**实验一　体验Nachos下的并发程序设计**

姓名：李波 学号：22920202204570

2023-3-25

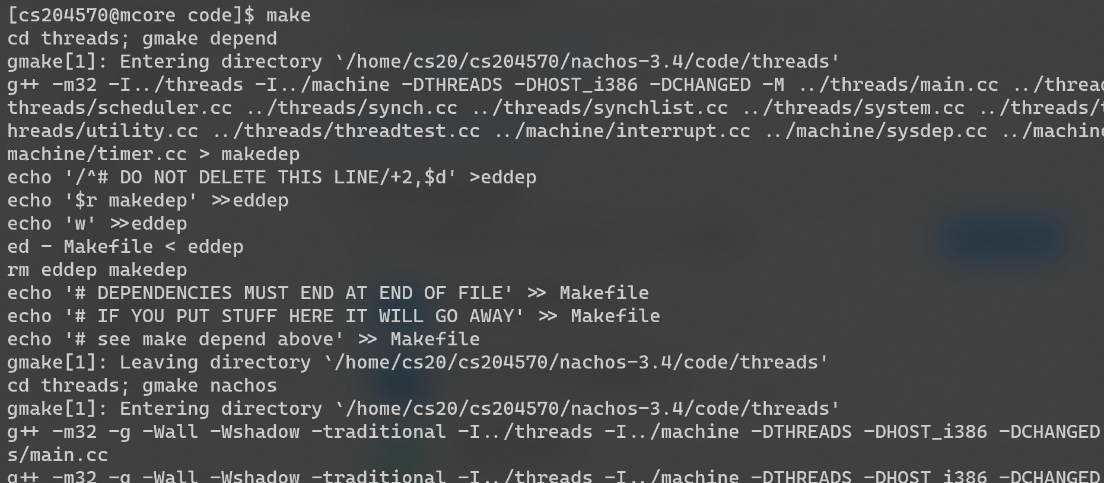
# 一、实验内容

本次实验的目的在于对nachos进行熟悉，并初步体验nachos下的并发程序设计。实验内容分三部分：安装nachos；用C++实现双向有序链表；在nachos系统中使用你所写的链表程序并演示一些并发错误。

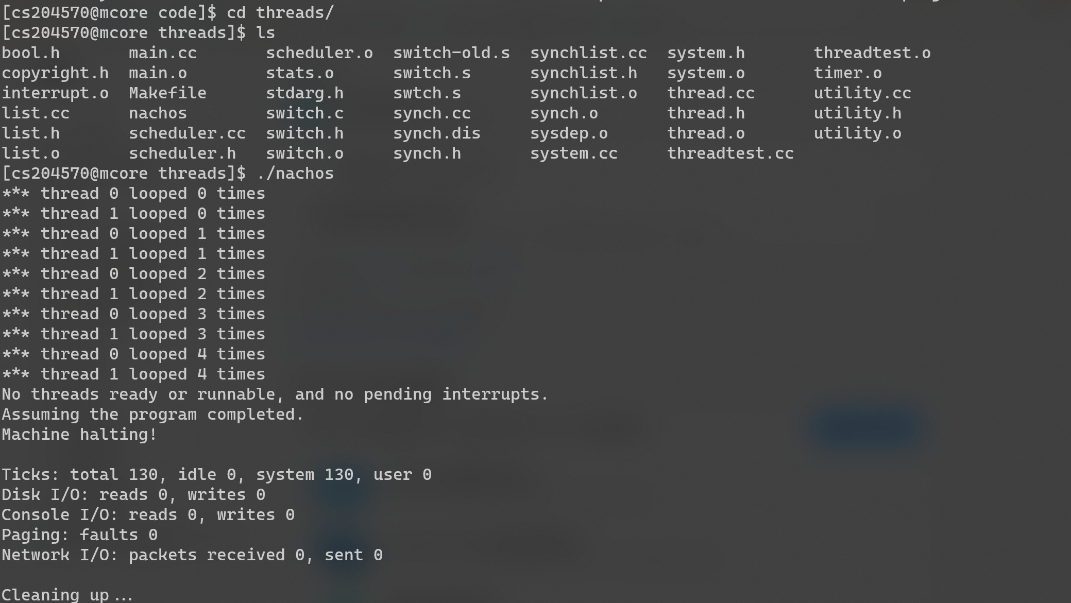
# 二、实验内容

## （一）安装nachos

将下载的nachos-linux64.tar.gz文件拷贝到实验平台的个人目录下，执行tar -zxvf nachos-linux64.tar.gz命令解压压缩包。进入nachos-3.4/code目录，执行命令make进行编译，可得到可执行文件nachos。



