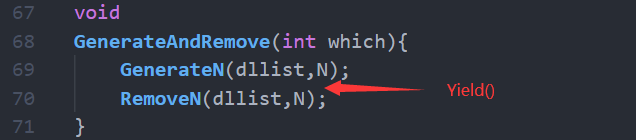
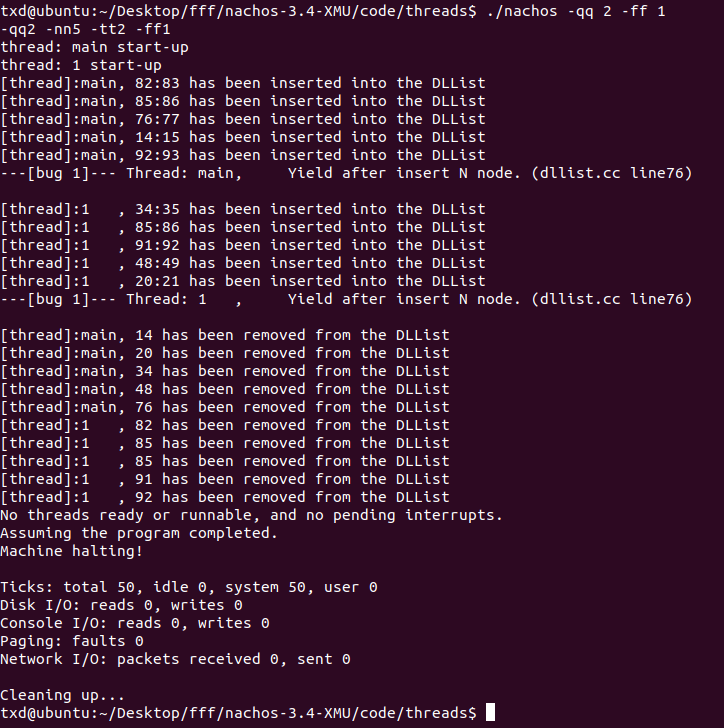
**bug1：**

修改了threadtest.cc文件中GenerateAndRemove

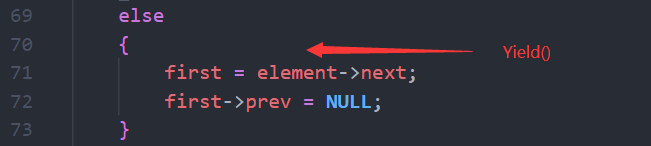


描述：线程只执行了插入函数，还未执行删除，就发生了线程切换。导致多个线程共用双向链表。在之后的删除过程中，每个线程删除的可能不是自己插入的节点。



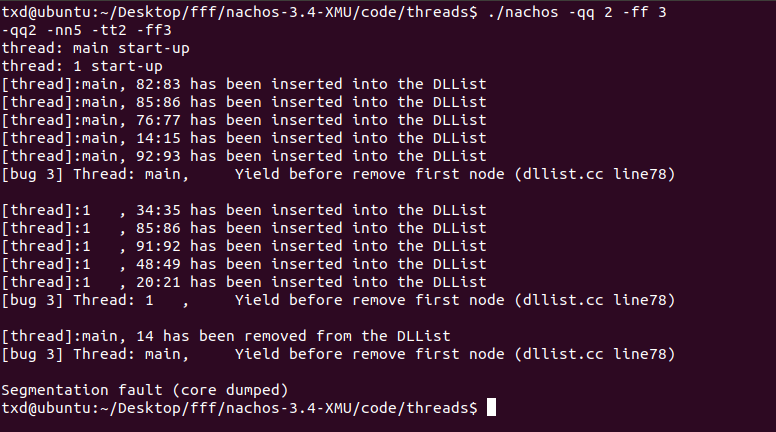
**bug2：**

修改了dllist.cc里Remove函数（这个函数的作用是删除第一个节点）



在刚要把first节点的next、prev改成NULL的时候，Yield了。

导致线程A要删除节点1的时候，这个节点1是first节点，刚要删就Yield了，之后线程B又往里加了一些节点，节点1已经不是端节点了，刚刚的线程又恢复，把节点1删了，导致链表断成两截。或者线程B把节点1删了，之后线程A恢复，仍然去访问节点1的prev、next，导致段错误。节点1被删两次，导致最后所有程序退出，链表里还有。

