浙江大學
ZHEJIANG UNIVERSITY

Deadlock Prevention: Practical Examples

Yajin Zhou (http://yajin.org)

Zhejiang University

Acknowledgement: some pages are based on the slides from Zhi Wang(fsu).

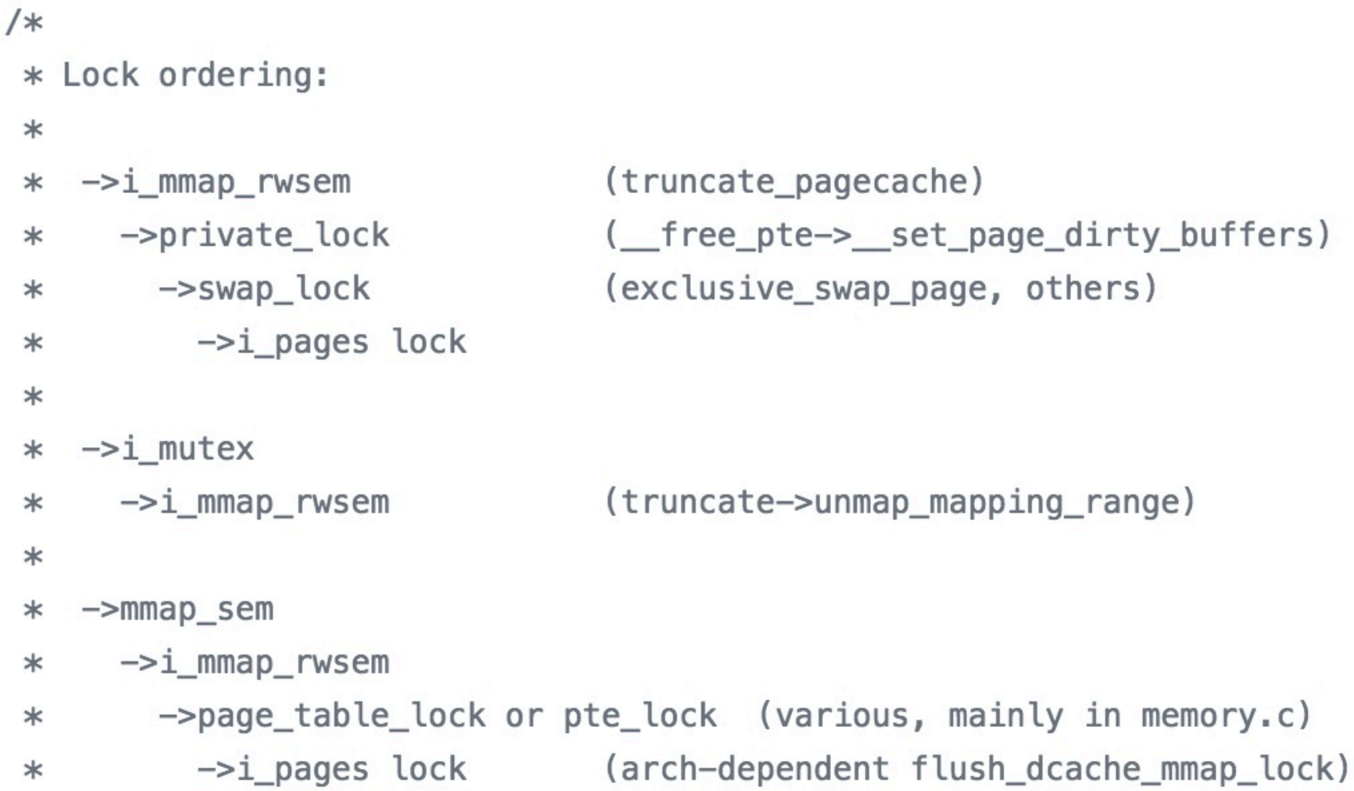
Conditions for Deadlock

- Mutual exclusion
- Hold and wait
- No preemption
- Circular wait

Circular Wait

- Most practical one, provide **a total ordering** to obtain the lock
- In complex systems, **partial ordering**