执行命令./nachos运行，可观察到0、1两个线程切换的现象，说明安装成功。



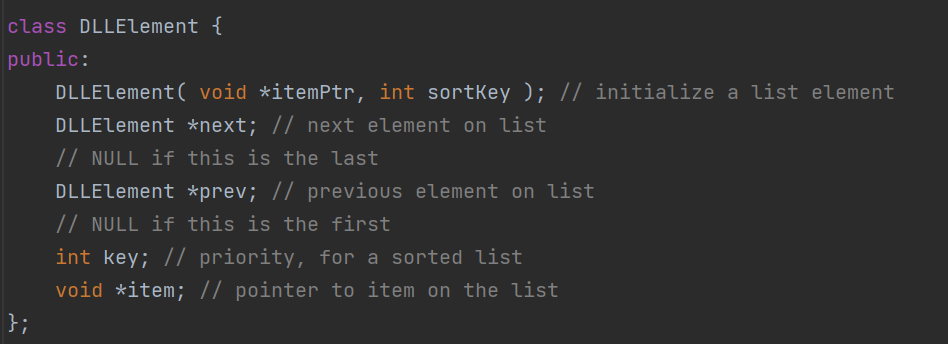
## （二）实现双向有序链表

### 1.双向有序链表的定义（dllist.h）

nachos-labs.pdf文件中的dllist.h定义了双向链表结点类和双向链表类的具体细节。

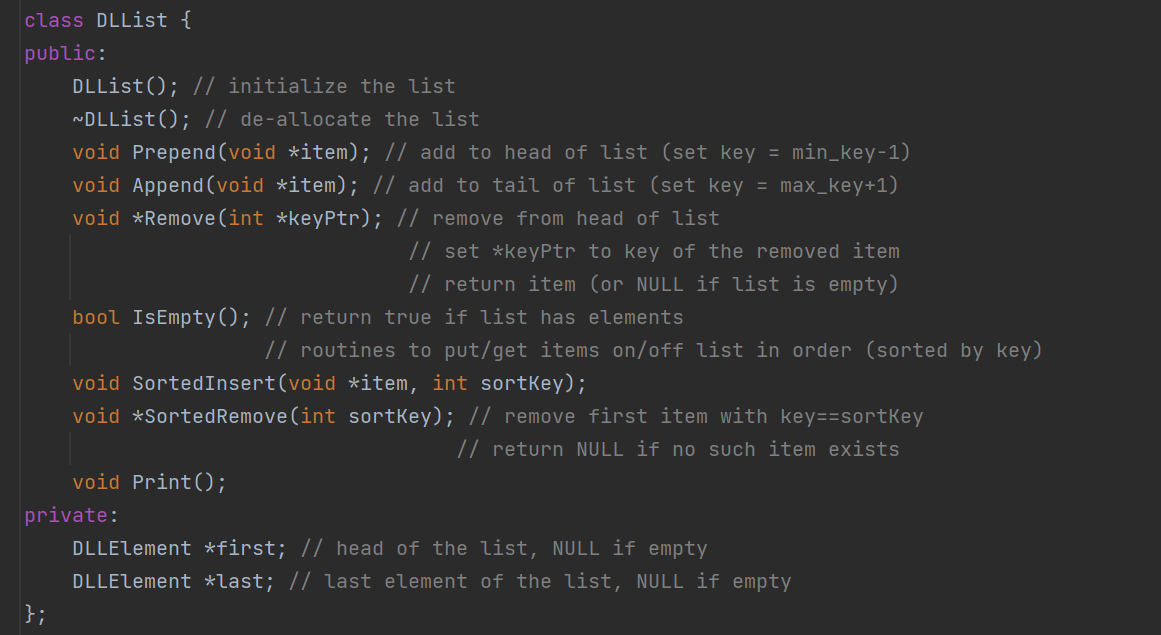
**（1）双向链表结点类定义**

双向链表结点类包含5个成员，首先是一个构造函数，用来初始化结点类，然后是两个指针prev和next，前者指向上一个结点，后者指向下一个结点，接着是key，用来表示排序优先级，最后是指向链表元素的指针item。



**（2）双向链表类定义**

在双向链表类中定义了链表的构造函数和析构函数，以及私有成员first和last，分贝用来指向链表的头部和尾部，同时定义了一系列函数，用来表示双向链表的各种操作，分别为从头部插入结点，从尾部插入结点，删除头结点结点，检测链表是否为空，按序插入结点，按序删除结点,链表输出。这里需要注意的一点是，链表不为空时IsEmpty函数返回true。

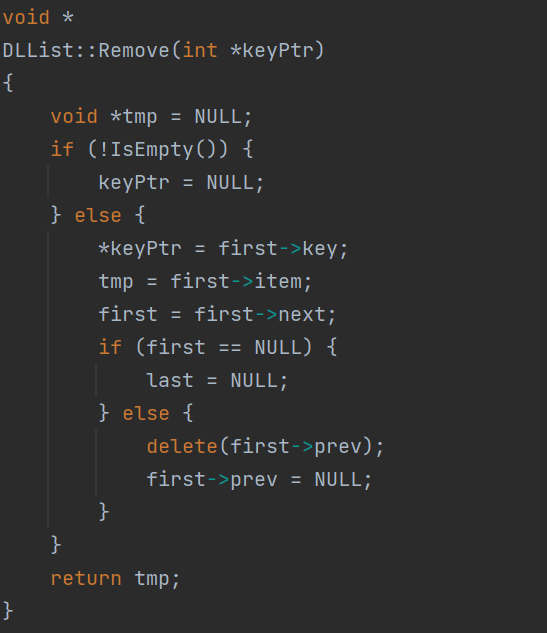


### 2. dllist.cc实现

双向链表结点类的构造函数以及双向链表类的所有函数均在dllist.cc中实现，这里仅展示Remove函数和SortedInsert函数。

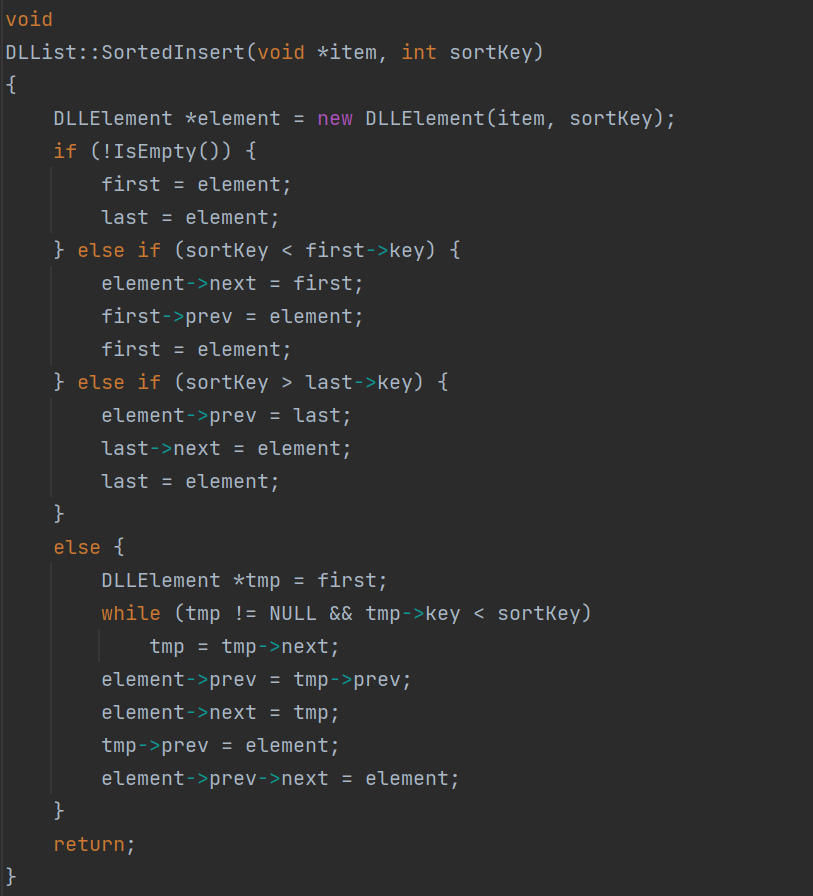
**（1）Remove函数实现**

Remove函数的功能是删除链表头结点，参数keyPtr指向删除结点key值的指针，如链表为空则设置为NULL，返回值为指向被删除结点的指针，如链表为空返回NULL。



