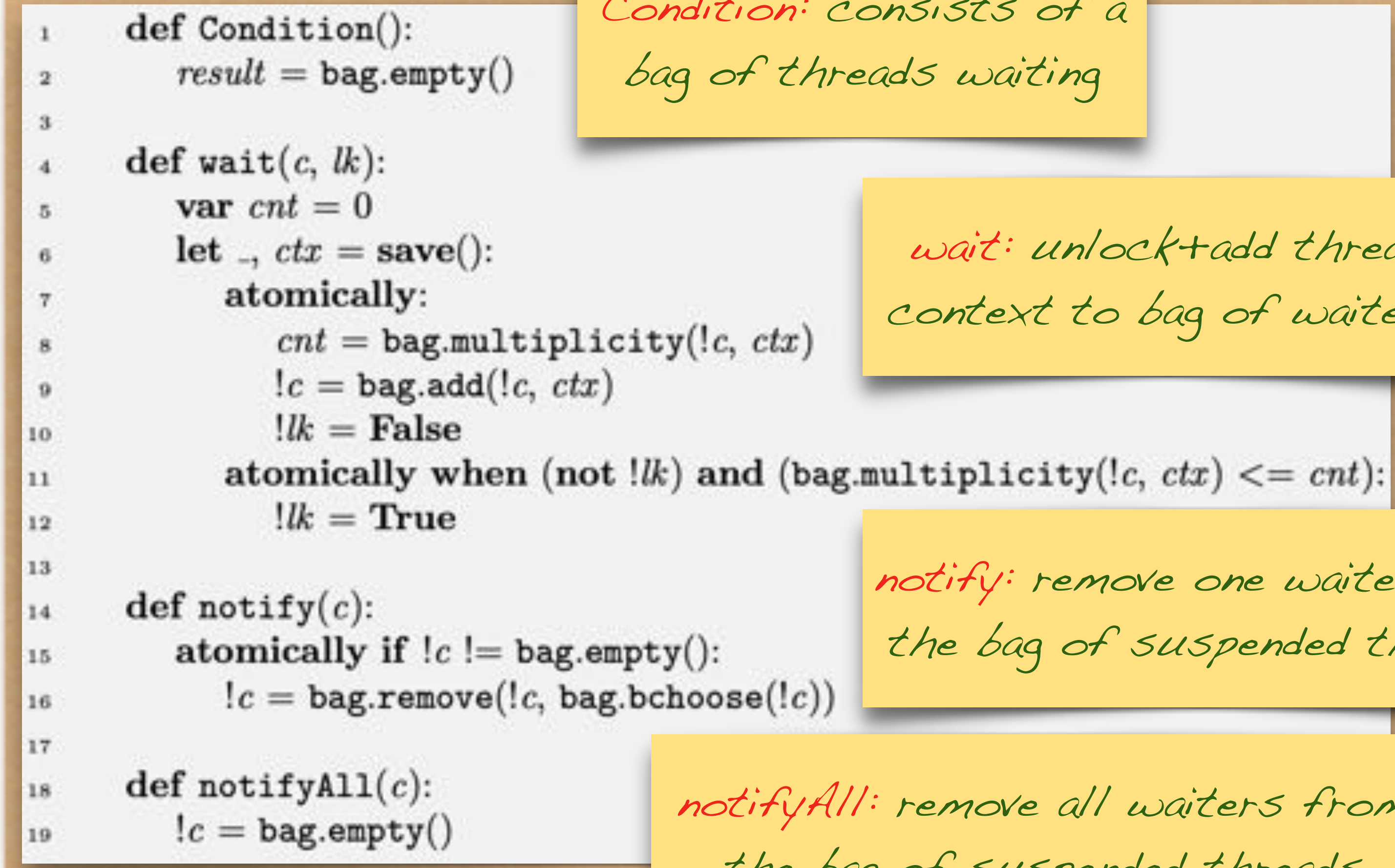
- notify() and notifyAll() are hints
  - adding them affects performance, never safety
- Shared state must be checked in a loop (the condition could have changed since the thread was notified!)
- Simple implementation
- Resilient to spurious wakeup

Hoare vs Mesa Monitors

Hoare Monitors	Mesa Monitors
Baton passing approach	If at first you don't succeed... sleep & try again!
signal passes baton	notify(all) moves waiting threads back to ready queue
used by most books	used by most real systems

Mesa monitors won the test of time...

