实验报告：pintos 实现优先级线程

课程名称：操作系统	**年级**：2023 级	**上机实践成绩**：
指导教师：张民	**姓名**：张建夫	
上机实践名称：**pintos 实现优先级线程**	**学号**：10235101477	**上机实践日期**：2024/10/28
上机实践编号：	**组号**：	**上机实践时间**：14:50~16:30 点

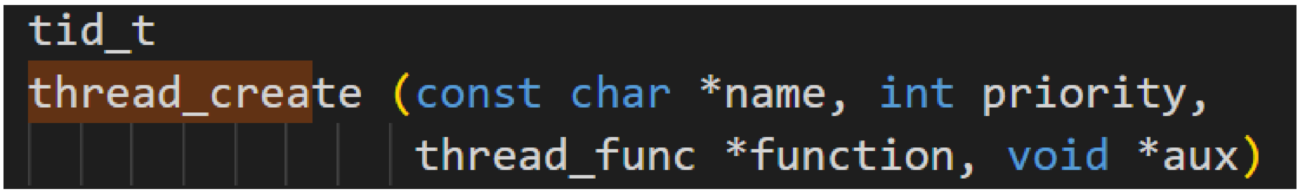
一、目的

明白线程之间有优先级，在线程创建并加入就绪队列时根据该线程的优先级加入，时了解测试在 pintos 中运行的整个流程。

二、内容与设计思想

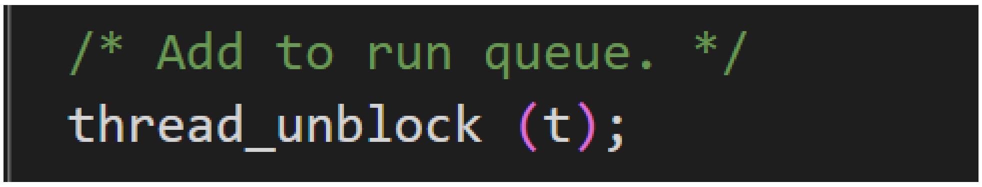
在 thread.c 文件中加入适当代码，使就绪队列的线程拥有优先级，且要实现优先级较高的线程优先调度，最后通过 alarm-priority 测试用例。

对于一个创建的进程（thread_create 函数），会有一个分配的优先级：



```
tid_t
thread_create (const char *name, int priority,
               thread_func *function, void *aux)
```

在该函数的最后，会将创建的进程加入就绪队列：



```
  /* Add to run queue. */
  thread_unblock (t);
```

在该函数中，会将该线程加入到就绪队列，但是线程加入的方式是直接将线程推入就绪队列，也就是说，优先级的传入并没有用，需要额外加入机制使优先级生效。对于 pintos 的线程系统，pintos 需要根据新线程的的优先级将新线程加入就绪队列（包括线程调度时,例如时间片用完，调用 thread_yield 函数），而且，pintos 是直接将就绪队列的首节点出队作为下一个要运行的线程，因此，就绪队列需要按照优先级顺序将线程排序，优先级高的在就绪队列前面，实现这一点，只要在每次有线程加入就绪队列时用插入排序的方式（找到第一个优先级小于要加入线程的优先级，将线程加到这个线程前面）即可。

三、使用环境

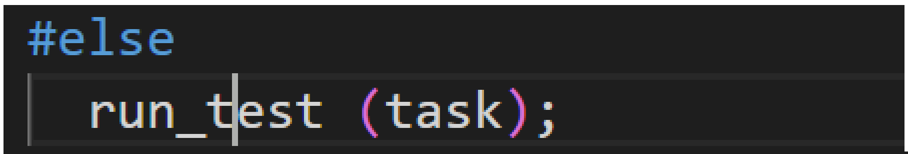
1. 主机 os：Windows-11
2. 虚拟机 os:Linux
3. 虚拟环境：docker 的 linux 镜像