**（2）SortedInsert()函数实现**

SortedInsert函数的功能是插入指定key值的结点，参数item是指向插入结点的指针，sortKey是需要插入的结点的key值。

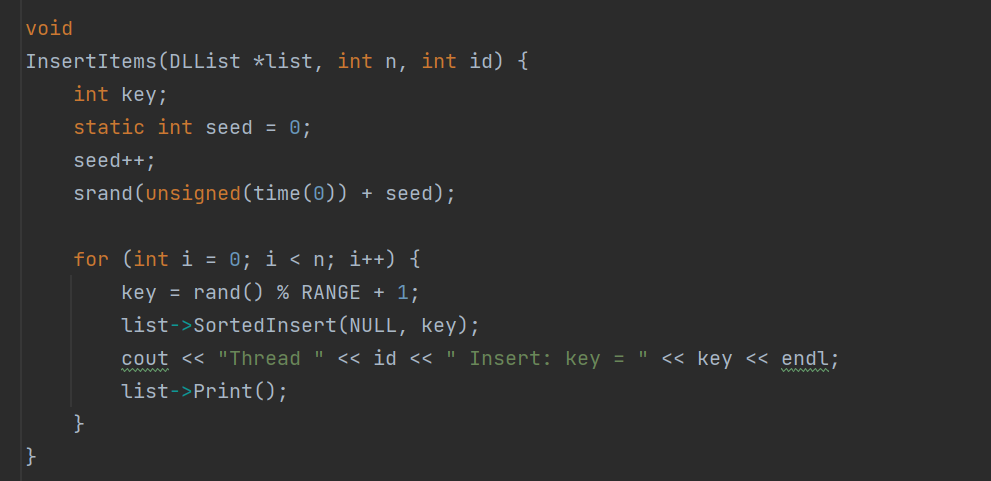


### 3. dllist-driver.cc实现

dllist-driver.cc需要提供两个供线程操作双向链表的函数，分别负责向双向链表中插入指定个随机元素和从表头开始删除指定个元素。

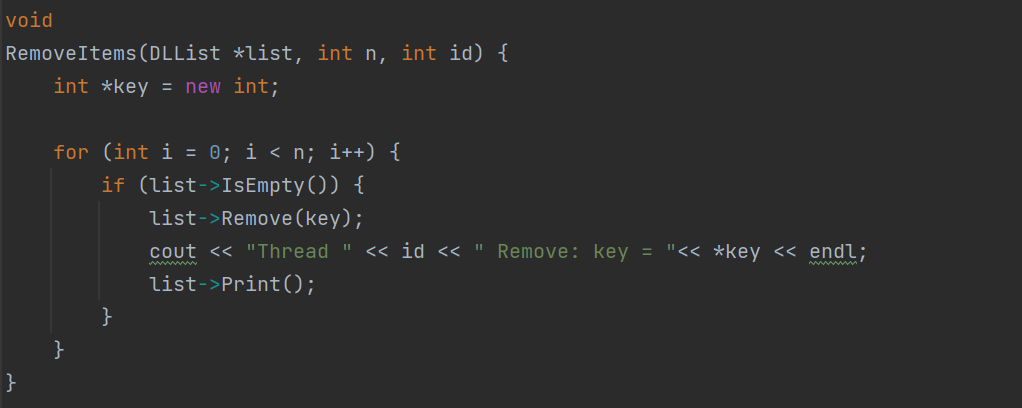
**（1）InsertItems函数实现**

InsertItems函数的功能是向双向链表中插入N个结点，通过多次调用SortedInsert函数实现，参数list是指向双向链表的类指针，n表示插入的结点数量，id表示线程号。



**（2）RemoveItems函数实现**

RemoveItems函数的功能是从头结点开始删除双向链表中N个结点，通过多次调用Remove函数实现，参数list是指向双向链表的类指针，n表示删除的结点数量，id表示线程号。



### 3.双向链表运行结果

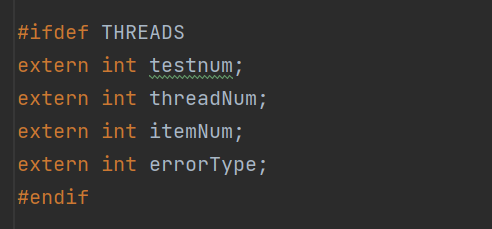
**（1）修改main.cc**

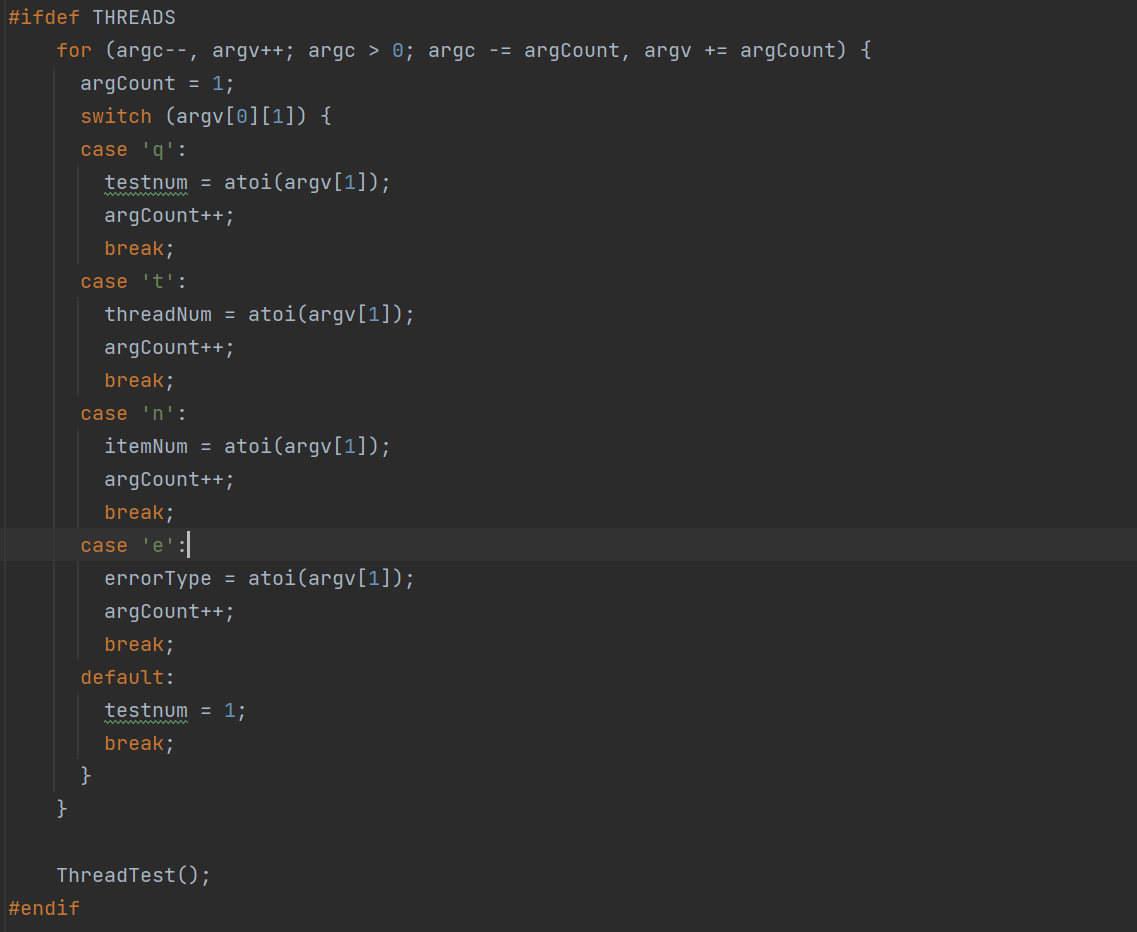
命令格式为

./nachos [-q testnum] [-e <errorType>] [-t <threadNum>] [-n <itemNum>]

