More Musings on Readers/Writers

- If readers and writers are waiting, and a writer exits, who goes first?
- Why do readers use a mutex?
- Why don't writers use a mutex?
- What if we move rcount_mutex.V() just above if (rcount = 1)?

```
int read() {
  rcount_mutex.P();
  rcount := rcount+1;
  if (rcount == 1) then
    rOw_lock.P();
  rcount_mutex.V();
  ...
  /* Perform read */
  ...
  rcount_mutex.P();
  rcount := rcount-1;
  if (rcount == 0) then
    rOw.lock.V();
  rcount_mutex.V();
}
```

```
void write() {
  rOw_lock.P();
  ...
  /* Perform write */
  ...
  rOw_lock.V();
}
```

```
Shared:
int rcount = 0;
Semaphore rcount_mutex (1)
Semaphore rOw_lock(1);
```


Classic Mistakes with Semaphores


```
P(S)
CS
P(S)
```
I

I stuck on 2nd P(). Subsequent processes hopelessly pile on 1st P()

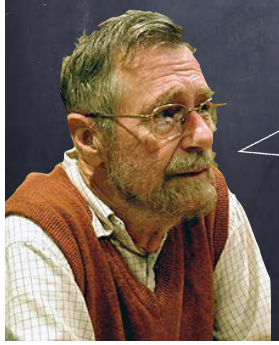
```
V(S)
CS
V(S)
```
J

Undermines mutex:
- J does not get permission via P()
- "extra" V() allows other processes into CS inappropriately

```
P(S)
if (x) return;
CS
V(S)
```
L

Conditional code can change code flow in the CS. Caused by code updates (bug fixes, etc.) by someone other than original author of code.

Edsger's perspective



"During system conception it transpired that we used the semaphores in two completely different ways. The difference is so marked that, looking back, one wonders whether it was really fair to present the two ways as uses of the very same primitives. On the one hand, we have the semaphores used for mutual exclusion, on the other hand, the private semaphores."

The structure of the 'THE'-Multiprogramming System" Communications of the ACM v. 11 n. 5 May 1968.

Semaphores considered harmful

- Semaphores are "low-level" primitives. Small errors
  - can introduce incorrect executions or grind the program to a halt
  - very difficult to debug
- Semaphores conflate two distinct uses
  - mutex
  - condition synchronization (e.g., bounded buffer)

Enter Monitors

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

How did Monitors come about?

- First introduced as an OO programming language construct
  - synchronization object + methods
  - calling a method defined in the monitor automatically acquires the lock
    - Mesa, Java (synchronized methods)
- A programming convention
  - can be defined in any language

Condition Variables

- An abstraction for conditional synchronization associated with a monitor
- Enable threads to wait inside a critical section by releasing the monitor lock
- A misnomer
  - can neither be read nor set to a value
  - think of them as a label associated with a condition on a resource and a queue
  - thread can wait in the queue (inside the CS) until they are notified that condition holds

How do I wait for thee? Let me count the ways...

- At the entry of the monitor
  - threads can queue on the mutex that protects the monitor, waiting for the thread that is currently in the monitor to exit (or to release the lock by waiting on a condition variable)
- On a condition variable
  - threads can queue waiting on the associated condition

Condition Variables: Operations

- Three operations on condition variable x
  - x.wait(lock)
    - Atomically: Release lock and go to sleep
    - sleep by waiting on the queue associated with x
  - x.notify (historically called x.signal())
    - wake up a waiter if any; otherwise no-op
    - wake up by moving waiter to the ready queue
  - x.notifyall (historically called x.broadcast())

Resource Variables

- Condition variables (unlike semaphores) are stateless
- Each condition variable should be associated with a resource variable (RV) tracking the state of that resource
  - 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

Notify() Semantics

- Which thread executed once notify() is called on CV?
  - if no thread is waiting on CV, notifier continues
  - if one or more thread waiting on CV:
    - at least two ready threads: notifier and thread(s) that are moved from the queue of the CV to the ready queue
    - only one can run...
    - ...but which one?

