# 实验目的：

重新实现timer\_sleep函数，避免忙则等待情况的发生。

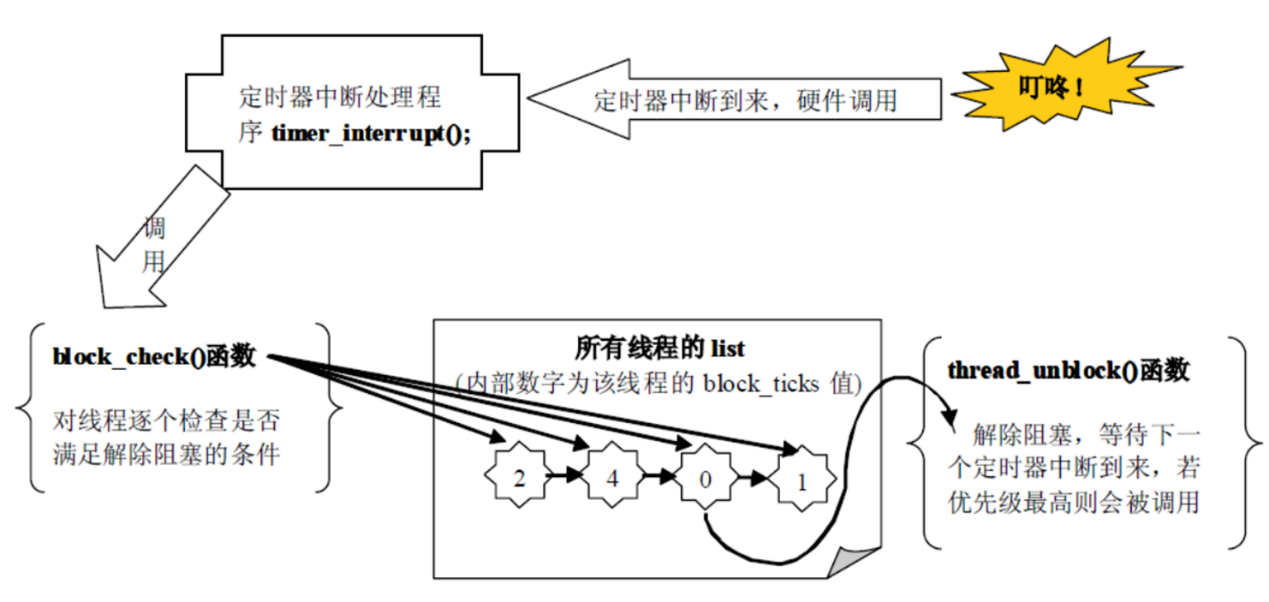
# 实验原理：

# 1.考虑在timer sleep()函数中让线程暂时阻塞(thread block()) ，然后等待 ticks个定时器中断周期结束以后再去唤醒它(thread\_ unblock())。

2.“等”不再使用循环来实现，而是在每个定时器中断到来后去检查一下wait list中的线程，看是否已经到了唤醒的时机。如果时机已到，马上将其唤醒进入 ready\_list()；如反之则不做任何操作。