4．代码编辑器:vscode

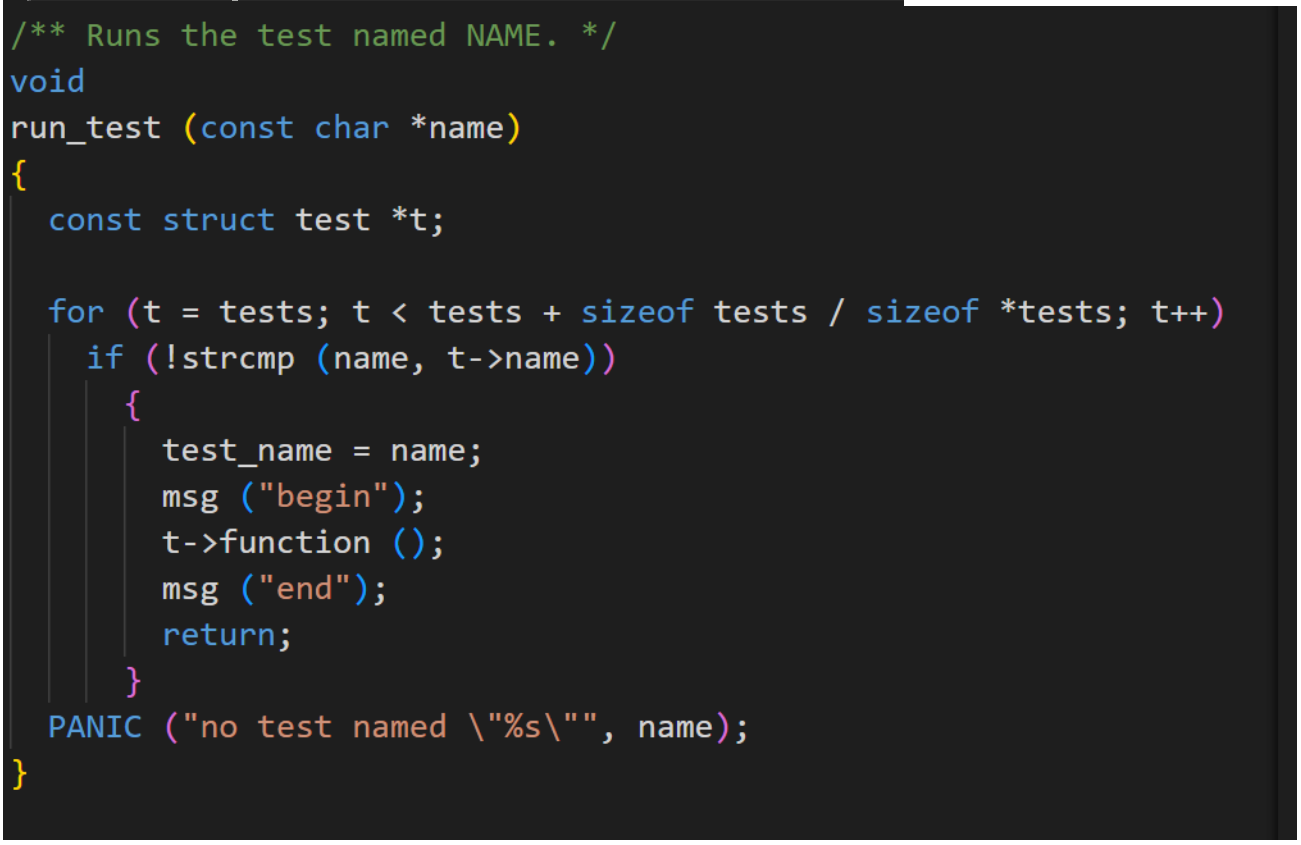
四、实验过程

1．了解测试用例的运行过程：

在 pintos 完成加载后，如果命令行有参数传进（即传入被测试程序），pintos_init 函数会调用 run_actions，根据传入的命令（例如 run）执行对应操作，对于用户程序，一般是 run。对于 run，run_actions 会调用 run_task，由于该用例不是 userprog，run_task 会调用 run_test，选取相应的测试案例运行：