Notify() semantics: Mesa vs. Hoare

- Mesa (or Brinch Hansen) semantics:
  - signaled thread is moved to ready list, but not guaranteed to run right away
- 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.
  - notifying thread is resumed if former waiter exits crucial section, or if it waits again

What are the implications?

Mesa/Brinch Hansen

- signal() and broadcast() are hints
  - adding them affects performance, never safety
- Shared state must be checked in a loop (could have changed! (tricky tricky...))
  - robust to spurious wakeups
- Simple implementation
- Used in most systems
- Sponsored by a Turing Award
  - Butler Lampson

Hoare

- Signaling is atomic with the resumption of waiting thread
  - shared state cannot change before waiting thread is resumed
- Shared state can be checked using an if statement
- Makes it easier to prove liveness
- Tricky to implement
  - interferes with scheduling
- Used in most books (but not yours!)
- Sponsored by a Turing Award
  - Tony Hoare

notify() vs notifyall() (signal() vs. broadcast())

- It is always safe to use notifyall() instead of notify()
  - only performance is affected
- notify() is preferable when
  - at most one waiting thread can make progress
  - any thread waiting on the condition variable can make progress
- notifyall() is preferable when
  - multiple waiting thread may be able to make progress
  - a single condition variable is used for multiple predicates
    - some waiting threads can make progress, others can't

Condition Variables vs Semaphores

- wait() vs P()
  - P() blocks threads only if value = 0
  - wait() always block and gives up monitor lock
- notify() vs V()
  - V is stateful - future thread does not wait on P()
  - if no waiting thread, notify() is a no op
  - condition variables are stateless
- Code that uses monitors is easier to read
  - Conditions for which threads are waiting are explicit

Producer-Consumer with Bounded Buffer


```
// add item to buffer
void produce(int item) {
  empty.P();
  mutex_in.P();
  buf[in%N] := item;
  in := in+1;
  mutex_in.V();
  full.V();
}
```

```
Shared:
int buf[N];
int in := 0, out := 0;
Semaphore mutex_in(1),
          mutex_out(1);
Semaphore empty(N), full(0);
```

Semaphores

```
// remove item from buffer
int consume() {
  full.P();
  mutex_out.P();
  int item := buf[out%N];
  out := out+1;
  mutex_out.V();
  empty.V();
  return(item);
}
```

Producer-Consumer with Bounded Buffer


```
// add item to buffer
void produce(int item) {
  lock.Acquire()
  while (n == N)
    wait(notFull);
  buf[in%N] := item;
  in := in+1;
  n := n+1;
  notify(notEmpty);
  lock.Release()
}
```

```
Monitor Producer Consumer {
char buf[N];
Lock lock;
int n := 0, in := 0, out := 0;
Condition notEmpty, notFull;
```

Monitor

```
// remove item from buffer
int consume() {
  lock.Acquire();
  while (n == 0)
    wait(nonEmpty);
  int item := buf[out%N];
  out := out+1;
  n:= n-1
  notify(notFull);
  lock.Release();
  return(item);
}
```


Kid and Cook Threads

```
kid_main() {
  dig_in_mud();
  BK.kid_eat();
  bathe();
  draw_on_walls();
  BK.kid_eat();
  facetime_Karthik();
  facetime_oma();
  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.signal();
     mlock.release()
  }
}
```

```
cook_main() {
  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();
}
```

Kid and Cook Threads

```
kid_main() {
  dig_in_mud();
  BK.kid_eat();
  bathe();
  draw_on_walls();
  BK.kid_eat();
  facetime_Karthik();
  facetime_oma();
  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.signal();
     mlock.release()
  }
}
```

```
cook_main() {
  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();
}
```

girl swapped in

Kid and Cook Threads

```
kid_main() {
  dig_in_mud();
  BK.kid_eat();
  bathe();
  draw_on_walls();
  BK.kid_eat();
  facetime_Karthik();
  facetime_oma();
  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.signal();
     mlock.release()
  }
}
```

```
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_oma();
   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.signal();
      mlock.release()
   }
}
```

```
cook_main() {

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

}
```

girl swapped out

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
   BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
   facetime_Karthik();
   facetime_oma();
   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.signal();
      mlock.release()
   }
}
```