3.没有了循环调用，其他线程就可以充分在CPU上运行。

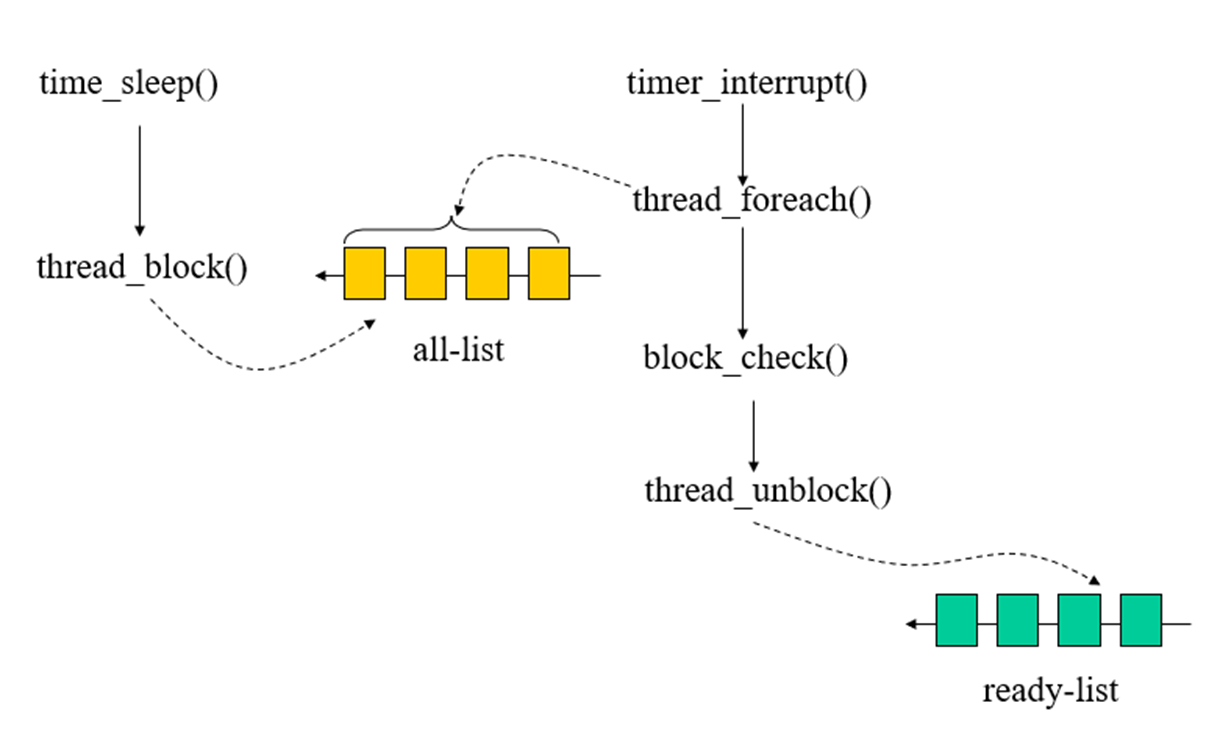
# 基本原理图：



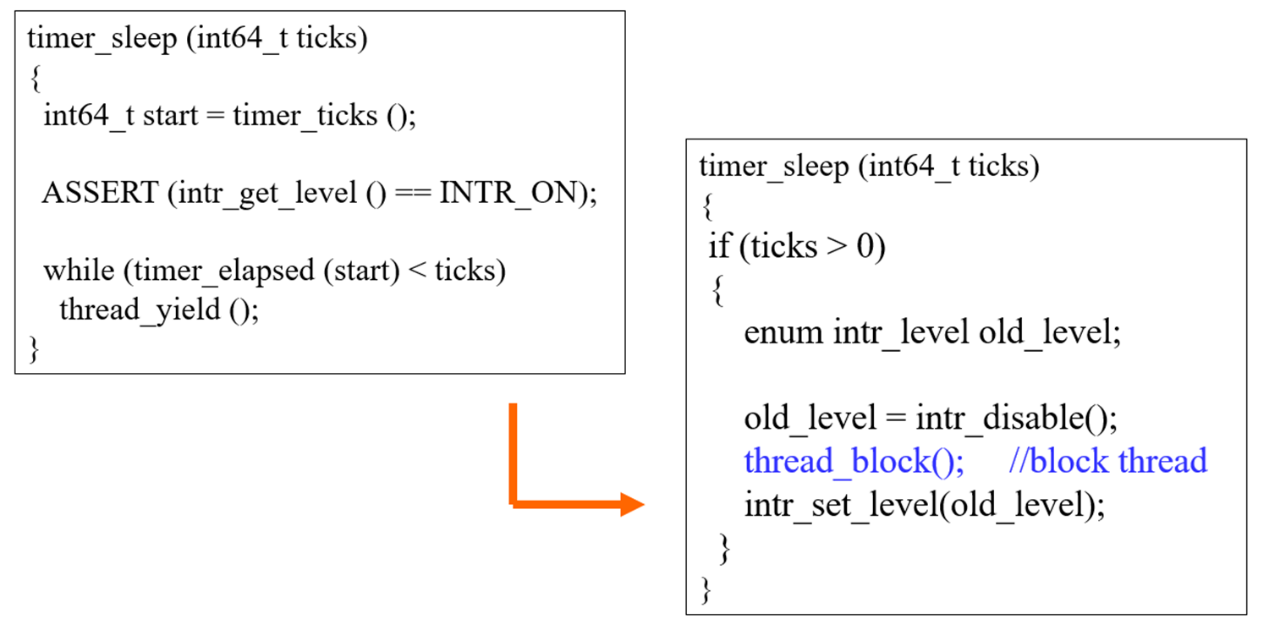
# 实验方案：

首先要在timer\_sleep函数中将线程阻塞，在线程结构体中添加表示阻塞时间的变量。interrupt作为定时器中断，每当经过一个时间片的时间，用foreach函数遍历阻塞队列，对各个阻塞线程执行check函数。如果其休眠时间结束，用unblock函数将其加入就绪队列，否则将其所需休眠的时间片减少1。

