本节课程目标

- 熟悉条件判断语句，如判断整数、判断字符串等
- 熟悉流程控制语句基本语法，如if...else...

一、条件判断语法结构

思考：何为真(true)？何为假(false)？

1. 条件判断语法格式

- 格式1： **test** 条件表达式
- 格式2： **[** 条件表达式 **]**
- 格式3： **[[** 条件表达式 **]]** 支持正则 =~

特别说明：

1）**[** 亲亲，我两边都有空格，不空打死你呦 **]** 😾

2）**[[** 亲亲，我两边都有空格，不空打死你呦 **]]**😾

3) 更多判断，`man test`去查看，很多的参数都用来进行条件判断

2. 条件判断相关参数

问：你要判断什么？

答：我要判断文件类型，判断文件新旧，判断字符串是否相等，判断权限等等...

(一) 判断文件类型

判断参数	含义
-e	判断文件是否存在（任何类型文件）
-f	判断文件是否存在并且是一个普通文件
-d	判断文件是否存在并且是一个目录
-L	判断文件是否存在并且是一个软连接文件
-b	判断文件是否存在并且是一个块设备文件
-S	判断文件是否存在并且是一个套接字文件
-c	判断文件是否存在并且是一个字符设备文件
-p	判断文件是否存在并且是一个命名管道文件
-s	判断文件是否存在并且是一个非空文件（有内容）

举例说明：

```
test -e file                只要文件存在条件为真
[ -d /shell01/dir1 ]        判断目录是否存在，存在条件为真
[ ! -d /shell01/dir1 ]      判断目录是否存在,不存在条件为真
[[ -f /shell01/1.sh ]]      判断文件是否存在，并且是一个普通的文件
```

㈡ 判断文件权限

判断参数	含义
-r	当前用户对其是否可读
-w	当前用户对其是否可写
-x	当前用户对其是否可执行
-u	是否有suid，高级权限冒险位
-g	是否sgid，高级权限强制位
-k	是否有t位，高级权限粘滞位

㈢ 判断文件新旧

说明：这里的新旧指的是文件的修改时间。

判断参数	含义
file1 -nt file2	比较file1是否比file2新
file1 -ot file2	比较file1是否比file2旧
file1 -ef file2	比较是否为同一个文件，或者用于判断硬连接，是否指向同一个inode

㈣ 判断整数

判断参数	含义
-eq	相等
-ne	不等
-gt	大于
-lt	小于
-ge	大于等于
-le	小于等于

㈤ 判断字符串

判断参数	含义
-z	判断是否为空字符串，字符串长度为0则成立
-n	判断是否为非空字符串，字符串长度不为0则成立
string1 = string2	判断字符串是否相等
string1 != string2	判断字符串是否相不等

㈥ 多重条件判断

判断符号	含义	举例
-a 和 &&	逻辑与	[1 -eq 1 -a 1 -ne 0] [1 -eq 1] && [1 -ne 0]
-o 和 ||	逻辑或	[1 -eq 1 -o 1 -ne 1]

特别说明：

&& 前面的表达式为真，才会执行后面的代码

|| 前面的表达式为假，才会执行后面的代码

; 只用于分割命令或表达式

① 举例说明

- 数值比较

```
[root@server ~]# [ $(id -u) -eq 0 ] && echo "the user is admin"
[root@server ~]$ [ $(id -u) -ne 0 ] && echo "the user is not admin"
[root@server ~]$ [ $(id -u) -eq 0 ] && echo "the user is admin" || echo "the user is not
admin"

[root@server ~]# uid=`id -u`
[root@server ~]# test $uid -eq 0 && echo this is admin
this is admin
[root@server ~]# [ $(id -u) -ne 0 ]  || echo this is admin
this is admin
[root@server ~]# [ $(id -u) -eq 0 ]  && echo this is admin || echo this is not admin
this is admin
[root@server ~]# su - stu1
[stu1@server ~]$ [ $(id -u) -eq 0 ]  && echo this is admin || echo this is not admin
this is not admin
```

- 类C风格的数值比较

```
注意：在(( ))中，=表示赋值；==表示判断
[root@server ~]# ((1==2));echo $?
[root@server ~]# ((1<2));echo $?
[root@server ~]# ((2>=1));echo $?
[root@server ~]# ((2!=1));echo $?
[root@server ~]# ((`id -u`==0));echo $?

[root@server ~]# ((a=123));echo $a
[root@server ~]# unset a
[root@server ~]# ((a==123));echo $?
```

