Теория и практика многопоточного программирования

Семинар 9

Неганов Алексей

Московский физико-технический институт (национальный исследовательский университет)
Кафедра теоретической и прикладной информатики

Москва 2020

Monitor

```
acquire(m); // Acquire this monitor's lock.
while (!p) { // While the condition/predicate/assertion that we are waiting for is not true...
^^Iwait(m, cv); // Wait on this monitor's lock and condition variable.
}
// ... Critical section of code goes here ...
signal(cv2); -- OR -- notifyAll(cv2); // cv2 might be the same as cv or different.
release(m); // Release this monitor's lock.
```

Seqlock

```
void thread_read(void) {
    do {
        while((c=lock->cnt)&1);
        /* read */
    } while(lock->cnt!=c);
}
```

```
void thread_write(void) {
    do {
        while((c=lock->cnt)&1);
    } while(!CAS(&lock->cnt,c,c+1);
    /* write */
    atomic_fetch_add(&lock->cnt, 1);
}
```

- нет указателей
- частые чтения
- запись редка, но обязана быть быстрой

Krefs

```
struct foo {
    struct kref kref;
}

struct foo *foo;
foo = kmalloc(sizeof(*foo), GFP_KERNEL);
kref_init(&foo->kref, foo_release);
kref_get(&foo->kref);
kref_put(&foo->kref);

struct kref *kref_get(struct kref *kref) {
    atomic_inc(&kref->refcount);
    return kref;
}

void foo_release(struct kref *kref) {
    struct foo *foo;
    foo = container_of(foo, struct foo, kref);
    kfree(foo);
}
```

Fastflow

```
Bool push(Element& item) {
    int nextTail = increment(tail);
    if (nextTail != head) {
        array[tail] = item;
        tail = nextTail;
        return true;
    }
    return false;
}

Bool pop(Element& item) {
    if (tail == head)
        return false;
    item = array[head];
    head = increment(head);
    return true;
}
```