实验四 实现简单的文件系统

1120152035 王奥博

运行环境及使用方法:

- bash4.4及以上版本

```
lab4 [master●] bash --version
GNU bash，版本 4.4.12(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2016 Free Software Foundation, Inc.
许可证 GPLv3+: GNU GPL 许可证第三版或者更新版本 <http://gnu.org/licenses/gpl.html>

本软件是自由软件，您可以自由地更改和重新发布。
在法律许可的情况下特此明示，本软件不提供任何担保。
lab4 [master●]
```

- 运行方法:
  - 首先赋给该脚本可执行权限,然后运行

```
lab4 [master●] chmod +x lab4.sh
lab4 [master●] ./lab4.sh
Welcome to M4x's fileSystem.
This is a simplified fileSystem for lab4 of OS of BIT.
~:
```

  - 或者直接使用**bash ./lab4.sh**,通过bash运行脚本

```
lab4 [master●] bash ./lab4.sh
Welcome to M4x's fileSystem.
This is a simplified fileSystem for lab4 of OS of BIT.
~:
```

设计思路

根据系统的要求可以看出,需要实现的功能和bash的功能非常类似,因此很自然的就想到了使用**shell编程**来完成这次实验,因此做出了如下的设计:

程序主要流程:

- 通过while true循环读取命令,对不同的命令进行不同的处理.
- 定义了一个**onCtrlC()**函数,用于捕获用户ctrl + C打断程序,此时将会删除目录下除了该shell脚本外的所有文件,以实现只有exit命令才能保存的目的,具体实现:

```
trap 'onCtrlC' INT
function onCtrlC () {
    echo "Something wrong occured. Your tasks won't be saved..."
    cd $workdir
    mv "$(basename $0)" ..
    rm * -r -f
    mv "../""$(basename $0)" .
    exit
}
```

- **new**:

  该程序使用一个文件夹表示文件系统,因此,new一个文件系统即为bash命令中的mkdir 文件夹,此时需要检测该文件夹是否存在,不存在时打印报错信息,具体实现:

```
    if [[ "$cmd" == new\ * ]]; then
        #echo "test"
        mkdir "${cmd:4}" 2>/dev/null || {
        echo "The fileSystem ""${cmd:4}"" already existed. Try another name!"
    }
```

- **sfs**:

  打开文件系统时,设置标记量**flag**为true,flag用于其他命令能否执行的检测,sfs命令需检测要打开的文件系统是否存在,具体实现:

```
    elif [[ "$cmd" == sfs\ * ]]; then
        flag=true
        cd "${cmd:4}" 2>/dev/null || {
        echo "There is no ""${cmd:4}""! Please check your fileSystem's name!"
    }
```

- **mkdir, rmdir, ls, cd**:

  对于这些命令,因为和bash中的用法相同,因此只需要检测这些命令本身的就合法性即可,具体实现

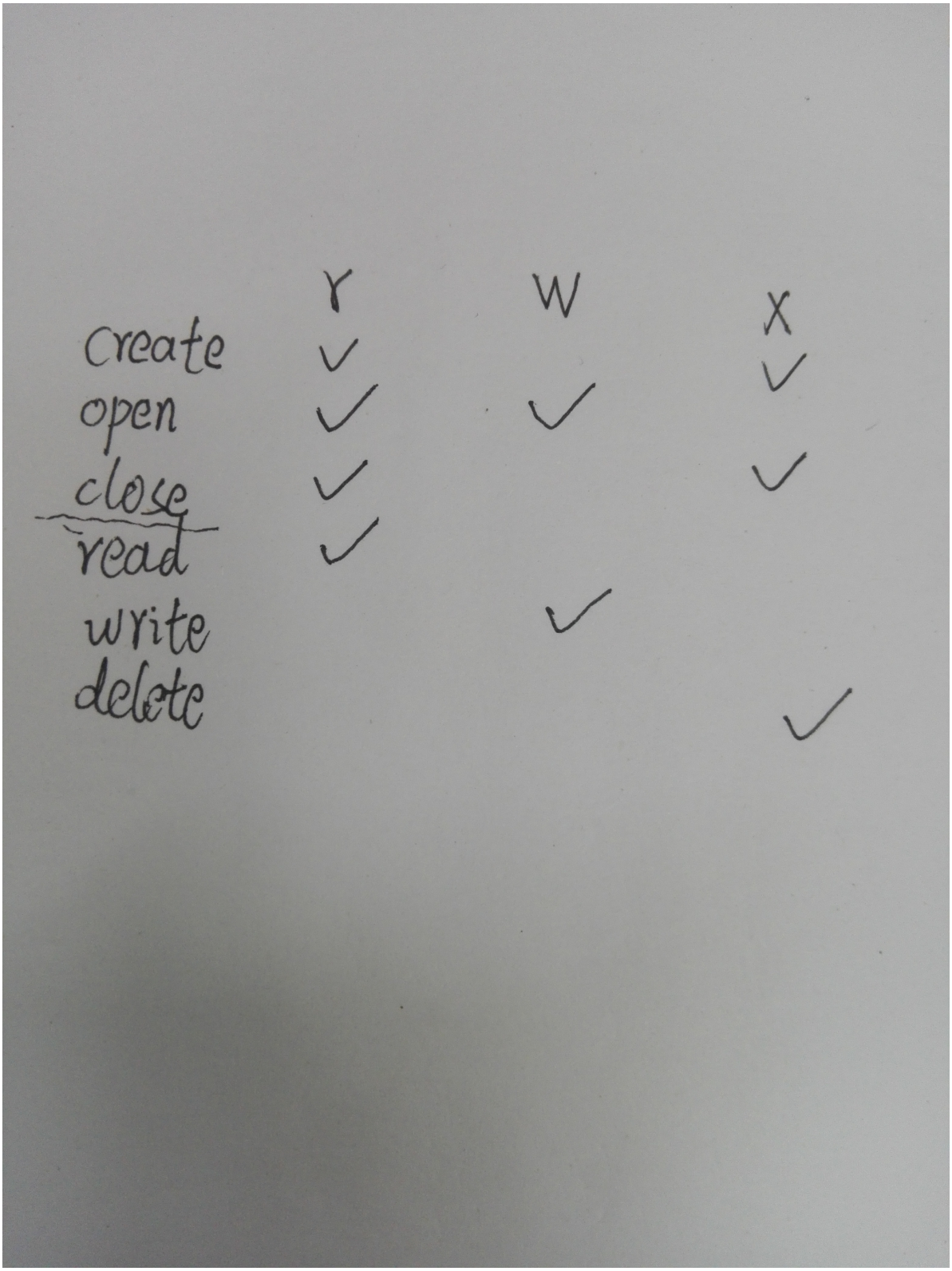
```
    elif [ "$flag" = true ] && ([[ "$cmd" == mkdir\ * ]] || [[ "$cmd" == rmdir\ * ]] ||
[[ "$cmd" == ls ]] || [[ "$cmd" == cd\ * ]]); then
        $cmd 2>/dev/null || {
        echo "Wrong command!"
    }
```

- **create, open, close**:

  需要满足:

  - 一个文件刚被创建时,只有可读和可执行的权限
  - 当一个文件被打开时,该文件有可读和可写的权限
  - 一个文件被关闭时,该文件有可读和可执行的权限

    即create,open,close三个命令需要控制的文件权限如下表:



对此三种操作,只需控制好文件权限,同时做出检测机制即可,具体实现:

```
    elif [ "$flag" = true ] && [[ "$cmd" == create\ *  ]]; then
        # create两次即为清空
        touch "${cmd:7}"
        # 此时文件有可读可执行权限
        chmod 555 "${cmd:7}" 2>/dev/null
    elif [ "$flag" = true ] && [[ "$cmd" == open\ * ]]; then
        # 此时文件具只有可写权限
        chmod 666 "${cmd:5}" 2>/dev/null || {
        echo "The file ""${cmd:4}"" doesn't existed!"
    }
    elif [ "$flag" = true ] && [[ "$cmd" == close\ * ]]; then
        # 此时文件具有可读可执行权限
        chmod 555 "${cmd:6}" 2>/dev/null || {
        echo "The file ""${cmd:4}"" doesn't existed!"
    }
```

- **read, write, delete**:

  需要满足:

  - read时,文件具有可读权限
  - write时,文件具有可写权限
  - delete时,文件具有可执行权限
  - write时,需要先检测要写入的文件是否存在,其余操作只需检测命令是否出错