4个参数依次为测试类型、错误类型、线程数、操作结点数。

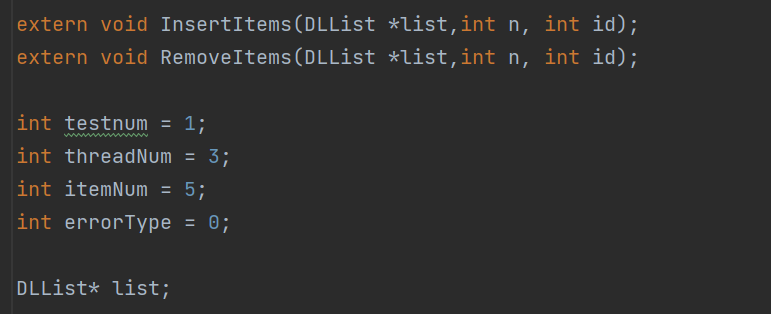
在main.cc中声明相关参数的全局变量，并增加命令行参数处理。



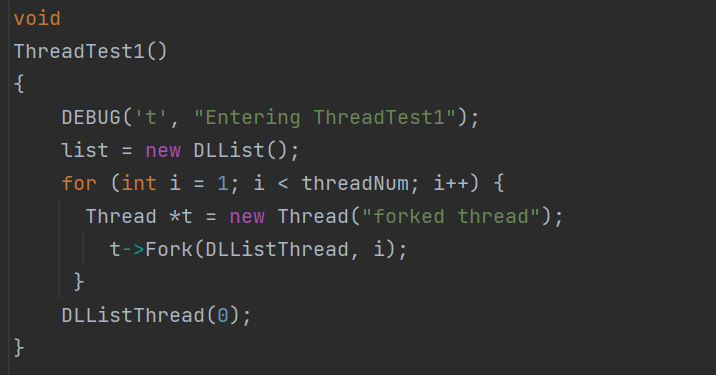


**（2）修改threadtest.cc**

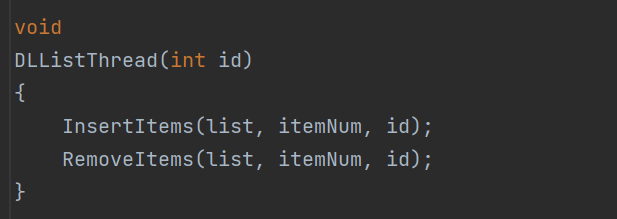
在全局变量中引入命令行参数并创建双向链表。



修改ThreadTest1()，初始化dllist并创建线程，每个线程调用DLListThread函数。

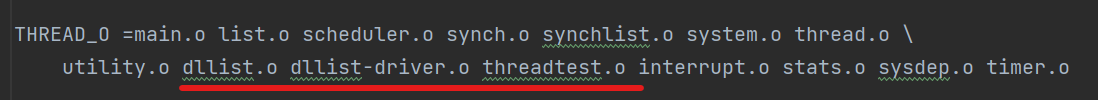


定义DLListThread()，其规定了每个线程执行的操作。



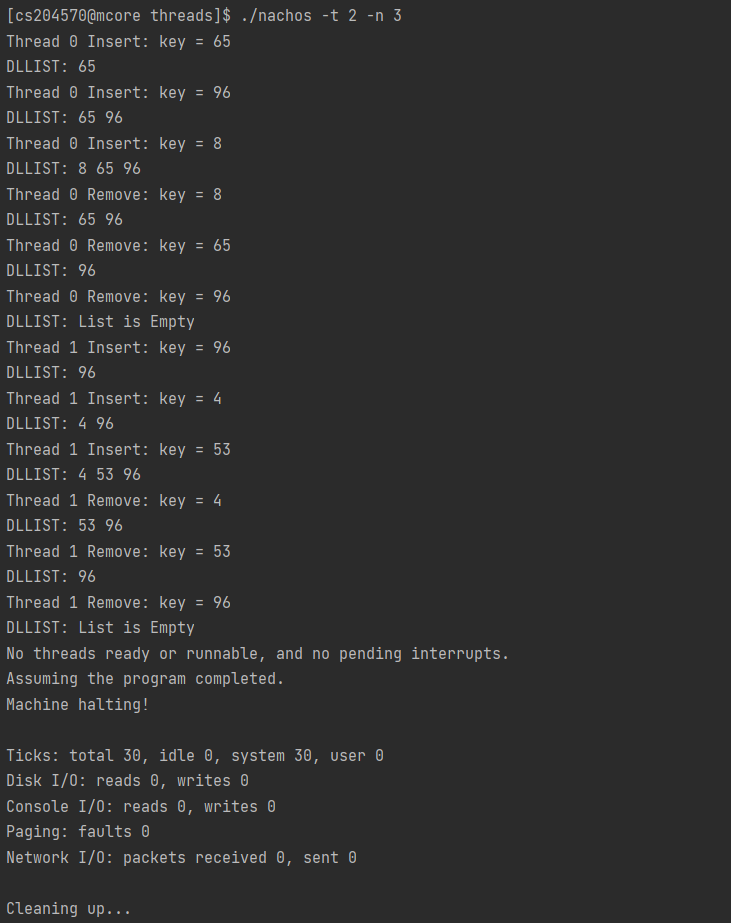
**（3）修改Makefile.common，重新编译nachos**



**（4）运行程序，观察****正常情况下的程序运行**

以命令./nachos -t 2 -n 3为例。

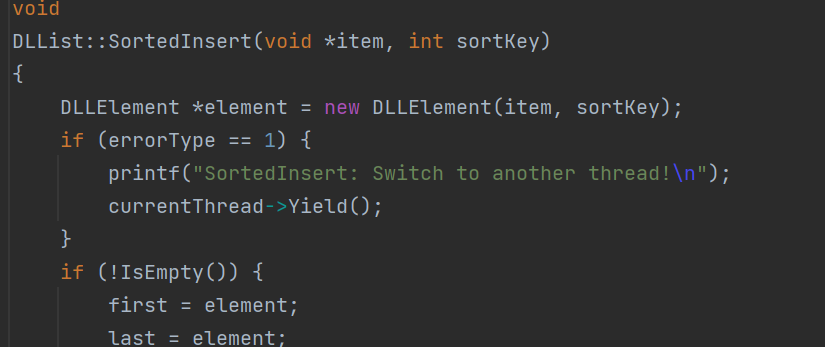


## （三）并发错误演示与分析

**1.错误类型1（errType = 1）：**

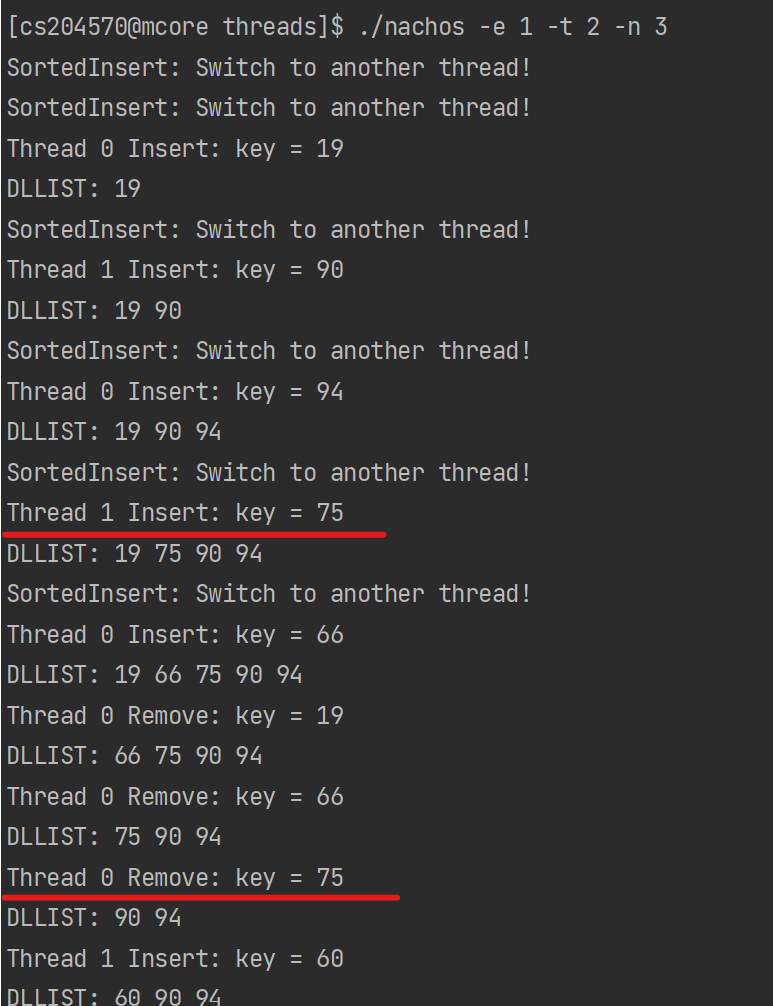