```
cook_main() {

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

}
```

cook swapped in

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
   BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
   facetime_Karthik();
   facetime_oma();
   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.signal();
      mlock.release()
   }
}
```

```
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_oma();
   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.signal();
      mlock.release()
   }
}
```

```
cook_main() {

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

}
```

cook swapped out

Kid and Cook Threads

```
kid_main() {

  dig_in_mud();
  BK.kid_eat();
  bathe();
  draw_on_walls();
  BK.kid_eat();
  facetime_Karthik();
  facetime_oma();
  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.signal();
     mlock.release()
  }
}
```

```
cook_main() {

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

}
```

boy swapped in

Kid and Cook Threads

```
kid_main() {

  dig_in_mud();
  BK.kid_eat();
  bathe();
  draw_on_walls();
  BK.kid_eat();
  facetime_Karthik();
  facetime_oma();
  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.signal();
     mlock.release()
  }
}
```

```
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_oma();
  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.signal();
     mlock.release()
  }
}
```

```
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_oma();
  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.signal();
     mlock.release()
  }
}
```

```
cook_main() {

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

}
```

boy gets monitor lock

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
   BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
   facetime_Karthik();
   facetime_oma();
   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.signal();
      mlock.release()
   }
}
```

```
cook_main() {

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

}
```

boy swapped out

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
   BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
   facetime_Karthik();
   facetime_oma();
   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.signal();
      mlock.release()
   }
}
```

```
cook_main() {

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

}
```

girl swapped in

Kid and Cook Threads

```
kid_main() {

   dig_in_mud();
   BK.kid_eat();
   bathe();
   draw_on_walls();
   BK.kid_eat();
   facetime_Karthik();
   facetime_oma();
   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.signal();
      mlock.release()
   }
}
```

```
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_oma();
   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.signal();
      mlock.release()
   }
}
```

```
cook_main() {

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

}
```

monitor has lock Queue

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.signal();
     mlock.release()
  }
}
```

```
kid_main() {

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

}
```

```
cook_main() {

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

}
```

Q:

Ready

Running

girl placed on lock 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.signal();
     mlock.release()
  }
}
```

```
kid_main() {

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

}
```

```
cook_main() {

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

}
```

Q:

Ready

Running

cook swapped in

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.signal();
     mlock.release()
  }
}
```

```
kid_main() {

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

}
```

```
cook_main() {

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

}
```

Q:

Ready

Running

cook executes

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.signal();
     mlock.release()
  }
}
```

```
kid_main() {

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

}
```

```
cook_main() {

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

}
```

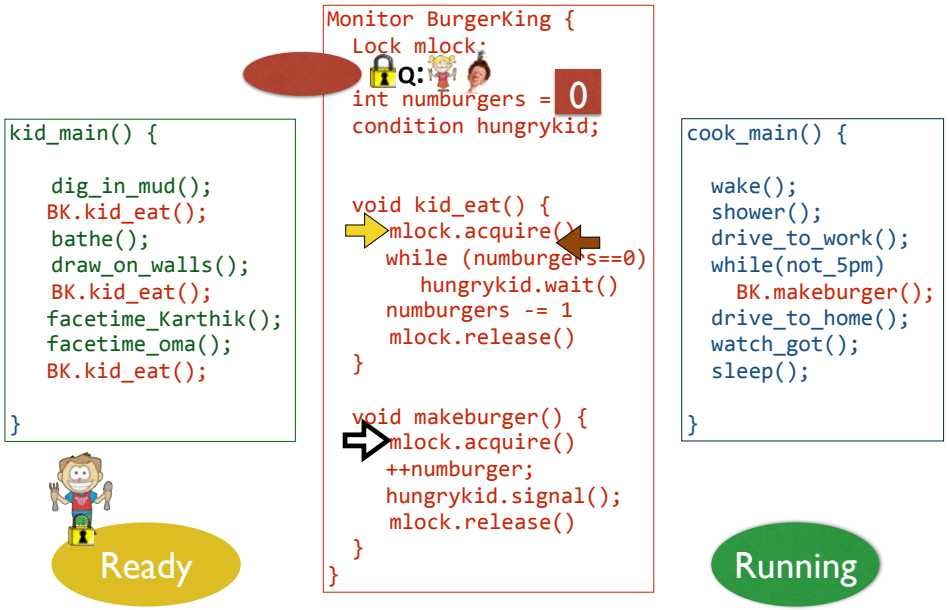
Q:

