# **实验项目二：Nachos进程管理**

## **2.1实验目的：**

完成Nachos的进程管理模块的扩展，掌握操作系统中进程管理和进程调度扩展实现。

## **2.2实验环境：**

操作系统:linux

机器配置：4G运行内存 20G硬盘

编译器版本：gcc9.5

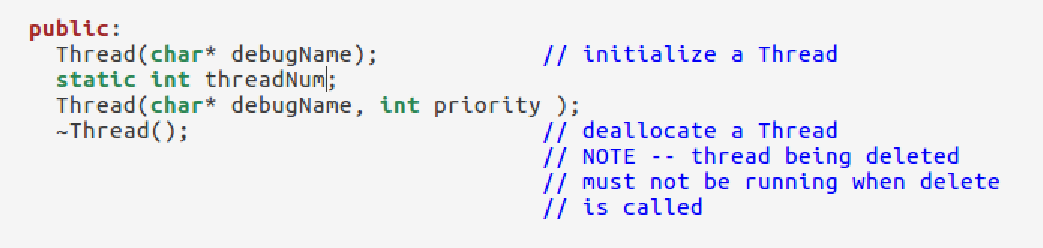
## **2.3实验内容：**

对实践过程的详细说明，如对Nachos平台的哪些代码进行了什么样的修改，采用了何种算法或思想等。

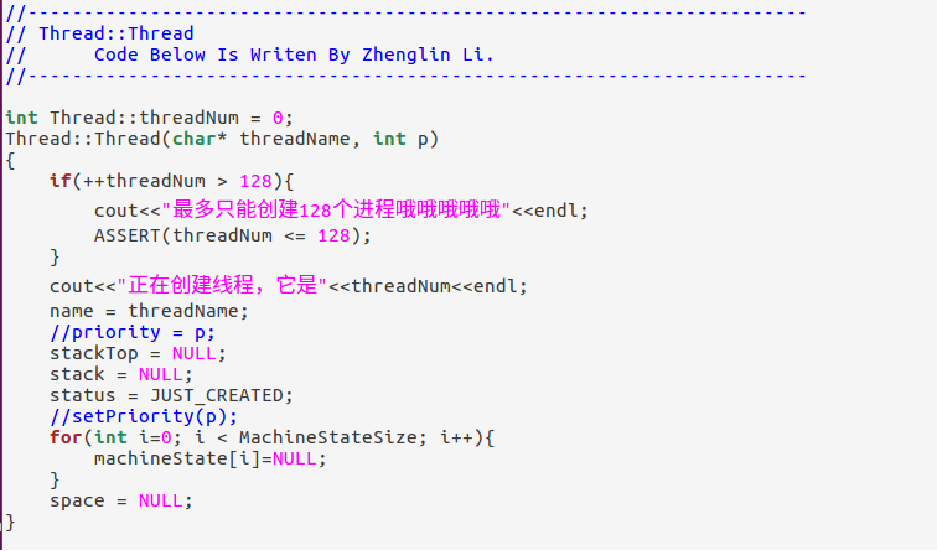
List核心的代码，如以文件为单位一一进行描述。

可以结合适当的流程图或者类图来辅助描述。

thread.h



thread.ccz中重载Thread::Thread函数



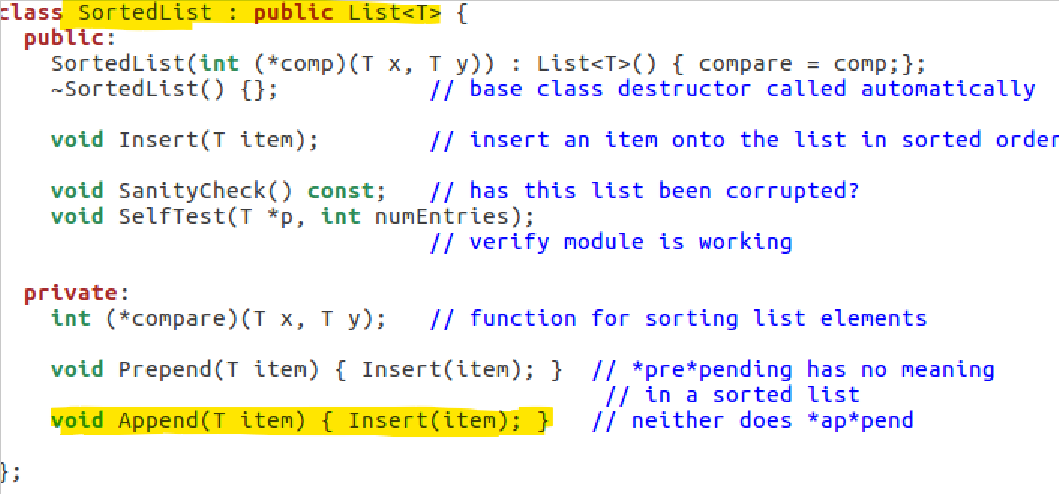
thread.cc中修改Thread::Yield（）函数



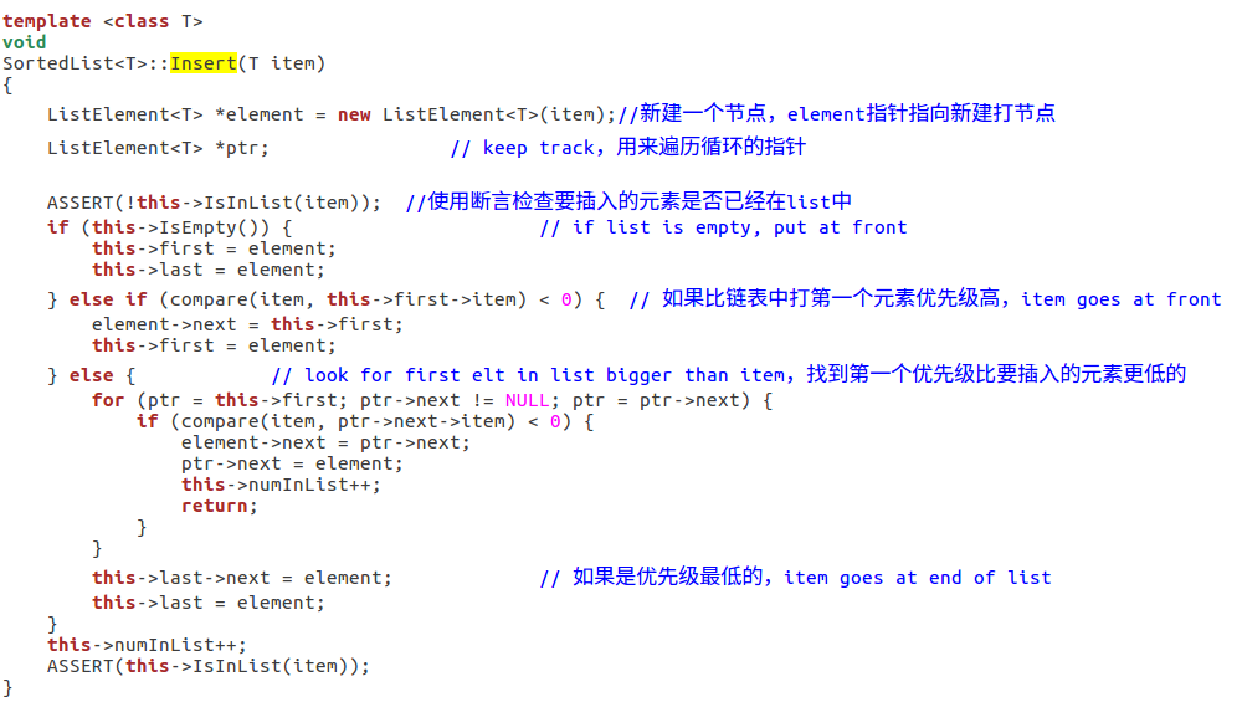
thread.cc中selfTest函数



