分布式系统作业

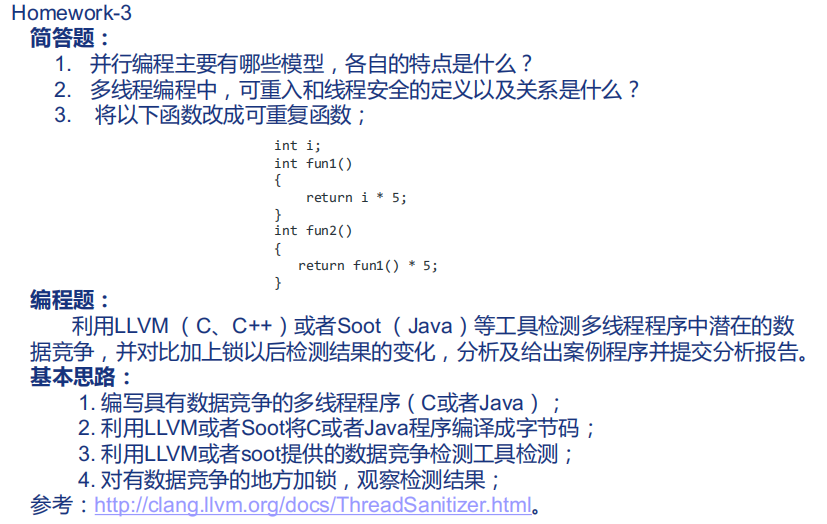
第3次作业

姓名：唐晨轩

班级：人工智能与大数据

学号：19335182

1. 问题描述



1. 解决方案

简答题：

1. 并行编程的模型有三个：
2. 消息传递：封装本地数据的独立任务，任务通过交换消息进行交互。
3. 共享内存：任务共享一个公共地址空间，任务通过异步读写此空间进行交互
4. 数据并行：任务执行一系列独立的操作，数据通常在任务之间均匀划分。
5. 线程安全的定义是：线程安全是多个线程访问时，采用了加锁机制，当一个线程访问该类的某个数据时，进行保护，其他线程不能进行访问直到该线程读取结束并且释放了锁，其他线程才可使用，保证了数据的一致性。

可重入的定义是：当一个函数被多次反复调用，其产生的结果不会变。

二者的关系是：可重入函数一定是线程安全的，但线程安全的函数不一定是可重入函数。

1. 改写成可重入函数

int fun1(int\* i)

{

return (\*i) \* 5;

}

int fun2(int\* i)

{

return fun1(i) \* 5;

}

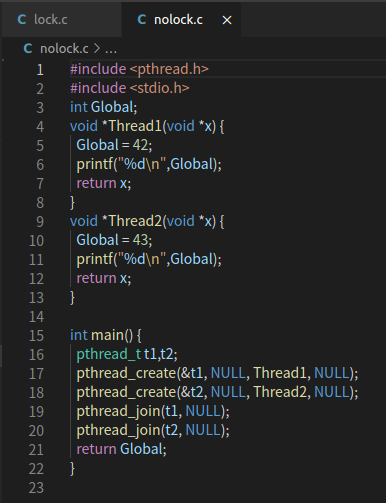
编程题：

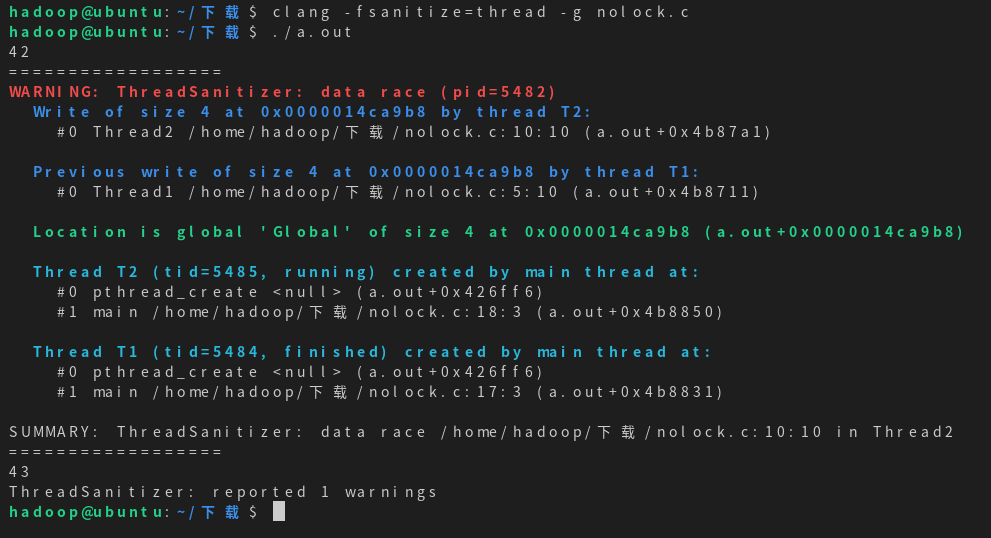
通过工具观察得知，无锁时多线程之间数据竞争明显，而且输出结果与预期不同；有锁时，多线程之间无数据竞争，且输出结果与预期一致。

