Go 记录日志——log包

        Golang's log模块主要提供了3类接口。分别是 “**Print**、**Panic**、**Fatal**”，对每一类接口其提供了3中调用方式，分别是 "Xxxx 、 Xxxx**ln** 、Xxxx**f**"，基本和fmt中的相关函数类似，下面是一个Print的示例：

package main

import (

"log"

)

func main(){

arr := []int {2,3}

log.Print("Print array ",arr,"\n")

log.Println("Println array",arr)

log.Printf("Printf array with item [%d,%d]\n",arr[0],arr[1])

}

会得到如下结果：

2016/12/15 19:46:19 Print array [2 3]

2016/12/15 19:46:19 Println array [2 3]

2016/12/15 19:46:19 Printf array with item [2,3]

        对于 log.Fatal 接口，会先将日志内容打印到标准输出，接着调用系统的 **os.exit(1)**接口，退出程序并返回状态 1 。但是有一点需要注意，由于是直接调用系统接口退出，defer函数不会被调用，下面是一个Fatal的示例：

package main

import (

"fmt"

"log"

)

func test\_deferfatal(){

defer func() {

fmt.Println("--first--")

}()

log.Fatalln("test for defer Fatal")

}

func main() {

test\_deferfatal()

}

        会得到如下结果：

2016/12/15 19:46:45 test for defer Fatal

        可以看到并没有调用defer 函数。

        对于log.Panic接口，该函数把日志内容刷到标准错误后调用 panic 函数，下面是一个Panic的示例：

package main

import (

"fmt"

"log"

)

func test\_deferpanic(){

defer func() {

fmt.Println("--first--")

if err := recover(); err != nil {

fmt.Println(err)

}

}()

log.Panicln("test for defer Panic")

defer func() {

fmt.Println("--second--")

}()

}

func main() {

test\_deferpanic()

}

        会得到如下结果：

2016/12/15 19:59:30 test for defer Panic

--first--

test for defer Panic

        可以看到首先输出了“test for defer Panic”，然后第一个defer函数被调用了并输出了“--first--”，但是第二个defer 函数并没有输出，可见在Panic之后声明的defer是不会执行的。

        你也可以自定义Logger类型， log.Logger提供了一个New方法用来创建对象：

func New(out io.Writer, prefix string, flag int) \*Logger

该函数一共有三个参数：

（1）输出位置out，是一个io.Writer对象，该对象可以是一个文件也可以是实现了该接口的对象。通常我们可以用这个来指定日志输出到哪个文件。  
（2）prefix 我们在前面已经看到，就是在日志内容前面的东西。我们可以将其置为 "[Info]" 、 "[Warning]"等来帮助区分日志级别。  
（3） flags 是一个选项，显示日志开头的东西，可选的值有：

Ldate = 1 << iota // 形如 2009/01/23 的日期

Ltime // 形如 01:23:23 的时间

Lmicroseconds // 形如 01:23:23.123123 的时间

Llongfile // 全路径文件名和行号: /a/b/c/d.go:23

Lshortfile // 文件名和行号: d.go:23

LstdFlags = Ldate | Ltime // 日期和时间

示例如下：

package main

import (

"log"

"os"

)

func main(){

fileName := "Info\_First.log"

logFile,err := os.Create(fileName)

defer logFile.Close()

if err != nil {

log.Fatalln("open file error")

}

debugLog := log.New(logFile,"[Info]",log.Llongfile)

debugLog.Println("A Info message here")

debugLog.SetPrefix("[Debug]")

debugLog.Println("A Debug Message here ")

}