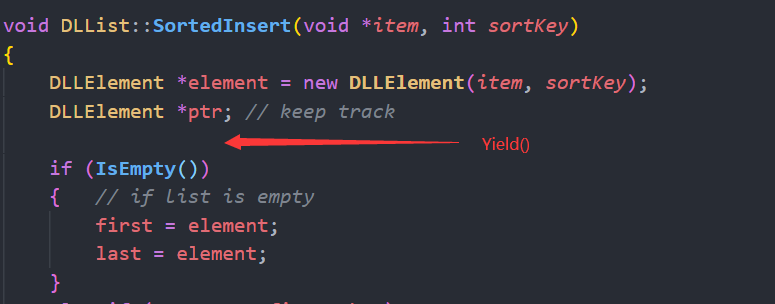


**bug3：**

修改了dllisst.cc里sortedInsert函数

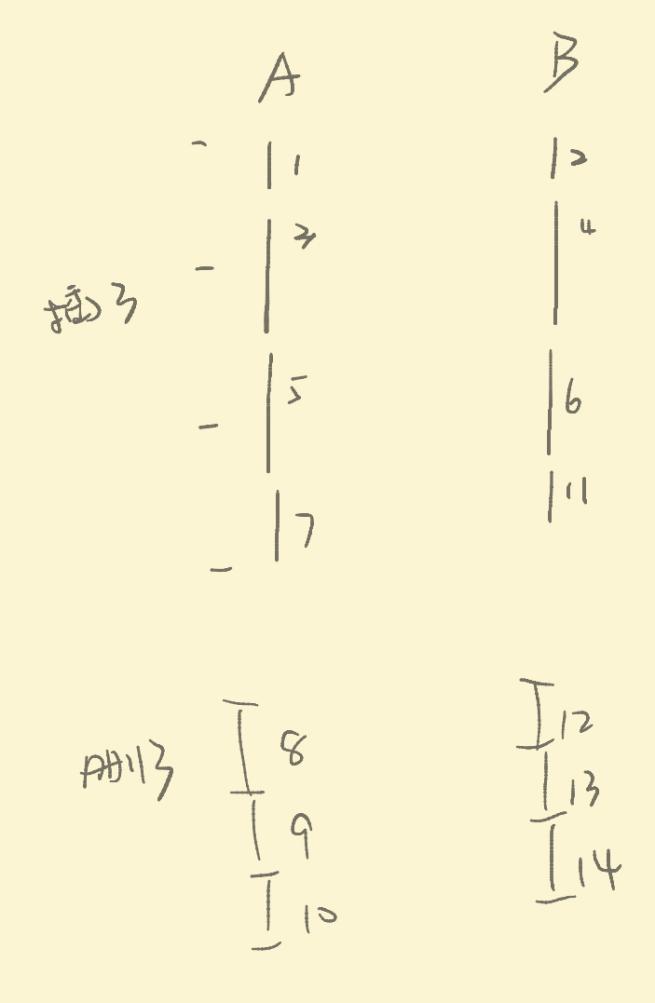
就是在插入节点时，刚完成分配空间、赋值，还未对next、prev修改时Yield。

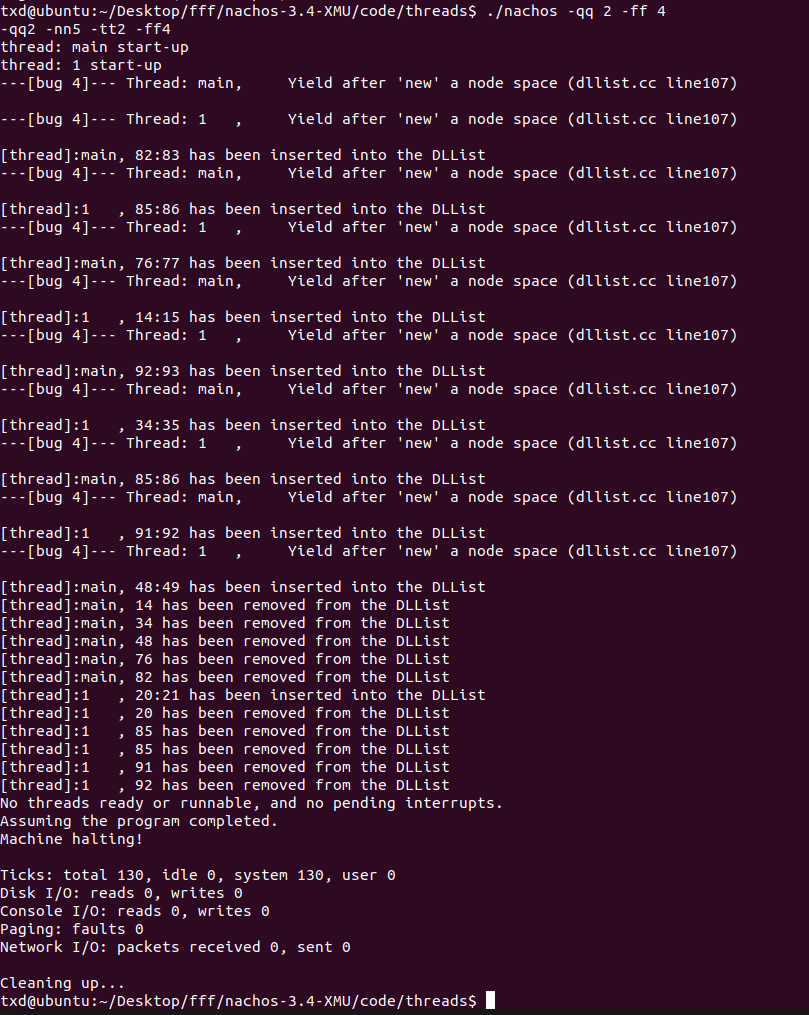
导致本来线程A要连续插3次，B再连续插3次。变成了各自交替插3次。导致删除的不是自己插入的。导致插入延迟，如下图，线程B在第6时就已经分配了新节点的空间，但是到第11时才成功修改next、prev，将节点插入链表中。



