Xidian University

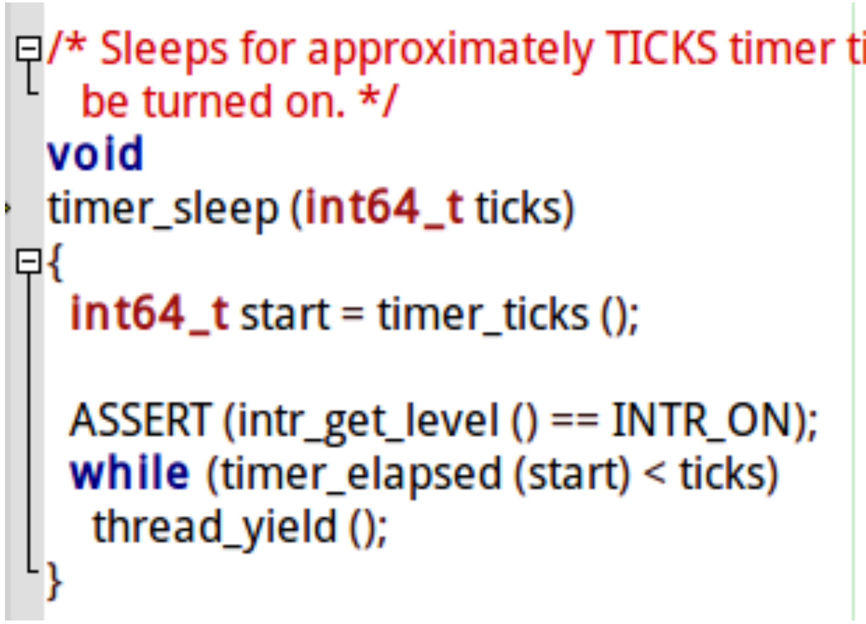
《操作系统课程设计》

Alarm-Clock之代码修改

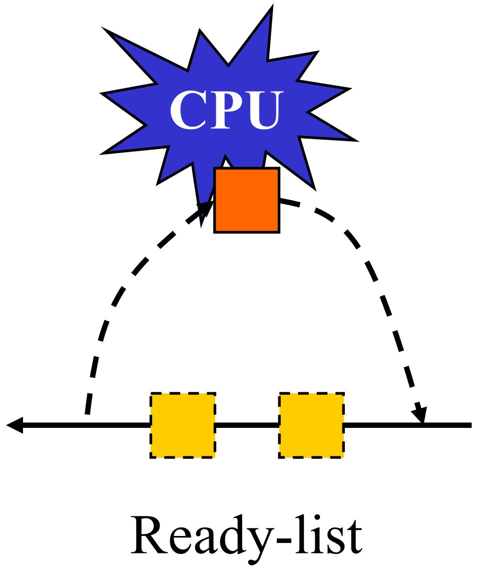
黄伯虎

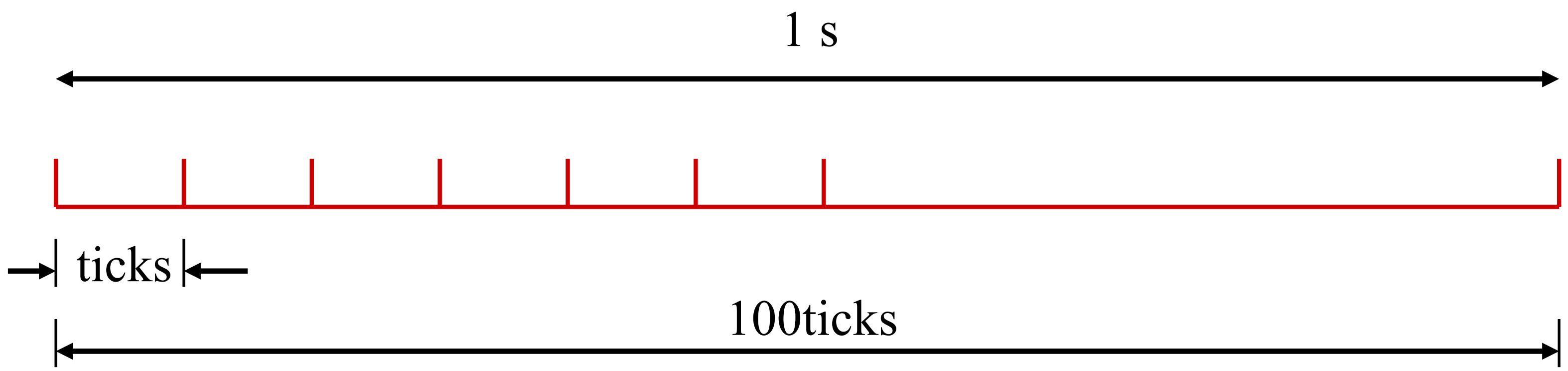
School of Computer Science & Technology

分析

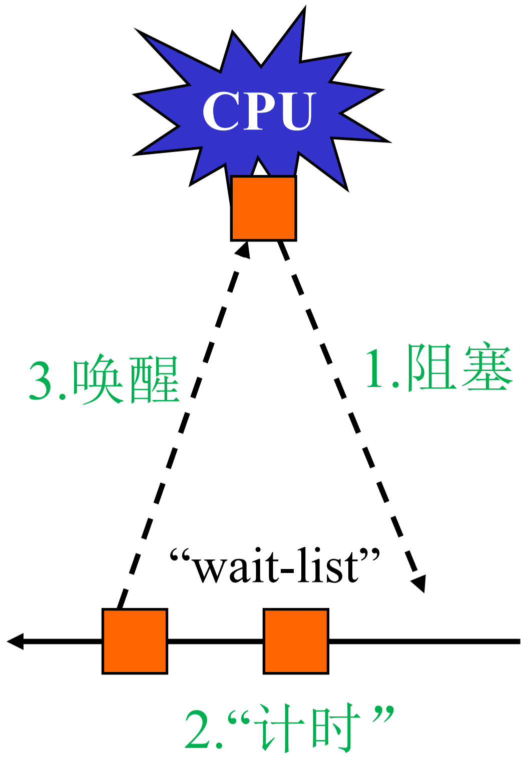


```
/* Sleeps for approximately TICKS timer ticks.  Interrupts must
   be turned on. */
void
timer_sleep (int64_t ticks)
{
  int64_t start = timer_ticks ();

  ASSERT (intr_get_level () == INTR_ON);
  while (timer_elapsed (start) < ticks)