Mesa Monitors in Harmony



```
def Condition():
    result = bag.empty()

def wait(c, lk):
    var cnt = 0
    let _, ctx = save():
        atomically:
            cnt = bag.multiplicity(!c, ctx)
            !c = bag.add(!c, ctx)
            !lk = False
        atomically when (not !lk) and (bag.multiplicity(!c, ctx) <= cnt):
            !lk = True

def notify(c):
    atomically if !c != bag.empty():
        !c = bag.remove(!c, bag.bchoose(!c))

def notifyAll(c):
    !c = bag.empty()
```

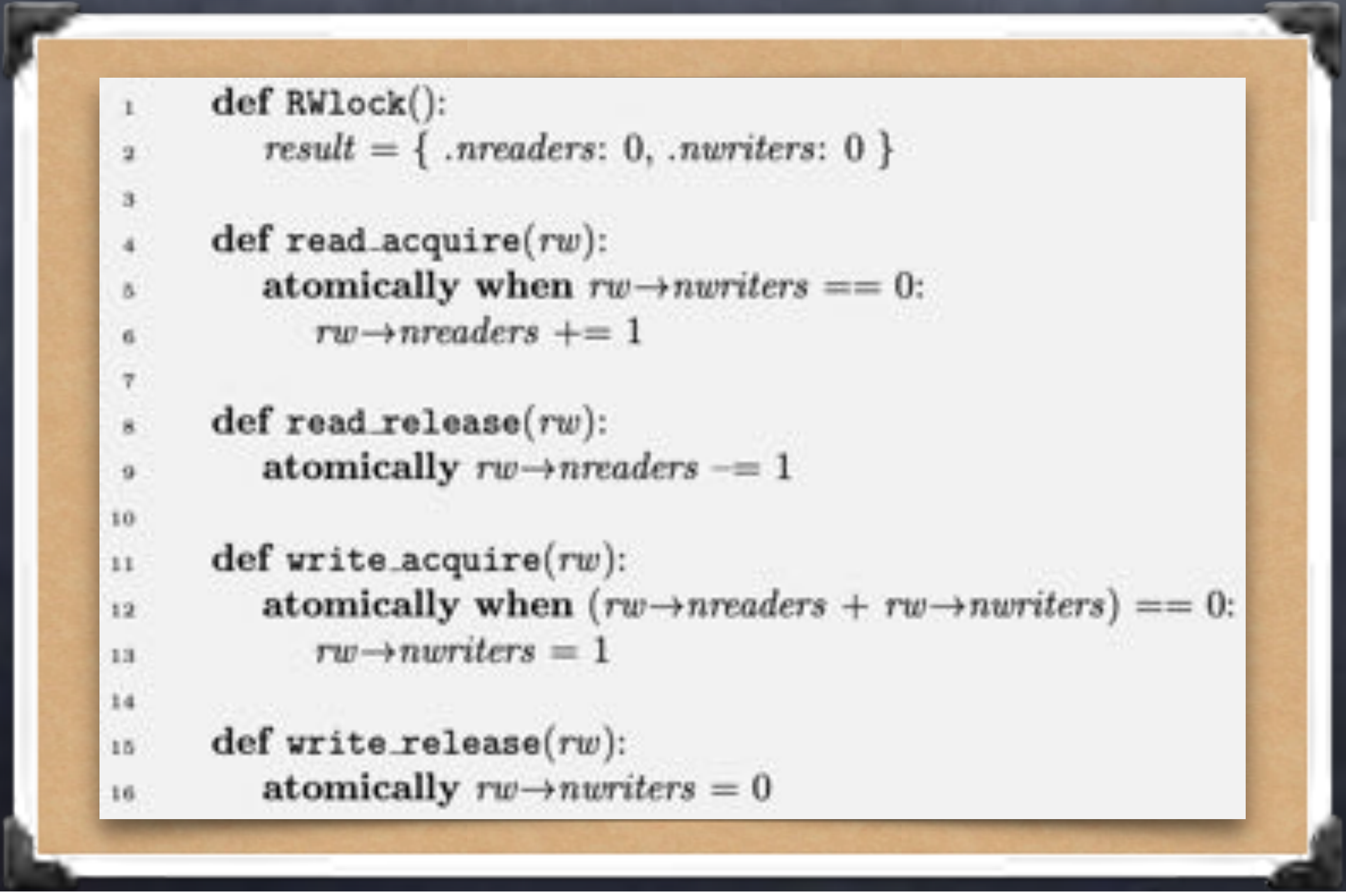
Condition: consists of a bag of threads waiting

