IO操作2

作者：少林之巅

目录

1. 文件打开和读写

2. 读取压缩文件

3. bufio原理和cat命令实现

4. defer详解

5. 课后作业

文件读写

1. 文件是存储在外部介质上的数据集合。

   A. 文件分类：文本文件和二进制文件

   B. 文件存取方式：随机存取和顺序存放

文件读写

2. 文件打开

```
package main

import (
	"bufio"
	"fmt"
	"io"
	"os"
)

func main() {
	//只读的方式打开
	inputFile, err := os.Open("input.dat")
	if err != nil {
		fmt.Printf("open file err:%v\n", err)
		return
	}

	defer inputFile.Close()
}
```

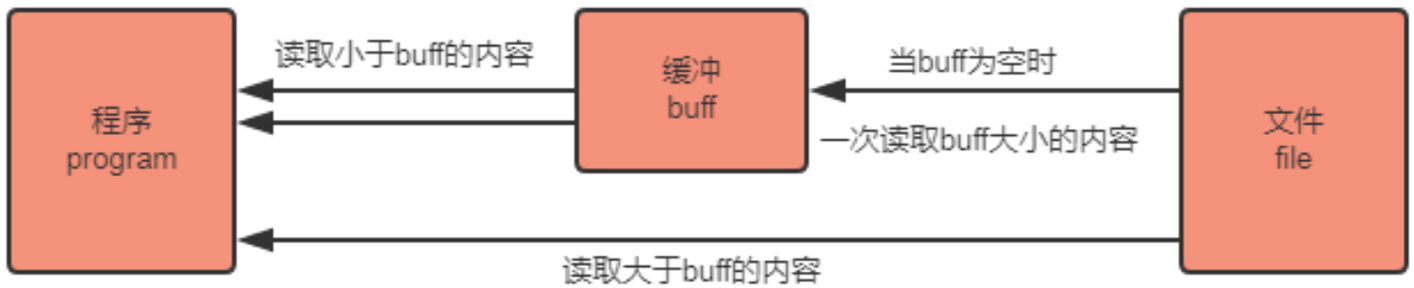
文件读写

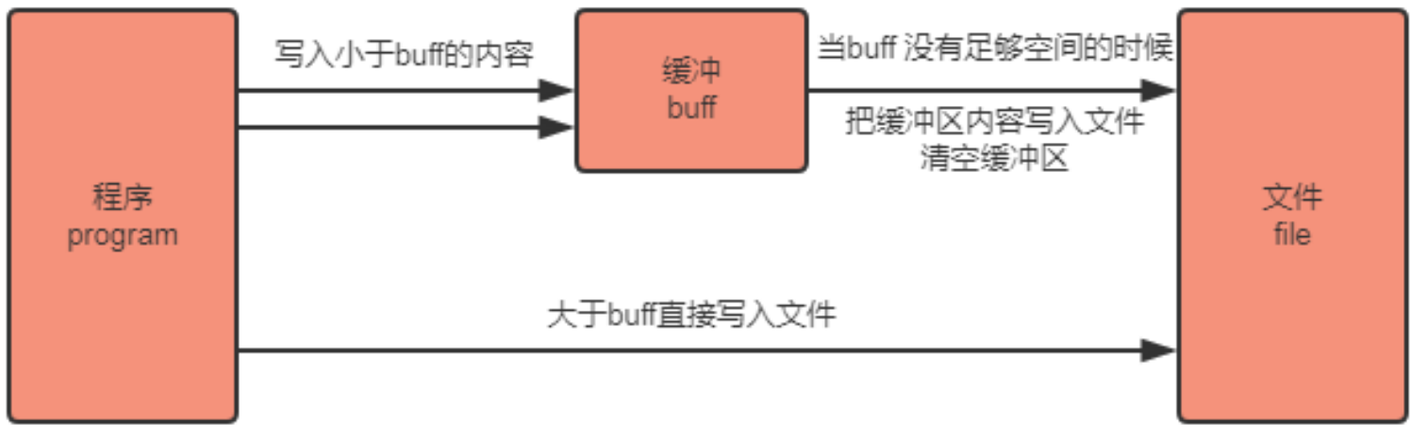
3. 文件读取, file.Read和file.ReadAt。读到文件末尾返回:io.EOF

```
package main

import (
        "bufio"
        "fmt"
        "io"
        "os"
)

func main() {
        //只读的方式打开
        inputFile, err := os.Open("input.dat")
        if err != nil {
                fmt.Printf("open file err:%v\n", err)
                return
        }
        var buf[128]byte
        inputFile.Read(buf[:])
        defer inputFile.Close()
}
```