引发方式：

在插入元素前发生线程切换。



运行结果：

执行命令./nachos -e 1 -t 2 -n 3



出现线程插入和删除的结点不一致的情况。

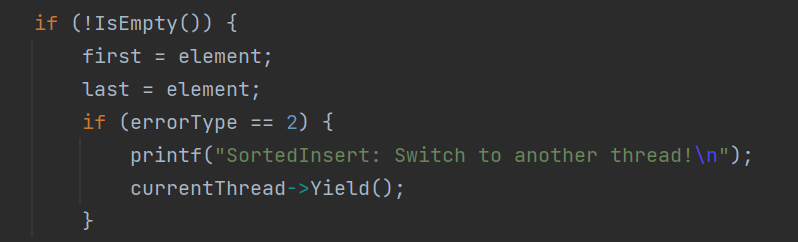
原因分析：

每个线程在插入前都会进行切换，而所有线程的内存是共享的，他们都在同一个链表上插入和删除，不同线程交叉进行插入和删除操作，插入时结点会按照大小排序，而删除时都是从链表头部删除，所以最终导致每个线程都有可能出现插入和删除的结点不一致的情况。

**2.错误类型2（errType = 2）：**

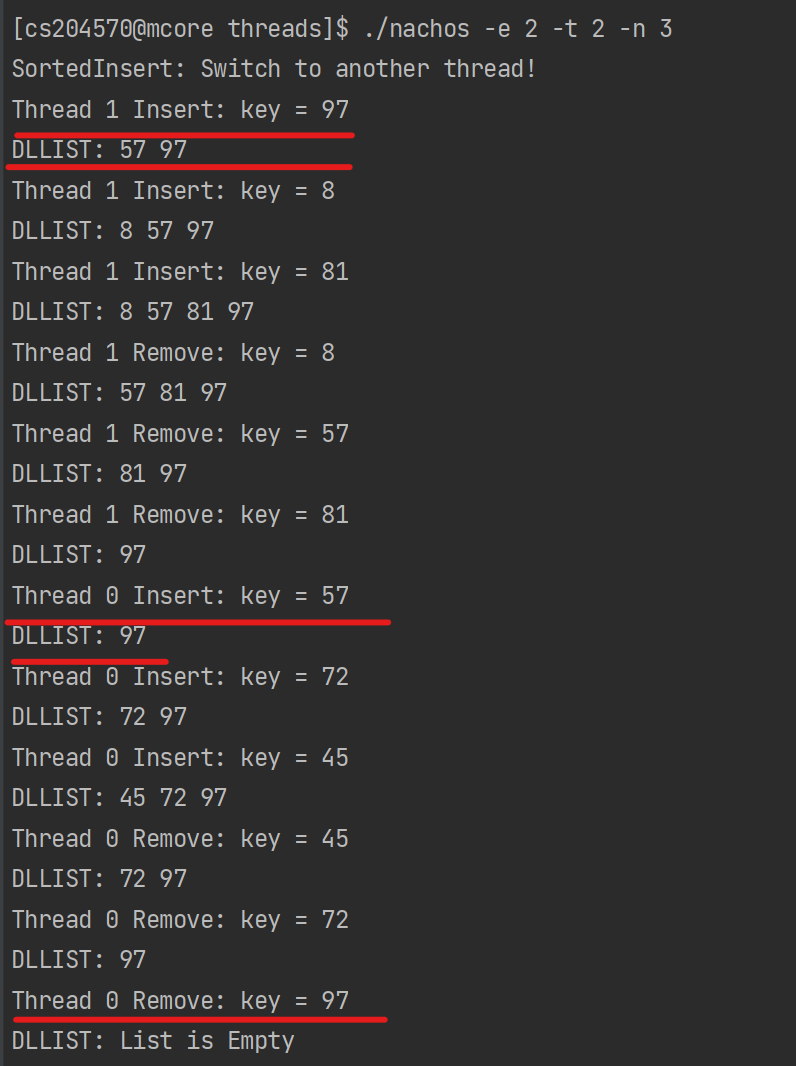
引发方式：

在链表为空时，插入完第一个元素后立刻切换线程。



运行结果：

执行命令./nachos -e 2 -t 2 -n 3



可以发现线程插入和删除的顺序出现了混乱，且每个线程插入和删除的元素并不相同。

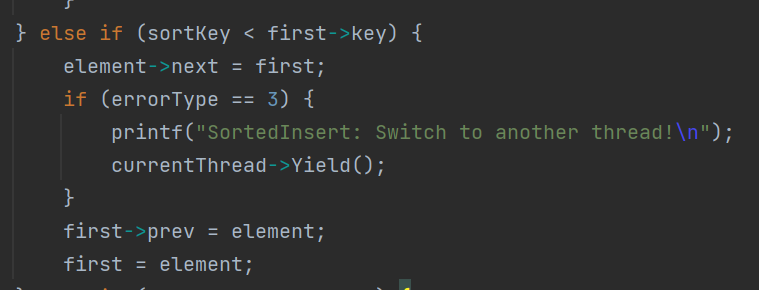
原因分析：

每个线程在插入元素之后都会立即切换，此时函数虽然尚未执行完但元素已经正常插入链表。这里线程1插入元素97之前，链表已经存在线程0插入的元素57。但元素97之后却被线程0删除。而之后线程0执行完插入元素57之后，链表中却是线程1先前插入的元素97。

