并行与分布式作业

“ 黄炜钊”
第三次作业

姓名：黄炜钊
班级：行政三班
学号：18340066

一、 问题描述

利用 LLVM （C、C++）或者 Soot （Java）等工具检测多线程程序中潜在的数 据竞争以及是否存在不可重入函数，给出案例程序并提交分析报告。

二、 解决方案

编写下面的程序:

```c
#include <pthread.h>
#include <stdio.h>

int Global;

void *Thread1(void *x) {
  Global++;
  return NULL;
}

void *Thread2(void *x) {
  Global--;
  return NULL;
}

int main() {
  pthread_t t[2];
  pthread_create(&t[0], NULL, Thread1, NULL);
  pthread_create(&t[1], NULL, Thread2, NULL);
  pthread_join(t[0], NULL);
  pthread_join(t[1], NULL);
}
```

然后使用 ThreadSanitizer 检测数据竞争，得到实验结果如下图一。

为了解决数据冲突的现象，定义一个互斥量 mutex，使得同一时间只有一个线程在执行临界区的代码，其他线程只能等

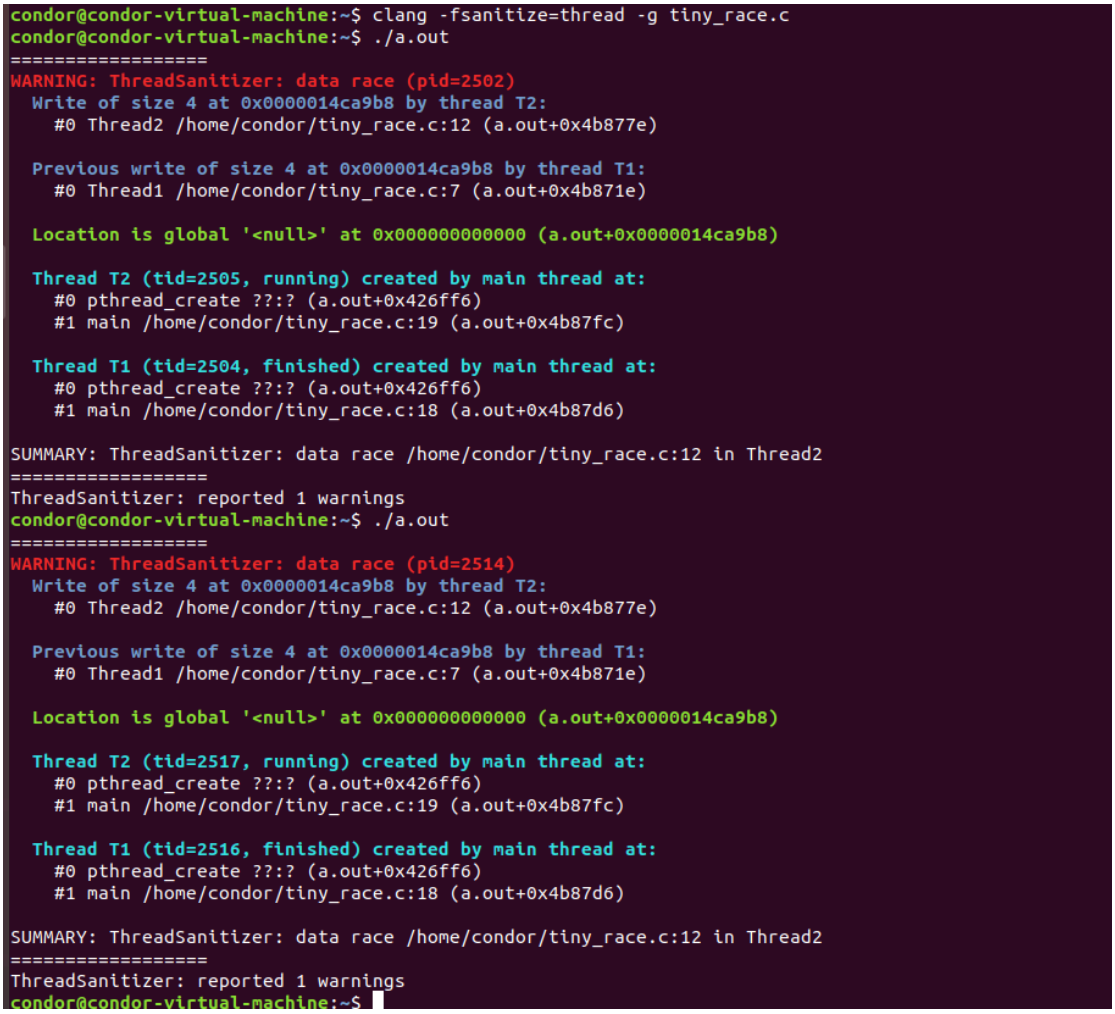
待。等当前线程执行完之后，会从等待的线程中随机选一个进入临界区。修改代码如下：

```c
#include <pthread.h>
#include <stdio.h>

int Global;
pthread_mutex_t mutex;

void *Thread1(void *x) {
  pthread_mutex_lock(&mutex);
  Global++;
  pthread_mutex_unlock(&mutex);
  return NULL;
}

void *Thread2(void *x) {
  pthread_mutex_lock(&mutex);
  Global--;
  pthread_mutex_unlock(&mutex);
  return NULL;
}

int main() {
  pthread_t t[2];
  pthread_create(&t[0], NULL, Thread1, NULL);
  pthread_create(&t[1], NULL, Thread2, NULL);
  pthread_join(t[0], NULL);
  pthread_join(t[1], NULL);

}
```