文件读写

4. bufio原理





文件读写

5. bufio读取文件

```
package main

import (
	"bufio"
	"fmt"
	"io"
	"os"
)

func main() {

	inputFile, err := os.Open("input.dat")
	if err != nil {
		fmt.Printf("open file err:%v\n", err)
		return
	}

	defer inputFile.Close()
	inputReader := bufio.NewReader(inputFile)
	for {
		inputString, readerError := inputReader.ReadString('\n')
		if readerError == io.EOF {
			return
		}
		fmt.Printf("The input was: %s", inputString)
	}
}
```

文件读写

6. 读取整个文件示例

```
package main

import (
	"fmt"
	"io/ioutil"
	"os"
)

func main() {

	inputFile := "products.txt"
	outputFile := "products_copy.txt"
	buf, err := ioutil.ReadFile(inputFile)
	if err != nil {
		fmt.Fprintf(os.Stderr, "File Error: %s\n", err)
		return
	}

	fmt.Printf("%s\n", string(buf))
}
```

文件读写

7. 读取压缩文件示例

```
package main

import (
	"bufio"
	"compress/gzip"
	"fmt"
	"os"
)
func main() {
	fName := "MyFile.gz"
	var r *bufio.Reader
	fi, err := os.Open(fName)
	if err != nil {
		fmt.Fprintf(os.Stderr, "%v, Can't open %s: error: %s\n", os.Args[0], fName, err)
		os.Exit(1)
	}
	fz, err := gzip.NewReader(fi)
	if err != nil {
		fmt.Fprintf(os.Stderr, "open gzip failed, err: %v\n", err)
		return
	}
	r = bufio.NewReader(fz)
	for {
		line, err := r.ReadString('\n')
		if err != nil {
			fmt.Println("Done reading file")
			os.Exit(0)
		}
		fmt.Println(line)
	}
}
```

文件读写

8. 文件写入

os.OpenFile(“output.dat”, os.O_WRONLY|os.O_CREATE, 0666)

第二个参数：文件打开模式

:

1. os.O_WRONLY：只写
2. os.O_CREATE：创建文件
3. os.O_RDONLY：只读
4. os.O_RDWR：读写
5. os.O_TRUNC ：清空
6. os. O_APPEND：追加

第三个参数：权限控制：

r ——> 004

w——> 002

x——> 001

9. 文件写入示例

file.Write()

file.WriteAt()

file.WriteString()

```
package main

import (
        "bufio"
        "fmt"
        "os"
)

func main() {
        outputFile, outputError := os.OpenFile("output.dat",
os.O_WRONLY|os.O_CREATE, 0666)
        if outputError != nil {
                fmt.Printf("An error occurred with file creation\n")
                return
        }
        str := “hello world”
        outputFile.Write([]byte(str))
        defer outputFile.Close()
}
```

5. 文件写入示例

```
package main

import (
	"bufio"
	"fmt"
	"os"
)

func main() {
	outputFile, outputError := os.OpenFile("output.dat",
os.O_WRONLY|os.O_CREATE, 0666)
	if outputError != nil {
		fmt.Printf("An error occurred with file creation\n")
		return
	}

	defer outputFile.Close()
	outputWriter := bufio.NewWriter(outputFile)
	outputString := "hello world!\n"
	for i := 0; i < 10; i++ {
		outputWriter.WriteString(outputString)
	}
	outputWriter.Flush()
}
```