时序图

（横线表示程序段，竖线表示“时钟周期”，竖线旁数字表示第几个“时钟周期”）

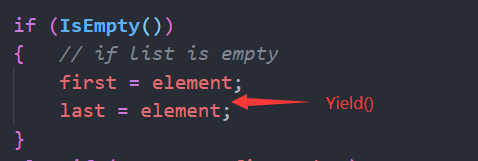




**bug4：**

修改了dllisst.cc里sortedInsert函数。

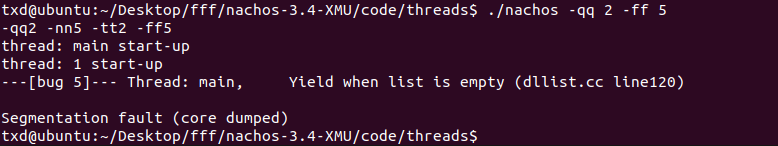
修改的是if(IsEmpty()){……}里的内容，即，程序对空链表插入时会触发这个bug。



**描述：**

线程A对空链表进行插入，修改完first=element;还没来得及修改last=element;时Yield();此时first指向第一个节点，last指向NULL;(所以之后如果有节点试图使用last的prev、next, 就会导致段错误）

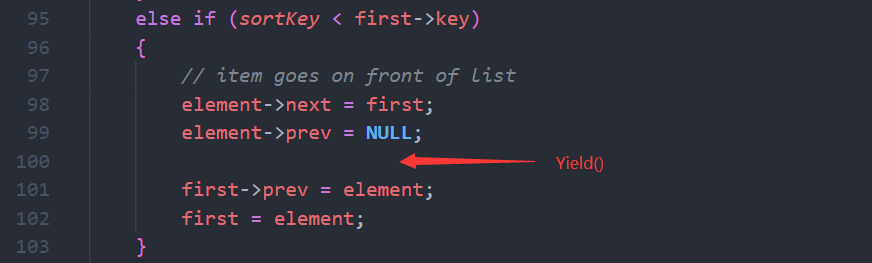
什么情况会使用last的prev、next呢？ 插入的节点要放在链表尾，这时就要修改ast。这时就会导致程序试图读取NULL的prev、next，导致段错误。（如果整个运行过程中，没有产生需要插入最后的节点，程序就会正常运行，不会报错）