  具体实现:

```
    elif [ "$flag" = true ] && [[ "$cmd" == read\ * ]]; then
        cat "${cmd:5}" 2>/dev/null || {
        echo "The file ""${cmd:4}"" doesn't existed!"
    }
    elif [ "$flag" = true ] && [[ "$cmd" == write\ * ]]; then
        # 检测文件是否存在
        if [ ! -e "$cmd{:6}" ]; then
            echo "The file ""${cmd:6}"" doesn't exist!"
        fi

        read -p "Please input your content: "  content
        echo -n "$content" > "${cmd:6}"
    elif [ "$flag" = true ] && [[ "$cmd" == delete\ * ]]; then
        rm "${cmd:7}" 2>/dev/null || {
        echo "The file ""${cmd:4}"" doesn't existed!"
    }
```

- **exit:**

  只需使脚本正常退出即可,具体实现:

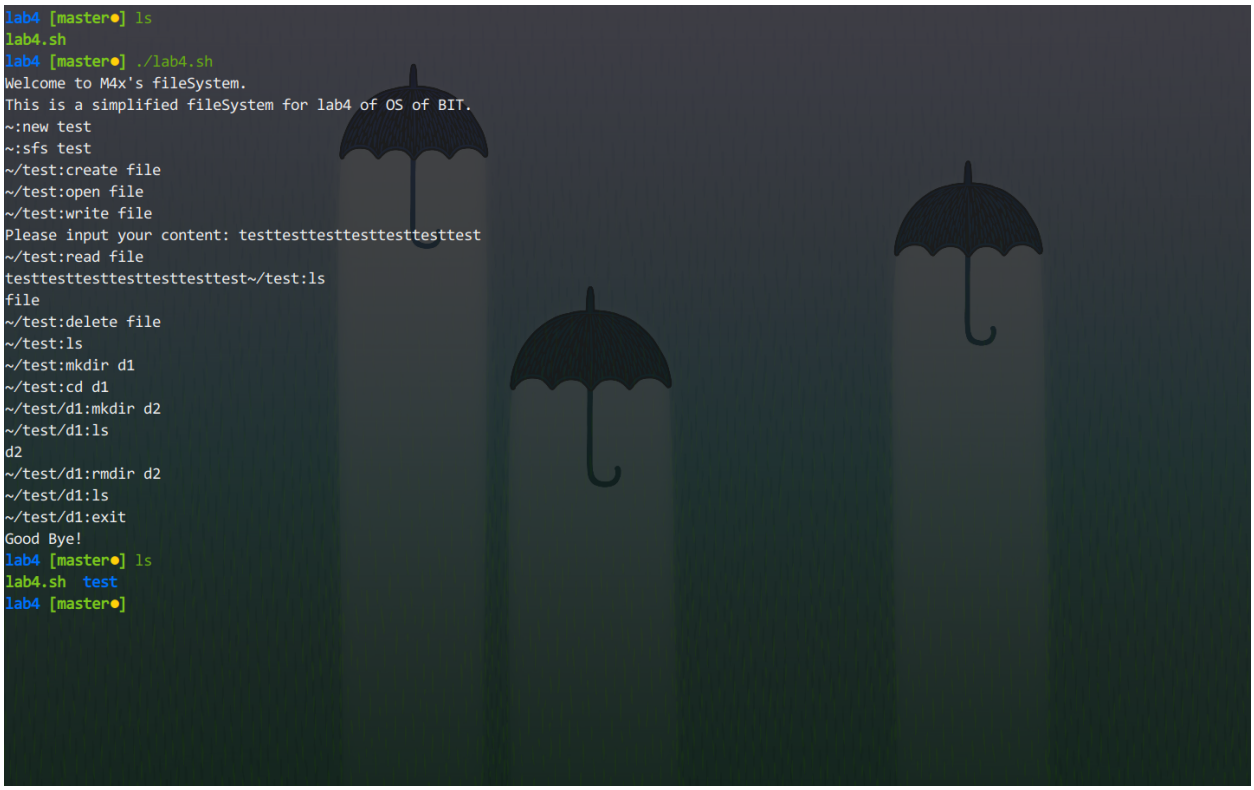
```
    elif [[ "$cmd" == exit ]]; then
        echo "Good Bye!"
        exit
```

具体细节:

- 为了检测不合法的执行,首先将stderr重定向到了/dev/null,这样命令在报错时就不会将错误信息打印出来
- shell中没有类似于python的try...except于是使用了{try} || {except}代替