文件读写

10.写入整个文件示例

```
package main

import (
	"fmt"
	"io/ioutil"
	"os"
)

func main() {
	inputFile := "products.txt"
	outputFile := "products_copy.txt"
	buf, err := ioutil.ReadFile(inputFile)
	if err != nil {
		fmt.Fprintf(os.Stderr, "File Error: %s\n", err)
		return
	}

	fmt.Printf("%s\n", string(buf))
	err = ioutil.WriteFile(outputFile, buf, 0x644)
	if err != nil {
		panic(err.Error())
	}
}
```

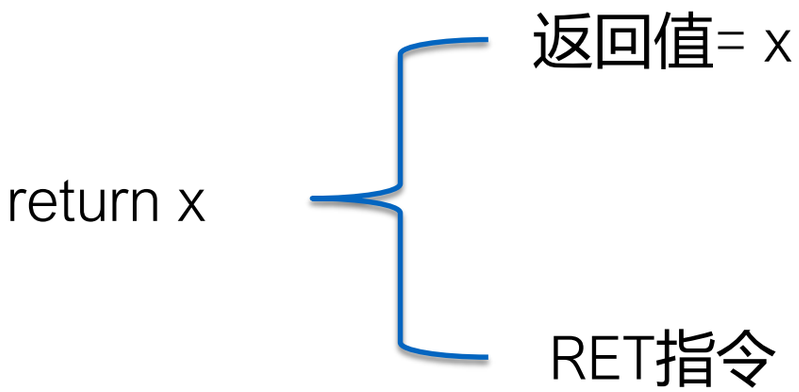
11.拷贝文件

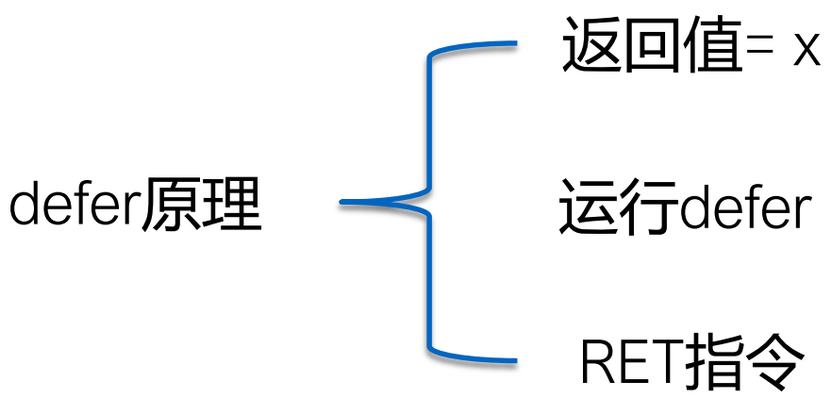
```
package main

import (
	"fmt"
	"io"
	"os"
)

func main() {

	CopyFile("target.txt", "source.txt")
	fmt.Println("Copy done!")
}

func CopyFile(dstName, srcName string) (written int64, err error) {
	src, err := os.Open(srcName)
	if err != nil {
		return
	}
	defer src.Close()
	dst, err := os.OpenFile(dstName, os.O_WRONLY|os.O_CREATE, 0644)
	if err != nil {
		return
	}
	defer dst.Close()
	return io.Copy(dst, src)
}
```

12.cat命令实现

```
package main

import (
	"bufio"
	"flag"
	"fmt"
	"io"
	"os"
)

func cat(r *bufio.Reader) {
	for {
		buf, err := r.ReadBytes('\n')
		if err == io.EOF {
			break
		}
		fmt.Fprintf(os.Stdout, "%s", buf)

		return
	}
}

func main() {
	flag.Parse()
	if flag.NArg() == 0 {
		cat(bufio.NewReader(os.Stdin))
	}
	for i := 0; i < flag.NArg(); i++ {
		f, err := os.Open(flag.Arg(i))
		if err != nil {
			fmt.Fprintf(os.Stderr, "%s:error reading from %s: %s\n",
				os.Args[0], flag.Arg(i), err.Error())
			continue
		}

		cat(bufio.NewReader(f))
	}
}
```

13.defer原理分析





14.defer案例1

```
package main

import (
    "fmt"
)

func funcA() int {
    x := 5
    defer func() {
        x += 1
    }()
    return x
}

func main() {
    fmt.Println(funcA())
}
```

