

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

一、问题描述

Homework-3

简答题：

1. 并行编程主要有哪些模型，各自的特点是什么？
2. 多线程编程中，可重入和线程安全的定义以及关系是什么？
3. 将以下函数改成可重复函数；

```
int i;
int fun1()
{
    return i * 5;
}
int fun2()
{
    return fun1() * 5;
}
```

编程题：

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，并对比加上锁以后检测结果的变化，分析及给出案例程序并提交分析报告。

基本思路：

1. 编写具有数据竞争的多线程程序（C或者Java）；
2. 利用LLVM或者Soot将C或者Java程序编译成字节码；
3. 利用LLVM或者soot提供的数据竞争检测工具检测；
4. 对有数据竞争的地方加锁，观察检测结果；

参考：<http://clang.llvm.org/docs/ThreadSanitizer.html>。

二、解决方案

简答题：

1. 并行编程的模型有三个：

(1) 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。

(2) 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

(3) 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。

2. 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。

3. 改写成可重入函数

```
int fun1(int* i)
{
    return (*i) * 5;
}
int fun2(int* i)
{
    return fun1(i) * 5;
}
```

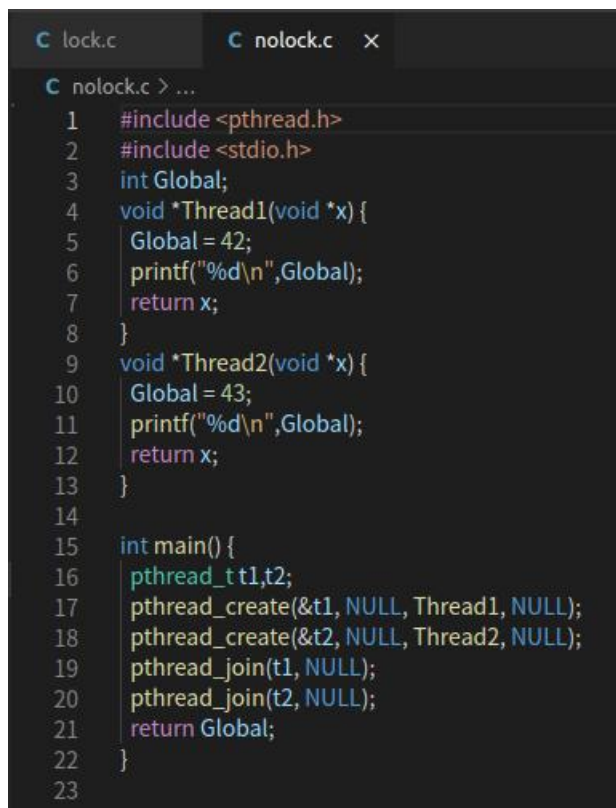
编程题：

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



```
lock.c  nolock.c x
C nolock.c > ...
1  #include <pthread.h>
2  #include <stdio.h>
3  int Global;
4  void *Thread1(void *x) {
5      Global = 42;
6      printf("%d\n",Global);
7      return x;
8  }
9  void *Thread2(void *x) {
10     Global = 43;
11     printf("%d\n",Global);
12     return x;
13 }
14
15 int main() {
16     pthread_t t1,t2;
17     pthread_create(&t1, NULL, Thread1, NULL);
18     pthread_create(&t2, NULL, Thread2, NULL);
19     pthread_join(t1, NULL);
20     pthread_join(t2, NULL);
21     return Global;
22 }
23
```

```

hadoop@ubuntu: ~/下载 $ clang -fsanitize=thread -g noloack.c
hadoop@ubuntu: ~/下载 $ ./a.out
42
=====
WARNING: ThreadSanitizer: data race (pid=5482)
  Write of size 4 at 0x0000014ca9b8 by thread T2:
    #0 Thread2 /home/hadoop/下载/noloack.c:10:10 (a.out+0x4b87a1)

  Previous write of size 4 at 0x0000014ca9b8 by thread T1:
    #0 Thread1 /home/hadoop/下载/noloack.c:5:10 (a.out+0x4b8711)

  Location is global 'Global' of size 4 at 0x0000014ca9b8 (a.out+0x0000014ca9b8)

  Thread T2 (tid=5485, running) created by main thread at:
    #0 pthread_create <null> (a.out+0x426ff6)
    #1 main /home/hadoop/下载/noloack.c:18:3 (a.out+0x4b8850)

  Thread T1 (tid=5484, finished) created by main thread at:
    #0 pthread_create <null> (a.out+0x426ff6)
    #1 main /home/hadoop/下载/noloack.c:17:3 (a.out+0x4b8831)

SUMMARY: ThreadSanitizer: data race /home/hadoop/下载/noloack.c:10:10 in Thread2
=====
43
ThreadSanitizer: reported 1 warnings
hadoop@ubuntu: ~/下载 $ █

```

有锁时结果：

无数据竞争现象

```
C lock.c  x  C nlock.c
C lock.c > main()
1  #include <pthread.h>
2  #include <stdio.h>
3  pthread_mutex_t mutex;
4  int Global;
5  void *Thread1(void *x) {
6      pthread_mutex_lock(&mutex);
7      Global = 42;
8      printf("%d\n", Global);
9      pthread_mutex_unlock(&mutex);
10     return x;
11 }
12 void *Thread2(void *x) {
13     pthread_mutex_lock(&mutex);
14     Global = 43;
15     printf("%d\n", Global);
16     return x;
17     pthread_mutex_unlock(&mutex);
18 }
19
20 int main() {
21     pthread_t t1, t2;
22     pthread_mutex_init(&mutex, NULL);
23     pthread_create(&t1, NULL, Thread1, NULL);
24     pthread_create(&t2, NULL, Thread2, NULL);
25     pthread_join(t1, NULL);
26     pthread_join(t2, NULL);
27     return Global;
28 }
29

问题  输出  终端  调试控制台
hadoop@ubuntu: ~/下载 $ clang -fsanitize=thread -g lock.c
hadoop@ubuntu: ~/下载 $ ./a.out
42
43
hadoop@ubuntu: ~/下载 $
```

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;
}
```