6 if条件语句

`if` 条件语句

有了条件测试，就要有**获得测试结果**的机制，并**根据测试结果运行不同的代码段**，这样程序就可以从简单的命令罗列变得更“智能”一些，从而实现程序的流程控制。

与其它编程语言类似，Shell 也**支持选择结构**，并且有两种形式，分别是 `if else` 语句和 `case in` 语句。本节先介绍 `if else` 语句。

一、`if` 单分支语句

`if` 单分支语句是**最简单**的判断结构，可以针对测试结果做相应处理：**如果测试结果为真则运行相关代码**。

其语法结构如下：

Shell | 复制代码

```shell
if [条件表达式]
    then
    代码块
fi
```

⚠ **最后必须以 `fi` 来闭合，** `fi` 就是 `if` 倒过来拼写。

代码块即使有多条语句也**不需要用 `{ }` 包围**起来，不同语句间用换行符隔开。

Shell中**缩进一般为4个空格**。

如果 `条件表达式` 成立（返回“真”），则执行 `then` 后边的代码块；

如果 `条件表达式` 不成立（返回“假”），则不执行任何语句。

⚠ 从本质上讲， `if` 检测的是命令的**退出状态码**。

为了简化代码，也可以将 `then` 和 `if` **写在一行**，注意 `then` **前必须有一个 `;`** 。

Shell 复制代码

```shell
if [条件表达式];then
    代码块
fi
```

案例：检测某文件是否存在

Bash 复制代码

```bash
[root@Shell ~]# vi file_exists.sh
#!/bin/bash
if [ -e /etc/hosts ];then
  echo "1"
fi
[root@Shell ~]# . file_exists.sh
1
```

以上代码用 `if` 单分支语句判断文件 `/etc/hosts` 是否存在，如果存在，则返回 `1` 。

案例：添加新用户

```shell
# 检测是否存在用户test1
[root@shell ~]# grep test1 /etc/passwd
[root@Shell ~]# vi add_user1.sh
#!/bin/bash
read -p "请输入用户名:" user
read -s -p "请输入密码:"   pass
if [[ ! -z "$user" && ! -z "$pass" ]];then
    useradd "$user"
    echo
    echo "$pass" | passwd --stdin "$user"
fi
# 运行脚本添加用户test1
[root@Shell ~]# . add_user1.sh
请输入用户名:test1
请输入密码:
Changing password for user test1.
passwd: all authentication tokens updated successfully.
# 用户test1添加成功
[root@shell ~]# grep test1 /etc/passwd
test1:x:1000:1000::/home/test1:/bin/bash
```

注意：如果用户重复，可以使用 `userdel -r 用户名` 命令删除用户。

案例：检测CPU厂家

```shell
[root@Shell ~]# vi detect_cpu.sh
#! /bin/bash
#grep的-q选项，可以让grep进入静默模式，不管过滤到数据还是没有，都不显示输出结果。
if grep -q AMD /proc/cpuinfo; then
    echo "AMD CPU"
fi
if grep -q Intel /proc/cpuinfo; then
    echo "Intel CPU"
fi
[root@Shell ~]# . detect_cpu.sh
Intel CPU
```

二、`if` 双分支语句

if 条件语句的双分支结构的含义为“如果……那么……否则……”。

if 条件语句的双分支结构语法格式为。

```shell
if [条件表达式]
  then
    代码块1
else
    代码块2
fi
```

如果 `条件表达式` 为真，那么执行代 `码块1` ，否则执行代 `码块2` 。

案例：判断定义的名字是否为空

```shell
[root@Shell ~]# vi detect_name.sh
#! /bin/bash
name=yang
if [ -z "$name" ]
  then
    echo yes
else
    echo no
fi
[root@Shell ~]# . detect_name.sh
no
```

案例：判断用户是否存在

```shell
[root@Shell ~]# vi user_exists.sh
#! /bin/bash
read -p "请输入用户名:" user
if id -u $user >/dev/null 2>&1; then
    echo "用户已存在"
else
    echo "用户不存在"
fi
[root@Shell ~]# . user_exists.sh
请输入用户名:test1
用户已存在
```