15.defer案例2

```
package main

import "fmt"

func funcB() (x int) {
    defer func() {
        x += 1
    }()
    return 5
}

func main() {
    fmt.Println(funcB())
}
```

16.defer案例3

```
package main

import "fmt"

func funcC() (y int) {
    x := 5
    defer func() {
        x += 1
    }()
    return x
}

func main() {
    fmt.Println(funcC())
}
```

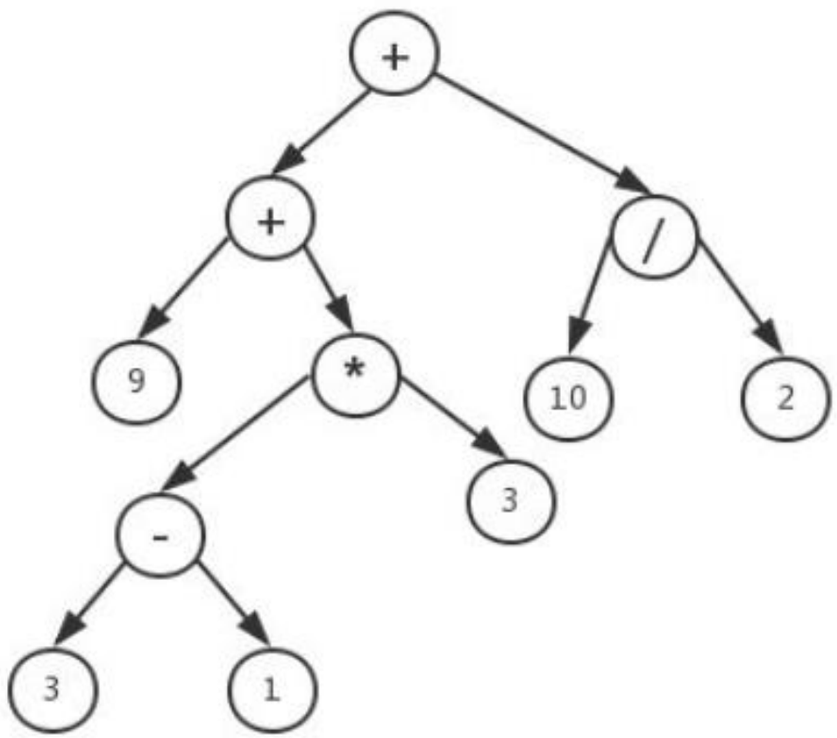
17.defer案例4

```
package main

import "fmt"

func funcD() (x int) {
    defer func(x int) {
        x += 1
    }(x)
    return 5
}

func main() {
    fmt.Println(funcD())
}
```

计算器作业讲解

1. 中缀表达式

9+(3-1)*3+10/2



计算器作业讲解

2. 后缀表达式，也叫逆波兰式

9+(3-1)*3+10/2

9 3 1 - 3 * 10 2 / + +

计算器作业讲解

3. 中缀表达式转后缀表达式

```
//一、 将中缀表达式转换成后缀表达式算法：
//1、从左至右扫描一中缀表达式。
//2、若读取的是操作数，则判断该操作数的类型，并将该操作数存入操作数堆栈
//3、若读取的是运算符
// (1) 该运算符为左括号“(”，则直接存入运算符堆栈。
// (2) 该运算符为右括号“)”，则输出运算符堆栈中的运算符到操作数堆栈，直到遇到左括号为止。
// (3) 该运算符为非括号运算符：
//       (a) 若运算符堆栈栈顶的运算符为括号，则直接存入运算符堆栈。
//       (b) 若比运算符堆栈栈顶的运算符优先级高，则直接存入运算符堆栈。
//       (c) 若比运算符堆栈栈顶的运算符优先级低或相等，则输出栈顶运算符到操作数堆栈，并将当前运算符压入运算符堆栈
```

课后练习

1. 实现一个类似 linux的tree 命令，输入tree.exe能够以树状的形式当前目录下所有文件，如下所示。