wait: unlock+add thread context to bag of waiters

notify: remove one waiter from the bag of suspended threads

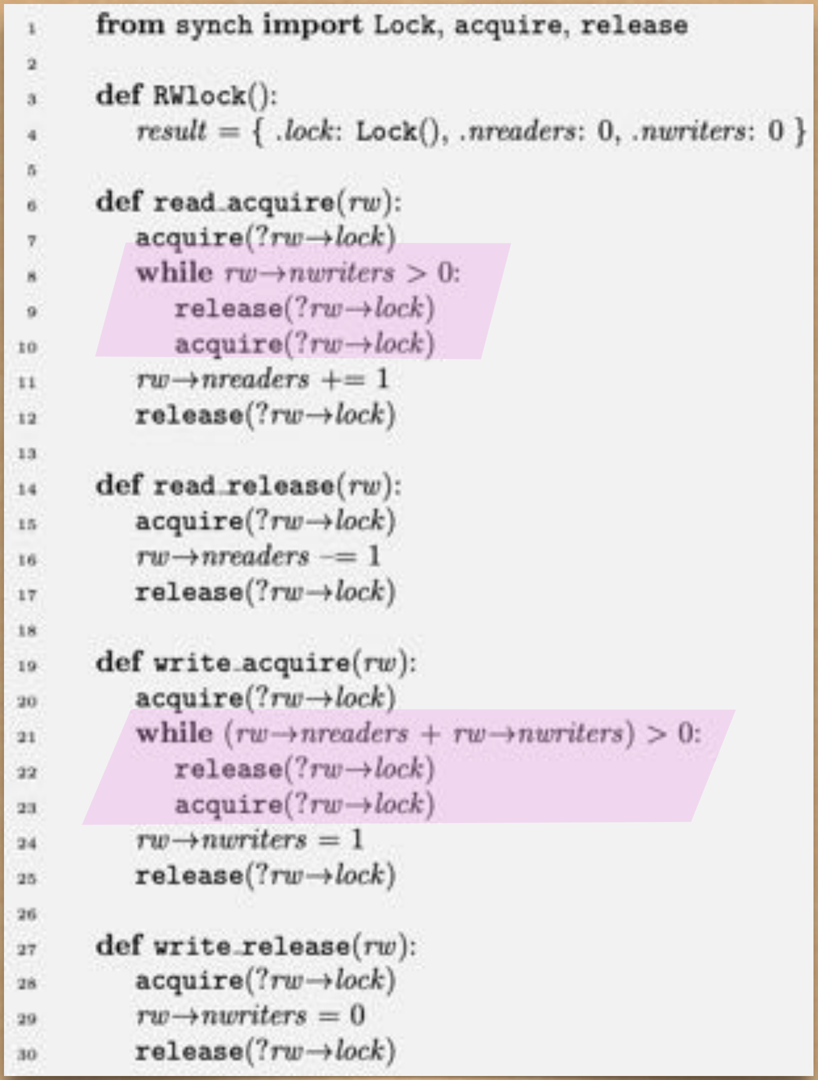
notifyAll: remove all waiters from the bag of suspended threads

Reader/Writer Lock Specification (again)



```
def RWlock():
    result = { .nreaders: 0, .nwriters: 0 }

def read_acquire(rw):
    atomically when rw→nwriters == 0:
        rw→nreaders += 1

def read_release(rw):
    atomically rw→nreaders −= 1

def write_acquire(rw):
    atomically when (rw→nreaders + rw→nwriters) == 0:
        rw→nwriters = 1

def write_release(rw):
    atomically rw→nwriters = 0
```

