《操作系统原理》实验报告

------实验三 Barriers

年级：2022 级　班级：智算大类十二班 学号：3523403837 姓名：邢俊杰

一、实验题目及要求

(1)题目：实现基于 pthread 库的屏障（Barriers）

(2)要求：在给定的代码框架中，实现一个能够通过条件变量和互斥锁实现的屏障功能。确保多个线程能够同步等待在特定的屏障点，直到所有线程到达之后才继续执行。

二、设计说明

实验中使用了 pthread 库，其中实现了一个基本的屏障机制，依赖互斥锁和条件变量来同步多个线程。屏障在所有指定线程到达后释放，允许它们继续执行。

三、编译、运行、测试说明

编译：$ gcc -g -O2 -pthread barrier.c

运行：$ ./a.out 2

测试说明：

测试使用不同数量的线程（例如，1 个、2 个、多个）来验证屏障的正常功能。通过观察输出和验证断言，确认是否所有线程都正确地在屏障点同步等待，并且在达到指定线程数后继续执行。

四、实验过程

（1）代码分析： 分析已给代码框架，理解互斥锁、条件变量的用法以及屏障功能的实现逻辑。

（2）修改代码： 编辑 barrier() 函数实现屏障功能，确保多个线程在达到指定数目后同步等待。

在该函数中首先获取互斥锁，确保在修改共享状态之前只有一个线程能够访问该代码块。在进入屏障的线程数加一，通过 if 检查是否所有线程都已经到达屏障，即 nthread 数量的线程已经达到屏障。如果达到了，重置 bstate.nthread 为 0，表示下一轮的屏障。增加 bstate.round，表示屏障的轮数已经加一。调用 pthread_cond_broadcast(&bstate.barrier_cond);，唤醒所有等待在条件变量上的线程，使它们继续执行。如果还没有达到指定的线程数量，代码将进入 else 部分：int current_round = bstate.round;：记录当前的屏障轮数。

while (current_round == bstate.round)：线程进入循环等待状态，使用 pthread_cond_wait 阻塞并等待条件变量的信号。

当另一个线程进入屏障并调用 pthread_cond_broadcast 时，唤醒了所有等待的线程。

唤醒后，线程将会检查 while 循环中的条件。如果 current_round == bstate.round 条件不再满足，说明屏障轮数已经更新，线程将退出 while 循环，解除阻塞状态，继续执行后续代码。

pthread_mutex_unlock(&bstate.barrier_mutex);：最后释放互斥锁，允许其他线程访问该屏障。

```
static void
barrier()
{
  pthread_mutex_lock(&bstate.barrier_mutex);
  bstate.nthread++;
  if (bstate.nthread >= nthread) {
    bstate.nthread = 0;
    bstate.round++;
    pthread_cond_broadcast(&bstate.barrier_cond);
  } else {
    int current_round = bstate.round;
    while (current_round == bstate.round) {
      pthread_cond_wait(&bstate.barrier_cond, &bstate.barrier_mutex);
    }
  }
  pthread_mutex_unlock(&bstate.barrier_mutex);
}
```

（3）编译运行： 使用给定的编译指令编译代码，运行程序，并在不同线程数量下进行测试。

```
[xingjunjie@localhost 作业3 Barriers]$ gcc -g -O2 -pthread barrier.c
[xingjunjie@localhost 作业3 Barriers]$ ls
a.out  barrier.c  作业3 Barriers.docx  作业3 Barriers.pdf
[xingjunjie@localhost 作业3 Barriers]$ ./a.out 2
```

（4）调试和验证： 观察输出信息，检查是否满足预期的同步等待效果。验证是否所有线程都能够在指定的屏障点同步等待。

```
a.out  barrier.c  作业3 Barriers.docx  作业3 Barriers.pdf
[xingjunjie@localhost 作业3 Barriers]$ ./a.out 2
OK; passed
```

五、实验总结

本次实验基于 pthread 库实现了一个简单的屏障机制。通过互斥锁和条件变量，所有线程能够在达到指定的屏障点同步等待。通过本次实验，加深了对线程同步机制和并发编程的理解，了解了互斥锁和条件变量在多线程编程中的作用。同时，实验过程中也加深了对多线程并发编程中同步机制的实际应用和调试方法的认识。

实验中还需要考虑更多边缘情况和并发问题，比如性能考量、死锁、饥饿等，并实施更全面的测试，以验证实现的正确性和健壮性。

完成日期：2023 年 10 月 28 日