shell

- 姓名：陈姝宇
- 学号：3017218119
- 班级：软工3班

实验内容

在本作业中，我们将探讨如何使用pthread库提供的条件变量来实现屏障。屏障是应用程序中所有线程都必须等待所有其他线程也到达该点的点。条件变量是一种序列协调技术，类似于xv6的睡眠和唤醒。

实验步骤

下载barrier.c，然后在我的电脑上编译:

```
➜  barriers gcc -g -O2 -pthread barrier.c
➜  barriers ./a.out 2
a.out: barrier.c:42: thread: Assertion `i == t' failed.
[1]    29356 abort (core dumped)  ./a.out 2
```

2指定在barrier上同步的线程数(barrier.c中的nthread)。每个线程都处于一个紧密的循环中。在每个循环迭代中，一个线程调用barrier()，然后休眠若干微秒。assert触发，因为一个线程在另一个线程到达barrier之前离开barrier。理想的行为是所有线程都应该阻塞，直到nthread调用barrier(这里的nthread是不是就是线程号i==nthread的那个线程?错，应该是本轮调用barrier的线程总数bstate.nthread)。

你的目标是实现这理想的行为。除了您以前见过的锁原语之外，您还需要以下新的pthread原语(有关详细信息，请参阅man pthreads):

```
pthread_cond_wait(&cond, &mutex);  // go to sleep on cond,
releasing lock mutex
pthread_cond_broadcast(&cond);     // wake up every thread
sleeping on cond
```

pthread_cond_wait在被调用时释放mutex，并在返回前重新获取mutex。

我们已经给出了barrier_init()。您的工作是实现barrier()，这样就不会发生恐慌。我们已经为你们定义了struct barrier;它的字段供您使用。

有两个问题会使你的任务复杂化:

你必须处理一连串的barrier calls，每一连串我们都称之为一轮(这里的一轮是指所有的线程都调用一次barrier?对)。bstate.round记录了当前轮。每个轮开始时你都应该增加bstate.round。 您必须处理这样一种情况：一个线程在其他线程退出barrier之前绕着循环运行。特别是，你正在重新使用bstate.nthread从一轮到下一轮。确保离开barrier并绕循环运行的线程不会增加bstate.nthread，当上一轮仍然在使用它。

```
27 static void
28 barrier()
29 {
30   pthread_mutex_lock(&bstate.barrier_mutex);
31   bstate.nthread++;
32   printf("in round %d as %d\n ",bstate.round, bstate.nthread);
33   if(bstate.nthread!=nthread)
34     pthread_cond_wait(&bstate.barrier_cond, &bstate.barrier_mutex);
35   else{
36     bstate.round++;
37     bstate.nthread=0;
38     //printf("The %d start\n",bstate.round);
39     //pthread_mutex_unlock(&bstate.barrier_mutex); why can't?
40     pthread_cond_broadcast(&bstate.barrier_cond);
41     //pthread_mutex_unlock(&bstate.barrier_mutex); why can't?
42   }
43   pthread_mutex_unlock(&bstate.barrier_mutex);
44 }
45
```

实验结果

设置线程数为1

```
in round 5653 as 1
in round 5654 as 1
in round 5655 as 1
in round 5656 as 1
in round 5657 as 1
in round 5658 as 1
in round 5659 as 1
in round 5660 as 1
in round 5661 as 1
in round 5662 as 1
in round 5663 as 1
in round 5664 as 1
in round 5665 as 1
in round 5666 as 1
in round 5667 as 1
```

设置线程数为2

```
in round 6583 as 1
in round 6583 as 2
in round 6584 as 1
in round 6584 as 2
in round 6585 as 1
in round 6585 as 2
in round 6586 as 1
in round 6586 as 2
in round 6587 as 1
in round 6587 as 2
in round 6588 as 1
in round 6588 as 2
in round 6589 as 1
in round 6589 as 2
```

设置线程数为3

```
 in round 9635 as 2
 in round 9635 as 3
 in round 9636 as 1
 in round 9636 as 2
 in round 9636 as 3
 in round 9637 as 1
 in round 9637 as 2
 in round 9637 as 3
 in round 9638 as 1
 in round 9638 as 2
 in round 9638 as 3
 in round 9639 as 1
 in round 9639 as 2
 in round 9639 as 3
^C
```