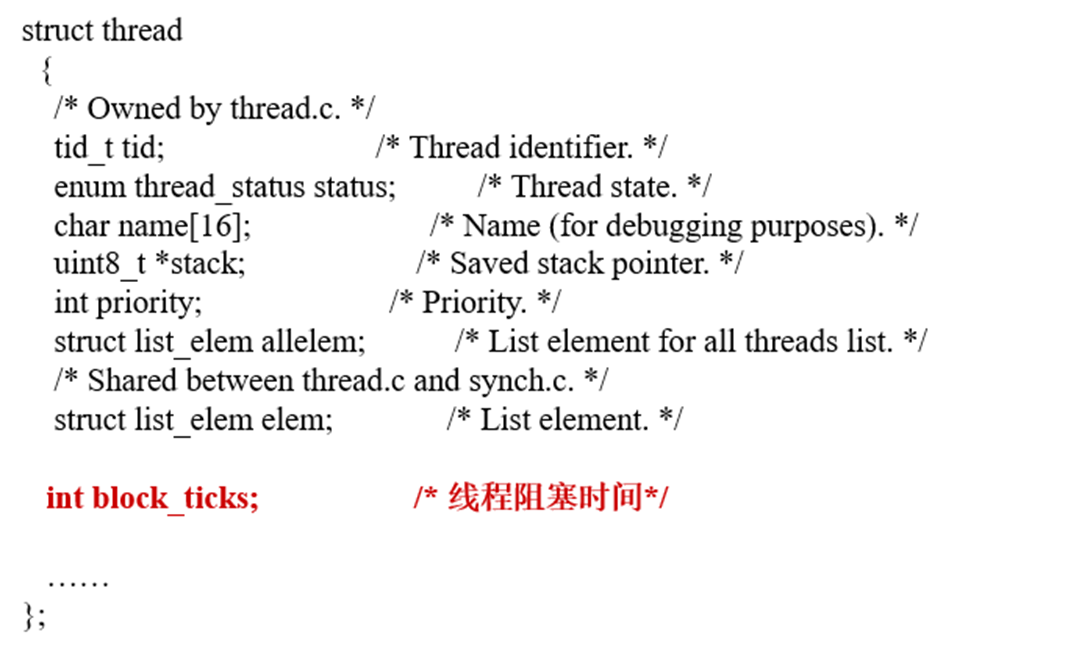
# 调用关系总结：



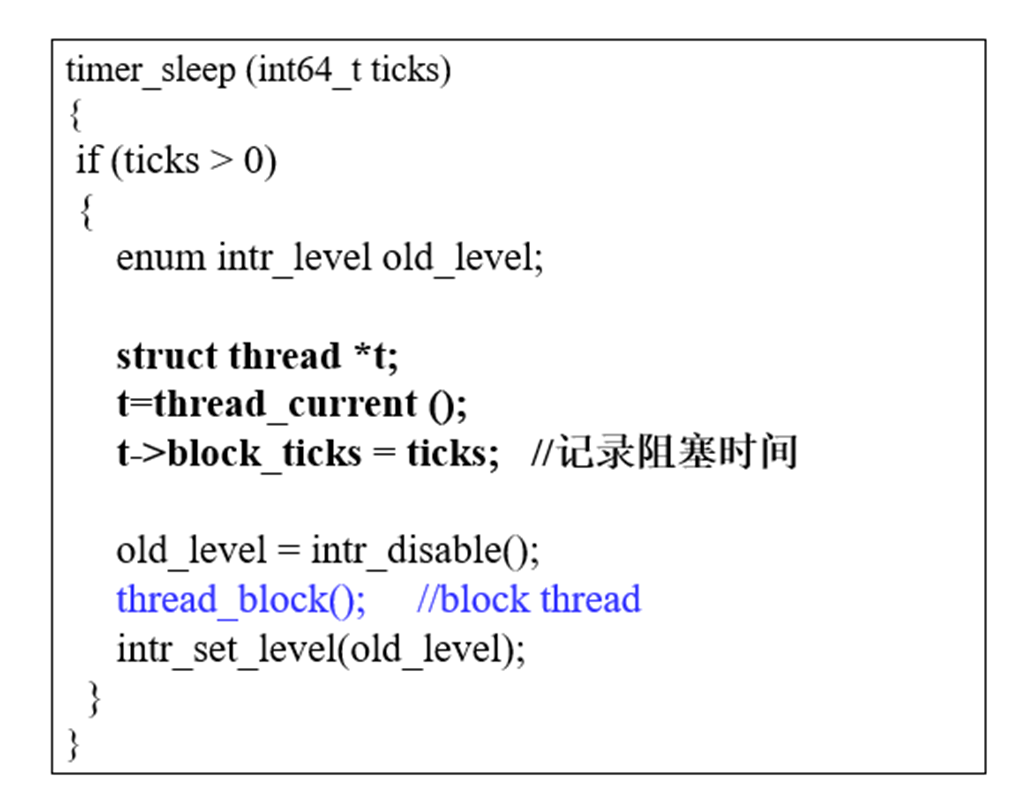
