Go语言学习(九)文件操作

原创 2016年03月27日 21:08:09

标签：

[go语言](http://so.csdn.net/so/search/s.do?q=go%E8%AF%AD%E8%A8%80&t=blog" \t "http://blog.csdn.net/mchenys/article/details/_blank)

4178

1.相关api介绍

1.1建立File内存地址api

//返回File的内存地址,错误信息,通过os库调用func Create(name string) (file \*File, err Error)//返回文件的内存地址,通过os库调用func NewFile(fd int, name string) \*File

1

2

3

4

1.2打开文件api

//返回File的内存地址,错误信息,通过os库调用func Open(name string)(file \*File,err Error)//返回File的内存地址,错误信息,通过os库调用func OpenFile(name string,flag int,perm unit32)(file \*File,err Error)

1

2

3

4

1.3写文件api

//写入一个slice,返回写的个数,错误信息,通过File的内存地址调用func (file \*File)Write(b []byte)(n int,err Error)//从slice的某个位置开始写入,返回写的个数,错误信息,通过File的内存地址调用func (file \*File)WriteAt(b []byte,off int64)(n int,err Error)//写入一个字符串,返回写的个数,错误信息,通过File的内存地址调用func (file \*File) WriteString(s string)(ret int,err Error)

1

2

3

4

5

6

1.4读文件api

//读取一个slice,返回读的个数,错误信息,通过File的内存地址调用func (file \*File) Read(b []byte)(n int, err Error)//从slice的某个位置开始读取,返回读到的个数,错误信息,通过File的内存地址调用func (file \*File) ReadAt(b []byte,off int64)(n int,err Error)

1

2

3

4

1.5删除文件api

//传入文件的路径来删除文件,返回错误个数func Remove(name string) Error

1

2

2.写文件的实例代码

package main

import (

"fmt"

"os"

)

func main() {

userFile := "d:/test.txt" //文件路径

fout,err := os.Create(userFile) //根据路径创建File的内存地址

defer fout.Close() //延迟关闭资源

if err != nil{

fmt.Println(userFile,err)

return

}

//循环写入数据到文件

for i:=0;i<10;i++{

fout.WriteString("Hello world!\r\n") //写入字符串

fout.Write([]byte("abcd!\r\n"))//强转成byte slice后再写入

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

输出的文件内容:

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

3.读文件的实例代码

package main

import (

"fmt"

"os"

)

func main() {

userFile := "d:/test.txt" //文件路径

fin,err := os.Open(userFile) //打开文件,返回File的内存地址

defer fin.Close() //延迟关闭资源

if err != nil{

fmt.Println(userFile,err)

return

}

buf := make([]byte,1024)//创建一个初始容量为1024的slice,作为缓冲容器

for{

//循环读取文件数据到缓冲容器中,返回读取到的个数

n,\_ := fin.Read(buf)

if 0==n{

break //如果读到个数为0,则读取完毕,跳出循环

}

//从缓冲slice中写出数据,从slice下标0到n,通过os.Stdout写出到控制台

os.Stdout.Write(buf[:n])

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

运行结果:   
G:\GoCodeHome\demo\bin>go build demo4

G:\GoCodeHome\demo\bin>demo4

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

Hello world!

abcd!

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

4.同时使用os.Open和os.Create操作文件

package main

import (

"io"

"os"

)

func main() {

fi, err := os.Open("d:/input.txt")//打开输入\*File

if err != nil { panic(err) }

defer fi.Close()

fo, err := os.Create("d:/output.txt")//创建输出\*File

if err != nil { panic(err) }

defer fo.Close()

buf := make([]byte, 1024)

for {

n, err := fi.Read(buf)//从input.txt读取

if err != nil && err != io.EOF { panic(err) }

if n == 0 { break }

if n2, err := fo.Write(buf[:n]); err != nil {//写入output.txt,直到错误

panic(err)

} else if n2 != n {

panic("error in writing")

}

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

5.使用bufio库

package main

import (

"bufio"

"io"

"os"

)

func main() {

fi, err := os.Open("input.txt")//打开输入\*File

if err != nil { panic(err) }

defer fi.Close()

r := bufio.NewReader(fi)//创建一个读取缓冲流

fo, err := os.Create("output.txt")//创建输出\*File

if err != nil { panic(err) }

defer fo.Close()

w := bufio.NewWriter(fo)//创建输出缓冲流

buf := make([]byte, 1024)

for {

n, err := r.Read(buf)

if err != nil && err != io.EOF { panic(err) }

if n == 0 { break }

if n2, err := w.Write(buf[:n]); err != nil {

panic(err)

} else if n2 != n {

panic("error in writing")

}

}

if err = w.Flush(); err != nil { panic(err) }

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

6.使用ioutil库

package main

import (

"io/ioutil"

)

func main() {

b, err := ioutil.ReadFile("input.txt")//读文件

if err != nil { panic(err) }

err = ioutil.WriteFile("output.txt", b, 0644)//写文件

if err != nil { panic(err) }

}

1

2

3

4

5

6

7

8

9

10

11

12

13

7.遍历文件夹

package mainimport (

"path/filepath"

"os"

"fmt"

"flag"

)

func getFilelist(path string) {

err := filepath.Walk(path, func(path string, f os.FileInfo, err error) error {

if ( f == nil ) {return err}

if f.IsDir() {return nil}

println(path)

return nil

})

if err != nil {

fmt.Printf("filepath.Walk() returned %v\n", err)

}

}

func main(){

flag.Parse()

root := flag.Arg(0)

getFilelist(root)

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

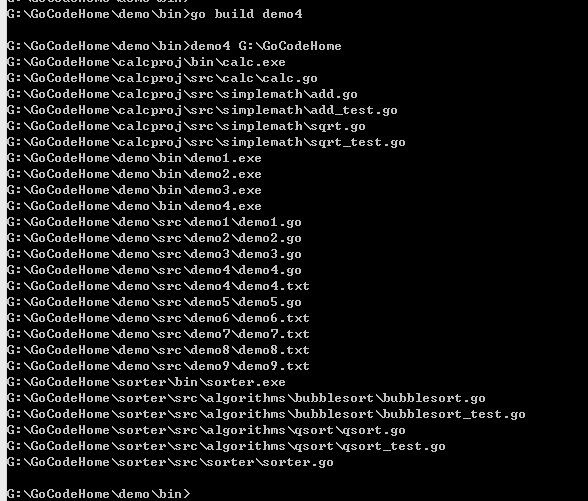
21

22

23

24

25

运行结果:   


文章来源：http://blog.csdn.net/mchenys/article/details/50993849

版权声明：本文为博主原创文章，未经博主允许不得转载。