要求：线程1打印1-3和8-10，线程2打印4-7，按数字顺序打印。

无锁时结果：

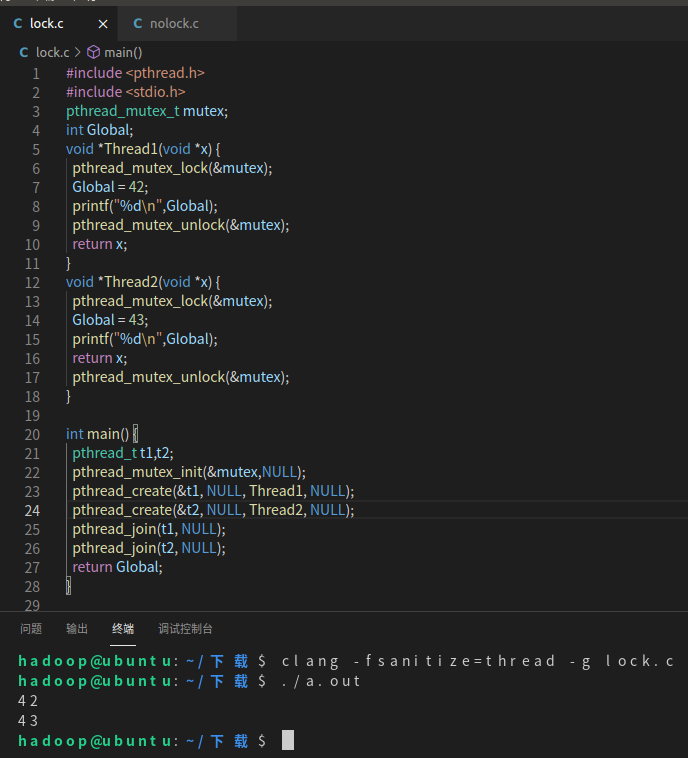
出现数据竞争现象





有锁时结果：

无数据竞争现象



备注：

**$** clang -fsanitize=thread -g lock.c

**$** ./a.out

代码：

Lock.c：

#include <pthread.h>

#include <stdio.h>

pthread\_mutex\_t mutex;

int Global;

void \*Thread1(void \*x) {

  pthread\_mutex\_lock(&mutex);

  Global = 42;

  printf("%d\n",Global);

  pthread\_mutex\_unlock(&mutex);

  return x;

}

void \*Thread2(void \*x) {

  pthread\_mutex\_lock(&mutex);

  Global = 43;

  printf("%d\n",Global);

  return x;

  pthread\_mutex\_unlock(&mutex);

}

int main() {

  pthread\_t t1,t2;

  pthread\_mutex\_init(&mutex,NULL);

  pthread\_create(&t1, NULL, Thread1, NULL);

  pthread\_create(&t2, NULL, Thread2, NULL);

  pthread\_join(t1, NULL);

  pthread\_join(t2, NULL);

  return Global;

}

Nolock.c：

#include <pthread.h>

#include <stdio.h>

int Global;

void \*Thread1(void \*x) {

  Global = 42;

  printf("%d\n",Global);

  return x;

}

void \*Thread2(void \*x) {

  Global = 43;

  printf("%d\n",Global);

  return x;

}

int main() {

  pthread\_t t1,t2;

  pthread\_create(&t1, NULL, Thread1, NULL);

  pthread\_create(&t2, NULL, Thread2, NULL);

  pthread\_join(t1, NULL);

  pthread\_join(t2, NULL);

  return Global;

}