Ready

Running

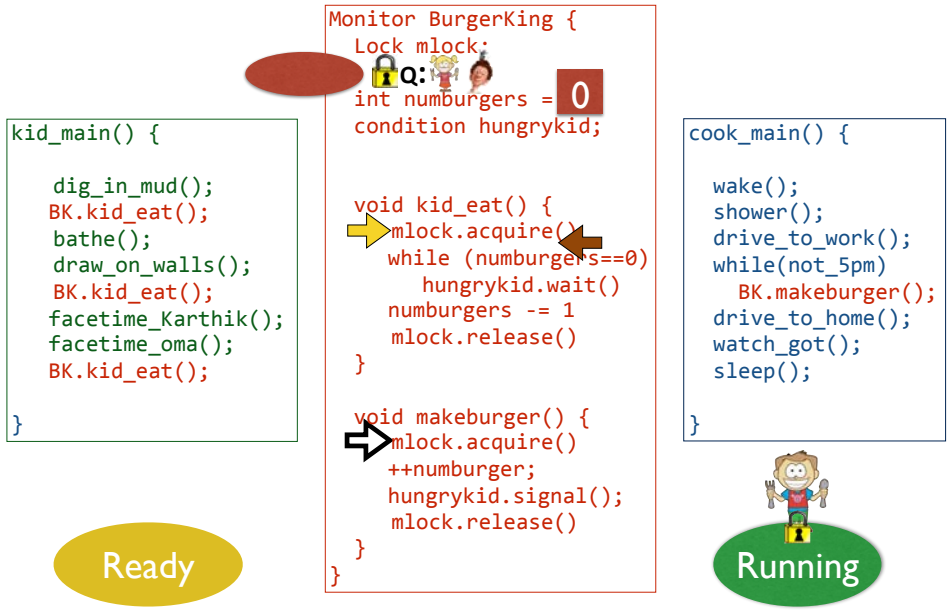
cook tries to enter monitor

Kid and Cook Threads



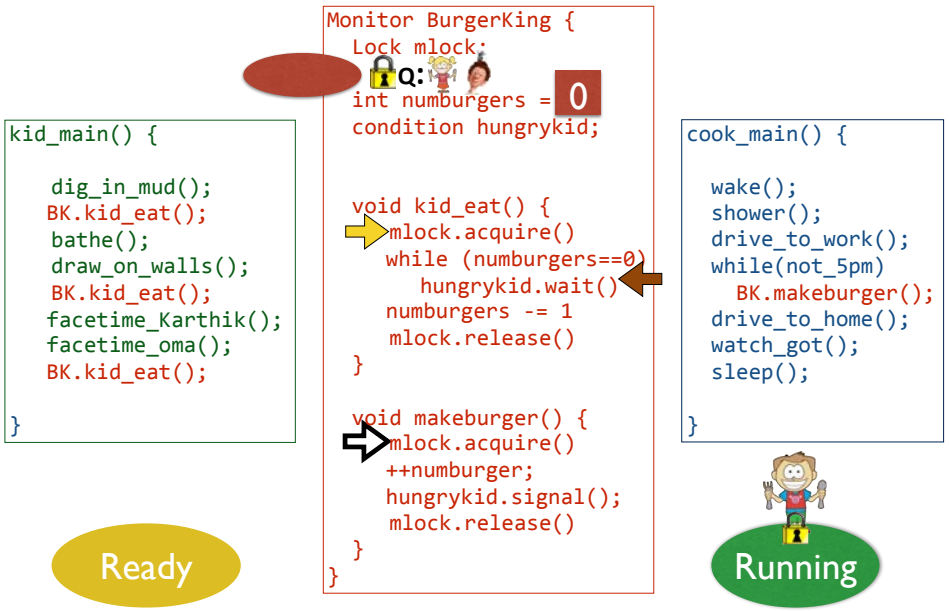
cook placed on lock Q