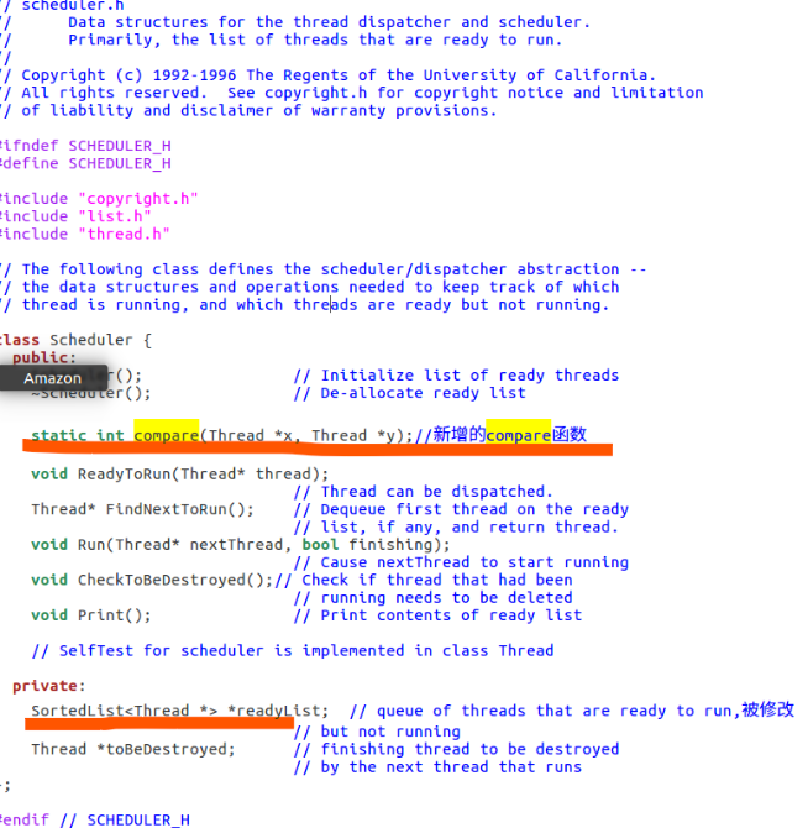
list.h中



list.cc中的insert函数



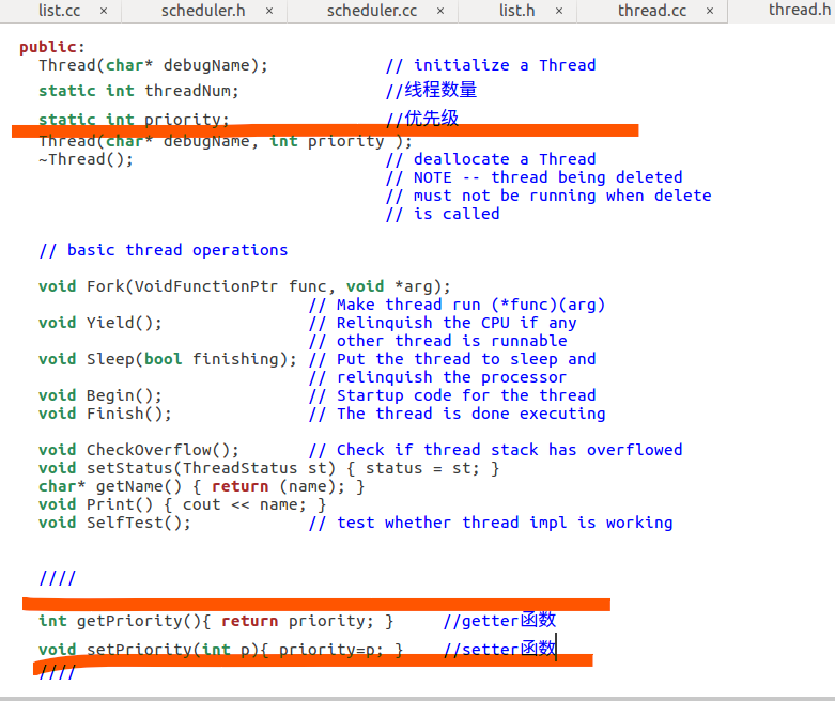
scheduler.h中



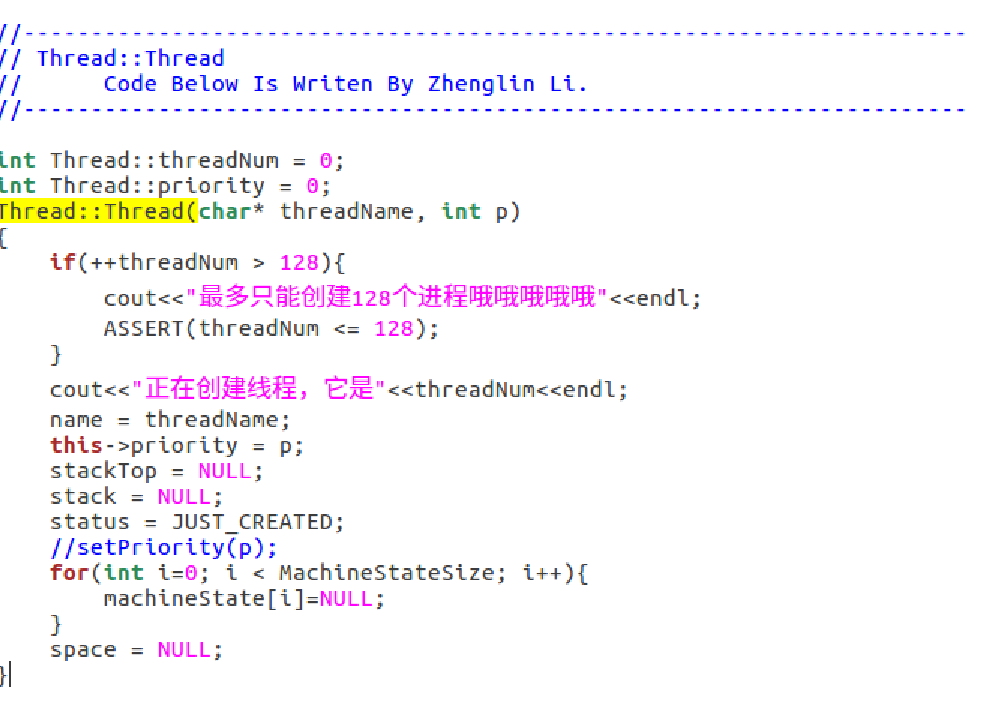
scheduler.cc中



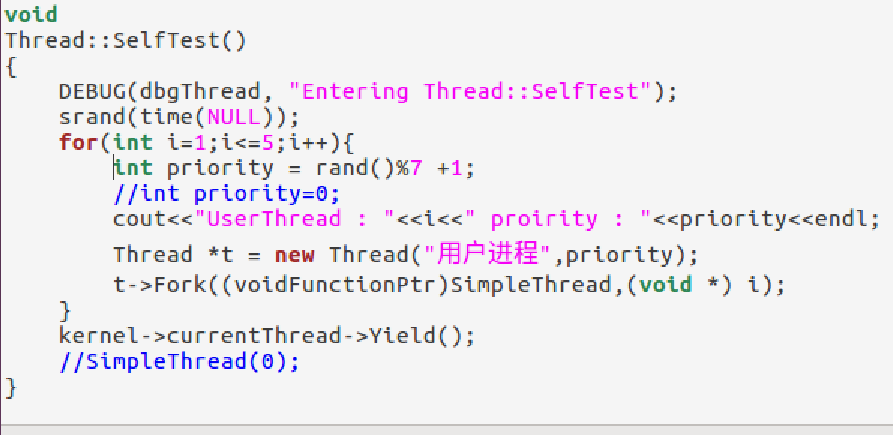
thread.h中增加getPriority函数与priority变量



Thread.cc中

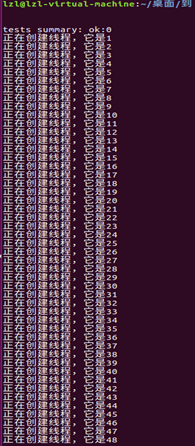


thread.cc



## **2.4实验结果：**

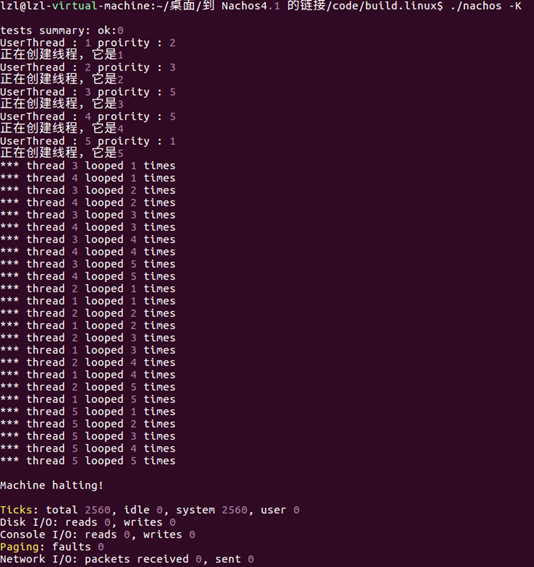
1、扩展Nachos线程管理模式，限制线程的数量(最多128个用户线程)





2、修改扩充Nachos的线程调度机制，改为“优先级调度”的抢占式调度

