University of BRISTOL

Computer Systems B

COMS20012

Introduction to Operating Systems and Security

bristol.ac.uk

University of BRISTOL

Implementing spinlocks

bristol.ac.uk

spinlock

```
void getTheMoney(account_t account, int amount) {
        int money = get_balance(account);
        money = money + amount;
        put_balance(account, money);
        return;
}
```

spinlock

```
int pay; //shared variable for test and set

void getTheMoney(account_t account, int amount) {
        test_and_set(&pay, 1);
        int money = get_balance(account);
        money = money + amount;
        put_balance(account, money);
        test_and_set(&pay, 0);
        return;
}
```

University of BRISTOL

Does this work?

bristol.ac.uk

University of BRISTOL

Does this work?

NO! How do we tell if another thread as already set pay?

bristol.ac.uk

spinlock

```
int pay; //shared variable for test and set

void getTheMoney(account_t account, int amount) {
        test_and_set(&pay, 1);
        int money = get_balance(account);
        money = money + amount;
        put_balance(account, money);
        test_and_set(&pay, 0);
        return;
}
```

spinlock

```
int pay; //shared variable for test and set

void getTheMoney(account_t account, int amount) {
        if (test_and_set(&pay, 1) == 1) {
                // it was already set. What do we do?
        }
        int money = get_balance(account);
        money = money + amount;
        put_balance(account, money);
        test_and_set(&pay, 0);
        return;
}
```

spinlock

```
int pay; //shared variable for test and set

void getTheMoney(account_t account, int amount) {
        while (test_and_set(&pay, 1) == 1) {
                ; // test again!
        }
        int money = get_balance(account);
        money = money + amount;
        put_balance(account, money);
        test_and_set(&pay, 0);
        return;
}
```

Busy wait!

- We are trying again, and again, and again
- Ok on multicore
  - Some other thread running on another core will change the value
- Terrible on single core
  - The value will never change!
  - **Preventing progress!**

Passive lock

- Active lock
  - Busy wait
- Passive lock
  - Sleep
  - … and try again when it wakes up
  - Let another thread make progress when it sleeps
- **Why would you ever use an active lock?**

Passive lock

- Switching between threads is not free
  - **Context Switch**
  - More in future videos
- When the critical section is **short**
  - It is **more expensive to switch context**
- When the critical section is **long**
  - It is **more expensive to busy wait**

University of BRISTOL

Thank you

bristol.ac.uk