    thread_yield ();
}
```





一种策略



解决方案

1.阻塞线程

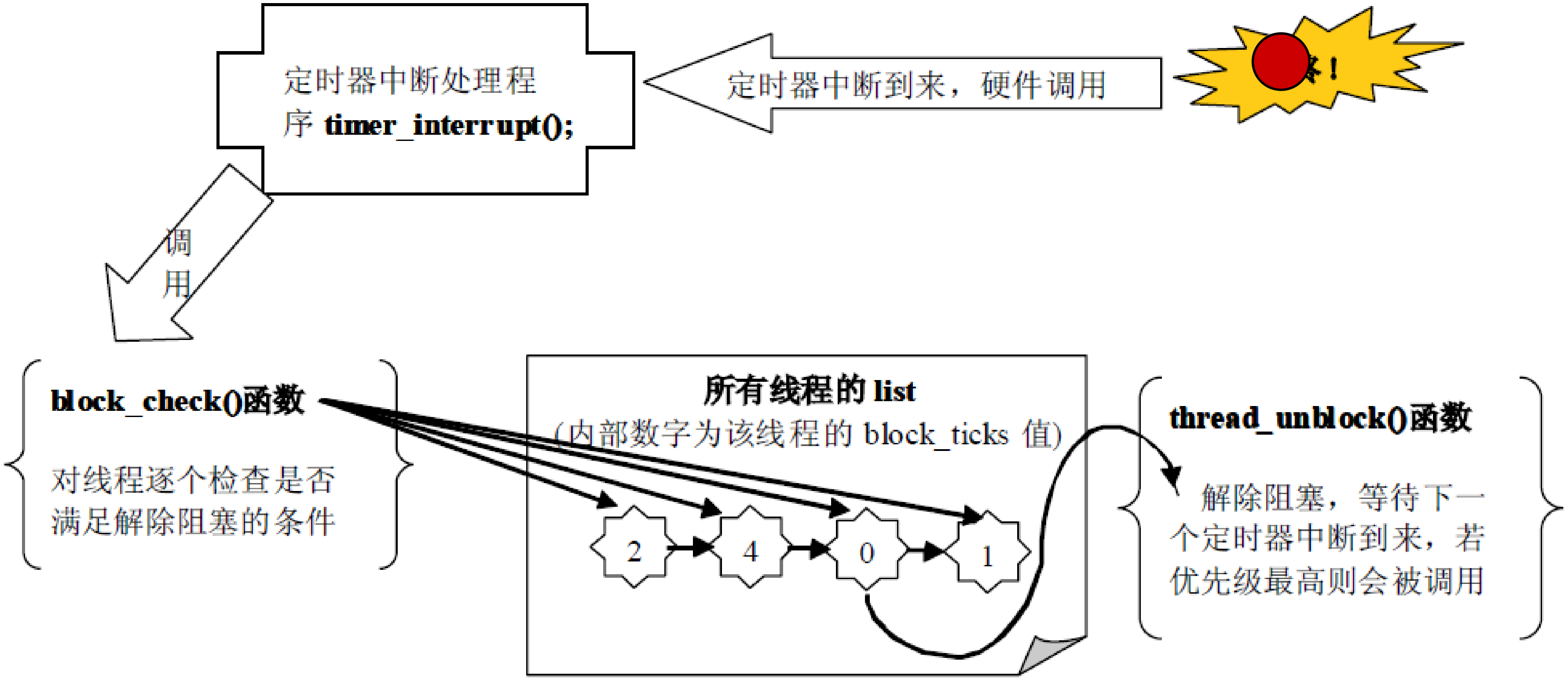
thread_block()

```
timer_sleep (int64_t ticks)
{
 int64_t start = timer_ticks ();

 ASSERT (intr_get_level () == INTR_ON);

 while (timer_elapsed (start) < ticks)
   thread_yield ();
}
```

```
timer_sleep (int64_t ticks)
{
 if (ticks > 0)
 {
     enum intr_level old_level;

     old_level = intr_disable();
     thread_block();     //block thread
     intr_set_level(old_level);
  }
}
```

