channer通道

why?

1：为了达到在多个goroutine发送和接受共享的数据，达到数据同步的目的。

方式和例子

方式： 使用关键字chan,加上Go语言内置的make函数 例子：

```
import "fmt"
func main(){
pipe :=make(chan string) //声明了一个string类型的通道
go func(){
    pipe<-"ju"

}()
x:=<-pipe
fmt.Printfln(x)
}
```

管道的缓冲，方向和选择器

相对于上面的例子，这样的管道是阻塞,也就是说如果我们在管道里面写了一些东西的话，那么这个管道就必须有一个接受的一端，如果没有的话那么就会出现错误。

管道的缓冲

方式：make(chan string,3)后面的3就代表了我们加入了三个缓冲 例子：

```
import "fmt"

func main(){
 mess := make(chan int,3)
 go func(){
     for val:=range mess{
         fmt.Println(val)
     }
 }()
for i:=0;i<3;i++{
    mess<-i
}//把这个for循环删除也不会出现问题了
}
```

管道的方向

引入的原因？ 如果说我们像定义一个只能读的通道或者只能写的通道该怎么做 例子：

```
import "fmt"
//这个函数的recv定义了只能从这个管道里面去取东西，send定义了只能从这个管道里面去存东西
func fun(recv <-chan string, send chan<- int)
{
    val ops int=0
    for val:= range recv{
        fmt.Println(val)
        send<-ops
        ops++
    }
    close(send)
}
func  setvalue(recv chan<- string,msg string)
{
  recv<-msg
}

func main(){
  recv:=make(chan string,3)
  send:=make(chan string,3)
  go fun(recv ,send)
  setvalue(recv,"ju")
  setvalue(recv,"wen")
  setvalue(recv,"jie")
  close(recv)
  for val:=range send{
      fmt.Println(val)
  }
}
```

通道选择器

引入的原因？ 主要就是为了可以进行多个通道的操作和处理的.如果select的多个分支都满足条件，则会随机的选取其中一个满足条件的分支

用法：

```
func main(){
    c1:=make(chan string)
    c2:=make(chan string)
    go func(){
        C1<-"JU"
    }()
    go func(){
        c2<-"LIU"
    }
    select{
        case msg1:=<-c1:
```

```
        fmt.Printfln(msg1)
        case msg2:=<-c2:
        fmt.Printfln(msg2)
    }
}
```