Lab03b: Shell 编程

1. 实验目的

能够进行 Shell 基础编程。

2. 实验指南

2.1. Shell 编程

2.1.1. 什么是 Shell?

通俗一点的解释就是**命令解析器**，用以接收用户输入的命令，然后调用相应的应用程序。

要注意我们敲命令的地方(也就是那个黑色框框)不叫 Shell！！！那个东西叫 terminal，我们在 terminal 中输入命令，然后 Shell 接收指令。Shell 可看成是一个进程，terminal 是这个进程的输入输出，与用户交互的部分。

详细可看知乎：终端、Shell、tty 和控制台（console）有什么区别？

2.1.2. 实验环境

这次实验统一使用`bash`，在 Shell 中输入`echo $SHELL`，输出应当是`/bin/bash`。如果不是，可使用`chsh`命令，之后输入`/bin/bash`将默认 Shell 更改为`bash`。

2.1.3. 如何运行 Shell?

以`helloworld.sh`为例

```
#!/bin/bash
echo "Hello, World!"
```

- `#!/bin/bash`被称为 shebang。具体其来源与作用见释伴：Linux 上的 Shebang 符号(#!)
- 运行该程序需要具有可执行权限，可通过`chmod +x helloworld.sh`进行赋权。
- `./helloworld.sh`即可看到输出。

2.1.4. 命令连接符

命令的执行是串行的，一条命令结束才能输入下一条命令，可以在命令之间加上`;`分割命令，从而可以一行输入所有命令。Shell 会挨个执行。

- `&&`连接符
  - 命令`1 && 命令2 && 命令3`，Shell 在判断出这个表达式的真假后就会停止执行。如果命令 1 为`false`，可以判断表达式 一定为假，执行停止。如果为`true`，那么还需要执行命令 2，一直执行到能判断真假为止或者执行完被`&&`连接的命令。
- `||`连接符
  - 同 `&&`，执行到能判断真假或者所有被连接命令被执行完为止。`&&`和`||`的计算方式同 c 语言中的`&& ||`。以上运算原则又被称作**短路原则**

2.1.5. Shell 变量

- 在 Shell 中使用变量无需定义，在使用的时候创建。并且变量不分类型，Shell 统一认为是字符串，需要的时候通过一些命令进行转换。
- 变量赋值: **变量名=值** ，等号左右不能够有空格。若字符串中包含空格，则需要用单/双引号括起来。
- 可以使用`readonly`将变量改为只读类型。
- 通过 `$` 引用变量值，`echo $SHELL`。
- 输入变量，`read`变量名。

引用格式	返回值
`$var`	返回变量值
`${#var}`	返回变量值的长度
`${var:start}`	返回从 start 下标到字符串末尾的子串
`${var:start:length}`	返回从 start 下标开始，长度为 length 的子串

实际上还有一些空值判断、字符串替换和正则匹配拆分字符串等，为了精简篇幅，这里不再列举，可自行查阅资料。

- 环境变量 (在前面已经介绍了)
  <!-- - 环境变量是用来定义系统运行环境的一些参数，比如每个用户不同的家目录`HOME`、邮件存放位置`MAIL`等。
  - 可以使用`env`命令来查看 Linux 系统中所有的环境变量。
  - `export 变量名`，将一个已经存在的本地变量修改为环境变量
  - `export 变量名 = 值`，定义一个环境变量。 -->
- 位置变量
  - 在执行 Shell 脚本的时候，可以传入参数，如当前有个脚本叫`test`，执行`sh test arg1 arg2 arg3`，那么在`test`中，`$0`代表脚本文件名，`$1`为第一个参数:`arg1`，以此类推。
  - 使用`shift`可以将参数左移，此时`$1`为`arg2`，`$2`为`arg3`，`$3`为空`$#`为参数数量。

特殊参数	含义
`$#`	传递到脚本的参数数量
`$?`	前一个命令执行情况，`0`成功，其他值失败
`$$`	运行当前脚本的进程`id`
`$!`	运行脚本最后一个命令
`$*`	传递给脚本或者函数的所有参数

