1. 实验目的：尝试多线程、多进程死锁的可能，理解C语言程序的内存结构，特别是全局数据，局部数据（栈）的使用方式与大小限制。建立内存分块及其管理的概念。
2. 实验目标：实现一个能够展示Linux多进程、多线程时，堆、栈内存分配 和 回收 方式 且能够展示死锁效果的实验程序。，至少要能够实现以下功能：
3. 创建两个以上的线程/进程，定义两个互斥体（或信号量），注意合理的初始化。
4. 两个线程/进程，以相反的顺序申请互斥体（或信号）。
5. 用循环尝试反复申请（malloc）1KB堆空间，在多线程时要考虑相互影响。但不释放（free），直到程序运行时崩溃。观察堆空间的最大可能值。
6. 定义一个递归函数，其中包含1KB的局部数组，设定初始值，令其反复递归，观察栈空间的最大可能值。
7. 对应第2点，以相反的顺序申请第二个互斥体。自己设计一种方案，显示出发生了死锁。

以上是带有破坏性（会破坏什么范围的内存？）的测试程序，可自己考虑是否释放空间和互斥体。

自己的扩展目标， 可供参考的内容：查资料，实现一种堆内存的管理方式。也就是假设你有一块大内存，自己设计一对malloc和free函数。

1. 实验原理：

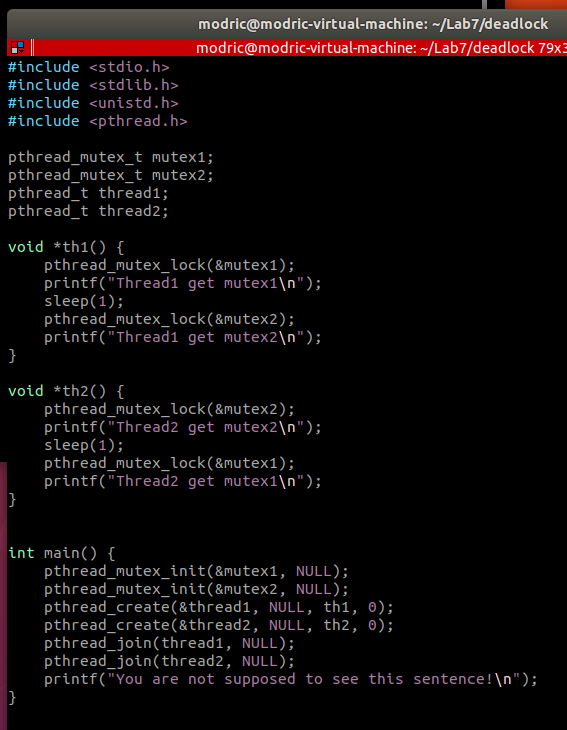
（1）线程1申请锁1，线程2申请锁2，都完成后线程1申请锁2，线程2申请锁1，进入死锁状态

（3）使用malloc申请堆空间，直到崩溃为止，查看堆空间的大小

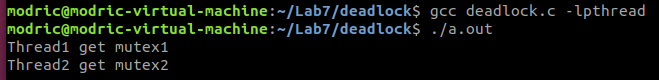
（4）使用函数内申请数组的方式分配栈空间，通过递归查看栈的大小

4. 实验方案

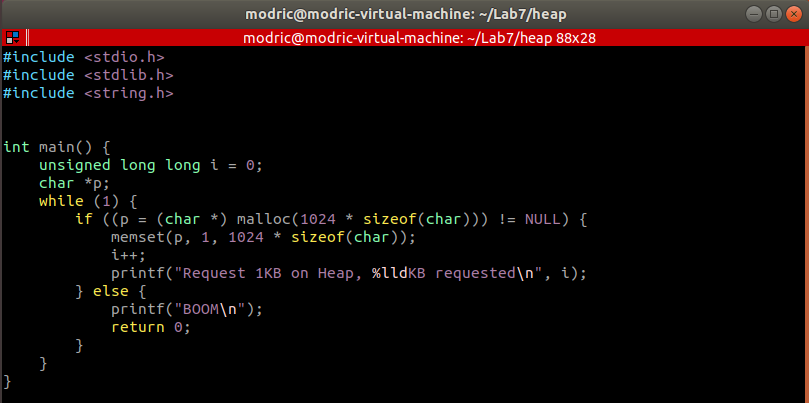
(1) 死锁实验代码如下：



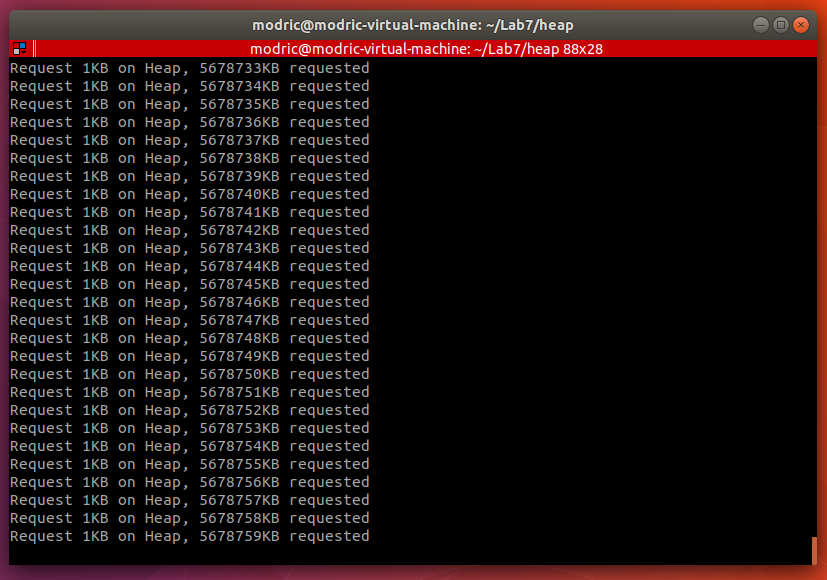
运行结果如下，两个线程各自得到一个锁，然后进入死锁状态



（2）堆内存分配代码如下：



运行截图：



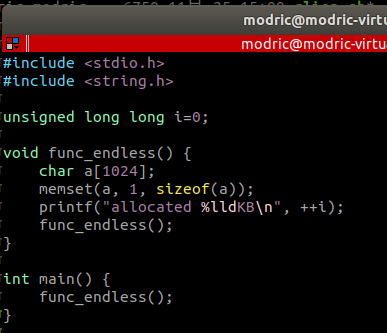
我给虚拟机设置了7G内存，在分配5.41G堆空间后，虚拟机中的Ubuntu系统卡死

重启后使用free命令查看空闲内存：

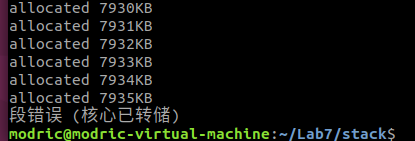


分配的堆空间大小基本与空余内存大小一致

（4）代码：



运行结果：



得出栈大小约为8M

5. 实验结论

实验（1）结合死锁的理论模型，观察了死锁的实际现象。

实验（3）说明堆空间的大小基本和剩余内存大小是符合的

实验（4）说明栈空间大小约为8M