- 字符串比较

```
注意：双引号引起来，看作一个整体；= 和 == 在 [ 字符串 ] 比较中都表示判断
[root@server ~]# a='hello world';b=world
[root@server ~]# [ $a = $b ];echo $?
[root@server ~]# [ "$a" = "$b" ];echo $?
[root@server ~]# [ "$a" != "$b" ];echo $?
[root@server ~]# [ "$a" !== "$b" ];echo $?        错误
[root@server ~]# [ "$a" == "$b" ];echo $?
[root@server ~]# test "$a" != "$b";echo $?


test  表达式
[ 表达式 ]
[[ 表达式 ]]

思考：[ ] 和 [[ ]] 有什么区别?

[root@server ~]# a=
[root@server ~]# test -z $a;echo $?
[root@server ~]# a=hello
```

```
[root@server ~]# test -z $a;echo $?
[root@server ~]# test -n $a;echo $?
[root@server ~]# test -n "$a";echo $?

# [ '' = $a ];echo $?
-bash: [: : unary operator expected
2
# [[ '' = $a ]];echo $?
0


[root@server ~]# [ 1 -eq 0 -a 1 -ne 0 ];echo $?
[root@server ~]# [ 1 -eq 0 && 1 -ne 0 ];echo $?
[root@server ~]# [[ 1 -eq 0 && 1 -ne 0 ]];echo $?
```

② 逻辑运算符总结

1. 符号;和&&和||都可以用来分割命令或者表达式
2. 分号（;）完全不考虑前面的语句是否正确执行，都会执行;号后面的内容
3. `&&`符号，需要考虑&&前面的语句的正确性，前面语句正确执行才会执行&&后的内容；反之亦然
4. `||`符号，需要考虑||前面的语句的非正确性，前面语句执行错误才会执行||后内容；反之亦然
5. 如果&&和||一起出现，从左往右依次看，按照以上原则

二、流程控制语句

关键词：选择（人生漫漫长路，我该何去何从🤔）

1. 基本语法结构

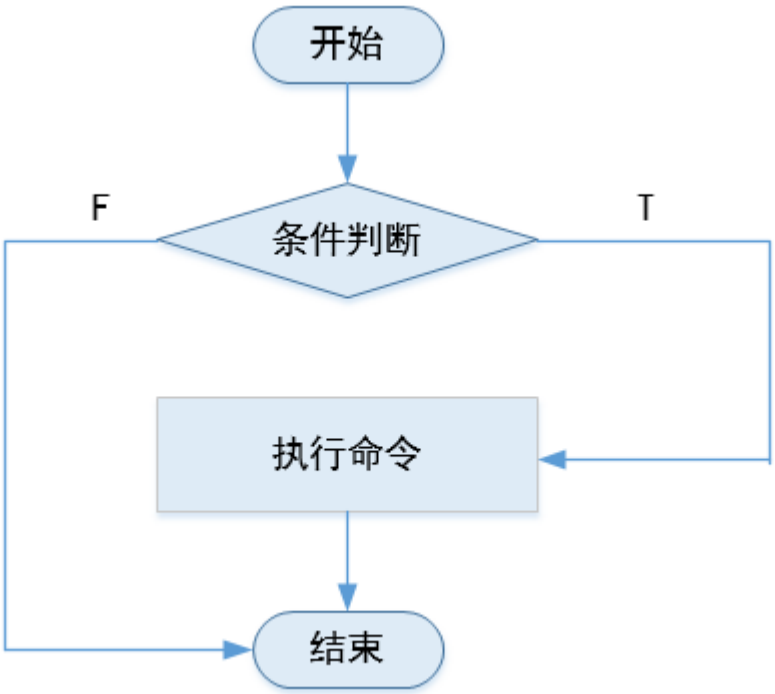
(一) if结构

箴言1：只要正确，就要一直向前冲👌

F:表示false，为假

T:表示true，为真

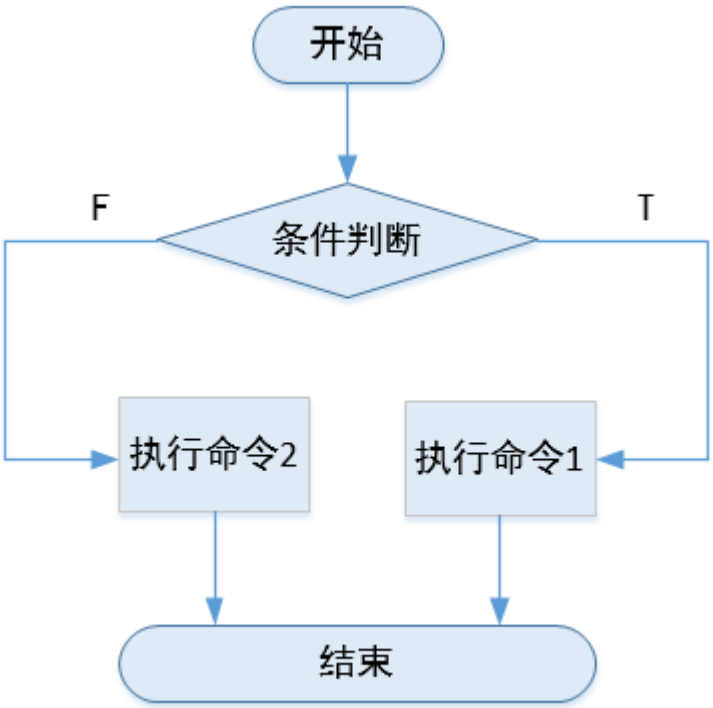
```
if [ condition ];then
       command
       command
fi

if test 条件;then
   命令
fi

if [[ 条件 ]];then
   命令
fi

[ 条件 ] && command
```