Kid and Cook Threads



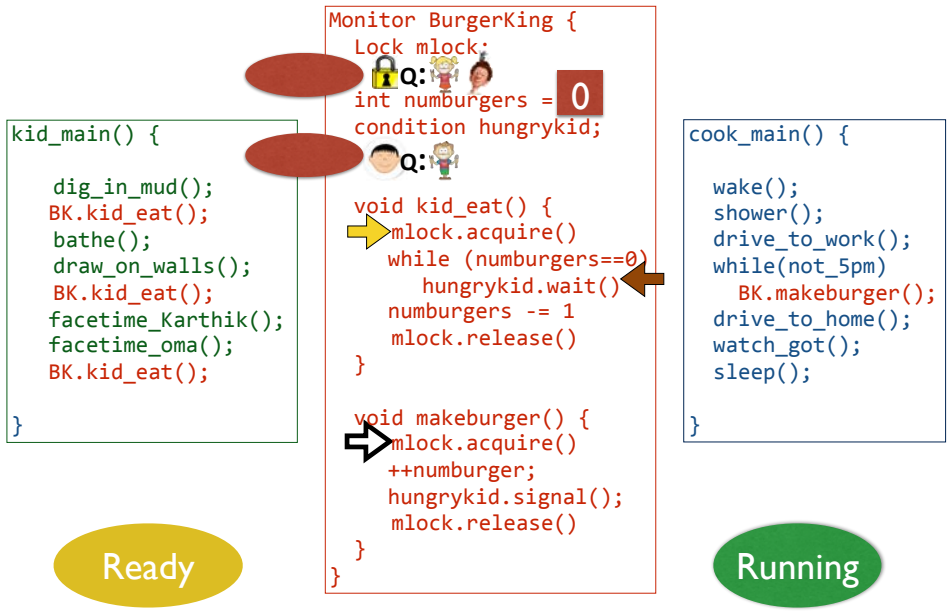
boy swapped in w/lock

Kid and Cook Threads



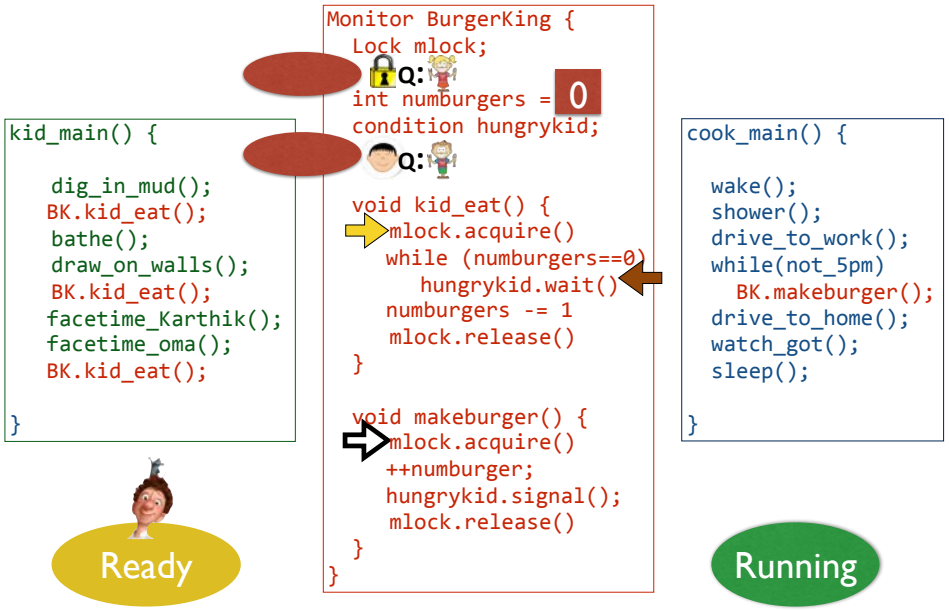
no burgers!