# 具体代码讲解：



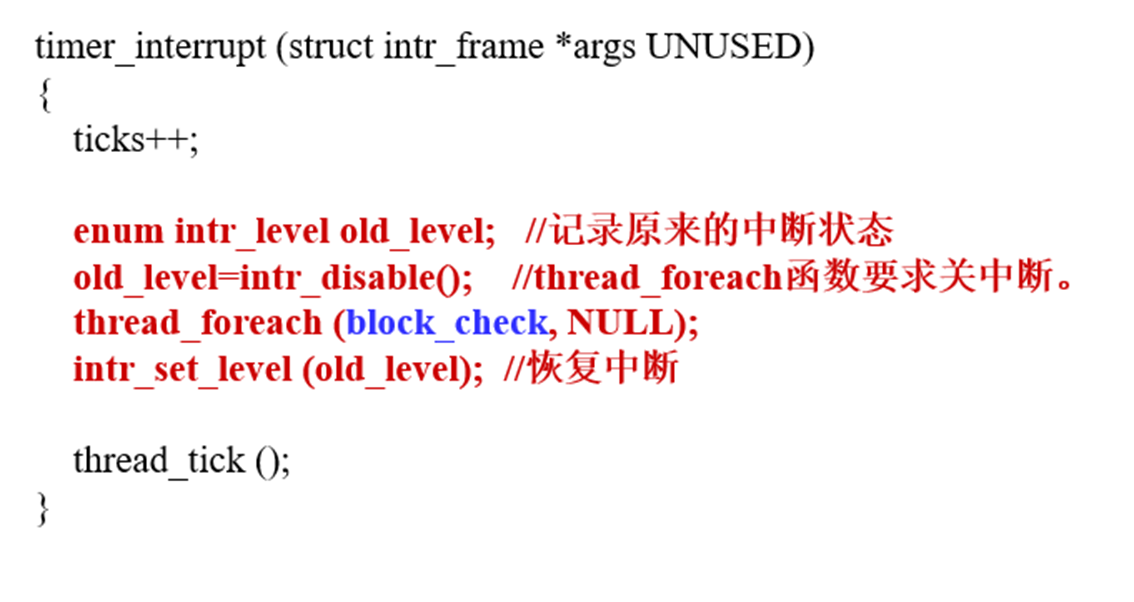
这一步将线程直接阻塞，是避免忙等待的关键。如果不将其阻塞，while循环将会一直执行，白白浪费CPU资源。