https://github.com/torvalds/linux/blob/master/mm/filemap.c

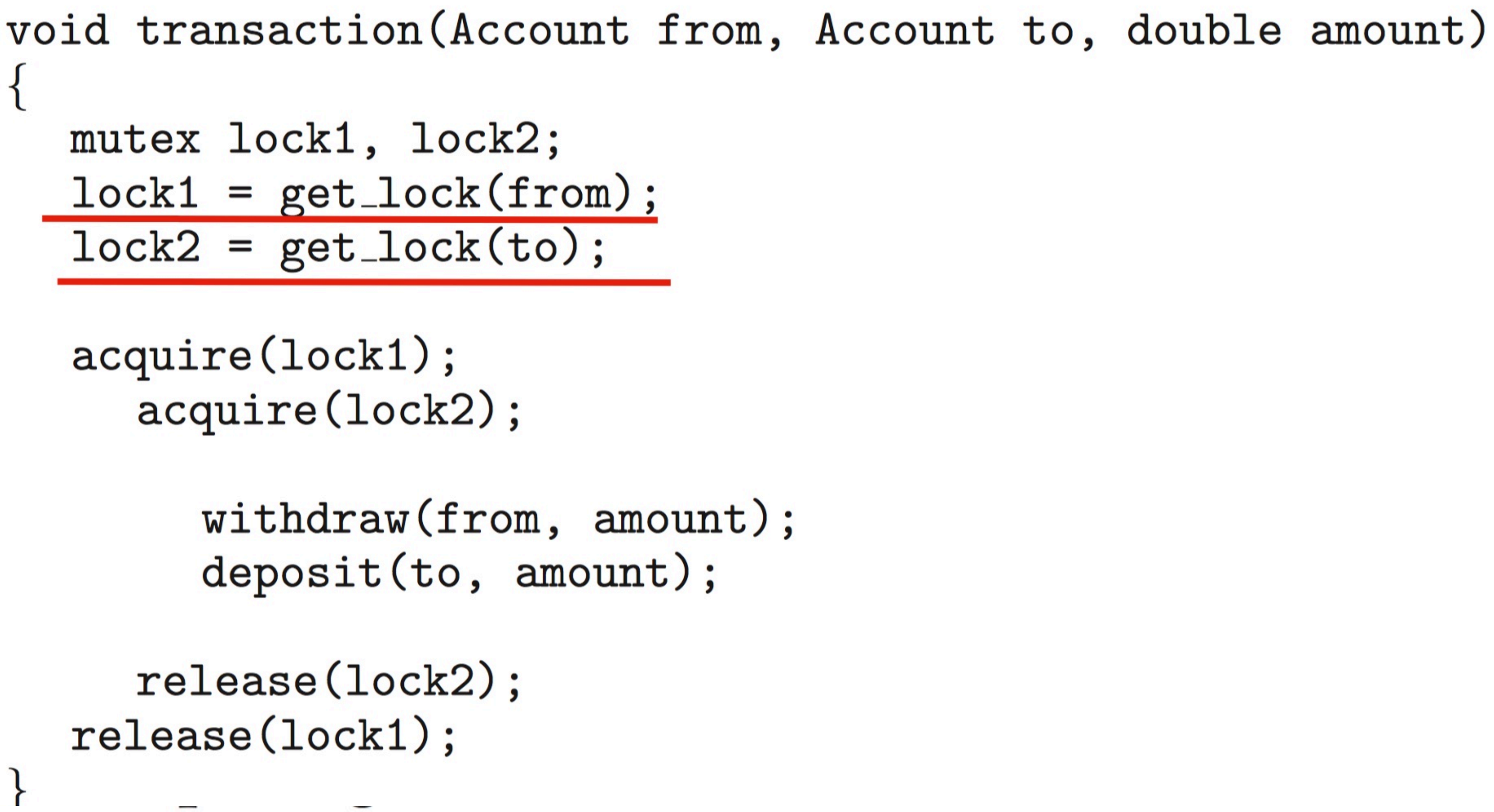


```
/*
 * Lock ordering:
 *
 *  ->i_mmap_rwsem             (truncate_pagecache)
 *    ->private_lock           (__free_pte->__set_page_dirty_buffers)
 *      ->swap_lock            (exclusive_swap_page, others)
 *        ->i_pages lock
 *
 *  ->i_mutex
 *    ->i_mmap_rwsem           (truncate->unmap_mapping_range)
 *
 *  ->mmap_sem
 *    ->i_mmap_rwsem
 *      ->page_table_lock or pte_lock  (various, mainly in memory.c)
 *        ->i_pages lock       (arch-dependent flush_dcache_mmap_lock)
```