实验结果见图二。

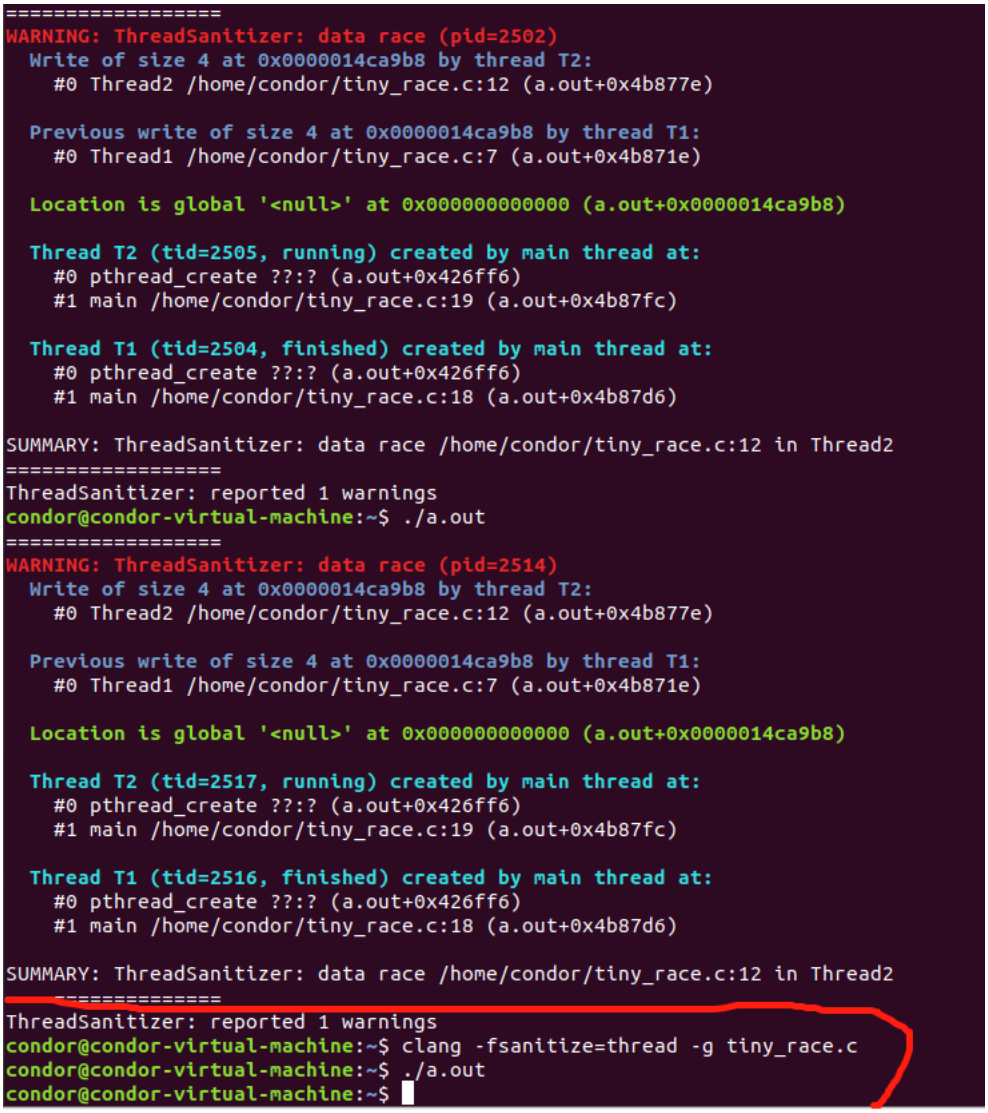
三、 实验结果

图一



可以发现，多线程对全局变量的值的修改确实引起了数据竞争。

图二



可以看到数据竞争的问题已经解决了。