**（3）错误类型3（errType = 3）**

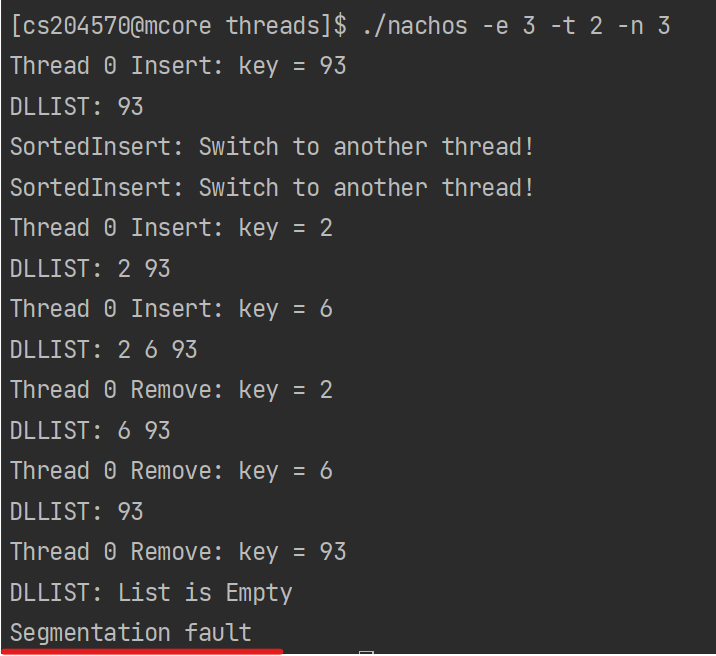
引发方式：

插入链表头部时只连接了element->next指针而未连接first->prev时发生切换。



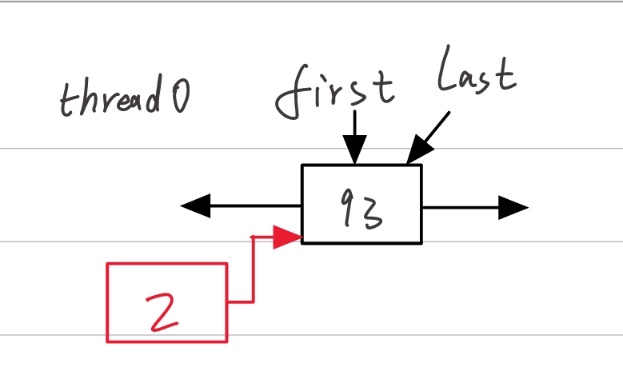
运行结果：

执行命令./nachos -e 3 -t 2 -n 3

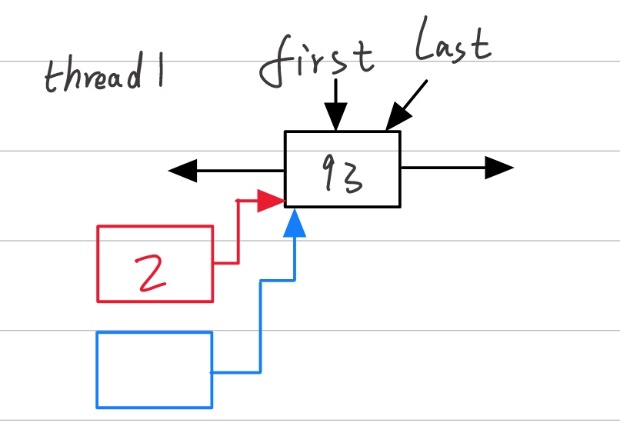


出现了段错误。

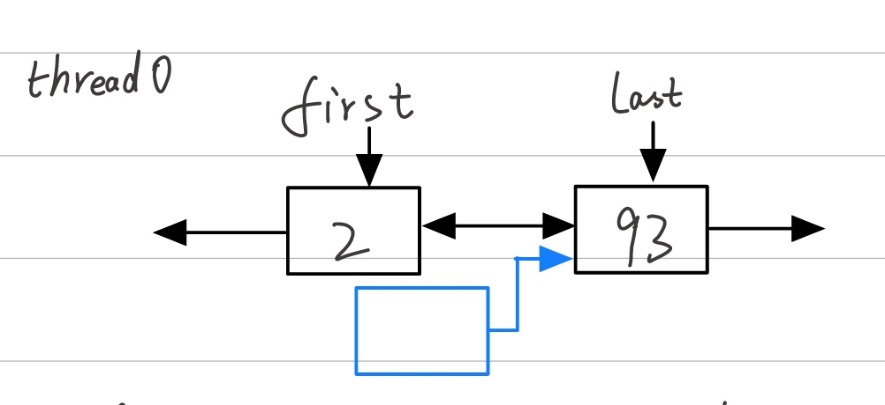
原因分析：



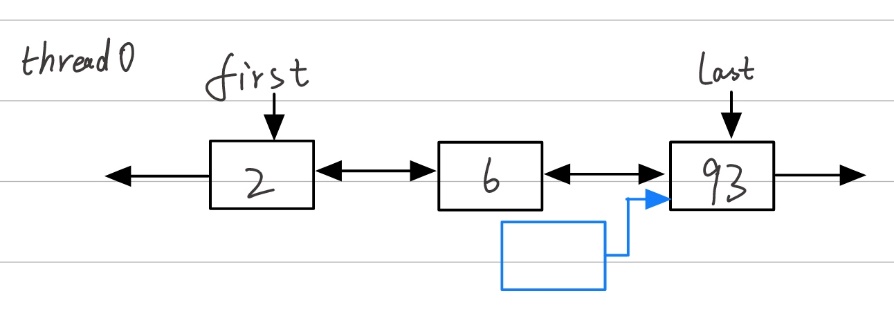
线程0插入结点93后，试图在头部插入一个结点2，将结点2的next指针指向头部后，发生了线程切换。



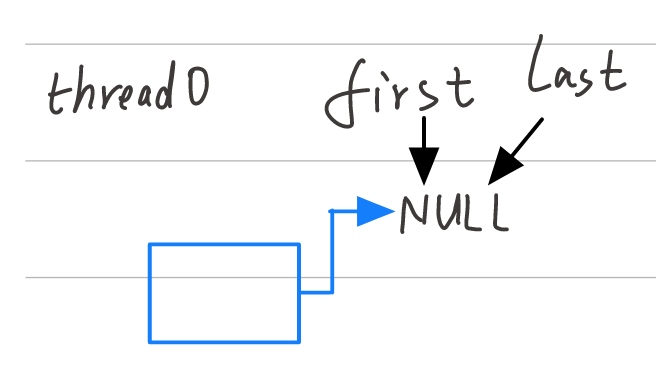
线程1试图在头部插入一个结点，但此时first指针未发生改变，所以新结点的next指针仍然指向了结点93，然后发生了线程切换。