在Nachos系统中，有一个线程就绪队列，其中是所有就绪线程。调度算法比较简单，就是取出第一个放在处理机运行即可。由于Nachos中线程没有优先级，所以线程就绪队列是没有优先级的。



## **2.5实验总结：**

总结本实验的完成情况，包括代码是否编写完成，是否调试通过，能否正常运行，实现了实验要求中要求的哪些项（对实验要求的满足程度；）；

总结本实验所涉及到的知识点。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 代码 | 完成编写  通过  正常 | | 知识点总结：  Nachos中没有进程和线程的区分。由于Nachos是运行在Linux上的一个进程，在源代码目录中Nachos进程对应的目录为threads，那么可以理解为每一个用户程序是一个独立的进程，但是运行在Nachos之上，每一个进程就会和Nachos内核态的每一个线程一一对应起来，经过如此转换，用户的每一个程序就相当于Nachos的一个线程在运行。  Thread:管理线程的相关数据结构  Scheduler:调度线程的运行  Switch:平台依赖的线程切换  Semaphore、Lock、Condition:线程同步、互斥的相关数据结构  我们主要修改的是thread，scheduler，list  在Nachos中，线程是最小的调度单位，在同一时间内，可以有几个线程处于就绪状态。Nachos的线程切换借助于宿主机的正文切换，由于这部分内容与机器密切相关，而且直接同宿主机的寄存器进行交道，所以这部分是用汇编来实现的。由于Nachos可以运行在多种机器上，不同机器的寄存器数目和作用不一定相同，所以在switch.s中针对不同的机器进行了不同的处理。读者如果需要将Nachos移植到其它机器上，就需要修改这部分的内容。  进程创建时调用构造函数，此时是justcreated状态  •Yield()：当前线程放入就绪队列，从scheduler就绪队列中的找到下一个线程上cpu，以达到放弃CPU的效果。  •Fork(VoidFunctionPtrfunc,intarg)：func是新线程运行的函数，arg是func函数的入参，Fork的实现包括分为几步：分配一个堆栈，初始化堆栈，将线程放入就绪队列。  •Finish()：不是直接收回线程的数据结构和堆栈，因为当前仍在这个堆栈上运行这个程。先将threadToBeDestroyed的值设为当前线程，在Scheduler的Run()内切换到新的线程时在销毁threadToBeDestroyed。Yield()、Sleep()。这里实现的方法大多是都是原子操作，在方法的一开始保存中断层次关闭中断，并在最后恢复原状态。  Scheduler是Nachos的调度器，最重要的功能是选择下一个能够运行的线程进行运行，在这个调度器中使用了List这个工具类，马上会说最重要的有三个函数，ReadyToRun, FindNextToRun, Run。  下面分别来介绍，ReadyToRun函数负责将一个线程的状态设置成READY，并且把它加入到readyList中，也就是说，调用ReadyToRun函数后，并不代表这个线程就要被运行了，而只是“候补”运行，只有当scheduler在调度时选择这个线程后，这个线程才会真正运行。  FindNextToRun函数就是很简单地将链表头部的元素（第一个线程）取出。  Run是真正的调度算法，执行用户线程。 | | | |
| 调试 |
| 程序运行 |
| 实现项 | 扩展Nachos线程管理模式，限制线程的数量为最多128个 | 修改扩充Nachos的线程调度机制，改为“优先级调度”的抢占式调度 |