Platinumc

命令行的解析

```
var name string
flag.StringVar(&name,"n","Client name")
var age uint
flag.UintVar(&age,"a","Client age")
flag.Parse()//Must be called after all flags are defined and before flags are accessed by the program.
```

单元测试

```
1:命名的格式
  name_test.go
2：函数的命名格式
func TestData(t testing.T){
}
```

二进制的序列化和反序列化

```
序列化
var value uint32
buff:=new(bytes.Buff)
binary.Write(buff,binary.BigEndian,value)
反序列化
var buf []byte
buff:=bytes.NewBuffer(buff)
binary.Read(buff,binary.BigEndian,&value)
通过对二进制的序列化和反序列化我认识到了在tcp层数据以流的方式流动时，我们要做到边界的处理，就要确定报头的大小
每次确定读取的数量，这样才不会造成错误的读取。
```