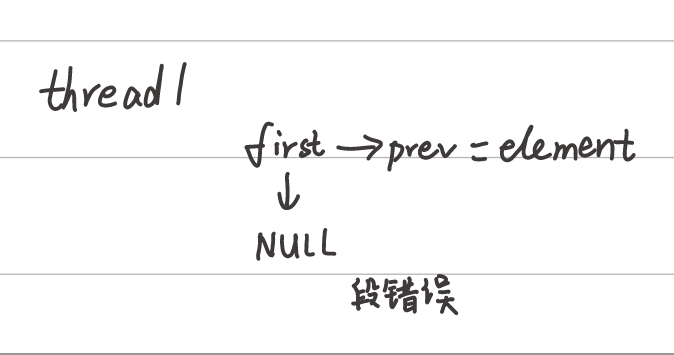
线程0接着之前未完成的工作，将结点2插入链表中。



线程0继续插入第3个结点即结点6，此时线程1要插入的结点的next指针仍指向结点93。



线程0自己创建的3个结点，此时first = last = NULL，由于之前线程1要插入的结点的next指针一直指向结点93，所以现在该结点也指向了NULL。

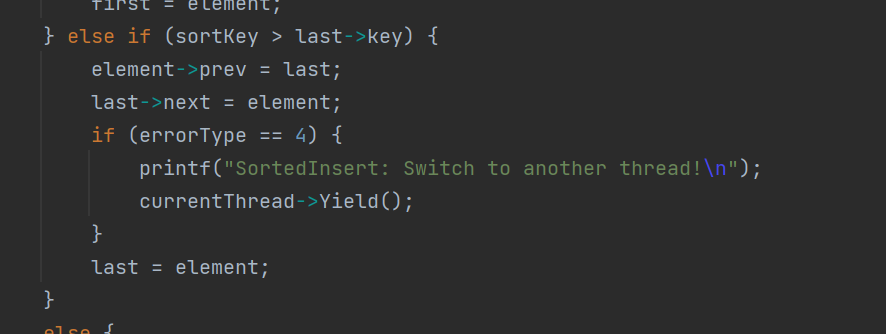


最后线程1试图完成前面没有完成的结点插入，但此时first指针值为空，执行first->prev=element语句发生段错误。

**（4）错误类型4（errType = 4）**

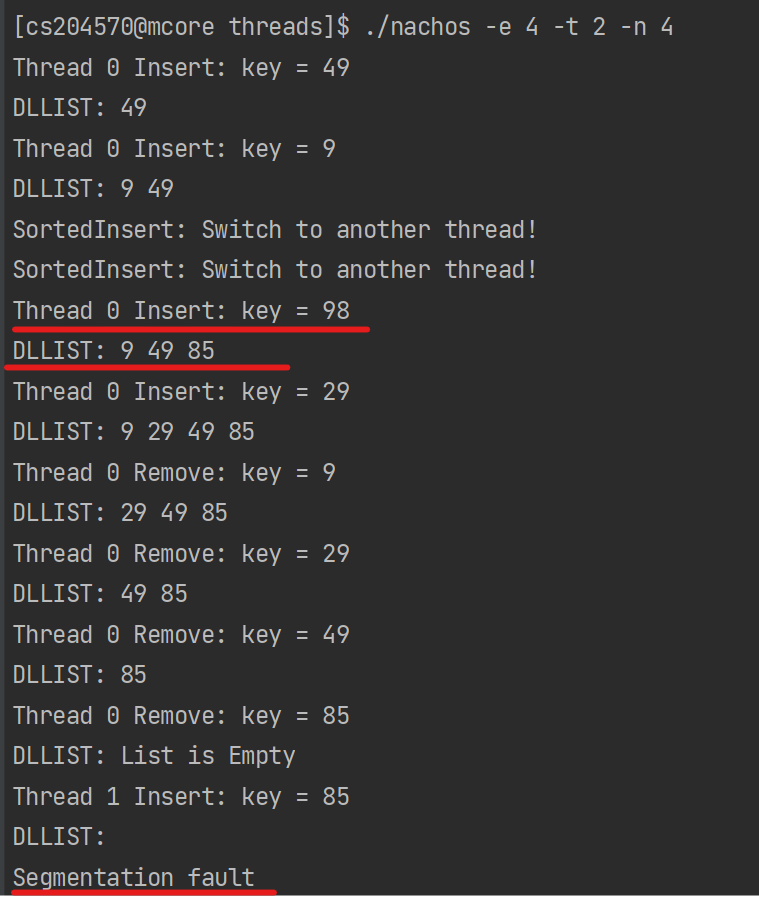
引发方式：

插入链表末尾时，指定了last->next指针但未将last移到最后一个元素时发生了切换。



运行结果：

执行命令./nachos -e 4 -t 2 -n 4



链表发生插入删除失序、结点丢失和段错误。

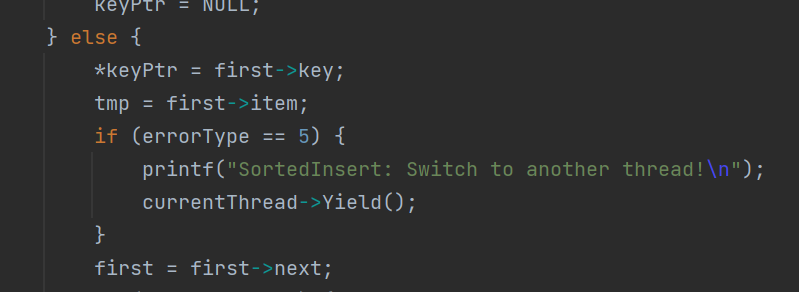
原因分析：

当向末尾插入元素时，此时已经插入了元素，但是还没有移动last指针。下一次再向末尾插入元素时，造成原先末尾的元素丢失。进行最后一次删除操作前，链表已空，因此产生段错误。