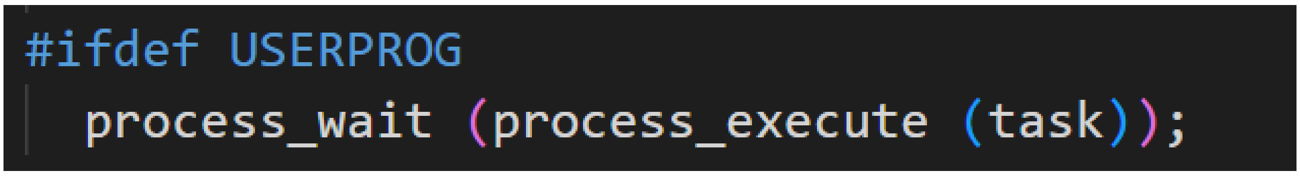


```
#else
  run_test (task);
```



```
/** Runs the test named NAME. */
void
run_test (const char *name)
{
  const struct test *t;

  for (t = tests; t < tests + sizeof tests / sizeof *tests; t++)
    if (!strcmp (name, t->name))
      {
        test_name = name;
        msg ("begin");
        t->function ();
        msg ("end");
        return;
      }
  PANIC ("no test named \"%s\"", name);
}
```

如果是用户程序，则接着会调用：

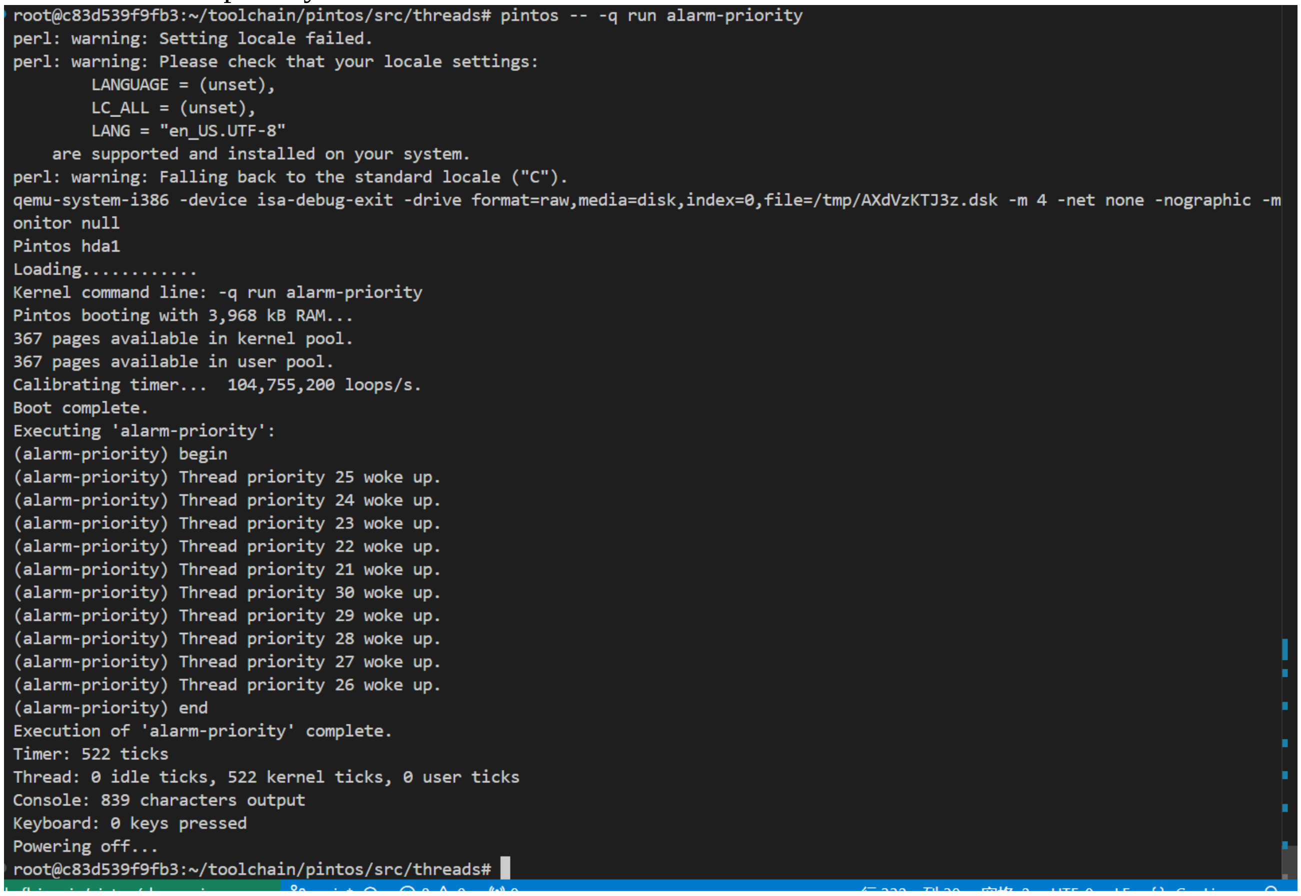


```
#ifdef USERPROG
  process_wait (process_execute (task));
```