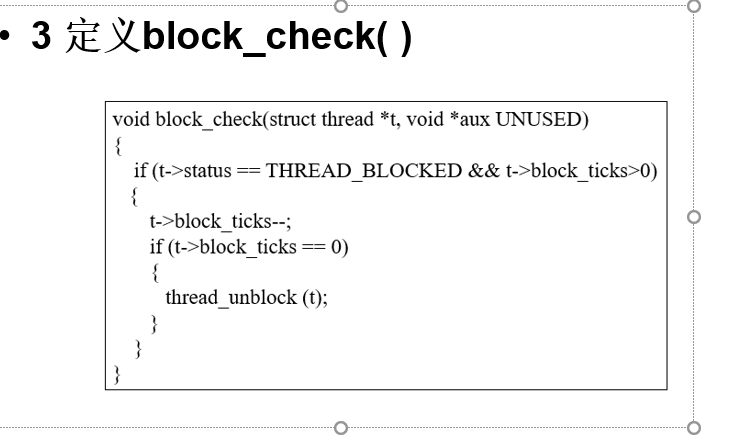
改造线程结构体，加入阻塞时间一项



在休眠函数中创建线程结构体指针，指向当前线程，并将其休眠时间赋值

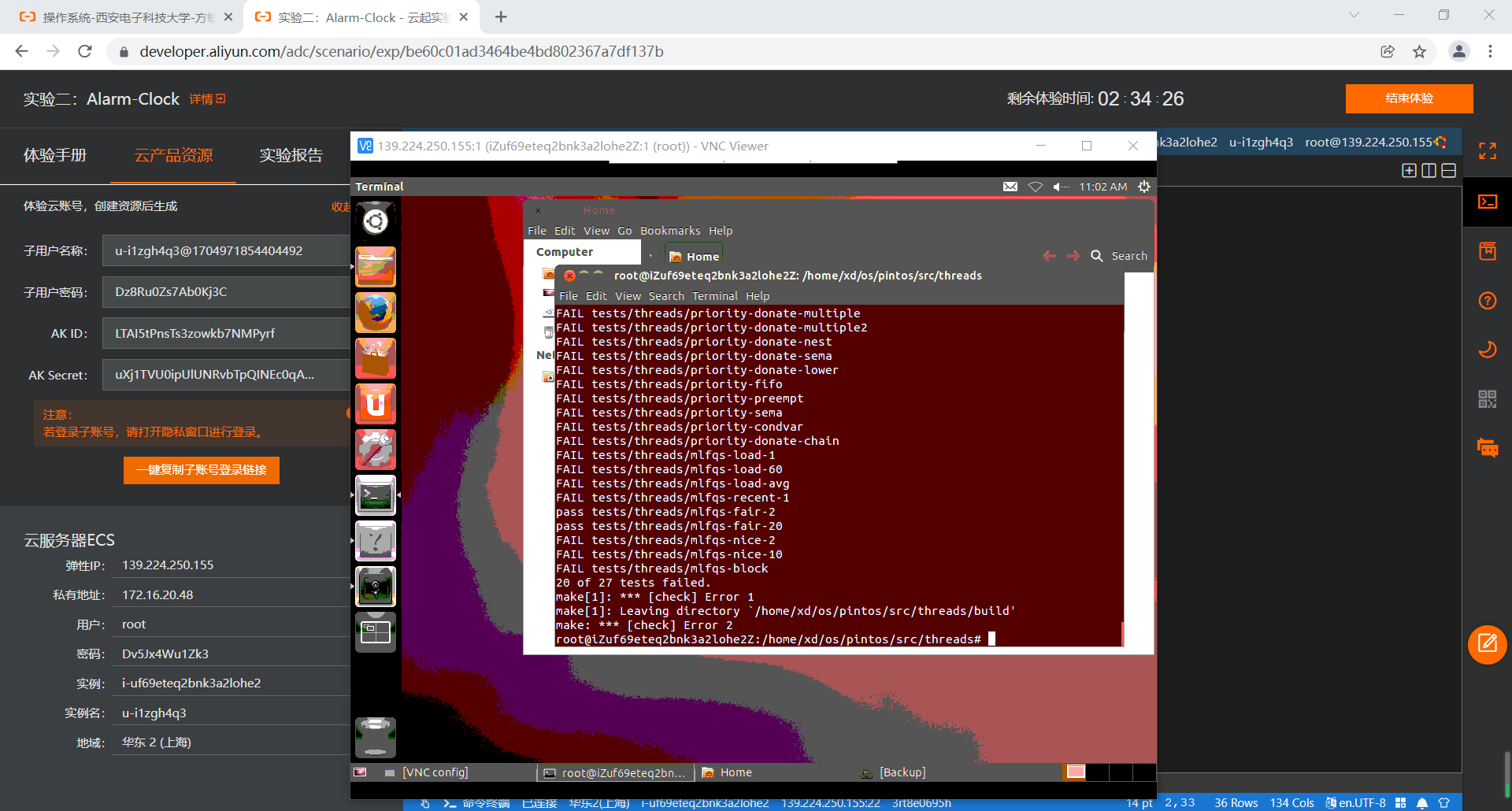


改造interrupt函数，让其调用foreach去查看每个阻塞线程。



改写check函数，如果剩余时间为零，调用unblock函数使其进入就绪队列，否则使其剩余休眠时间片减少1

# 实验结果：

有七个测试点通过