锁与原子操作CAS的底层实现

课程提纲

- 思考题
- 原子操作
- 无锁CAS
- Volatile
- 死锁
- Epoll实现分析

课程提纲

思考题

原子操作

无锁CAS

Volatile

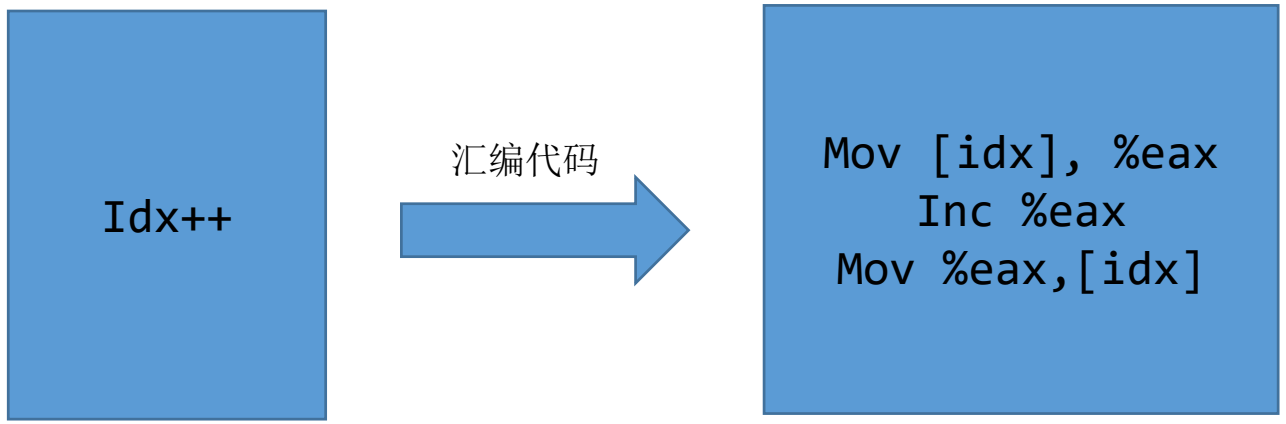
死锁

Epoll实现分析

原子操作

```
Global int idx; //全局定义

idx++; //多线程
```



线程1	线程2
Mov [idx], %eax	
Inc %eax	
Mov %eax, [idx]	
	Mov [idx], %eax
	Inc %eax
	Mov %eax, [idx]

原子操作

线程1	线程2
Mov [idx], %eax	
	Mov [idx], %eax
	Inc %eax
	Mov %eax, [idx]
Inc %eax	
Mov %eax, [idx]	

线程1	线程2
Mov [idx], %eax	
Inc %eax	
	Mov [idx], %eax
	Inc %eax
	Mov %eax, [idx]
Mov %eax, [idx]	

该如何处理？


```
Global int idx; //全局定义

lock (mtx);
idx ++; //多线程
unlock (mtx)
```

会出现线程挂起，线程切换

课程提纲

- 思考题
- 原子操作
- 无锁CAS
- Volatile
- 死锁
- Epoll实现分析

原子操作

```
00004: unsigned long cmpxchg(void *addr, unsigned long _old, unsigned long _new) {
00005:     int *a = addr;
00006:
00007:     if (*a == _old) {
00008:         *a = _new;
00009:     }
00010:
00011:     return _old;
00012: }
```

```
00019: unsigned long cmpxchg(void *addr, unsigned long _old, unsigned long _new) {
00020:
00021:     unsigned long prev;
00022:     volatile unsigned int *_ptr = (volatile unsigned int *)addr;
00023:
00024:     __asm__ volatile (
00025:         "lock; cmpxchg %1, %2"
00026:         : "=a" (prev)
00027:         : "r" (_new), "m" (*_ptr), "0" (_old)
00028:         : "memory"
00029:     );
00030:
00031:     return prev;
00032: }
00033:
```