process_execute 会创建一个子线程（pintos 中并没有进程的概念，pintos 中任何进程都是由线程模拟的），创建完成后，这个新子线程会运行 start_process 函数，设置该子进程相应堆栈信息（例如设置栈指针，将命令行参数推入栈等），从子进程调用 start_process 开始，子进程和创建它的进程就开始并发进行（尽管这里调用了 process_wait,但实际上这个函数尚未实现，要实现了 wait 系统调用才有用），start_process 中通过模拟从中断返回来执行正式用户进程（即自己写的程序）。在用户程序运行完之后，会调用 process_exit,结束进程。

2．实现线程优先级调度：

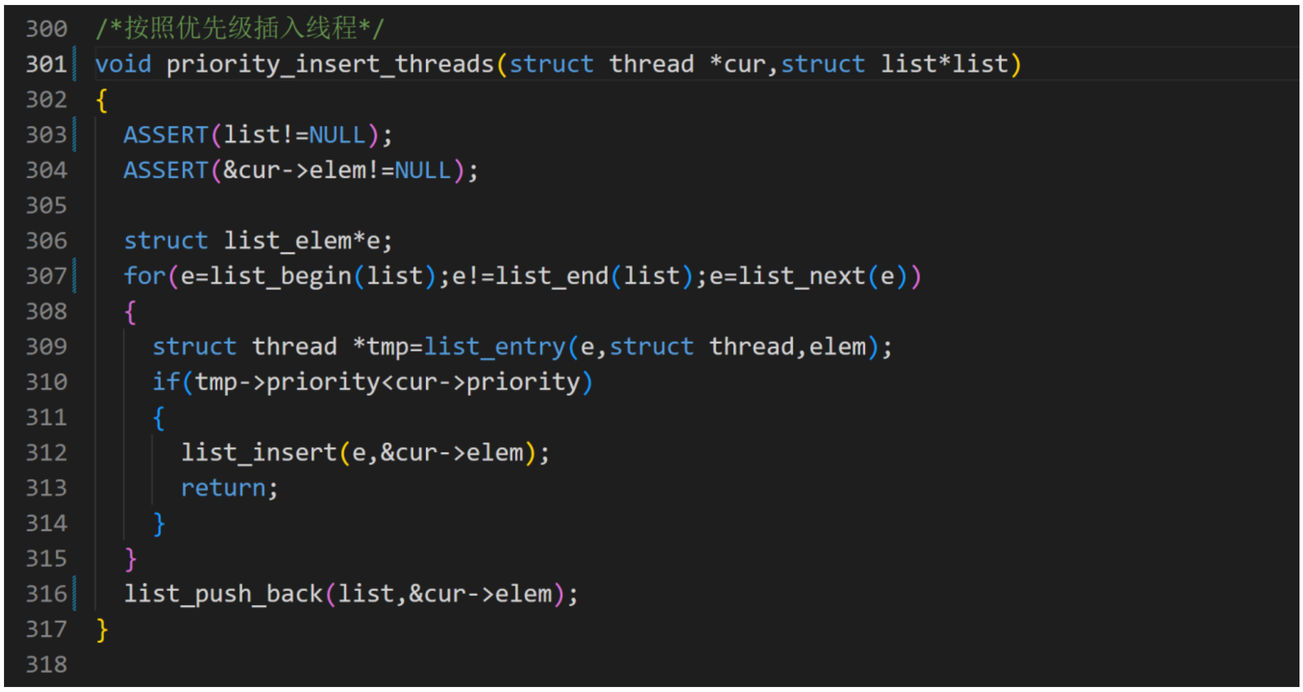
先跑一下 alarm-priority：



```
root@c83d539f9fb3:~/toolchain/pintos/src/threads# pintos -- -q run alarm-priority
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
qemu-system-i386 -device isa-debug-exit -drive format=raw,media=disk,index=0,file=/tmp/AXdVzKTJ3z.dsk -m 4 -net none -nographic -m
onitor null
Pintos hda1
Loading...........
Kernel command line: -q run alarm-priority
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  104,755,200 loops/s.
Boot complete.
Executing 'alarm-priority':
(alarm-priority) begin
(alarm-priority) Thread priority 25 woke up.
(alarm-priority) Thread priority 24 woke up.
(alarm-priority) Thread priority 23 woke up.
(alarm-priority) Thread priority 22 woke up.
(alarm-priority) Thread priority 21 woke up.
(alarm-priority) Thread priority 30 woke up.
(alarm-priority) Thread priority 29 woke up.
(alarm-priority) Thread priority 28 woke up.
(alarm-priority) Thread priority 27 woke up.
(alarm-priority) Thread priority 26 woke up.
(alarm-priority) end
Execution of 'alarm-priority' complete.
Timer: 522 ticks
Thread: 0 idle ticks, 522 kernel ticks, 0 user ticks
Console: 839 characters output
Keyboard: 0 keys pressed
Powering off...
root@c83d539f9fb3:~/toolchain/pintos/src/threads# 
```