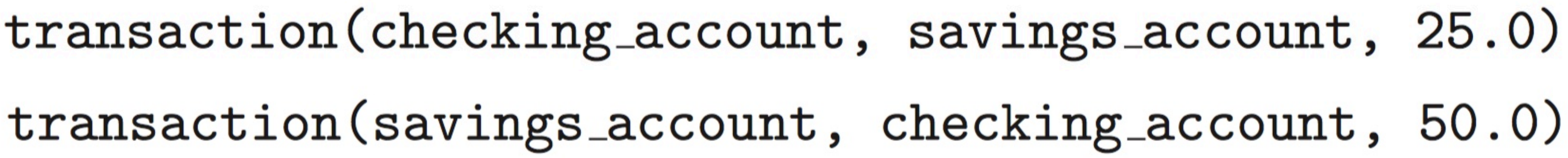
Any tool to check this?

Circular Wait

- Dynamic lock

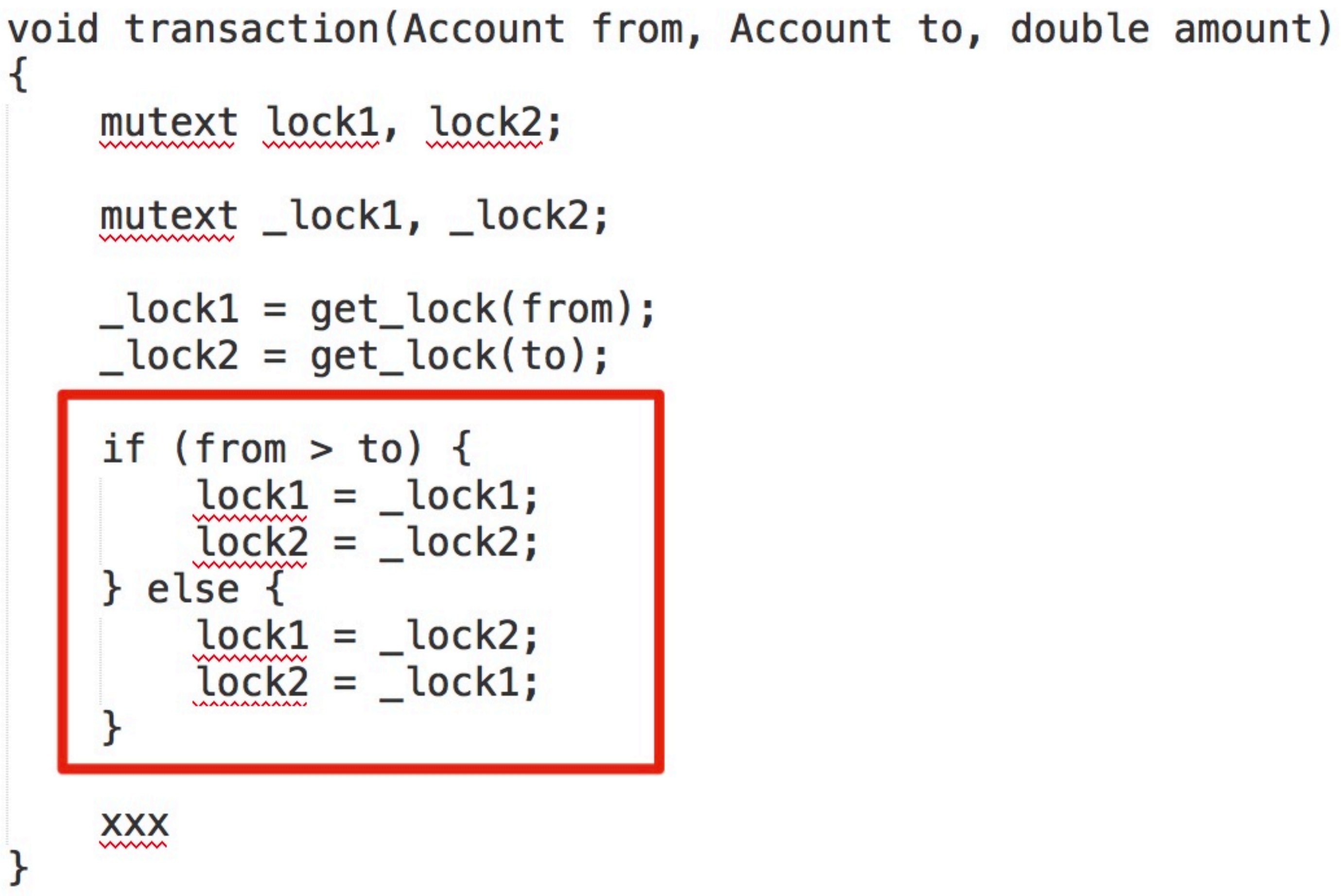


```
void transaction(Account from, Account to, double amount)
{
   mutex lock1, lock2;
   lock1 = get_lock(from);
   lock2 = get_lock(to);

   acquire(lock1);
      acquire(lock2);

         withdraw(from, amount);
         deposit(to, amount);

      release(lock2);
   release(lock1);
}
```