扩展习题：使用shell脚本添加新用户，要求在添加前能够检测用户是否存在。

```shell
[root@Shell ~]# vi add_user2.sh
#! /bin/bash
read -p "请输入用户名:" user

if  id -u $user >/dev/null 2>&1;then
    echo "用户已存在"
else
    read -s -p "请输入密码:"   pass
    if [ ! -z "$user" -a ! -z "$pass" ] ;then
        useradd "$user"
        echo ""
        echo "$pass" | passwd --stdin "$user"
    fi
fi
[root@Shell ~]# . add_user2.sh
请输入用户名:test1
用户已存在
```

三、`if` 多分支语句

不论是 `if` 结构的单分支结构，还是 `if/else` 的双分支结构，实际上都不能满足需要，现实中的判断往往有多种可能，在这种情况下可以通过 `if/else` 的语法嵌套完成**多向选择**。

if 条件语句的多分支结构语法格式为。

```shell
if [条件表达式1];then
    代码块1
elif [条件表达式2];then
    代码块2
elif [条件表达式3];then
    代码块3
else
    代码块4
fi
```

如果条件表达式1为真，那么执行代码块1，或者条件代码块2为真，就执行代码块2，或者条件表达式3为真，就执行代码块3，否则执行代码块4。

在多分支结构中，依次检测条件表达式，遇到表达式为真值后，后续的表达式不再检测。

当所有条件表达式都不成立时，执行 `else` 后的代码块。

注意：每个 `elif` 都要带有 `then` ，最后结尾的 `else` 后面没有 `then` 。

案例：根据年龄判断人生阶段

Shell | 复制代码

```
[root@Shell ~]# vi juge_life_stage.sh
#!/bin/bash
read -p "请输入年龄: " age
if (( $age <= 2 )); then
    echo "婴儿"
elif (( $age <= 8 )); then
    echo "幼儿"
elif (( $age <= 17 )); then
    echo "少年"
elif (( $age <=25 )); then
    echo "成年"
elif (( $age <= 40 )); then
    echo "青年"
elif (( $age <= 60 )); then
    echo "中年"
else
    echo "老年"
fi

[root@Shell ~]# . juge_life_stage.sh
请输入年龄: 40
青年
```

案例：根据操作系统版本设置不同的yum源

Shell | 复制代码

```shell
[root@shell ~]# vi yum.sh
#!/bin/bash
# 设置yum源服务器
yum_server=127.0.0.1
# 查询系统版本号
os_version=$(cat /etc/redhat-release |awk '{print $4}' |awk -F"." '{print $1"."$2}')
# 根据CentOS版本设置yum源
if [ "$os_version" = "7.9" ];then
                cat >/etc/yum.repos.d/CentOS7u9.repo <<-EOF
                [CentOS7u9]
                name=CentOS7u9
                baseurl=ftp://$yum_server/CentOS7u9
                gpgcheck=0
EOF
                echo "7.9 yum configure…"
elif [ "$os_version" = "6.8" ];then
                cat >/etc/yum.repos.d/CentOS6u8.repo <<-EOF
                [CentOS6u8]
                name=CentOS6u8
                baseurl=ftp://$yum_server/CentOS6u8
                gpgcheck=0
EOF
elif [ "$os_version" = "5.9" ];then
                cat >/etc/yum.repos.d/centos5u9.repo <<-EOF
                [CentOs5u9]
                name=CentOS5u9
                baseurl=ftp://$yum_server/CentOs5u9
                gpgcheck=0
EOF
else
                echo "error"
fi
[root@shell ~]# . yum.sh
7.9 yum configure…
```

💬 *脚本中应用了Here Document来输入多行字符串， Here Document是Shell 中的一种特殊的重定向方式，用来将输入重定向到一个交互式 Shell 脚本或程序。*

小结

- if单分支：格式（then,fi）
- if双分支：格式
- if多分支：运行逻辑

课程目标

- 知识目标：熟练掌握if条件语句的基本语法。
- 技能目标：能够根据实际需求利用if语句实现条件流程控制。

课外拓展

- 进一步了解if条件语句的应用场景。
- Here Document语法。

参考资料

- `if` 命令帮助： `help if`
- 《Linux Shell核心编程指南》，丁明一，电子工业出版社