2.1.6. 其余 Shell 语法

Shell 编程还可以使用一些比如`if`语句等其余语句，这些语句和咱们学过的`C`中的对应语句功能基本一致，只是写法可能有点不一样，这里就不列举具体的功能了，仅给出一些例子以供参考。

```
#!/bin/bash
clear
echo;echo
echo "为 $0 添加执行权限"
chmod +x $0
echo

# 重定向
echo 输出重定向 > __re.txt
cat __re.txt
echo

gcc tan90.file 2>__error.log
cat __error.log
echo

# 管道
ls -l /etc | grep pa
echo

# 定义变量
var=string

# 只读变量不可修改
readonly test
# test=6 去掉注释报错

# 引用变量
echo "var的值为: "$var
echo "var长度为: "${#var}
echo "var2~4为:  "${var:2:3}
echo

# 传入参数
echo "文件名为$0"
echo '$1: '$1
echo '$2: '$2
echo '$1: '$3
echo "传入参数有:  "$*
echo "传入参数数量: "$#
```

```
echo

shift;echo "移位后参数为："
echo '$0: '$0
echo '$1: '$1
echo

# 运算
i=1
((i+=5))
echo i=1,执行 '((i+=5))'
echo 'i=: '$i
echo

# if
echo -e "输入一个文件名，测试是否存在: \c" # -e能解释转义字符，详情man echo
read filename
if [[ -f $filename && -s $filename ]]; then
    echo "$filename 文件存在且不为空"
else
    echo "$filename 文件不存在或为空"
fi
echo
#======================
if [ -d /boot ]; then
    echo "存在/boot目录"
else
    echo "不存在/boot目录"
fi
echo
#==================
if ls -l /boot/efi; then
    echo "访问/boot/efi 成功"
else
    echo "访问/boot/efi 失败"
fi
echo
#======
echo
data=3
data2=2
if [ $data -lt $data2 ]; then
    echo $data \< $data2
elif [ $data -gt $data2 ]; then
    echo $data \> $data2
else
    echo $data = $data2
fi

echo

# select
echo "输入编号选择："
select subject in Math ComputerScience Chinese Moyu
do
    echo "我最喜欢$subject"
    break
done
```

```
echo

# case
case $subject in
    'Moyu' ) echo 'Moyu美滋滋';; #2个分号
    '*' )  echo "好好学习天天向上"
esac
echo

# for
for var in  1 2 3 4 5 6
do
    echo -e $var '\c '
done
echo;echo

# while
count=1
sum=0
while [ $count -lt 101 ]
do
    ((sum+=count))
    # count=`expr $count + 1` 这种写法不推荐
    ((count=expr+1))
done
echo sum is $sum
echo

# until 略

function helloworld # 或者 helloworld()
{
    echo 参数个数为 $#
    echo 传入参数为 $*
}
helloworld 1 2 3


# 一个综合一些的例子
f[1]=1
function fib () {
    for ((i = $1; i < $2; i++)); do
    if (( i > 6 && i > 8)); then
        break
    fi
    ((f[i] ++ ))
    # f[i]=$((${f[i]}+1))
    # f[i]=`expr ${f[i]} + 1`
    # 这三种方式都可以进行赋值，但expr的方法不推荐(lint会报错)
    echo ${f[i]}
done
}
fib 3 10 # $0: fib, $1: 3, $2: 10

rm __error.log
rm __re.txt
exit 0  # 脚本成功退出
```

3. 实验习题

- 假如在脚本的第一行放入`#!/bin/rm`或者在普通文本文件中第一行放置`#!/bin/more`，然后将文件设为可执行权限执行，看看会发生什么，并解释为什么。
- 编写一个 bash 脚本，执行该脚本文件将得到两行输出，第一行是你的学号，第二行是当前的日期（考虑使用`date`命令）。对该脚本文件的要求是
  - 文件名为`date-${你的学号}`，比如`date-15131049`
  - 用户可以在任意位置**只需要输入文件名**就可以执行该脚本文件
  - **不破坏除用户家目录之外的任何目录结构**，即不要在家目录之外的任何地方增删改任何文件

  请详细叙述你的操作过程以及操作过程的截图，并给出你所编写的脚本文件的代码。
- 完成LeetCode: 193. Valid Phone Numbers | leetcode-cn (体会`grep -E`, `grep -P`, `egrep`, `awk`等的差异)
- 完成LeetCode: 195. Tenth Line | leetcode-cn，给出你的代码和 AC 截图。(提示：怎么读取每一行)
- 完成一个简单的交互设计，根据用户输入输出对应内容，具体交互内容随意，要求至少用上`select`，`case`和`read`。
- 编写 Shell 脚本`addowner.sh`将某目录下面所有的文件名后面加上文件所有者的名字。比如`a.txt`和`file`的所有者都为 owner，文件名修改后分别为`a[owner].txt`和`file[owner]`。
  - 使用用法：`./addowner 目录名称`。（若无目录名称这一参数，则默认为当前目录）
  - 对于子目录，名称不变。
  - （提示：为了测试效果，请通过`useradd`创建若干用户，并可通过`chown`改变文件所有者。另外，网上的答案是错的。）