Busy-Waiting Implementation

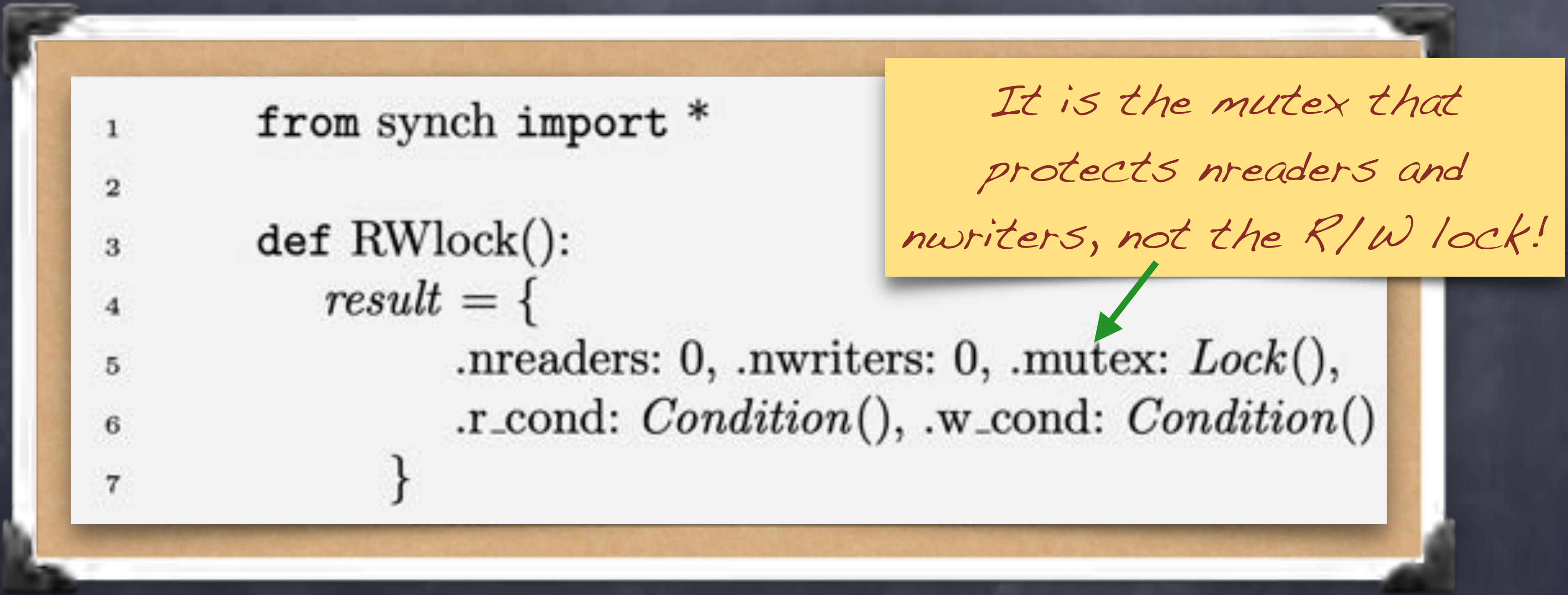


```
from synch import Lock, acquire, release

def RWlock():
    result = { .lock: Lock(), .nreaders: 0, .nwriters: 0 }

def read_acquire(rw):
    acquire(?rw→lock)
    while rw→nwriters > 0:
        release(?rw→lock)
        acquire(?rw→lock)
    rw→nreaders += 1
    release(?rw→lock)

def read_release(rw):
    acquire(?rw→lock)
    rw→nreaders −= 1
    release(?rw→lock)

def write_acquire(rw):
    acquire(?rw→lock)
    while (rw→nreaders + rw→nwriters) > 0:
        release(?rw→lock)
        acquire(?rw→lock)
    rw→nwriters = 1
    release(?rw→lock)

def write_release(rw):
    acquire(?rw→lock)
    rw→nwriters = 0
    release(?rw→lock)
```

It has the same behaviors as the implementation!