**（5）错误类型5（errType = 5）**

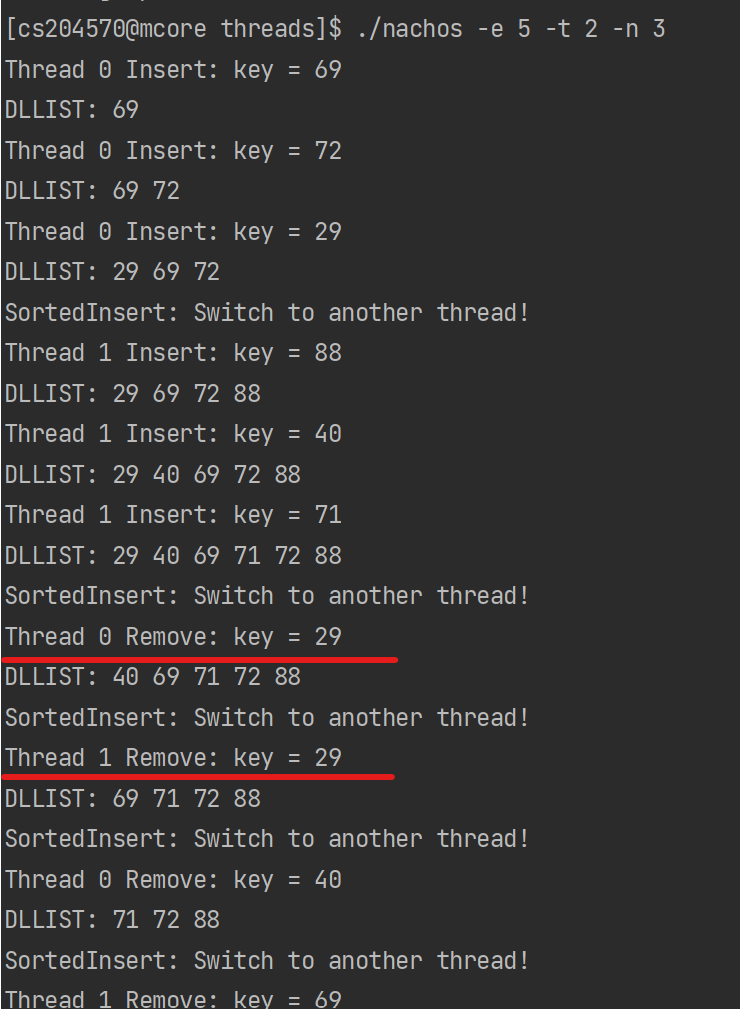
引发方式：

在修改first指针之前切换线程。



运行结果：

执行命令./nachos -e 5 -t 2 -n 3



发生重复删除错误。

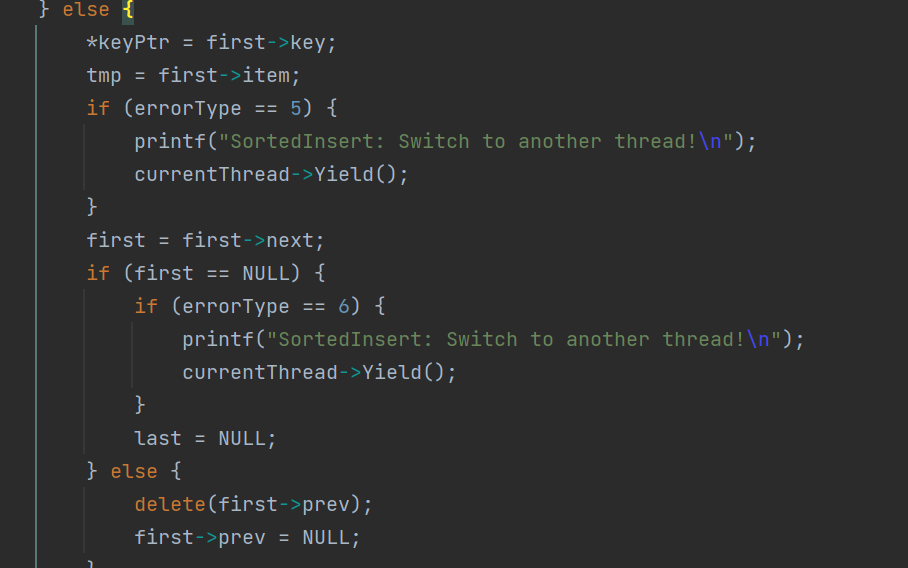
原因分析：

在修改first指针之前切换线程，此时first指针仍然指向被删除的结点，删除操作并未实际发生，切换后的线程认为并没有进行删除，所以发生重复删除。

**（6）错误类型6（errType = 6）**

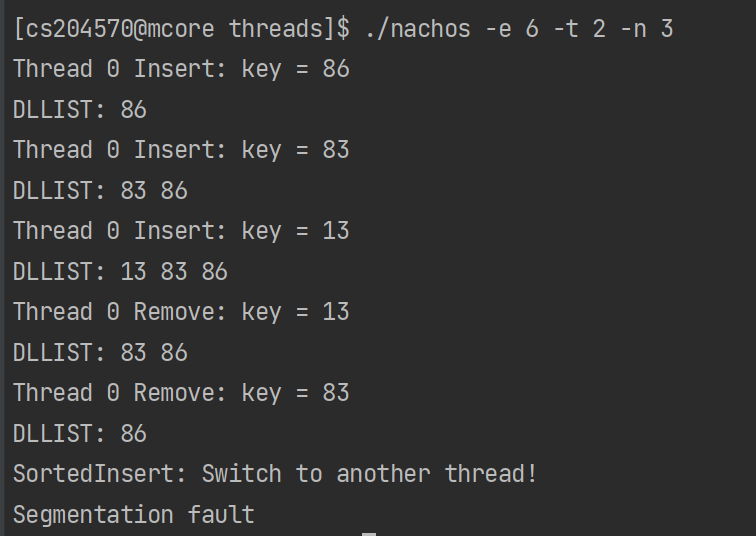
引发方式：

若删除结点后链表为空，在将last指针改为NULL之前切换线程。



运行结果：

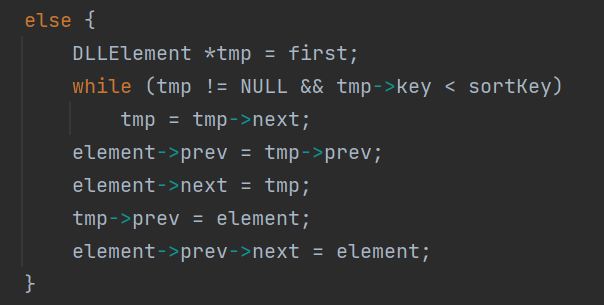
执行命令./nachos -e 6 -t 2 -n 3



发生段错误。

原因分析：

在将last指针改为NULL之前切换线程，切换后的线程调用IsEmpty函数判断链表是否为空时，由于只有在first和last同时为NULL时函数才返回false，所以切换后的线程认为此时链表并不为空，所以向链表内插入结点。



此时tmp被赋值为NULL，所以在执行tmp->prev语句时发生段错误。

# 三、实验总结

本次实验的主题是实现双向链表并体验Nachos的线程系统。

在实验过程中，我首先完成了 Nachos 的安装，了解了Nachos编译安装的基本过程，对makefile文件有了一个基本的理解。

然后我实现了一个双向有序链表的C++程序，该程序编写的重难点是多文件的编写，之前编写的程序绝大多数都是单文件，而这次的程序编写需要分别编写dllist.h文件，dllist.cc文件和dllist-driver.cc文件，文件之间的跨文件变量与函数的声明和使用是需要重点关注的。在一个项目中必须保证函数、变量、枚举等在所有的源文件中保持一致，extern关键字表明该变量在所有模块中作为一种全局变量，只能被定义一次，否则会出现连接错误。但是可以声明多次，且声明必须保证类型一致。在编写完成后，还需要并将其集成到Nachos系统中，需要修改Makefile.common文件。

最后，我进行了并发错误的设计，通过在不同位置设置线程的切换，我了解了在并发程序设计中可能出现的问题，由于并发的存在，在单线程条件下能正常运行的程序却出现了乱序、丢失、段错误等问题。在不断地分析中，我加深了对线程系统工作原理的理解，让我认识到了在多线程情况下互斥和同步的重要性。