实验3 系统分析工具 Valgrind

背景介绍

1. Valgrind是一个开源的内存调试工具，用于检测内存泄漏、内存错误、线程问题等。它最初是由 Julian Seward 在 1999 年开发的。
2. Valgrind提供了一系列工具，包括Memcheck、Cachegrind、Helgrind等。
3. 在本实验中，同学们将学习如何使用 Helgrind 工具进行多线程程序的检测，以及如何使用锁等同步机制来解决数据竞争问题。通过本实验，同学们可以深入了解并发编程的挑战和复杂性，发展计算机系统设计思维，培养解决并发和互斥问题的能力。

Valgrind介绍

- Valgrind 提供了一系列工具，包括 Memcheck、Cachegrind、Helgrind 等。
  - Memcheck：检查内存访问错误，如内存泄漏、未初始化的内存使用等。
  - Cachegrind：缓存分析工具，用于模拟和分析程序对CPU缓存的使用情况，可以检测缓存命中、缓存未命中和缓存冲突。
  - Helgrind：检测并发编程中的线程同步问题，可以检测出数据竞争、死锁等常见的并发错误。
- 如果使用自己的虚拟机，则可用以下命令安装：
  - sudo apt-get install valgrind

实验环境

1. 软件环境：Linux操作系统，多核处理器

如果报错： WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!

执行： ssh-keygen -R "[10.140.32.159]:分配的服务器端口号"

清除原有信息后即可成功连接

2. 要求：

① 请大家拿到后先修改登录口令，不能使用其他人的账号登录，也不得替他人登录练习，严禁使用账号进行与学习无关或攻击他人及恶意消耗系统资源等行为，一旦发现，按违规处理，后果严重；

② 大家认真进行练习，练习量将作为实验课达标的依据之一（不评优劣，达到基本练习量即可）

实验目标

1. 了解多线程编程，理解Race Condition问题；

2.掌握Valgrind的使用，学会使用Valgrind检测内存问题；

3. 理解如何使用锁解决互斥问题。

实验安排

知识介绍与环境配置		10分钟
实验内容 1		25分钟
实验内容 2		40分钟
实验内容 3		10分钟
实验内容 4		10分钟
布置课下实验		5分钟

实验内容1 多线程编程

1. 编译ph.c，并单线程运行

> gcc -g -o ph ph.c -pthread

> ./ph 1

实验说明：

1）ph程序的参数表示ph程序对hash表执行put和get操作时所启用的线程数量。1就表示单线程。

2）程序在put阶段，将NKEYS插入到哈希表中，在get阶段将它们全部查找出来。

3）观察：运行后我们能看到两个阶段的运行时间，观察并记录。

实验内容1 多线程编程

2. 使用2个线程运行ph程序

> ./ph 2

实验说明：

1）思考：

代码使用C语言编写，C语言是底层系统编程默认的语言，pthreads函数库可用于创建多线程，在多核环境下多线程可以并行运行。

当用2个线程运行ph时，在put和get阶段都是2线程在并行操作。

在理想的并行下，2个线程应该都花一半时间完成1/2的任务。

2）观察：结果是这样吗？两个阶段的耗时又有何差别呢？

实验内容1 多线程编程

3. 实验结果

completion time for put phase = ?

completion time for get phase = ?

实验说明：

1）观察：我们看到的结果可能与期待的有挺大差距。失望……

2）思考：主要问题发生在哪个阶段？到底是隐藏着什么问题呢……

3）分析：继续观察输出，“xx keys missing”有何含义？阅读源代码中的put函数和get函数，再阅读get_thread函数的最后5行代码，理解“xx keys missing”的含义。

4）思考：keys missing为什么会发生呢？**

标注的内容在实验练习题上作答。**

实验内容1 多线程编程

4. 使用4个线程运行ph.c文件

> ./ph 4

实验说明：

1）在思考并做出简单的分析之后，使用4个线程运行ph。

2）观察：

4线程运行ph的结果是否与你的分析相吻合呢?

key丢失的现象是否有所变化?

实验内容2：使用Valgrind进行检测

使用Valgrind检测竞态条件（race condition）

> valgrind --tool=helgrind ./ph 2

实验说明：

1）ph.c程序的问题可能很容易通过人工检查发现，但对于复杂的程序就需要借助工具来自动发现问题了。

2）helgrind就可以用于发现这类问题，它的思路与我们的文献阅读“Eraser”相同，也是Valgrind工具箱中的主要组件之一。

实验内容2：使用Valgrind进行检测

探索研究**

1. 研究helgrind输出和ph.c程序，找出put()函数的问题原因
2. 用互斥锁在put()函数中插入lock和unlock语句，确保没有遗漏的键值
3. 重新编译ph.c文件，并使用Valgrind检测代码是否正确

讨论**

为什么这些更改能确保正确性?

标注的内容在实验练习题上作答。**

实验内容3：put操作

实验要求

- 多次运行修改后的ph程序（避免偶然误差）

讨论

- 对于修改后的put操作，双线程版本是否比单线程版本更快？为什么？**

实验内容4：get操作

实验要求

- 从ph.c中的get()中删除锁，重新编译并重新运行程序

讨论

- 为什么ph的get阶段可以在多核上加速？**

课后实验内容及思考

- 拓展学习（根据个人情况进行学习）

1. Valgrind还可以做什么？了解Memcheck、Cachegrind等工具的使用，并尝试用它们解决问题
2. Valgrind的最初作者Julian Seward于2006年由于Valgrind上的工作获得了第二届Google-O'Reilly开源代码奖。了解一下该奖励颁发给了哪些人的哪些成果?
3. Valgrind遵守GNU的GPL开源协议。了解该协议，以及与其他协议的不同。