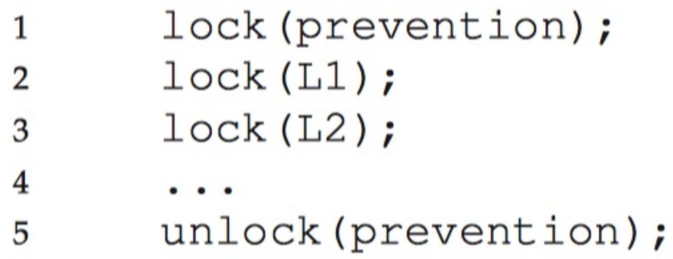
```
transaction(checking_account, savings_account, 25.0)

transaction(savings_account, checking_account, 50.0)
```

Circular Wait



```
void transaction(Account from, Account to, double amount)
{
    mutext lock1, lock2;

    mutext _lock1, _lock2;

    _lock1 = get_lock(from);
    _lock2 = get_lock(to);

    if (from > to) {
        lock1 = _lock1;
        lock2 = _lock2;
    } else {
        lock1 = _lock2;
        lock2 = _lock1;
    }

    xxx
}
```

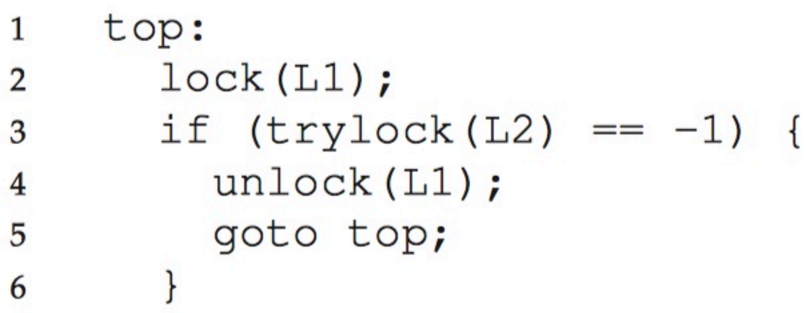
Hold and wait



```
lock(prevention);
lock(L1);
lock(L2);
...
unlock(prevention);
```