Kid and Cook Threads



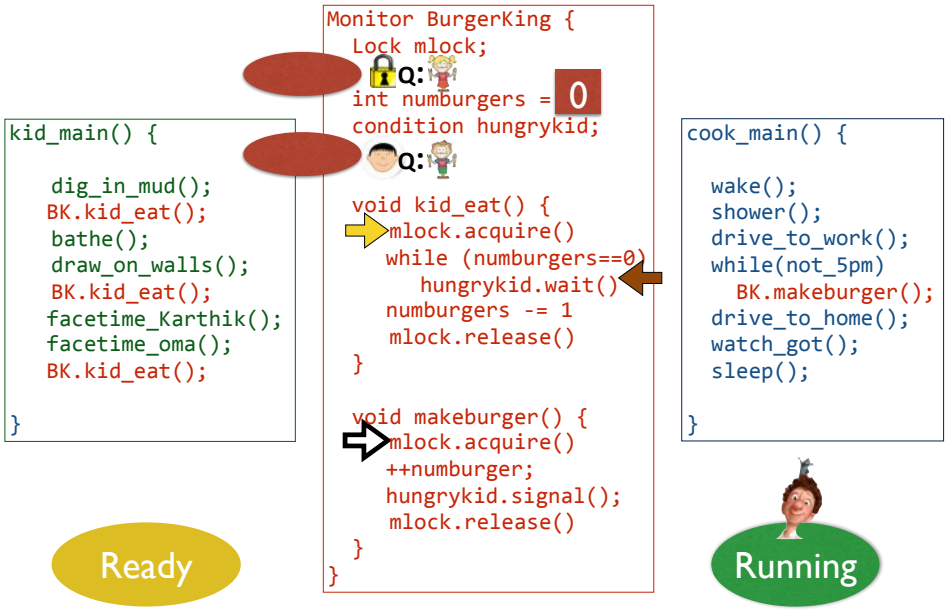
boy releases monitor lock & waits for hungrykid signal

Kid and Cook Threads



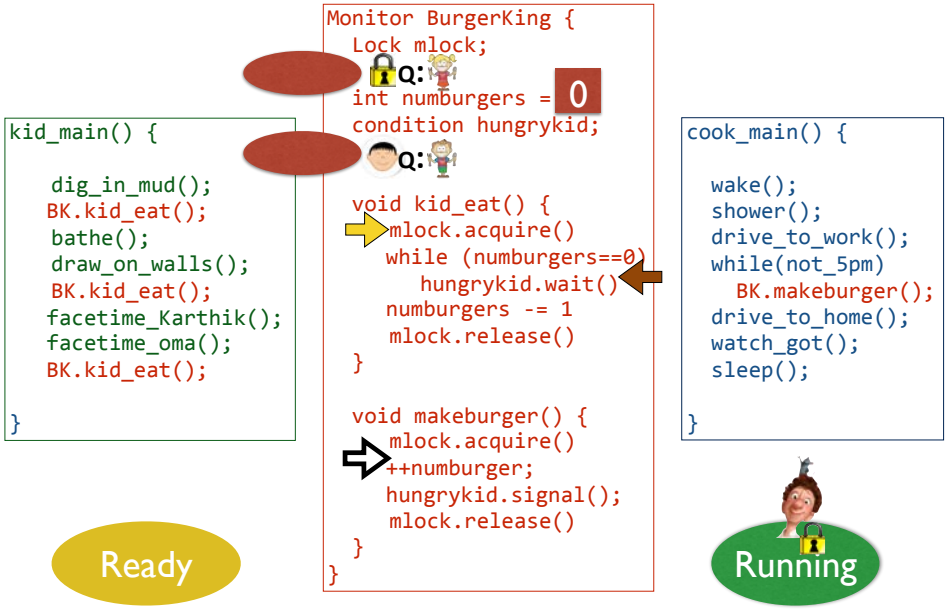
cook made Ready with release of monitor lock

