作业二: shell 文件下的分支结构与循环结构

嵇敏君
信息与计算科学 3200103322

2022 年 6 月 29 日

shell 文件下的分支结构与循环结构同 python,C++ 等的结构有较大的区别, 下面便大致区分一下 shell 文件的分支结构与循环结构同 python,C++ 的区别

1 分支结构

分支结构主要是 if 语句与 case 语句

1.1 if 语句

python 的 if 语句结构体如下:

```
if condition
  statements
elif condition
  statements
else
  statements
```

而在 shell 文件下,if 结构体变成了:

```
if condition
then
  statements
elif condition
```

```
then
  statements
else
  statements
fi
```

这里

```
if/elif condition
then
```

也可写成

```
if/elif condition; then
```

可以看出, 在 shell 文件下,if 语句在判断之后,`if` 与`elif` 的后面都接上了一个`then`, 同时, 在整个 if 语句结束之后,shell 文件最后一行还需要加上一个`fi`, 而且在 shell 文件中,condition 的判断需要用 [] 接着, 我们用书本 35 页中的一个案例来测试一下, 测试代码如下:

```
#!/bin/bash

echo "Is it morning? Please answer yes or no"
read timeofday
if [ $timeofday = "yes" ]
then
  echo "Good morning"
elif [ $timeofday = "no" ]; then
  echo "Good afternoon"
else
  echo "Sorry,$timeofday not recognized.Enter yes or no"
  exit1
fi

exit 0
```

这里我们要注意条件判断时,`$timeofday="yes"` 与`"$timeofday"="yes"` 的区别

1.2 case 语句

在 python 中,case 语句的结构体为:

```
switch (variable){
case pattern1: statements;
case pattern2: statements;
...
case patternn: statements;
default: statemens;}
```

case 结构体如下:

```
case variable in
  pattern [ | pattern] ...) statements;;
  pattern [ | pattern] ...) statements;;
  ...
esac
```

可以看到, shell 文件的 case 语句与 python 中的语句差异较大,python 中 case 会与 switch 一起使用, 每个 statements 后都有一个;, 结尾处有一个 default 来表示其他所有情况. 而在 shell 文件中,case 作为开头与 in 一起使用, 每个 statements 后都有两个;(;;), 同时结尾处用了一个 esac(case 反过来写) 表示结束下面我们用书本中的几个案例来体会一下 case case1:

```
#!/bin/bash

echo "Is it morning? Please answer yes or no"
read timeofday

case "$timeofday" in
    yes) echo "Good morning";;
    no ) echo "Good afternoon";;
    y  ) echo "Good morning";;
    n  ) echo "Good afternoon";;
    *  ) echo "Sorry, answer not recognized";;
esac
```

```
exit 0
```

可以看出, 在 shell 文件中, 用 * 来表示其他所有的情况，但是, 这样子不能尽可能多的作出正确的判断, 如对于大小写的判断就不够灵敏, 于是便有如下两种方法 case2:

```
#!/bin/bash

echo "Is it morning? Please answer yes or no"
read timeofday

case "$timeofday" in
    yes | y | Yes | YES) echo "Good morning";;
    n* | N* ) echo "Good afternoon";;
    *  ) echo "Sorry, answer not recognized";;
esac

exit 0
```

case3:

```
#!/bin/bash

echo "Is it morning? Please answer yes or no"
read timeofday

case "$timeofday" in
    yes | y | Yes | YES)
           echo "Good morning"
           echo "Up bright and early this morning";;
    [nN]* )
           echo "Good afternoon";;
    *)
           echo "Sorry, answer not recognized"
           echo "Please answer yes or no"
```

```
            exit 1;;
esac

exit 0
```

2 循环结构

2.1 for 语句

在 shell 文件中, 由于输入与进行判断的都是字符串, 而不是像 python 中的一个个数字, 故 for 语句也只能进行字符串的替换，结构体如下:

```
for ... in srt1 str2 str3
do
  statements
done
```

相比与 python, 可以看到我们在 for 语句后面多了一个do, 在结束时加上了一个done 同样, 我们用书本 37 页的案例进行测试，测试代码如下:

```
#!/bin/bash

for foo in bar fud 43
do
  echo $foo
done
exit 0
```

输出的结果为:

```
bar
fud
43
```

但是，若我们将代码换成:

```
for foo in "bar fud" 43
do
```

```
  echo $foo
done
exit 0
```

此时输出为:

```
bar fud
43
```

可以知, 当两个字符串被一个"" 包裹在一起时, 他们便构成了一个新的字符串,for 语句是将 in 后面的所有字符串一个一个替换到 foo 中去, 然后执行 do 后面的语句

2.2 while 语句

shell 文件中的 while 结构体与 python 中相差不大, 就是最判断结束时多了一个done, 具体结构体如下:

```
while condition do
  statements
done
```

我们用 39 页的一个代码测试:

```
#!/bin/bash

echo "Enter password"
read trythis

while [ "$trythis" != "secret" ]; do
  echo "Sorry, try again"
  read trythis
done
echo "Pass"
exit 0
```

2.3 until 语句

until 语句的情况与 while 类似, 只不过一个是当满足 condition 时, 执行 statements(while), 一个是当满足 condition 时结束循环 (until),until 结构体如下:

```
until condition
do
  statements
done
```

同样, 用书本 40 页中的代码进行测试:

```
#!/bin/bash

until who |grep "$1" > /dev/null
do
  sleep 60
done

# now ring the bell and announce the expected user.

echo -e '\a'
echo ***** $1 has just logged in *****

exit 0
```