分布式上机作业

1 小组成员

冯吕　　201928013229158
刘丁玮　2019E8013261029
孙佳钰　2019E8013261049
解伟凡　201928013229128

2 快照算法

- 源程序：*snapshot/snapshot.c*
- 编译运行：*./snapshot/brun.sh*
- 说明：两个进程 p 和 q 通过两个通道轮转消息 *message*，当进程 p 状态为 101，即收到消息 101 次，之后把消息再次发送出去，然后开始快照算法，算法会记录两个进程的状态和通道状态。

3 锁服务

- 锁服务实现代码：*LockService/include*, *LockService/src*
- 编译：*./LockService/build.sh r*

3.1 API

3.1.1 锁服务器

```
#include <iostream>

#include <LockServer.h>

int main()
{
    LockServer ls(8080, 5);
    ls.init();
    ls.run();
}
```

锁服务器的创建非常简单。例如，上面的代码创建了一个锁服务器 *ls*，第一个参数为服务器监听的端口号，第二个参数为可选参数，设置服务器的工作线程数目为 5，默认值为 2；*ls.init*() 进行服务器初始化工作；*ls.run*() 开始运行服务器。

3.1.2 锁客户端

```
1  #include <iostream>
2  #include <string>
3  #include <thread>
4
5  #include "LockClient.h"
6
7  int share_number = 0;
8  LockID lock_id = 1;
9  void worker(PID pid)
10 {
11     LockClient lc("localhost", 8080);
12     Status s;
13     for (size_t i = 0; i < 10; ++i)
14     {
15         s = lc.acquire(lock_id, pid);
16
17         std::cout << "Pid = " << pid << ", "
18                   << "share_number = " << share_number << std::endl;
19         ++share_number;
20
21         s = lc.release(lock_id, pid);
22     }
23 }
24
25
26 int main()
27 {
28     std::thread t1(worker, 1);
29     std::thread t2(worker, 2);
30     std::thread t3(worker, 3);
31     t1.join();
32     t2.join();
33     t3.join();
34     std::cout << "The final value of share_number is "
35               << share_number << std::endl;
36     return 0;
37 }
```

锁客户端的创建同样非常简单，如上述第 11 行代码所示，创建 client 时需要指定服务器的 *ip* 地址和端口号，上面的代码指定服务器为本地服务器，端口号为 8080。Client 提供两个方法：*acquire* 和 *release*。

- *acquire*：申请锁，参数为锁 *id* 和进程 *id*；
- *release*：释放锁，参数同样为锁 *id* 和进程 *id*。

上述代码为三个线程通过锁服务互斥访问一个共享变量，每次访问将共享变量的值加一。上述程序运行的结果如下图所示：

```
fenglv@segmentfault:~/上机作业/LockService/build/bin$ ./client
Pid = 1, share_number = 0
Pid = 2, share_number = 1
Pid = 3, share_number = 2
Pid = 1, share_number = 3
Pid = 3, share_number = 4
Pid = 2, share_number = 5
Pid = 1, share_number = 6
Pid = 3, share_number = 7
Pid = 2, share_number = 8
Pid = 1, share_number = 9
Pid = 2, share_number = 10
Pid = 3, share_number = 11
Pid = 1, share_number = 12
Pid = 2, share_number = 13
Pid = 3, share_number = 14
Pid = 1, share_number = 15
Pid = 2, share_number = 16
Pid = 3, share_number = 17
Pid = 1, share_number = 18
Pid = 2, share_number = 19
Pid = 3, share_number = 20
Pid = 1, share_number = 21
Pid = 2, share_number = 22
Pid = 3, share_number = 23
Pid = 1, share_number = 24
Pid = 2, share_number = 25
Pid = 1, share_number = 26
Pid = 2, share_number = 27
Pid = 3, share_number = 28
Pid = 3, share_number = 29
The final value of share_number is 30
```