哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验设计报告

开课学期：2022 年秋季

课程名称：操作系统

实验名称：Thread

实验性质：额外实验

实验时间：10.30 地点：T2

学生班级：11

学生学号：200111132

学生姓名：吴桐

评阅教师：

报告成绩：

实验与创新实践教育中心印制

2022 年 9 月

一、 实验详细设计

注意不要照搬实验指导书上的内容，请根据你自己的设计方案来填写

1. Uthread: switching between threads

自己手动写切换线程的时候，如何交换上下文信息。

自己先写好上下文交换的数据结构（内存中）context：内容主要是不易失寄存器的内容。

在 thread_create 中为 context 的 ra 附上传入函数地址，sp 为栈顶

在 thread_schedule 中，调用 `thread_switch((uint64)&t->context, (uint64)&current_thread->context);`
交换两个进程的 context。

这个函数实现是利用汇编代码写的：uthread_switch.S

主要利用 sd 和 ld，利用我们传入的数据结构 context 各个属性计算偏移的位置，把寄存器的值传递到下一个 context。

2. Using threads

将锁转为锁数组 locks[NBUCKET]

由于是对一个哈希桶的操作，所以我们只需要对每个桶的操作互斥即可。这里相当于改成在每次操作的时候进行判断，只要自己哈希对应的锁没有被拿到，就可以操作。而不在抢占一个大锁。

还有初始化的时候用循环初始化每一个 lock

3. Barrier

Emmm 其实就是写了一个计数器一样的东西，不过使用了🔒进行保护一下。然后根据题目提示进行 broadcast 等操作而已。

二、 实验结果截图


make[1]: Leaving directory '/root/myxv6/github/MIT6.S081-2020-labs'
== Test uthread ==
$ make qemu-gdb
uthread: OK (3.7s)
== Test answers-thread.txt == answers-thread.txt: OK
== Test ph_safe == make[1]: Entering directory '/root/myxv6/github/MIT6.S081-2020-labs'
gcc -o ph -g -O2 notxv6/ph.c -pthread
make[1]: Leaving directory '/root/myxv6/github/MIT6.S081-2020-labs'
ph_safe: OK (9.3s)
== Test ph_fast == make[1]: Entering directory '/root/myxv6/github/MIT6.S081-2020-labs'
make[1]: 'ph' is up to date.
make[1]: Leaving directory '/root/myxv6/github/MIT6.S081-2020-labs'
ph_fast: OK (19.6s)
== Test barrier == make[1]: Entering directory '/root/myxv6/github/MIT6.S081-2020-labs'
gcc -o barrier -g -O2 notxv6/barrier.c -pthread
make[1]: Leaving directory '/root/myxv6/github/MIT6.S081-2020-labs'
barrier: OK (2.8s)
== Test time ==
time: OK
Score: 60/60