Process continuously scheduled to try to get the lock even if it is not available

Reader/Writer lock with Mesa monitors



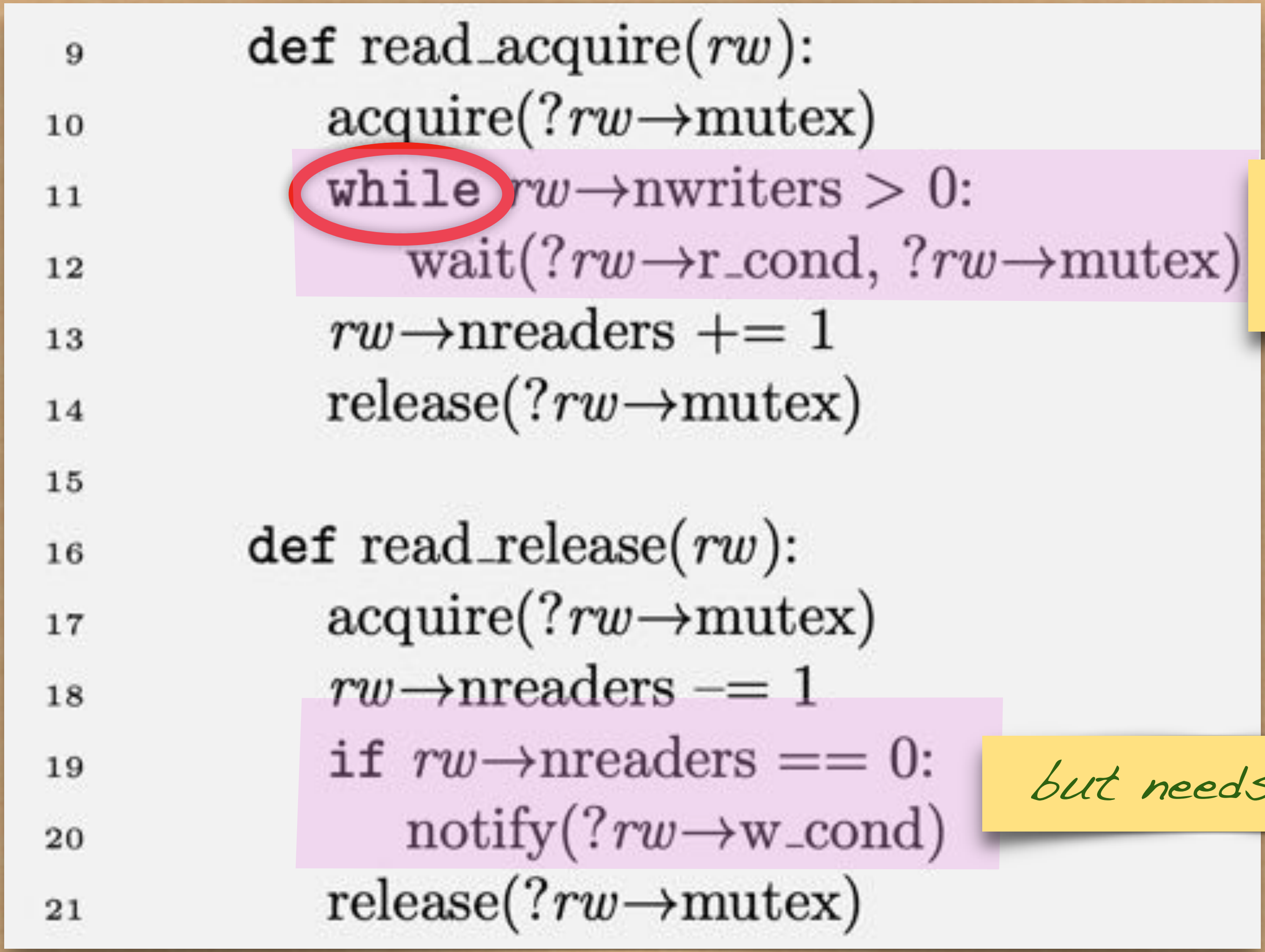
```
from synch import *

def RWlock():
    result = {
        .nreaders: 0, .nwriters: 0, .mutex: Lock(),
        .r_cond: Condition(), .w_cond: Condition()
    }
```

It is the mutex that protects nreaders and nwriters, not the R/W lock!

