# Linux 命令解释程序设计与实现

19281030-张云鹏

2022-03-18

## Linux 命令解释程序设计与实现

### 实验环境

- Ubuntu 20.04 LTS
- go 1.17.8

### 测试命令

```
git clone git@github.com:bruce2233/os.git && ./os/shell/main
```

### 实验结果

### 命令行提示符

1. 系统调用 Getcwd() 返回读取字节数 n

```
n, _ := syscall.Getcwd(cwd)
```

2. 截取切片返回 string

```
cwdstr := string(cwd[0 : n-1])
```

### 内部命令

**ls** 命令

1. 系统调用 Open() 返回文件描述符

```
fd, err = syscall.Open(myPath, syscall.O_RDONLY, 0)
```

2. 系统调用 ReadDirent(), 读取文件夹数据到 buf

```
direntBytes := make([]byte, bufSize)
syscall.ReadDirent(fd, direntBytes)
```

3. 解析 Dirent 结构体数据，返回目录下的文件名 string 数组

```
syscall.ParseDirent(direntBytes, 100, fileNames)
```

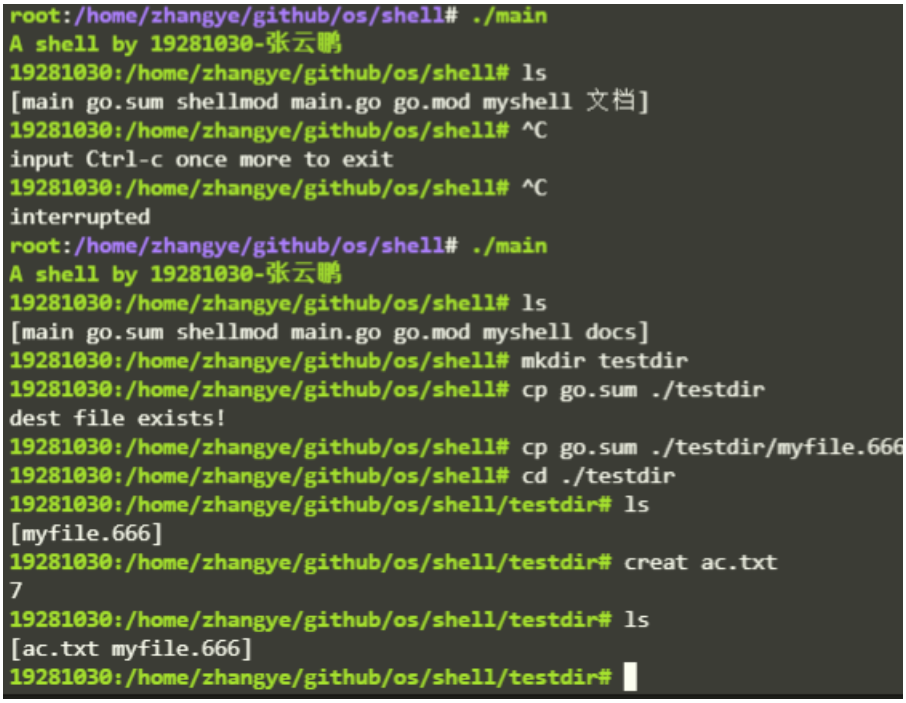


Figure 1: lab1-result

**cd** 命令

1. 系统调用 Chdir(), 切换工作目录

```
err := syscall.Chdir(c.Args.String("cdpath"))
```

2. 更新 prompt 提示，系统调用 Getcwd()

```
c.App.SetPrompt("19281030:" + gocall.Getcwd() + "# ")
```

**copy** 命令

1. 系统调用 Open(), 获取源文件和目标文件的文件描述符，同时判断异常

```
sourcefd, err := syscall.Open(sourcePath, syscall.O_RDONLY, 0)
destfd, err := syscall.Open(destPath, syscall.O_RDONLY, 0)
```

2. 创建缓冲区，系统调用 Read().Write(), 循环从源文件读取到 buf，再写入到文件

```
for {
        p := make([]byte, 2048)
        n, _ := syscall.Read(sourcefd, p)
        syscall.Write(destfd, p[0:n])
        if n < 2048 {
            break
        }
    }
```

**mkdir** 命令

1. 系统调用 Mkdir()

```
err := syscall.Mkdir(dirpath, syscall.O_WRONLY)
```

2. 异常判断输出

```
if err != nil {
            println("mkdir fail")
        }
```

**creat** 命令

1. 系统调用 Creat(), 或者通过 Open 创建模式打开文件。

```
fd, err := syscall.Creat(file, 0)
```

2. 异常判断输出

```
if err != nil {
    println("create file fail")
}
```

外部命令

1. 系统调用 Getenv(), 拼接字符串，获得文件绝对路径

```
path := gocall.Getenv("PATH")
```

2. 系统调用 fork 和 exec，创建子进程后立即执行文件

```
pid, err = syscall.ForkExec(path+"/"+file, nil, &procAttr)
```

环境变量

1. 系统调用 Setenv(), 设置环境变量 PATH

```
syscall.Setenv("PATH", "/etc/mypath")
```

2. 获取当前用户名，设置环境变量 HOME

```
if u.Username == "root" {
    syscall.Setenv("HOME", "/root")
} else {
    syscall.Setenv("HOME", "/home/"+u.Username)
}
```