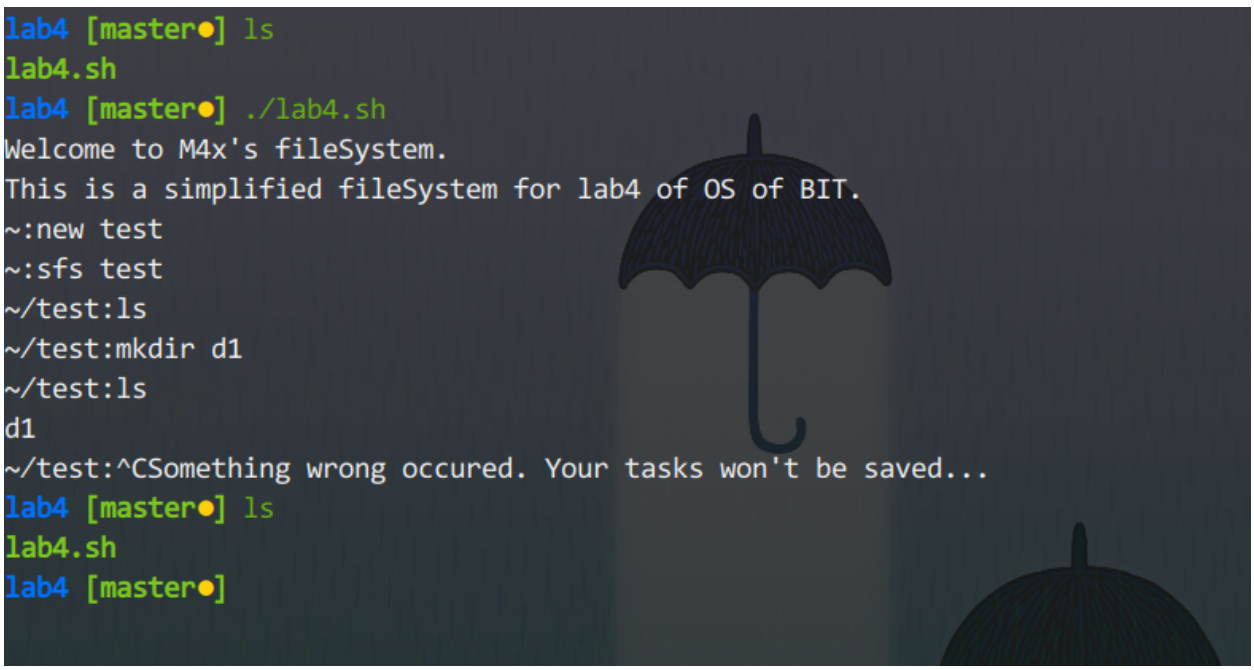
测试:

- 正常退出时:



如上图,正常退出时保存了运行结果

- 费正常退出时:



如上图,非正常退出时,结果不被保存

两点提示：

- 对于shell脚本的debug,可以使用**set -x**,set -x可以打印命令以及该条命令的运行结果,如下:

```
68 |   |   # 检测文件是否存在
69 |   |   # set -x
70 |   |   if [ ! -e "${cmd:6}" ]; then
71 |   |   |   echo "The file ""${cmd:6}"" doesn't exist!"
72 |   |   fi
73 |   |   # set +x
```

```
lab4 [master●] ./lab4.sh
Welcome to M4x's fileSystem.
This is a simplified fileSystem for lab4 of OS of BIT.
~:new test
~:sfs test
~/test:create file
~/test:open file
~/test:write file
+ '[' '!' -e file ']'
+ set +x
Please input your content: debugdebugdebug
~/test:
```

- 除此之外,还可以使用shellcheck这个工具检查语法错误

> $sudo apt-get install shellcheck

```
lab4
lab4.sh
lab4 [master●] vim lab4.sh
lab4 [master●] shellcheck lab4.sh

In lab4.sh line 10:
        cd $workdir
        ^-- SC2164: Use 'cd ... || exit' or 'cd ... || return' in case cd fails.
           ^-- SC2086: Double quote to prevent globbing and word splitting.


In lab4.sh line 11:
        mv "$(basename $0)" ..
                   ^-- SC2086: Double quote to prevent globbing and word splitting.


In lab4.sh line 12:
        rm * -r -f
           ^-- SC2035: Use ./*glob* or -- *glob* so names with dashes won't become options.


In lab4.sh line 13:
        mv "../""$(basename $0)" .
                        ^-- SC2086: Double quote to prevent globbing and word splitting.


In lab4.sh line 18:
workdir="$(cd $(dirname $0); pwd)"
          ^-- SC2164: Use 'cd ... || exit' or 'cd ... || return' in case cd fails.
             ^-- SC2046: Quote this to prevent word splitting.
                      ^-- SC2086: Double quote to prevent globbing and word splitting.


In lab4.sh line 28:
        read -p "~""${path:len}"":" cmd
```

