```
package main

import (
        "fmt"
        "log"
        "net"
        "net/rpc"
        "sync"
)

//
// RPC request/reply definitions
//

const (
        OK       = "OK"
        ErrNoKey = "ErrNoKey"
)

type Err string

type PutArgs struct {
        Key   string
        Value string
}

type PutReply struct {
        Err Err
}

type GetArgs struct {
        Key string
}

type GetReply struct {
        Err   Err
        Value string
}

//
// Client
//

func connect() *rpc.Client {
        client, err := rpc.Dial("tcp", ":1234")
        if err != nil {
                log.Fatal("dialing:", err)
        }
        return client
}

func get(key string) string {
        client := connect()
        args := GetArgs{"subject"}
        reply := GetReply{}
        err := client.Call("KV.Get", &args, &reply)
        if err != nil {
                log.Fatal("error:", err)
        }
        client.Close()
        return reply.Value
}

func put(key string, val string) {
        client := connect()
        args := PutArgs{"subject", "6.824"}
        reply := PutReply{}
        err := client.Call("KV.Put", &args, &reply)
        if err != nil {
                log.Fatal("error:", err)
        }
        client.Close()
}

//
// Server
```

```
//

type KV struct {
        mu   sync.Mutex
        data map[string]string
}

func server() {
        kv := new(KV)
        kv.data = map[string]string{}
        rpcs := rpc.NewServer()
        rpcs.Register(kv)
        l, e := net.Listen("tcp", ":1234")
        if e != nil {
                log.Fatal("listen error:", e)
        }
        go func() {
                for {
                        conn, err := l.Accept()
                        if err == nil {
                                go rpcs.ServeConn(conn)
                        } else {
                                break
                        }
                }
                l.Close()
        }()
}

func (kv *KV) Get(args *GetArgs, reply *GetReply) error {
        kv.mu.Lock()
        defer kv.mu.Unlock()

        val, ok := kv.data[args.Key]
        if ok {
                reply.Err = OK
                reply.Value = val
        } else {
                reply.Err = ErrNoKey
                reply.Value = ""
        }
        return nil
}

func (kv *KV) Put(args *PutArgs, reply *PutReply) error {
        kv.mu.Lock()
        defer kv.mu.Unlock()

        kv.data[args.Key] = args.Value
        reply.Err = OK
        return nil
}

//
// main
//

func main() {
        server()

        put("subject", "6.824")
        fmt.Printf("Put(subject, 6.824) done\n")
        fmt.Printf("get(subject) -> %s\n", get("subject"))
}
```