go+区块链培训 讲师:张长志

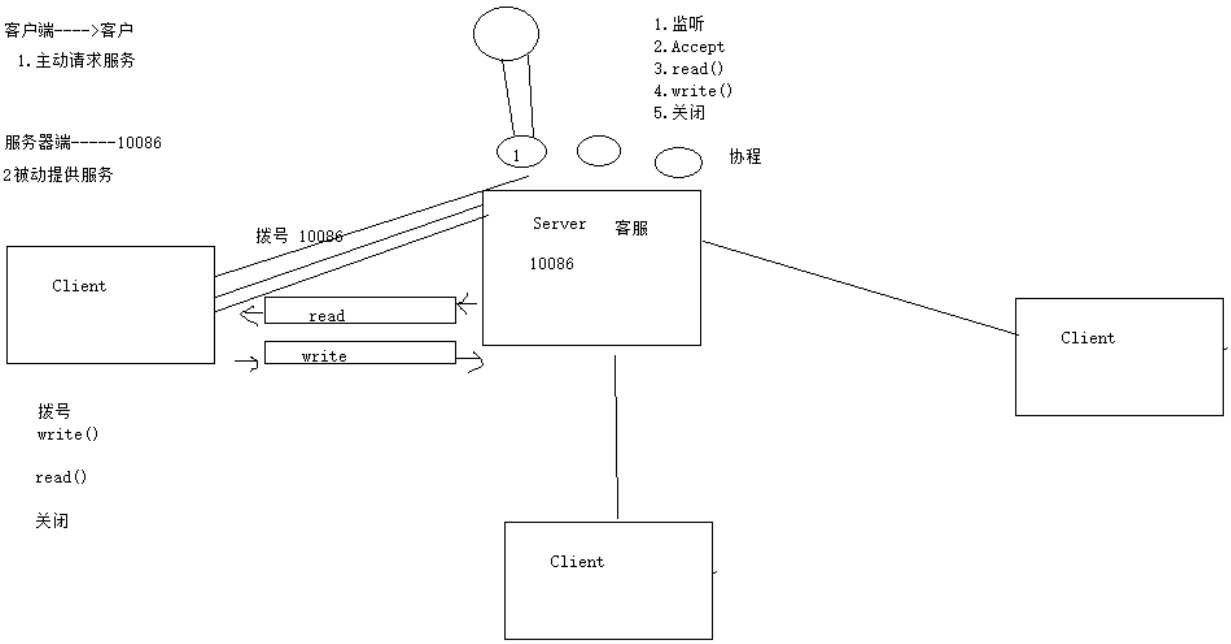
socket

什么是socket编程

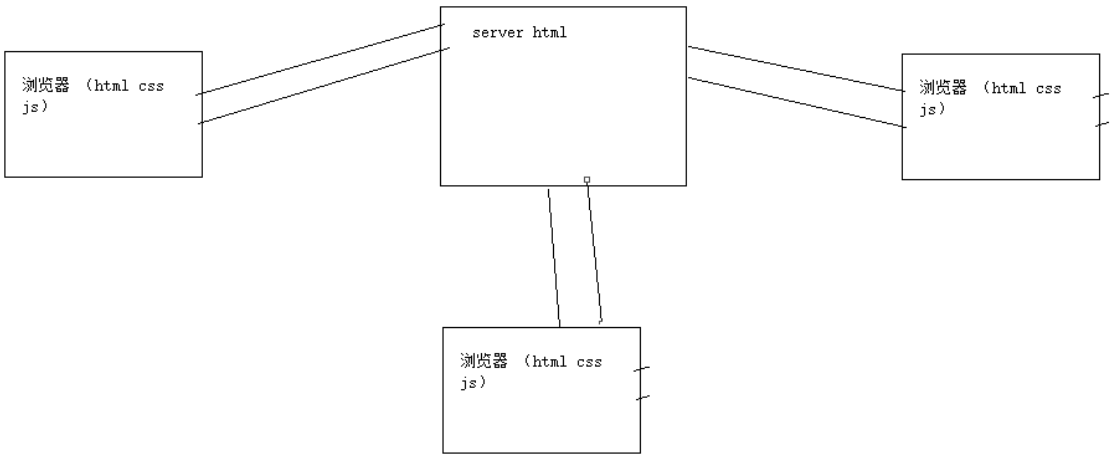
socket起源于Unix 而Unix基本哲学是**一切皆文件**，可以用 “打开 open--->读写write/read--->关闭close”模式来操作。Socket就是该模式一个实现。网络的socket数据传输其实就是一种特殊的I/o.Socket也是一种文件描述符，socket也是具有一个类似文件打开 函数读写调用 文件关闭操作。 Socket() 该函数返回一个整体的socket描述符，随后建立连接 数据传输等操作都是通过socket实现。

socket分类：流式Socket(SOCK_STREAM)和数据报式socket（SOCK_DGRAM）.流式Socket是一种面向连接的socket，准对TCP服务应用， 数据报式socket 是一种无连接的socket，对应UDP。