实验总结

本次试验相当于一个简易的文件系统,用shell编程可以很容易的实现,本次试验中收获最多的除了对文件系统更深刻的理解,还有对shell脚本编写的进一步掌握.

完整代码

> 代码已上传至:https://github.com/M4xW4n9/personal_repository/blob/master/OS/lab4/lab4.sh

```
#!/usr/bin/env bash

set -u
# set -x #debug
echo "Welcome to M4x's fileSystem."
echo "This is a simplified fileSystem for lab4 of OS of BIT."

trap 'onCtrlC' INT
function onCtrlC () { echo "Something wrong occured. Your tasks won't be saved..."
    cd $workdir
    mv "$(basename $0)" ..
    rm * -r -f
    mv "../""$(basename $0)" .
    exit
}

# workdir保存脚本所在目录路径
workdir="$(cd $(dirname $0); pwd)"
# echo $workdir
#转化为相对路径
len=${#workdir}
# echo $len
#标记是否打开了文件系统
flag=false
while true
do
    path=$(pwd)
    read -p "~""${path:len}"":" cmd
    # 字符串比较注意空格的使用
    # new,转义空格
    if [[ "$cmd" == new\ * ]]; then
        #echo "test"
        mkdir "${cmd:4}" 2>/dev/null || {
        echo "The fileSystem "" ${cmd:4}"" already existed. Try another name!"
    }

    elif [[ "$cmd" == sfs\ * ]]; then
        flag=true
        cd "${cmd:4}" 2>/dev/null || {
        echo "There is no "" ${cmd:4}""! Please check your fileSystem's name!"
    }

    elif [ "$flag" = true ] && ([[ "$cmd" == mkdir\ * ]] || [[ "$cmd" == rmdir\ * ]] || [[
"$cmd" == ls ]] || [[ "$cmd" == cd\ * ]]); then
        $cmd 2>/dev/null || {
        echo "Wrong command!"
    }

    elif [ "$flag" = true ] && [[ "$cmd" == create\ *  ]]; then
        # create两次即为清空
        touch "${cmd:7}"
        # 此时文件有可读可执行权限
        chmod 555 "${cmd:7}" 2>/dev/null
```

```
    elif [ "$flag" = true ] && [[ "$cmd" == open\ * ]]; then
        # 此时文件具只有可写权限
        chmod 666 "${cmd:5}" 2>/dev/null || {
        echo "The file ""${cmd:4}"" doesn't existed!"
    }
    elif [ "$flag" = true ] && [[ "$cmd" == close\ * ]]; then
        # 此时文件具有可读可执行权限
        chmod 555 "${cmd:6}" 2>/dev/null || {
        echo "The file ""${cmd:4}"" doesn't existed!"
    }
    elif [ "$flag" = true ] && [[ "$cmd" == read\ * ]]; then
        cat "${cmd:5}" 2>/dev/null || {
        echo "The file ""${cmd:4}"" doesn't existed!"
    }
    elif [ "$flag" = true ] && [[ "$cmd" == write\ * ]]; then
        # 检测文件是否存在
        # set -x
        if [ ! -e "${cmd:6}" ]; then
            echo "The file ""${cmd:6}"" doesn't exist!"
        fi
        # set +x

        read -p "Please input your content: "  content
        echo -n "$content" > "${cmd:6}" 2>/dev/null || {
        echo "The file ""${cmd:6}"" hasn't been opened!"
    }
    elif [ "$flag" = true ] && [[ "$cmd" == delete\ * ]]; then
        rm "${cmd:7}" 2>/dev/null || {
        echo "The file ""${cmd:4}"" doesn't existed!"
    }
    elif [[ "$cmd" == exit ]]; then
        echo "Good Bye!"
        exit
    else
        echo "Wrong command!"
    fi

done
```