原子操作

C语言中嵌入汇编

```
__asm__ volatile (
Assembler template
: output operands
: input operands
: clobbered registers list
)
```

```
00019: unsigned long cmpxchg(void *addr, unsigned long _old, unsigned long _new) {
00020:
00021:     unsigned long prev;
00022:     volatile unsigned int *_ptr = (volatile unsigned int *)addr;
00023:
00024:     __asm__ volatile (
00025:         "lock; cmpxchg %1, %2"
00026:         : "=a" (prev)
00027:         : "r" (_new), "m" (*_ptr), "0" (_old)
00028:         : "memory"
00029:     );
00030:
00031:     return prev;
00032: }
00033:
```

“lock; cmpxchg %1 %2” 是操作指令
%1 是冒号以后按照0,1,2的顺序。
cmpxchg _new, *_ptr
0(_old): 是存储在eax上面的。

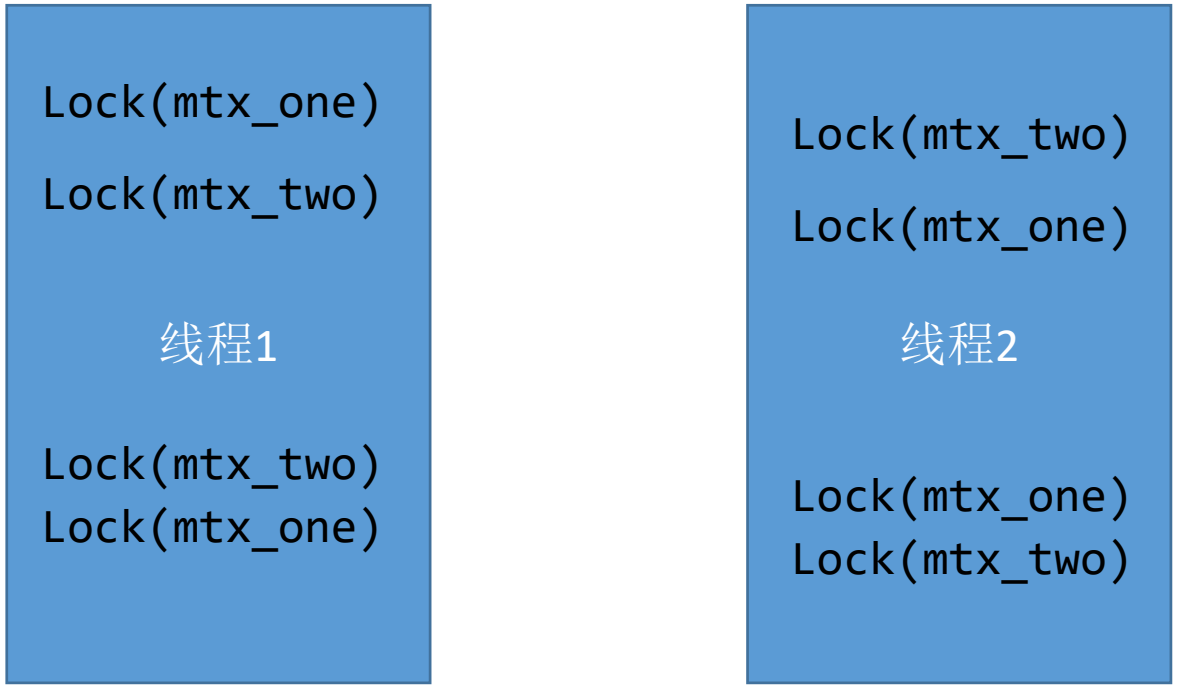
课程提纲

- 思考题
- 原子操作
- 无锁CAS
- **Volatile**
- 死锁
- Epoll实现分析

课程提纲

- 思考题
- 无锁CAS
- Volatile
- 死锁
- Epoll实现分析

死锁


Lock(mtx_one)
Lock(mtx_two)
线程1
Lock(mtx_two)
Lock(mtx_one)
Lock(mtx_two)
Lock(mtx_one)
线程2
Lock(mtx_one)
Lock(mtx_two)

零声学院

www.0voice.com

一切只为渴望更优秀的你！

非常感谢您的观看！