Invariants

- If n readers in the critical section, then $nreaders \geq n$
- If n writers in the critical section, then $nwriters \geq n$
- $(nreaders \geq 0 \wedge nwriters = 0) \vee (nreaders = 0 \wedge nwriters = \leq 1)$

R/W Lock, Reader

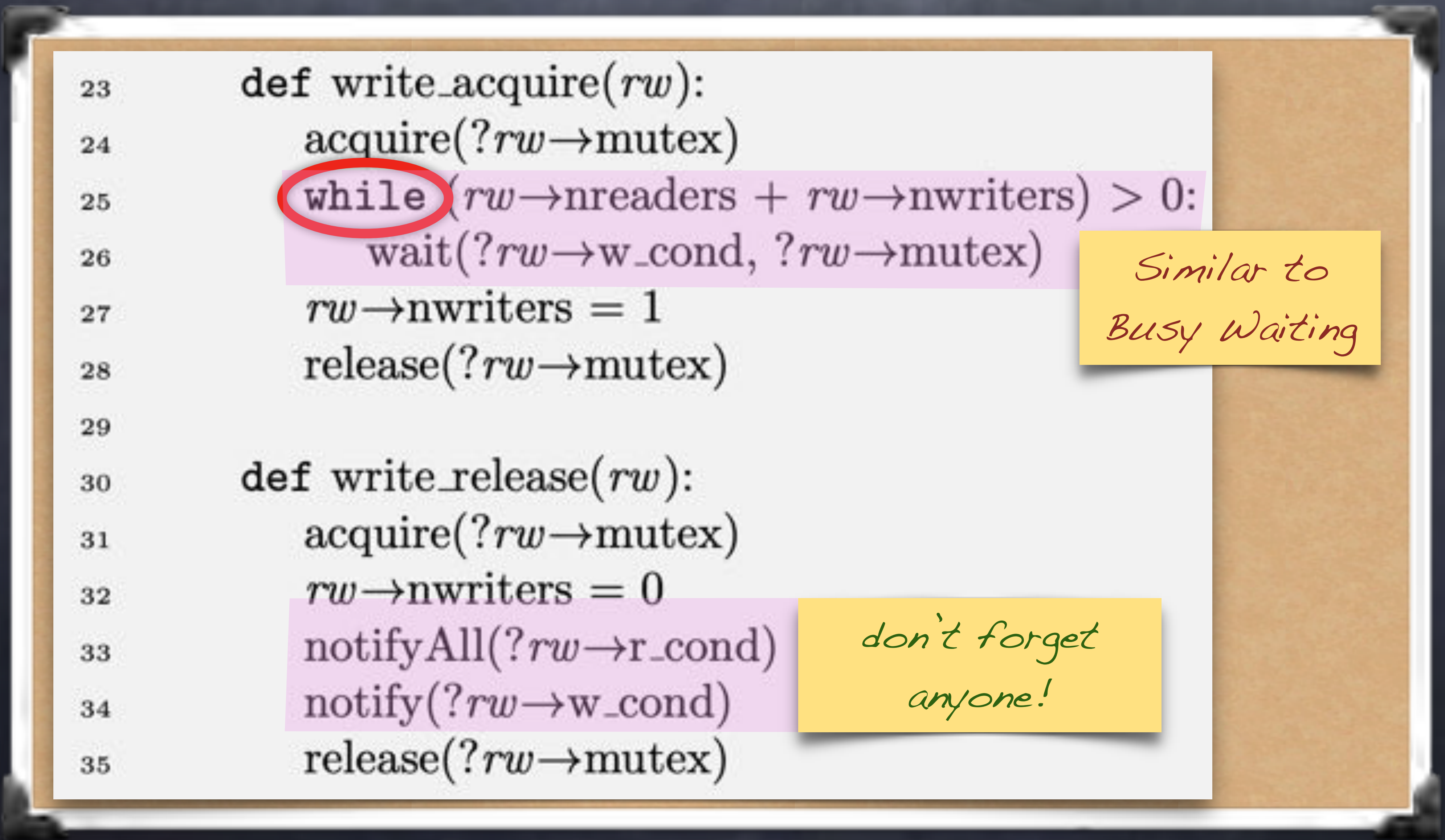


```
def read_acquire(rw):
    acquire(?rw→mutex)
    while rw→nwriters > 0:
        wait(?rw→r_cond, ?rw→mutex)
    rw→nreaders += 1
    release(?rw→mutex)

def read_release(rw):
    acquire(?rw→mutex)
    rw→nreaders −= 1
    if rw→nreaders == 0:
        notify(?rw→w_cond)
    release(?rw→mutex)
```