No Preemption

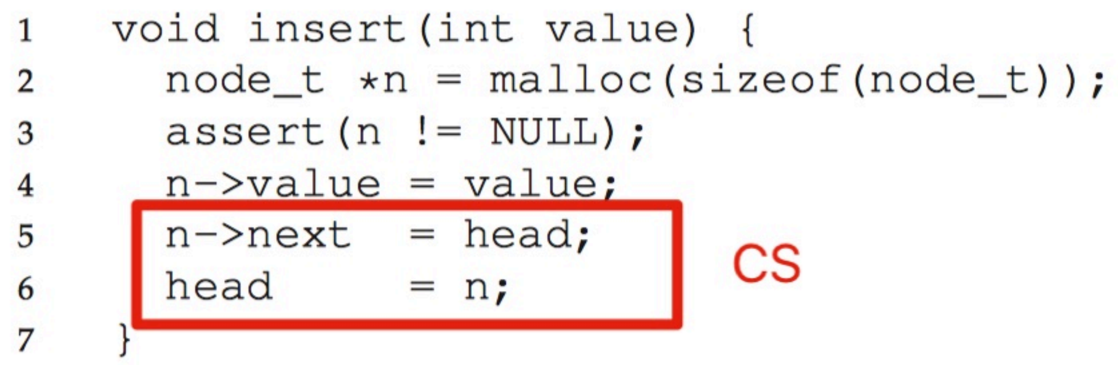
- actually does not add preemption, but **giving up** the lock



```
top:
  lock(L1);
  if (trylock(L2) == -1) {
    unlock(L1);
    goto top;
  }
```

