并行与分布式作业

第三次作业

姓名：陈琮昊
学号：18340013
方向：人工智能与大数据

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：

使用老师给出的参考，即 ThreadSanitizer，首先由 LLVM 将程序转为 IR code，然后使用 Clang 编译器结合 ThreadSanitizer 自动生成分析，由此便可得到结果。

三、实验结果：

1. 对全局变量的访问：

根据所给代码文件 3.cpp，其中 sum 为全局变量，也表明 work 函数是不可重入函数。

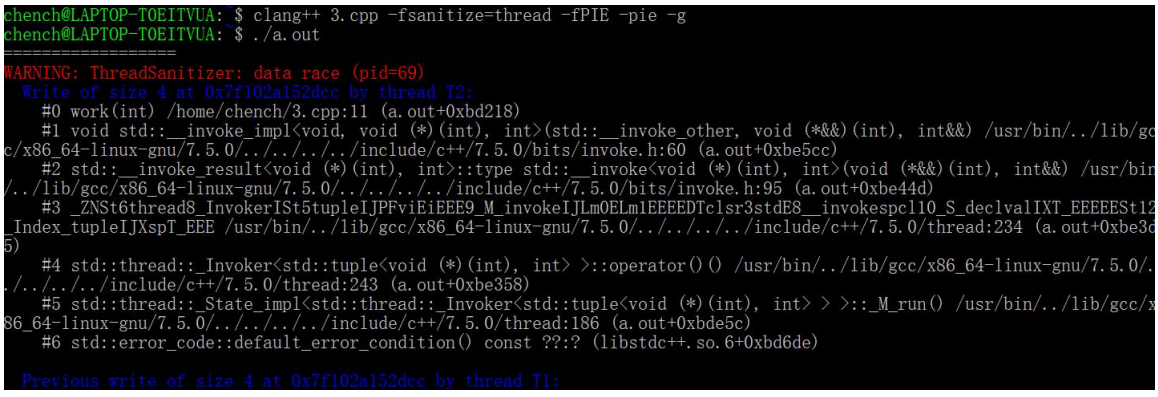
```
int sum = 0;

void work(int index)
{
    for (int i = 0; i < 100; i++) {
        sum++;
    }
}
```

使用如下命令编译：

clang++ test.cpp -fsanitize=thread -fPIE -pie -g

结果如图片所示：



```
  Location is global '<null>' at 0x000000000000 (a.out+0x000001152dcc)

  Thread T2 (tid=72, running) created by main thread at:
    #0 pthread_create ??:? (a.out+0x28ad6)
    #1 std::thread::_M_start_thread(std::unique_ptr<std::thread::_State, std::default_delete<std::thread::_State> >, voi
d (*)()) ??:? (libstdc++.so.6+0xbd994)
    #2 main /home/chench/3.cpp:19 (a.out+0xbd2d3)

  Thread T1 (tid=71, finished) created by main thread at:
    #0 pthread_create ??:? (a.out+0x28ad6)
    #1 std::thread::_M_start_thread(std::unique_ptr<std::thread::_State, std::default_delete<std::thread::_State> >, voi
d (*)()) ??:? (libstdc++.so.6+0xbd994)
    #2 main /home/chench/3.cpp:19 (a.out+0xbd2d3)

SUMMARY: ThreadSanitizer: data race /home/chench/3.cpp:11 in work(int)
```

可以看到显示其存在数据竞争及不可重入函数。

2. 加锁：

如果在线程函数访问全局变量前后加锁，即将 work 函数改为如下：

```
int sum = 0;
mutex m;
void work(int index)
{
    for (int i = 0; i < 100; i++) {
        m.lock();
        sum++;
        m.unlock();
    }
}
```

然后进行编译，结果如图片所示：

```
chench@LAPTOP-TOEITVUA: $ ls
3.cpp   cs231n   MPI   P2PChat
chench@LAPTOP-TOEITVUA: $ clang++ 3.cpp -fsanitize=thread -fPIE -pie -g
chench@LAPTOP-TOEITVUA: $ ls
3.cpp   a.out   cs231n   MPI   P2PChat
chench@LAPTOP-TOEITVUA: $ ./a.out
200
```

可以看到没有 warning，即不存在冲突。

四、遇到的问题及解决办法：

无。