Similar to Busy Waiting

but needs this

R/W Lock, Writer



```
def write_acquire(rw):
    acquire(?rw→mutex)
    while (rw→nreaders + rw→nwriters) > 0:
        wait(?rw→w_cond, ?rw→mutex)
    rw→nwriters = 1
    release(?rw→mutex)

def write_release(rw):
    acquire(?rw→mutex)
    rw→nwriters = 0
    notifyAll(?rw→r_cond)
    notify(?rw→w_cond)
    release(?rw→mutex)
```

Similar to Busy Waiting

don't forget anyone!

Conditional Critical Sections

Let me count the ways…

Busy Waiting	Split Binary Semaphores	Mesa Monitors
Use a lock and a loop	Use a collection of binary semaphores	Use a lock, a collection of condition variables, and a loop
Easy to write the code	Just follow the recipe	Notifying is tricky
Easy to understand the code	Tricky to understand if you don't know the recipe	Easy to understand the code
OK-ish for true multicore, but bad for virtual threads	Good for virtual threading. Thread only runs when it can make progress	Good for both multicore and virtual threading

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
                                0
  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Running

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

Running

girl back on ready Q

Kid and Cook Threads

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
kid_main() {

   dig_in_mud();
   BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

Ready

Running

cook gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;

   int numburgers = 0;
   condition hungrykid;


   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook back on ready Q

Kid and Cook Threads

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
kid_main() {

   dig_in_mud();
   BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

boy gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy tries to enter monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy tries to enter monitor

Kid and Cook Threads

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Running

boy back on ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

cook gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl tries to enter monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;


   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

monitor has lock Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Waiting

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

girl placed on lock Q

Kid and Cook Threads

Waiting

Q:

0

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook gets the CPU

Kid and Cook Threads

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Waiting

Q:

0

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

cook executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

Waiting

```
Monitor BurgerKing {
  Lock mlock;
  Q:
  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

girl tries to enter monitor

Kid and Cook Threads

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Waiting

0

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

Ready

Running

girl placed on lock Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

Waiting

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

boy (with lock) gets the CPU

Kid and Cook Threads

Waiting

0

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

Ready

Running

no burgers

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

Waiting

Waiting

```
Monitor BurgerKing {
  Lock mlock;
      Q:
  int numburgers = 0;
  condition hungrykid;
      Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

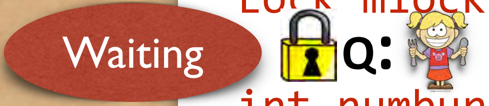
boy releases monitor lock & waits for hungrykid signal

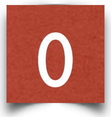
Kid and Cook Threads

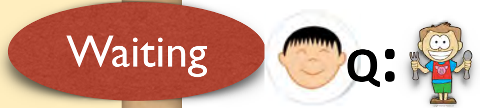
```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

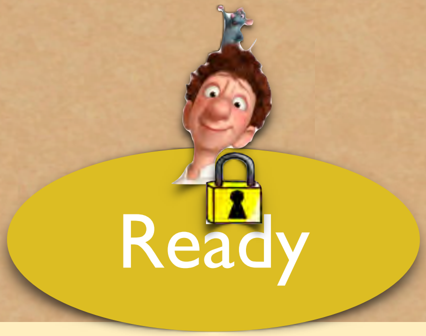
```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```









cook joins ready Q with release of monitor lock

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

Waiting

Waiting

```
Monitor BurgerKing {
  Lock mlock;
  Q:
  int numburgers = 0;
  condition hungrykid;
  Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

Ready

Running

cook gets the CPU

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
  BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
  facetime_Karthik();
  facetime_Gma();
  BK.kid_eat();

}
```