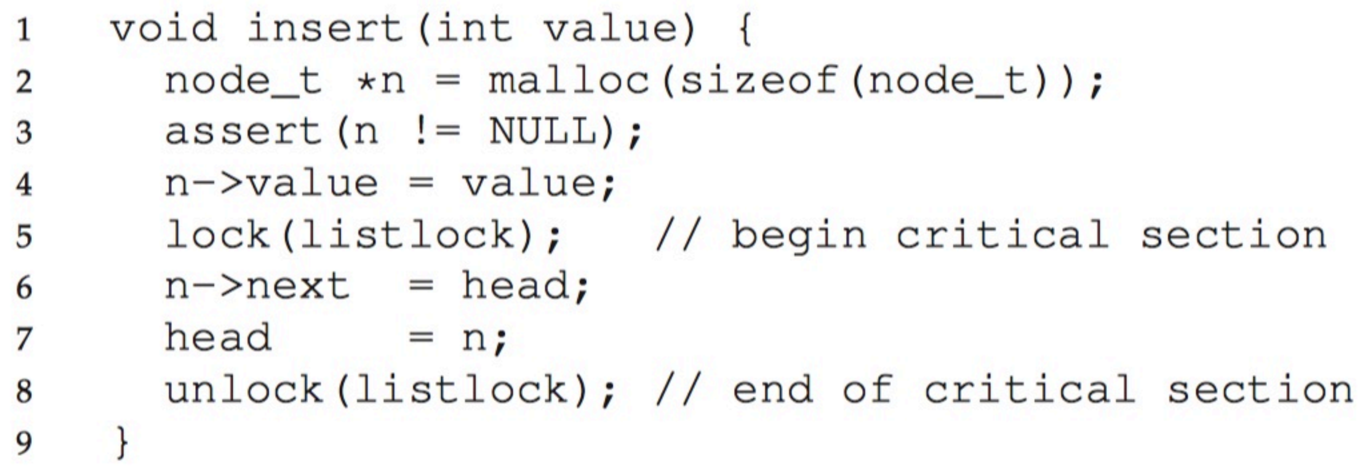
Mutual Exclusion

- Lock-free: use powerful hardware instruction



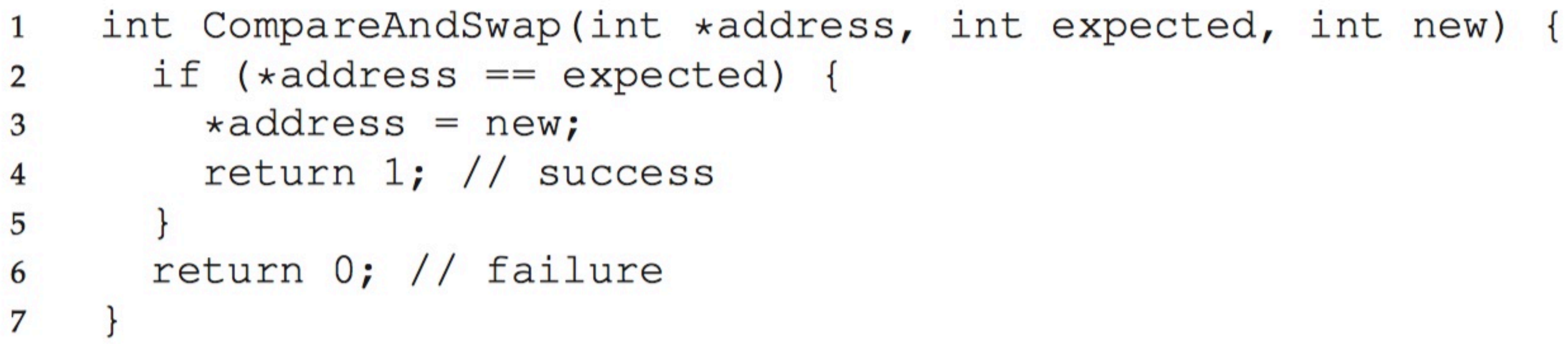
```
void insert(int value) {
  node_t *n = malloc(sizeof(node_t));
  assert(n != NULL);
  n->value = value;
  n->next   = head;
  head      = n;
}
```

CS

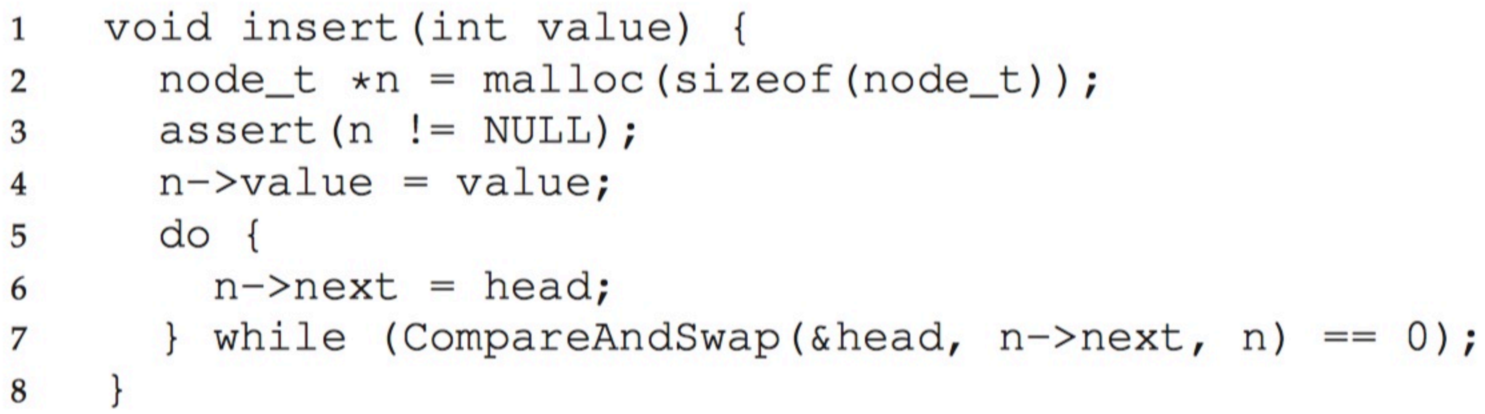


```
void insert(int value) {
  node_t *n = malloc(sizeof(node_t));
  assert(n != NULL);
  n->value = value;
  lock(listlock);    // begin critical section
  n->next   = head;
  head      = n;
  unlock(listlock); // end of critical section
}
```

Mutual Exclusion



```
int CompareAndSwap(int *address, int expected, int new) {
  if (*address == expected) {
    *address = new;
    return 1; // success
  }
  return 0; // failure
}
```



```
void insert(int value) {
  node_t *n = malloc(sizeof(node_t));
  assert(n != NULL);
  n->value = value;
  do {
    n->next = head;
  } while (CompareAndSwap(&head, n->next, n) == 0);
}
```