线程唤醒顺序不对，应该按照优先级唤醒。

因此，我们先将根据优先级加入就绪队列的实现方式（第二部分已说明实现的方式）封装成一个函数，如下：

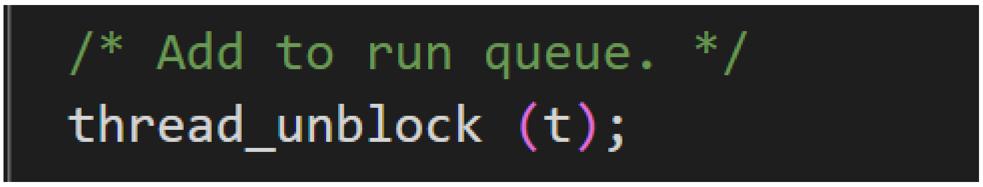


```c
/*按照优先级插入线程*/
void priority_insert_threads(struct thread *cur,struct list*list)
{
  ASSERT(list!=NULL);
  ASSERT(&cur->elem!=NULL);

  struct list_elem*e;
  for(e=list_begin(list);e!=list_end(list);e=list_next(e))
  {
    struct thread *tmp=list_entry(e,struct thread,elem);
    if(tmp->priority<cur->priority)
    {
      list_insert(e,&cur->elem);
      return;
    }
  }
  list_push_back(list,&cur->elem);
}