Kid and Cook Threads



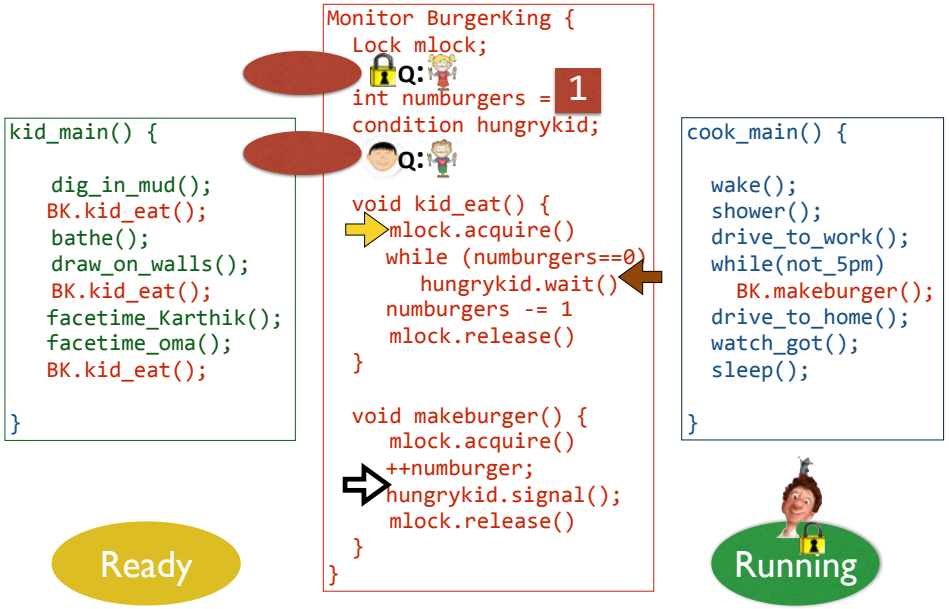
cook swapped in

Kid and Cook Threads



cook acquires monitor lock

Kid and Cook Threads



cook makes a burger

Kid and Cook Threads

```
kid_main() {

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

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 1
  condition hungrykid;

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

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

```
cook_main() {

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

}
```

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_oma();
    BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 1
  condition hungrykid;

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

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

```
cook_main() {

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

}
```

cook releases monitor lock, girl made Ready

Kid and Cook Threads

```
kid_main() {

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

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 1
  condition hungrykid;

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

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

```
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_oma();
    BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 1
  condition hungrykid;

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

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

```
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_oma();
    BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 1
  condition hungrykid;

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

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

```
cook_main() {

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

}
```

cook swapped out

Kid and Cook Threads

```
kid_main() {

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

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 1
  condition hungrykid;

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

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

```
cook_main() {

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

}
```

girl swapped in

Kid and Cook Threads

```
kid_main() {

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

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 1
  condition hungrykid;

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

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.signal();
     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_oma();
    BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
  int numburgers = 1
  condition hungrykid;

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

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

```
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_oma();
    BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;
  Q:
  int numburgers = 1
  condition hungrykid;
  Q:
  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

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

```
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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
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

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
    facetime_Karthik();
    facetime_oma();
    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.signal();
     mlock.release()
  }
}
```

```
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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
cook_main() {

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

}
```

girl swapped out

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
    facetime_Karthik();
    facetime_oma();
    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.signal();
     mlock.release()
  }
}
```

```
cook_main() {

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

}
```

boy swapped in

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
    facetime_Karthik();
    facetime_oma();
    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.signal();
     mlock.release()
  }
}
```

```
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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
cook_main() {

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

}
```

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_oma();
    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.signal();
     mlock.release()
  }
}
```

```
cook_main() {

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

}
```

cook swapped in