

Feb 04, 13 19:28

l01.go

Page 1/4

```

1 //
2 // *** a simple application using the lock service
3 //
4
5 func main() {
6     primary_port := os.Args[1]
7     backup_port := os.Args[2]
8     clerk := lockservice.MakeClerk(primary_port,
9                                     backup_port)
10
11     for clerk.Lock("car keys") == false {
12         // wait
13     }
14
15     // it's my turn to drive the car...
16
17     clerk.Unlock("car keys")
18 }
19

```

Feb 04, 13 19:28

l01.go

Page 2/4

```

19 //
20 // *** client.go -- the application calls
21 // these library "stubs"
22 //
23
24 type Clerk struct {
25     servers [2]string // primary port, backup port
26 }
27
28 func MakeClerk(primary string, backup string) *Clerk {
29     ck := new(Clerk)
30     ck.servers[0] = primary
31     ck.servers[1] = backup
32     return ck
33 }
34
35 //
36 // ask the lock service for a lock.
37 // returns true if the lock service
38 // granted the lock, false otherwise.
39 //
40 func (ck *Clerk) Lock(lockname string) bool {
41     args := &LockArgs{} // RPC arguments
42     args.Lockname = lockname
43     var reply LockReply // space for RPC reply
44
45     // send an RPC request, wait for the reply.
46     ok := call(ck.servers[0], "LockServer.Lock",
47               args, &reply)
48     return ok && reply.OK
49 }
50

```

Feb 04, 13 19:28

l01.go

Page 3/4

```

50 //
51 // *** server.go
52 //
53 //
54 //
55 // a lock server's state
56 //
57 type LockServer struct {
58     mu sync.Mutex
59     l net.Listener
60
61     am_primary bool // am I the primary?
62     backup string // backup's port
63
64     // for each lock name, is it locked?
65     locks map[string]bool
66 }
67
68 //
69 // server Lock() RPC handler
70 //
71 //
72 func (ls *LockServer) Lock(args *LockArgs,
73                          reply *LockReply) error {
74     ls.mu.Lock()
75     defer ls.mu.Unlock()
76
77     locked, _ := ls.locks[args.Lockname]
78
79     if locked {
80         reply.OK = false
81     } else {
82         reply.OK = true
83         ls.locks[args.Lockname] = true
84     }
85
86     return nil
87 }
88

```

Feb 04, 13 19:28

l01.go

Page 4/4

```

88 //
89 // start a lock server
90 //
91 func StartServer(primary string, backup string,
92                 am_primary bool) *LockServer {
93     ls := new(LockServer)
94     ls.backup = backup
95     ls.am_primary = am_primary
96     ls.locks = map[string]bool{}
97
98     // tell net/rpc about our RPC server and handlers.
99     rpcs := rpc.NewServer()
100    rpcs.Register(ls)
101
102    my_port := ""
103    if am_primary {
104        my_port = primary
105    } else {
106        my_port = backup
107    }
108
109    // prepare to receive connections from clients.
110    ls.l, _ := net.Listen("unix", my_port);
111
112    // thread to accept RPC connections from clients.
113    go func() {
114        for {
115            conn, _ := ls.l.Accept()
116            go rpcs.ServeConn(conn)
117        }
118    }()
119
120    return ls
121 }

```