```

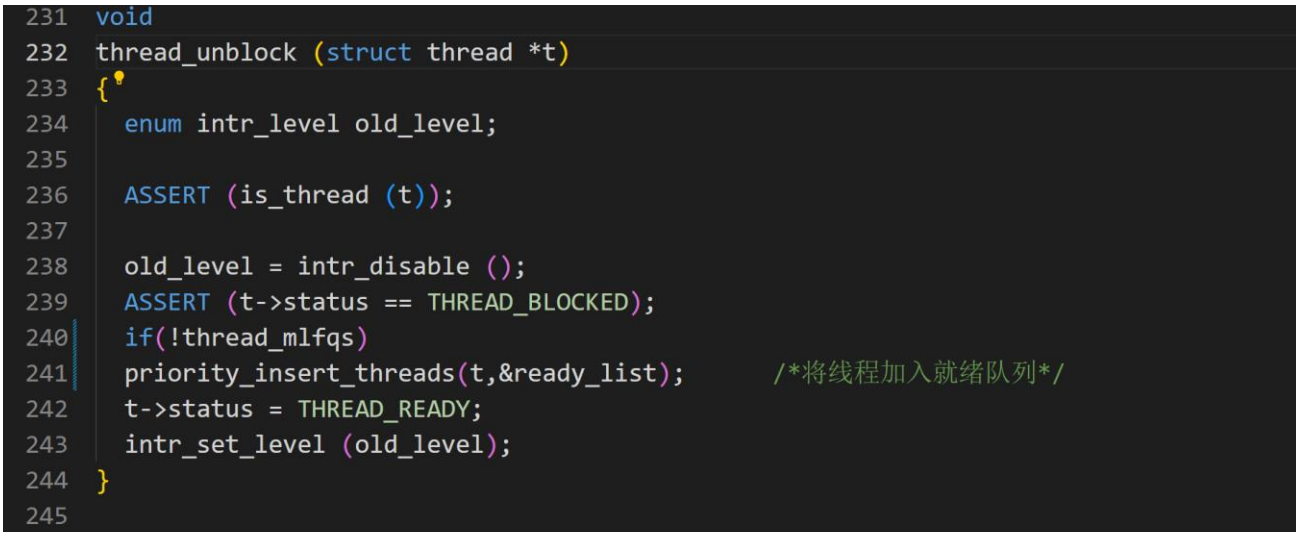
该函数将一个线程加入就绪队列（我没使用 ppt 的函数， ppt 里的那个线程优先级比

较函数实在有点“脱裤子放屁”，而且那个比较函数只能比较线程的优先级，无法在其他地方复用）

接下来，我们在需要将线程加入就绪队列的地方调用这个函数，一个是创建线程的时候：

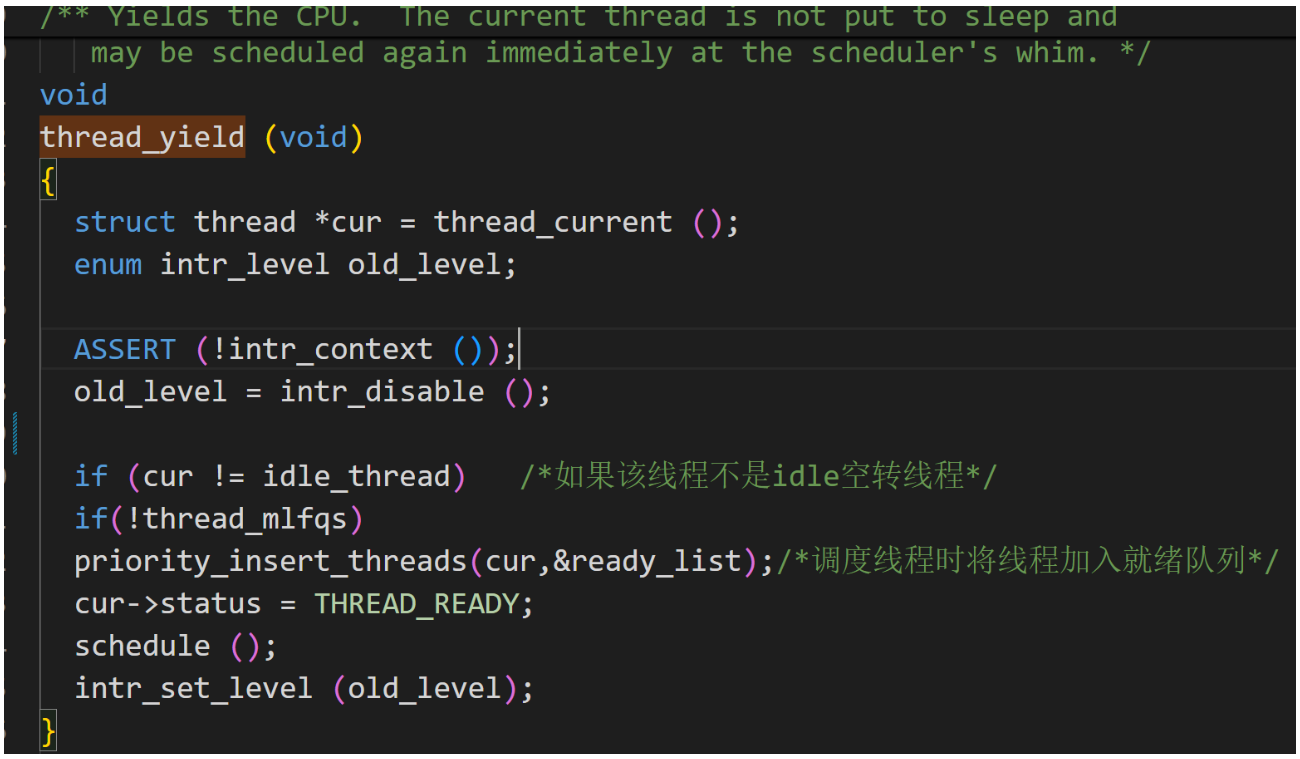


```
/* Add to run queue. */
thread_unblock (t);
```



```
void
thread_unblock (struct thread *t)
{
  enum intr_level old_level;

  ASSERT (is_thread (t));

  old_level = intr_disable ();
  ASSERT (t->status == THREAD_BLOCKED);
  if(!thread_mlfqs)
  priority_insert_threads(t,&ready_list);      /*将线程加入就绪队列*/
  t->status = THREAD_READY;
  intr_set_level (old_level);
}
```