**bug5**

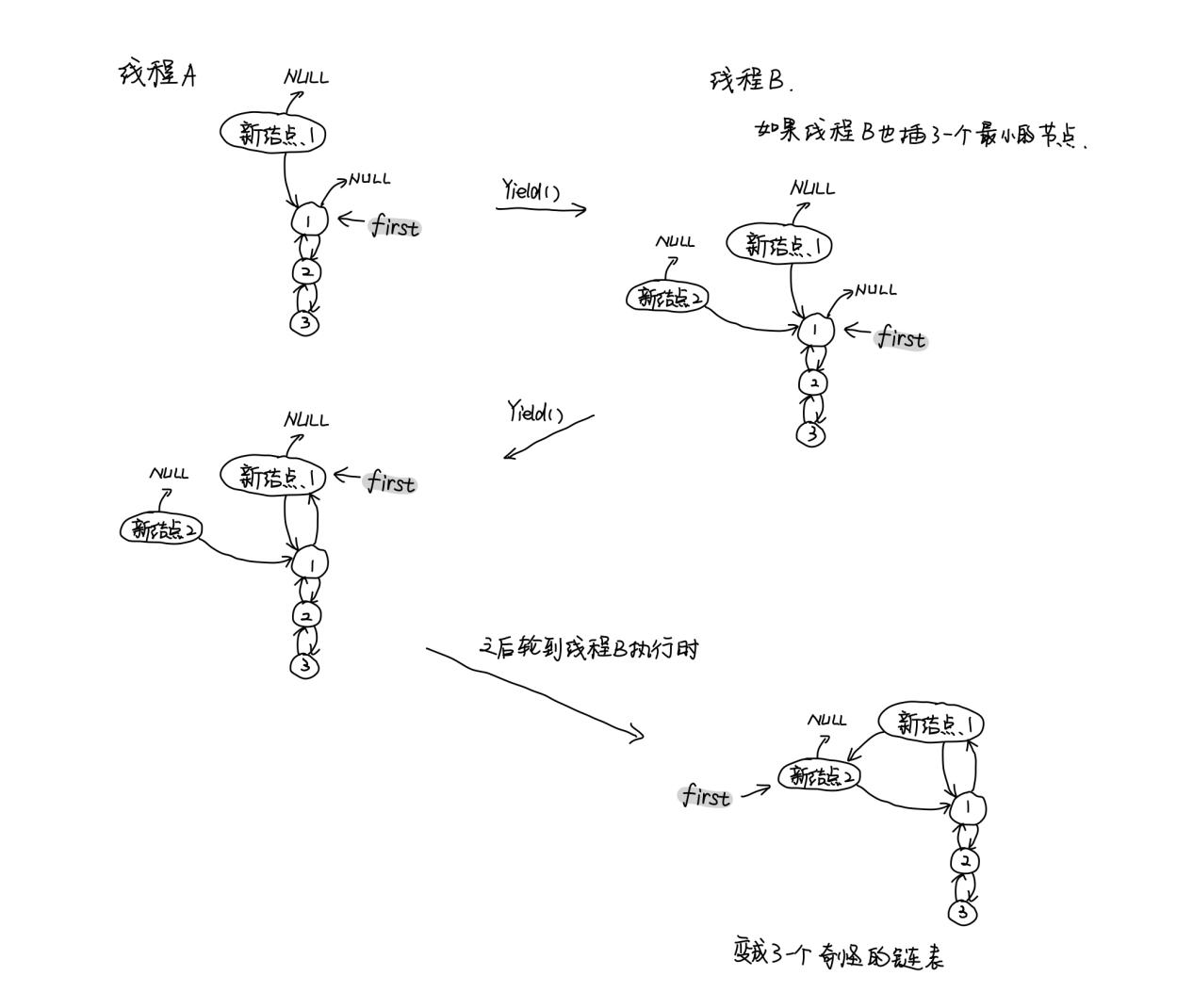
修改了dllisst.cc里sortedInsert函数。

新插入的节点是最小的，是新的头节点。

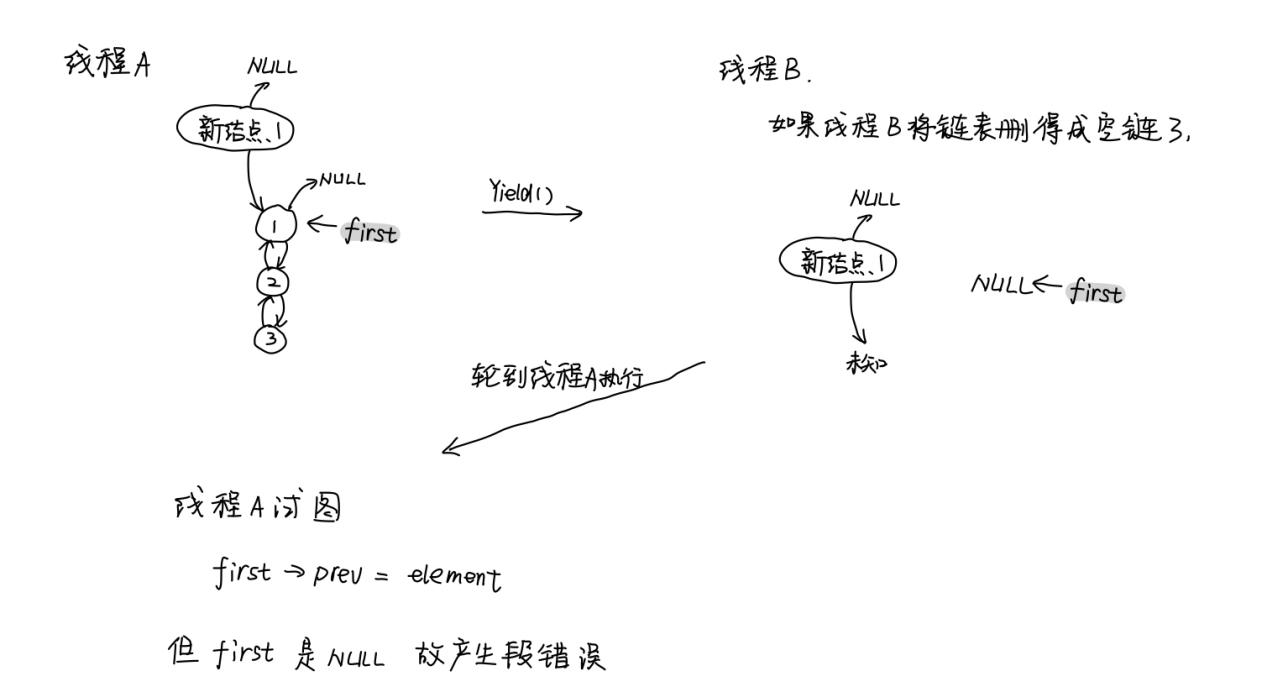


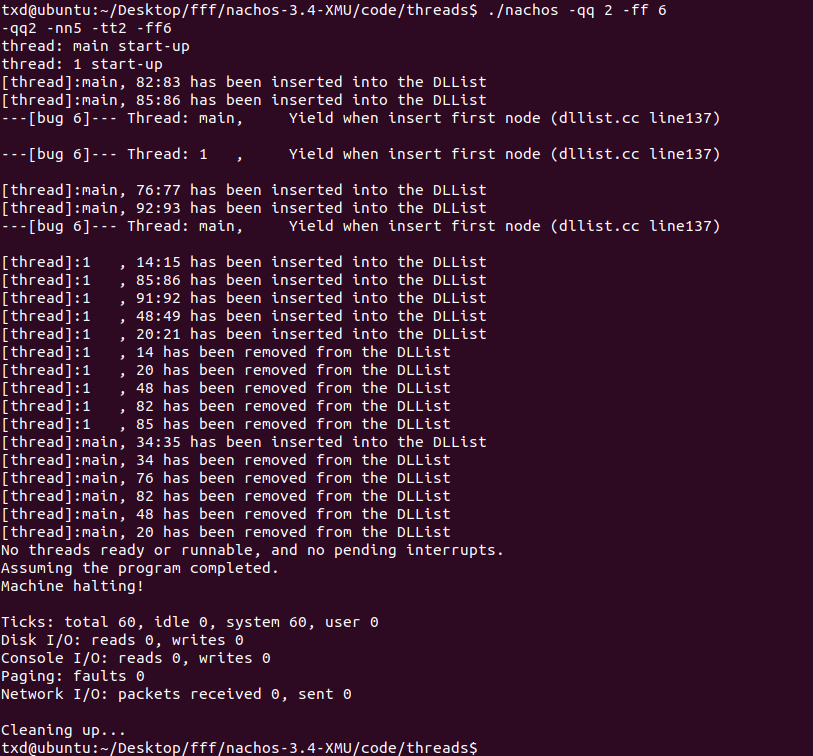
把新节点的prev、next改了，但是之前的first节点的prev没改，first指针也没改，就Yied了。

情况1：



情况2

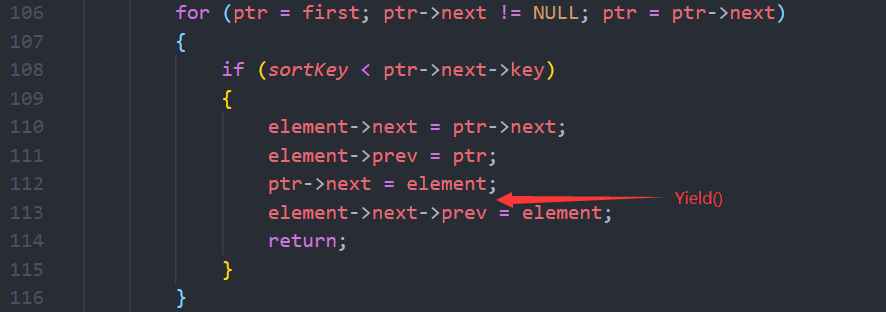




**bug6**

修改了dllisst.cc里sortedInsert函数

新插入的节点在链表中间。



类似bug7，一样是变成一个类似“图”的奇怪链表。