CS模型和BS模型

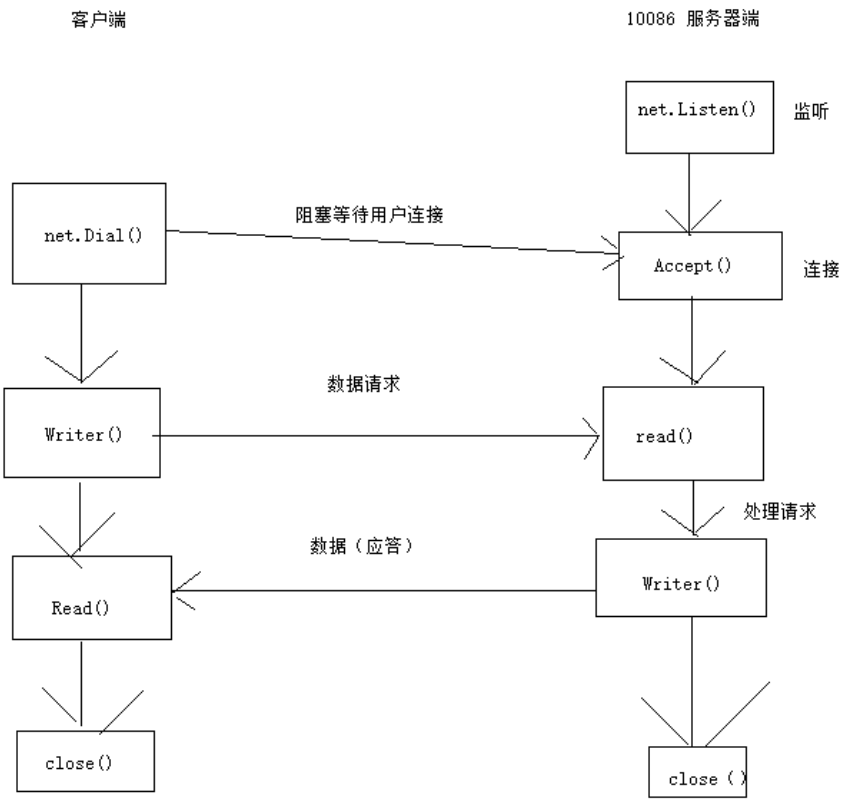


BS



TCP模型

socket TCP模型



服务器端代码

http://docscn.studygolang.com/pkg/net/

```
package main
```

```
import (
   "net"
   "fmt"
)

/**
服务器端代码
 */
func main(){
   listener,err := net.Listen("tcp","127.0.0.1:8000")
   if err != nil{
         fmt.Println("err=",err)
         return
   }
   defer listener.Close()

   //阻塞等待用户的连接
  conn,err := listener.Accept()
  if err!=nil{
   fmt.Println("err=",err)
   return
  }

  //接收用户的请求
  buf := make([]byte,1024) //1024大小进行缓冲

   n,err :=conn.Read(buf)
   if err !=nil{
      fmt.Println("err=",err)
      return
   }
   fmt.Println("buf=",string(buf[:n]))
   defer  conn.Close()



}
```

多客户端读写案例

```
package main

import (
   "net"
   "fmt"

   "strings"
)
```

```
func HandleConn(conn net.Conn)  {
   //函数调用完毕，自动关闭conn
   defer conn.Close()

   //获取客户端网络地址信息
   addr := conn.RemoteAddr().String()
   fmt.Println(addr," connect sucessful")
   buf := make([]byte,2048)

   for{
       n,err := conn.Read(buf)
       if err != nil{
          fmt.Println("err=",err)
          return
      }
      fmt.Printf("[%s]:%s\n",addr,string(buf[:n]))
      fmt.Println("len=",len(string(buf[:n])))

      if "exit" == string(buf[0:n-2]) {  //"\r\n"
         fmt.Println(addr,"exit")
         return
      }

      //把用户小写转换成大写 在发送给用户
      conn.Write([]byte(strings.ToUpper(string(buf[:n]))))
   }



}

func main()  {
   //监听
   listener,err := net.Listen("tcp","127.0.0.1:8000")
   if err != nil{
      fmt.Println("err=",err)
      return
   }
   defer listener.Close()
   //接收多个用户
   for {
         conn,err := listener.Accept()
         if err != nil{
          fmt.Println("err=",err)
          return
     }
     //处理用户请求，新建立一个协程
     go  HandleConn(conn)
   }

}
```

```go
package main

import (
    "net"
    "fmt"
    "os"
)

func main()  {
    //主动连接服务器
    conn,err := net.Dial("tcp","127.0.0.1:8000")
    if err != nil{
        fmt.Println("net.dial err=",err)
        return
    }
    defer conn.Close()

    //从键盘输入 我给它一个协程

    go func() {
        //从键盘输入，给服务器发送内容
        str := make([]byte,1024)
        for{
            n,err := os.Stdin.Read(str) //从键盘输入，放入到str里面
            if err != nil{
                fmt.Println("os.stdin.err=",err)
                return
            }
            //把内容发给服务器
            conn.Write(str[:n])
        }
    }()

    //接收服务器回复的数据
    buf := make([]byte,1024)
    for{
        n,err := conn.Read(buf) //接收服务器请求
        if err !=nil{
            fmt.Println("conn.Read err=",err)
            return
        }
        fmt.Println(string(buf[0:n]))
    }

}
```

获取文件属性

```go
package main

import (
   "os"
   "fmt"
)

func main()  {
   list := os.Args

   fmt.Println(len(list))

   if len(list) != 2{
      fmt.Println("useage:xxx file")
      return
   }
   fileName := list[1]

   info,err := os.Stat(fileName)
   if err != nil{
      fmt.Println("err=",err)
      return
   }

   fmt.Println("name=",info.Name())
   fmt.Println("size=",info.Size())
}
```