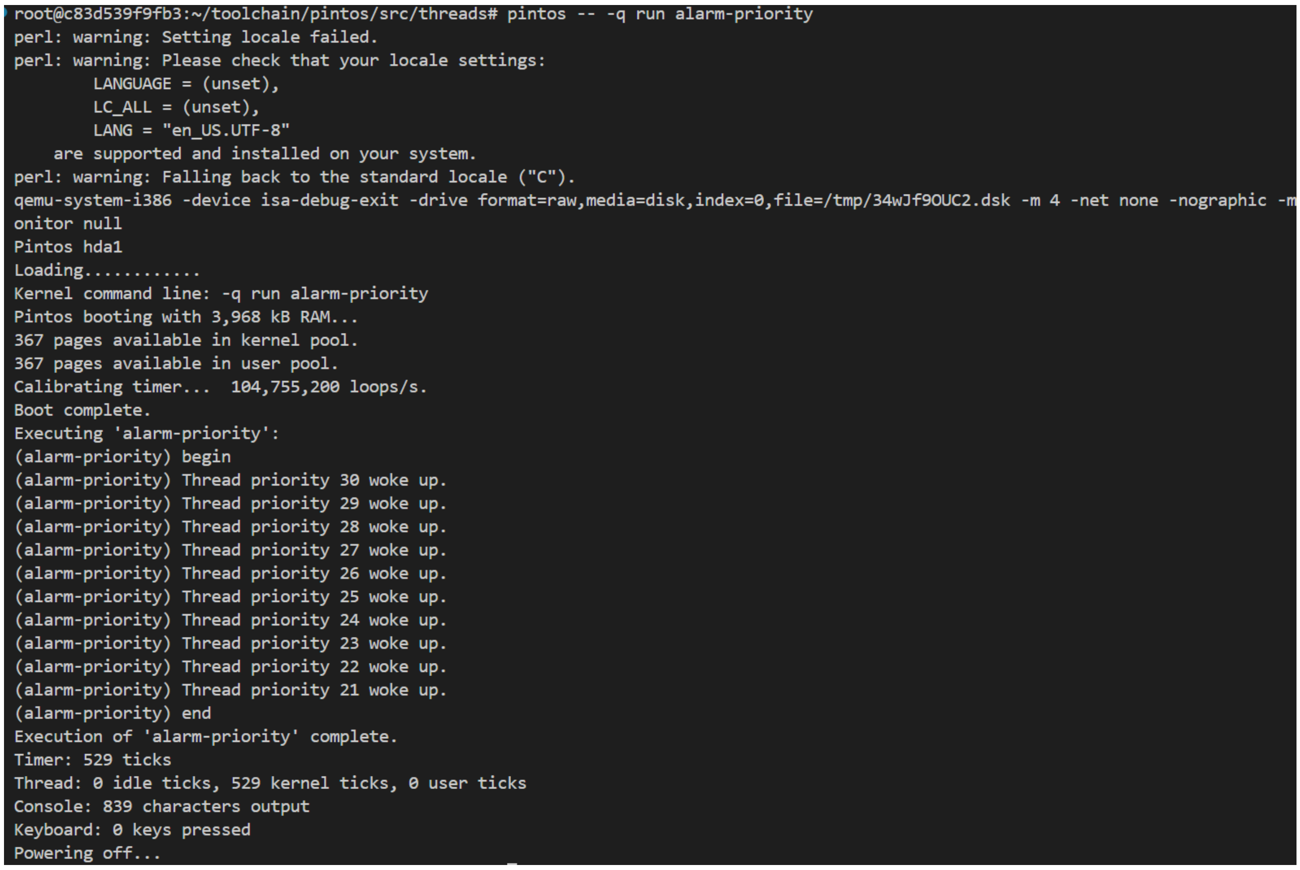
二是线程调度的时候：



```
/** Yields the CPU.  The current thread is not put to sleep and
   may be scheduled again immediately at the scheduler's whim. */
void
thread_yield (void)
{
  struct thread *cur = thread_current ();
  enum intr_level old_level;

  ASSERT (!intr_context ());
  old_level = intr_disable ();

  if (cur != idle_thread)   /*如果该线程不是idle空转线程*/
  if(!thread_mlfqs)
  priority_insert_threads(cur,&ready_list);/*调度线程时将线程加入就绪队列*/
  cur->status = THREAD_READY;
  schedule ();
  intr_set_level (old_level);
}
```