解决方案

2.计时



定时器中断处理程序 timer_interrupt();
定时器中断到来，硬件调用
调用
block_check()函数
对线程逐个检查是否满足解除阻塞的条件
所有线程的 list
(内部数字为该线程的 block_ticks 值)
2
4
0
1
thread_unblock()函数
解除阻塞，等待下一个定时器中断到来，若优先级最高则会被调用

解决方案

（1）改造thread结构体(thread.h)

```
struct thread
  {
   /* Owned by thread.c. */
   tid_t tid;                          /* Thread identifier. */
   enum thread_status status;          /* Thread state. */
   char name[16];                      /* Name (for debugging purposes). */
   uint8_t *stack;                     /* Saved stack pointer. */
   int priority;                       /* Priority. */
   struct list_elem allelem;           /* List element for all threads list. */
   /* Shared between thread.c and synch.c. */
   struct list_elem elem;              /* List element. */

  int block_ticks;                     /* 线程阻塞时间*/

  ……
};
```

解决方案

```
timer_sleep (int64_t ticks)
{
 if (ticks > 0)
 {
     enum intr_level old_level;

     struct thread *t;
     t=thread_current ();
     t->block_ticks = ticks;   //记录阻塞时间

     old_level = intr_disable();
     thread_block();     //block thread
     intr_set_level(old_level);
  }
}
```

解决方案

(2)定义block_check()