Waiting

Waiting

```
Monitor BurgerKing {
  Lock mlock;
                Q:
  int numburgers = 0;
  condition hungrykid;
                Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook acquires monitor lock

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
  BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
  facetime_Karthik();
  facetime_Gma();
  BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
  Q:
  int numburgers = 0;
  condition hungrykid;
  Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

1

Waiting

Waiting

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook makes a burger

Kid and Cook Threads

Waiting

```
Monitor BurgerKing {
  Lock mlock;
  Q:
  int numburgers = 0;
  condition hungrykid;
  Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

I

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

Ready

Running

cook signals a hungry kid

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
     Q:
  int numburgers = 0;
  condition hungrykid;
     Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

I

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

cook releases monitor lock; girl goes back to ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

I

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

cook leaves monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

1

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

cook executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

I

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Running

cook moved to ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
      Q:
   int numburgers = 0;
   condition hungrykid;
      Q:

   void kid_eat() {
       mlock.acquire()
       while (numburgers==0)
           hungrykid.wait()
       numburgers -= 1
       mlock.release()
   }

   void makeburger() {
       mlock.acquire()
       ++numburger;
       hungrykid.notify();
       mlock.release()
   }
}
```

I

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

girl gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
      Q:
   int numburgers = 0;
   condition hungrykid;
      Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

girl acquires monitor lock

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
     Q:
  int numburgers = 0;
  condition hungrykid;
     Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

I

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

I

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Mmmm... burgers...

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

0

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

girl eats burger

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
     Q:
  int numburgers = 0;
  condition hungrykid;
     Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl releases monitor lock

Kid and Cook Threads

```
Monitor BurgerKing {
   Lock mlock;
      Q:
   int numburgers = 0;
   condition hungrykid;
      Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

0

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

girl leaves monitor

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
  BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
  facetime_Karthik();
  facetime_Gma();
  BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
    Q:
  int numburgers = 0;
  condition hungrykid;
    Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

0

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

Running

girl moved to ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy acquires monitor lock

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
      Q:
   int numburgers = 0;
   condition hungrykid;
      Q:

   void kid_eat() {
       mlock.acquire()
       while (numburgers==0)
          hungrykid.wait()
       numburgers -= 1
       mlock.release()
   }

   void makeburger() {
       mlock.acquire()
       ++numburger;
       hungrykid.notify();
       mlock.release()
   }
}
```

0

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

boy returns from wait

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
     Q:
  int numburgers = 0;
  condition hungrykid;
     Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy executes

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
   BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
     Q:
   int numburgers = 0;
   condition hungrykid;
     Q:

   void kid_eat() {
      mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
   }

   void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
   }
}
```

0

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

no burgers

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
   Lock mlock;
        Q:                    0
   int numburgers = 0;
   condition hungrykid;
        Q:

   void kid_eat() {
       mlock.acquire()
       while (numburgers==0)
           hungrykid.wait()
       numburgers -= 1
       mlock.release()
   }

   void makeburger() {
       mlock.acquire()
       ++numburger;
       hungrykid.notify();
       mlock.release()
   }
}
```

Waiting

```
cook_main() {

   wake();
   shower();
   drive_to_work();
   while(not_5pm)
     BK.makeburger();
   drive_to_home();
   watch_got();
   sleep();

}
```

Running

boy releases monitor lock and waits for hungrykid signal

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
  BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
  facetime_Karthik();
  facetime_Gma();
  BK.kid_eat();

}
```

Waiting

```
Monitor BurgerKing {
  Lock mlock;
     Q:
  int numburgers = 0;
  condition hungrykid;
     Q:

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook gets the CPU

and so forth...