此处有个 if(!thread_mlfqs)的原因是方便后面实验多级队列的线程调度实现，暂且不谈

接着再跑一下 alarm-priority 用例：



```
root@c83d539f9fb3:~/toolchain/pintos/src/threads# pintos -- -q run alarm-priority
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
qemu-system-i386 -device isa-debug-exit -drive format=raw,media=disk,index=0,file=/tmp/34wJf9OUC2.dsk -m 4 -net none -nographic -m
onitor null
Pintos hda1
Loading...........
Kernel command line: -q run alarm-priority
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  104,755,200 loops/s.
Boot complete.
Executing 'alarm-priority':
(alarm-priority) begin
(alarm-priority) Thread priority 30 woke up.
(alarm-priority) Thread priority 29 woke up.
(alarm-priority) Thread priority 28 woke up.
(alarm-priority) Thread priority 27 woke up.
(alarm-priority) Thread priority 26 woke up.
(alarm-priority) Thread priority 25 woke up.
(alarm-priority) Thread priority 24 woke up.
(alarm-priority) Thread priority 23 woke up.
(alarm-priority) Thread priority 22 woke up.
(alarm-priority) Thread priority 21 woke up.
(alarm-priority) end
Execution of 'alarm-priority' complete.
Timer: 529 ticks
Thread: 0 idle ticks, 529 kernel ticks, 0 user ticks
Console: 839 characters output
Keyboard: 0 keys pressed
Powering off...
```

线程按顺序唤醒，实验成功。

代码修改结果：https://github.com/saydontgo/school_OS_course/

五、总结

本次实验较为简单，只要理解整个线程创建过程和就绪队列加入的方式（按优先级插入）即可解决，除此之外，我知道了如何提高代码复用性，使代码能够解决不同队列的优先级插入问题，最后就是我认为可以加一些 pintos 中 list 库使用的作业，很多人要是没做过 cs162 或斯坦福的 os 课程的话都不知道这几个 list 相关的函数是怎么来的，更别说用了（虽然也给了具体实现）。