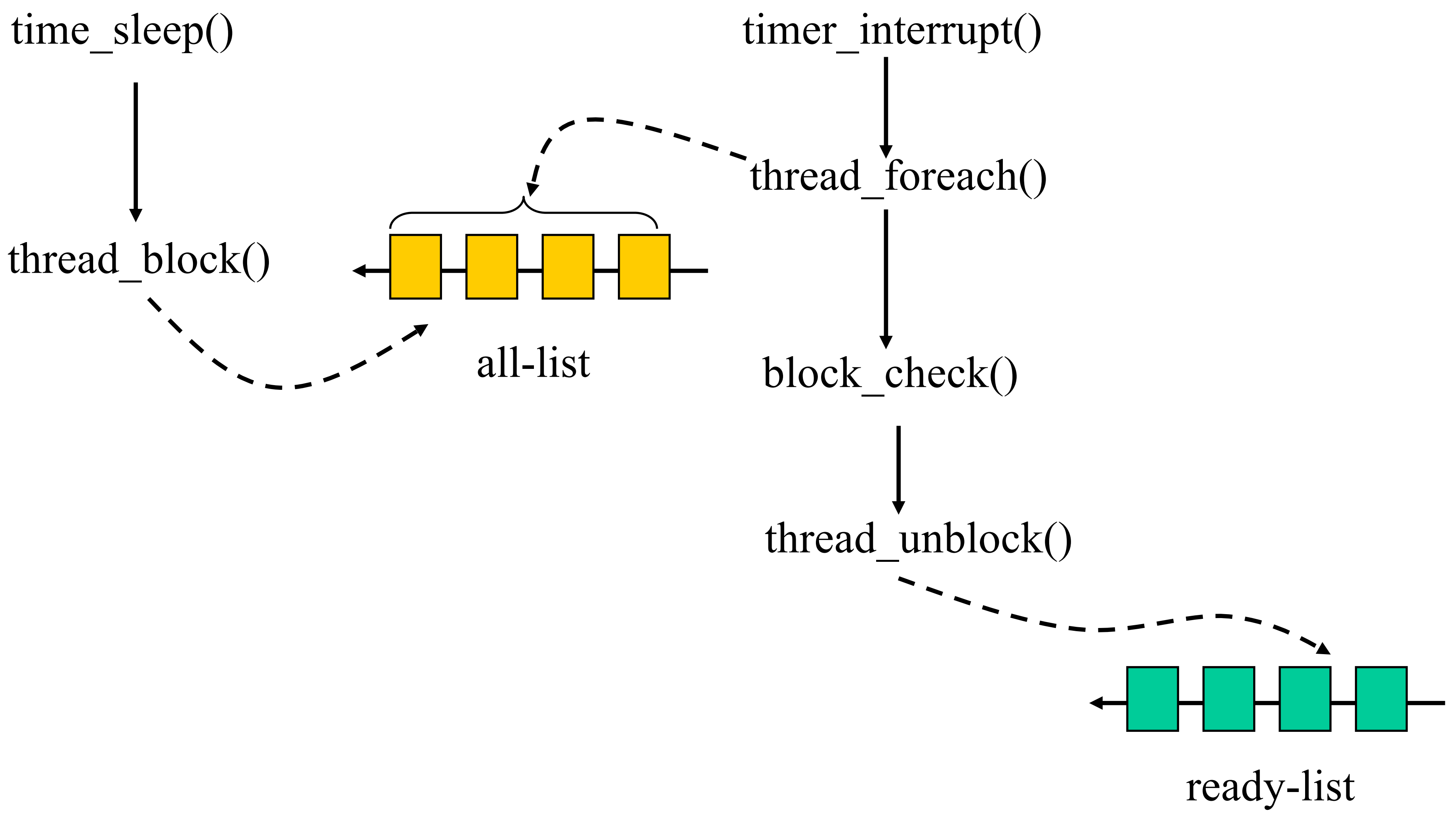
```
void block_check(struct thread *t, void *aux UNUSED)
{
    if (t->status == THREAD_BLOCKED && t->block_ticks>0)
    {
        t->block_ticks--;
        if (t->block_ticks == 0)
        {
            thread_unblock (t);
        }
    }
}
```

解决方案

调用block_check()

