



## Deadlock Prevention: Practical Examples

---

Yajin Zhou (<http://yajin.org>)

Zhejiang University



# 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)
{
    mutex lock1, lock2;

    mutex _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

---

```
1  lock (prevention);  
2  lock (L1);  
3  lock (L2);  
4  ...  
5  unlock (prevention);
```



# No Preemption

---

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

```
1 top:
2   lock(L1);
3   if (trylock(L2) == -1) {
4     unlock(L1);
5     goto top;
6   }
```



# Mutual Exclusion

- Lock-free: use powerful hardware instruction

```
1 void insert(int value) {
2     node_t *n = malloc(sizeof(node_t));
3     assert(n != NULL);
4     n->value = value;
5     n->next = head;
6     head = n;
7 }
```

CS

```
1 void insert(int value) {
2     node_t *n = malloc(sizeof(node_t));
3     assert(n != NULL);
4     n->value = value;
5     lock(listlock); // begin critical section
6     n->next = head;
7     head = n;
8     unlock(listlock); // end of critical section
9 }
```





# Mutual Exclusion

---

```
1  int CompareAndSwap(int *address, int expected, int new) {
2      if (*address == expected) {
3          *address = new;
4          return 1; // success
5      }
6      return 0; // failure
7  }
```

```
1  void insert(int value) {
2      node_t *n = malloc(sizeof(node_t));
3      assert(n != NULL);
4      n->value = value;
5      do {
6          n->next = head;
7      } while (CompareAndSwap(&head, n->next, n) == 0);
8  }
```