**[golang语言中os包的学习与使用(文件，目录，进程的操作)](https://www.cnblogs.com/jkko123/p/7146474.html)**

os中一些常用函数的使用：

|  |  |
| --- | --- |
| 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  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103 | package main;    import (      "os"      "fmt"      "time"      "strings"  )    //os包中的一些常用函数    func main() {        //获取主机名      fmt.Println(os.Hostname());        //获取当前目录      fmt.Println(os.Getwd());        //获取用户ID      fmt.Println(os.Getuid());        //获取有效用户ID      fmt.Println(os.Geteuid());        //获取组ID      fmt.Println(os.Getgid());        //获取有效组ID      fmt.Println(os.Getegid());        //获取进程ID      fmt.Println(os.Getpid());        //获取父进程ID      fmt.Println(os.Getppid());        //获取环境变量的值      fmt.Println(os.Getenv("GOPATH"));        //设置环境变量的值      os.Setenv("TEST", "test");        //改变当前工作目录      os.Chdir("C:/");      fmt.Println(os.Getwd());        //创建文件      f1, \_ := os.Create("./1.txt");      defer f1.Close();        //修改文件权限      if err := os.Chmod("./1.txt", 0777); err != nil {          fmt.Println(err);      }        //修改文件所有者      if err := os.Chown("./1.txt", 0, 0); err != nil {          fmt.Println(err);      }        //修改文件的访问时间和修改时间      os.Chtimes("./1.txt", time.Now().Add(time.Hour), time.Now().Add(time.Hour));        //获取所有环境变量      fmt.Println(strings.Join(os.Environ(), "\r\n"));        //把字符串中带${var}或$var替换成指定指符串      fmt.Println(os.Expand("${1} ${2} ${3}", func(k string) string {          mapp := map[string]string{              "1": "111",              "2": "222",              "3": "333",          };          return mapp[k];      }));        //创建目录      os.Mkdir("abc", os.ModePerm);        //创建多级目录      os.MkdirAll("abc/d/e/f", os.ModePerm);        //删除文件或目录      os.Remove("abc/d/e/f");        //删除指定目录下所有文件      os.RemoveAll("abc");        //重命名文件      os.Rename("./2.txt", "./2\_new.txt");        //判断是否为同一文件      //unix下通过底层结构的设备和索引节点是否相同来判断      //其他系统可能是通过文件绝对路径来判断      fs1, \_ := f1.Stat();      f2, \_ := os.Open("./1.txt");      fs2, \_ := f2.Stat();      fmt.Println(os.SameFile(fs1, fs2));        //返回临时目录      fmt.Println(os.TempDir());  } |

os中一些常用文件函数的使用：

|  |  |
| --- | --- |
| 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  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86 | package main;    import (      "os"      "fmt"      "strconv"  )    //os包中关于文件的操作函数    func main() {      //创建文件，返回一个文件指针      f3, \_ := os.Create("./3.txt");      defer f3.Close();        //以读写方式打开文件，如果不存在则创建文件，等同于上面os.Create      f4, \_ := os.OpenFile("./4.txt", os.O\_RDWR|os.O\_CREATE|os.O\_TRUNC, 0666);      defer f4.Close();        //打开文件，返回文件指针      f1, \_ := os.Open("./1.txt");      defer f1.Close();        //修改文件权限，类似os.chmod      f1.Chmod(0777);        //修改文件所有者，类似os.chown      f1.Chown(0, 0);        //返回文件的句柄，通过NewFile创建文件需要文件句柄      fmt.Println(f1.Fd());        //从文件中读取数据      buf := make([]byte, 128);      //read每次读取数据到buf中      for n, \_ := f1.Read(buf); n != 0; n, \_ = f1.Read(buf) {          fmt.Println(string(buf[:n]));      }        //向文件中写入数据      for i := 0; i < 5; i++ {          f3.Write([]byte("写入数据" + strconv.Itoa(i) + "\r\n"));      }        //返回一对关联的文件对象      //从r中可以读取到从w写入的数据      r, w, \_ := os.Pipe();      //向w中写入字符串      w.WriteString("写入w");      buf2 := make([]byte, 128);      //从r中读取数据      n, \_ := r.Read(buf);      fmt.Println(string(buf2[:n]));        //改变工作目录      os.Mkdir("a", os.ModePerm);      dir, \_ := os.Open("a");      //改变工作目录到dir，dir必须为一个目录      dir.Chdir();      fmt.Println(os.Getwd());        //读取目录的内容，返回一个FileInfo的slice      //参数大于0，最多返回n个FileInfo      //参数小于等于0，返回所有FileInfo      fi, \_ := dir.Readdir(-1);      for \_, v := range fi {          fmt.Println(v.Name());      }        //读取目录中文件对象的名字      names, \_ := dir.Readdirnames(-1);      fmt.Println(names);        //获取文件的详细信息，返回FileInfo结构      fi3, \_ := f3.Stat();      //文件名      fmt.Println(fi3.Name());      //文件大小      fmt.Println(fi3.Size());      //文件权限      fmt.Println(fi3.Mode());      //文件修改时间      fmt.Println(fi3.ModTime());      //是否是目录      fmt.Println(fi3.IsDir());  } |

os中关于进程的操作：

|  |  |
| --- | --- |
| 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  35  36  37  38  39  40  41  42  43 | package main;    import (      "os"      "fmt"      "time"  )    //os包中关于进程的操作函数    func main() {      //设置新进程的属性      attr := &os.ProcAttr{          //files指定新进程继承的活动文件对象          //前三个分别为，标准输入、标准输出、标准错误输出          Files: []\*os.File{os.Stdin, os.Stdout, os.Stderr},          //新进程的环境变量          Env: os.Environ(),      }        //win下通过记事本打开1.txt      //开始一个新进程      p, err := os.StartProcess("C:\\Windows\\System32\\notepad.exe", []string{"C:\\Windows\\System32\\notepad.exe", "D:\\1.txt"}, attr);      if err != nil {          fmt.Println(err);      }      fmt.Println(p);      fmt.Println("进程ID：", p.Pid);        //通过进程ID查找进程      p2, \_ := os.FindProcess(p.Pid);      fmt.Println(p2);        //等待10秒，执行函数      time.AfterFunc(time.Second\*10, func() {          //向p进程发送退出信号          p.Signal(os.Kill);      });        //等待进程p的退出，返回进程状态      ps, \_ := p.Wait();      fmt.Println(ps.String());  } |