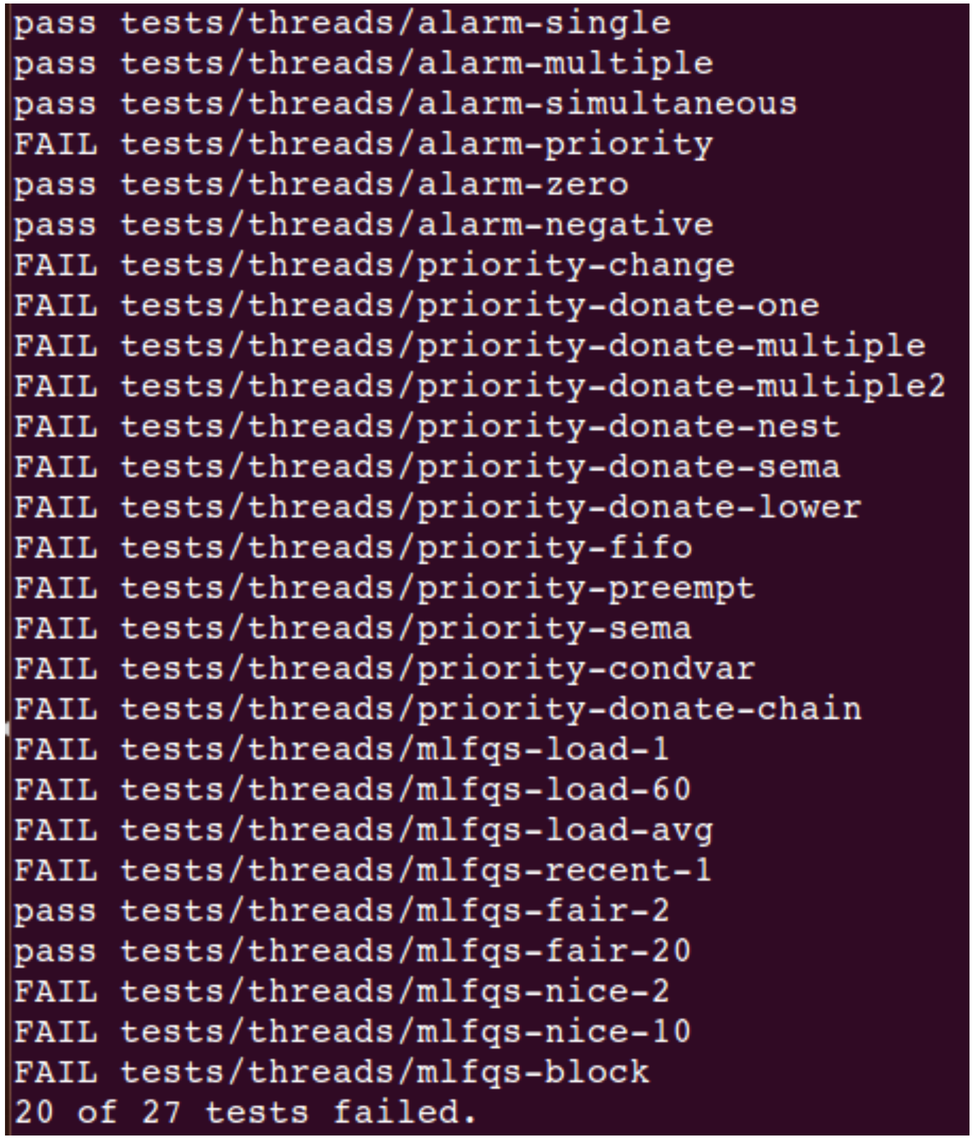
```
timer_interrupt (struct intr_frame *args UNUSED)
{
    ticks++;

    enum intr_level old_level;   //记录原来的中断状态
    old_level=intr_disable();    //thread_foreach函数要求关中断。
    thread_foreach (block_check, NULL);
    intr_set_level (old_level);  //恢复中断

    thread_tick ();
}
```

调用关系总结


time_sleep()
thread_block()
all-list
timer_interrupt()
thread_foreach()
block_check()
thread_unblock()
ready-list

测试

- 进入
  - ../printos/src/threads/目录，运行#make check命令，会有如下5个相关检测通过:
    - **Alarm-single**
    - **Alarm-multiple**
    - **Alarm-simultaneous**
    - **Alarm-zero**
    - **Alarm-negative**



```
pass tests/threads/alarm-single
pass tests/threads/alarm-multiple
pass tests/threads/alarm-simultaneous
FAIL tests/threads/alarm-priority
pass tests/threads/alarm-zero
pass tests/threads/alarm-negative
FAIL tests/threads/priority-change
FAIL tests/threads/priority-donate-one
FAIL tests/threads/priority-donate-multiple
FAIL tests/threads/priority-donate-multiple2
FAIL tests/threads/priority-donate-nest
FAIL tests/threads/priority-donate-sema
FAIL tests/threads/priority-donate-lower
FAIL tests/threads/priority-fifo
FAIL tests/threads/priority-preempt
FAIL tests/threads/priority-sema
FAIL tests/threads/priority-condvar
FAIL tests/threads/priority-donate-chain
FAIL tests/threads/mlfqs-load-1
FAIL tests/threads/mlfqs-load-60
FAIL tests/threads/mlfqs-load-avg
FAIL tests/threads/mlfqs-recent-1
pass tests/threads/mlfqs-fair-2
pass tests/threads/mlfqs-fair-20
FAIL tests/threads/mlfqs-nice-2
FAIL tests/threads/mlfqs-nice-10
FAIL tests/threads/mlfqs-block
20 of 27 tests failed.
```