(二) if...else结构

箴言2：分叉路口，二选一

```
if [ condition ];then
       command1
   else
       command2
fi

[ 条件 ] && command1 || command2
```



小试牛刀：

让用户自己输入字符串，如果用户输入的是hello，请打印world，否则打印“请输入hello”

1. `read定义变量`
2. if....else...

```
#!/bin/env bash

read -p '请输入一个字符串:' str
if [ "$str" = 'hello' ];then
    echo 'world'
 else
    echo '请输入hello!'
fi
```

```
#!/bin/env bash

read -p "请输入一个字符串:" str
if [ "$str" = "hello" ]
then
    echo world
else
    echo "请输入hello!"
fi
```

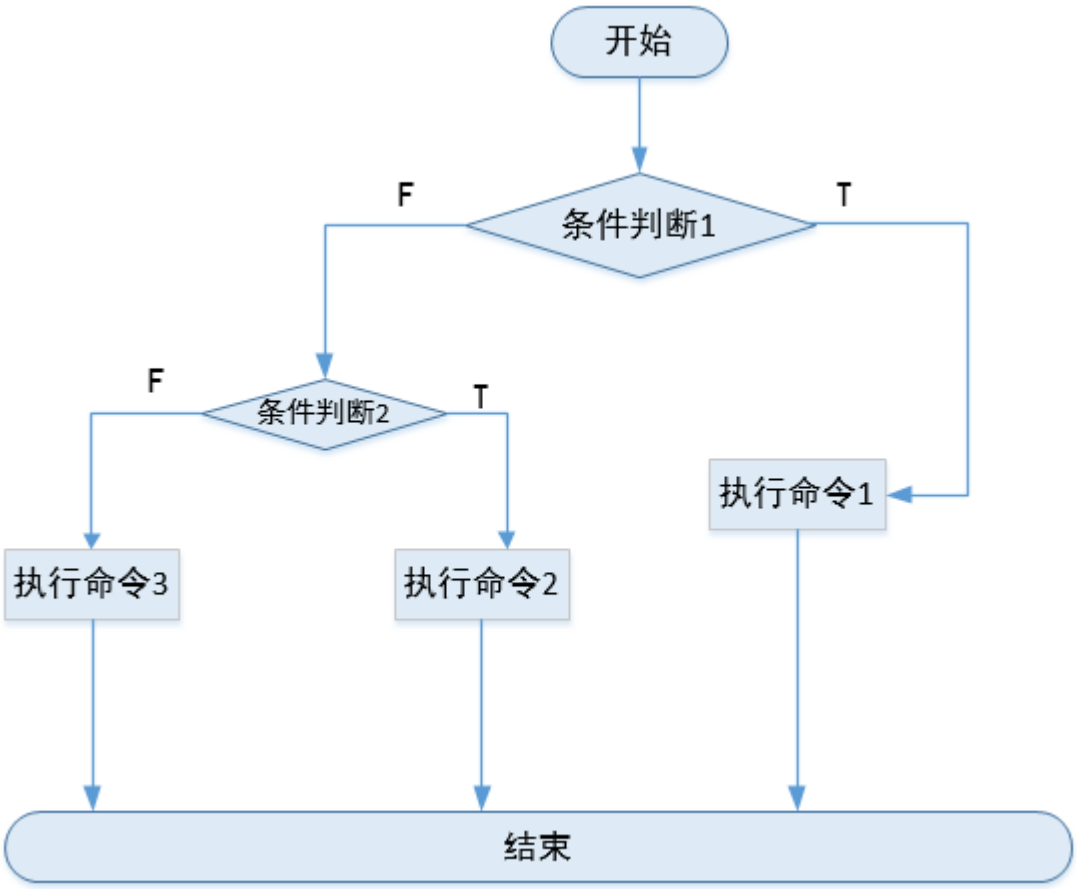
```
echo "该脚本需要传递参数"
if [ $1 = hello ];then
        echo "hello"
else
        echo "请输入hello"
```

```
  5 fi

#!/bin/env bash

A=hello
B=world
C=hello

if [ "$1" = "$A" ];then
        echo "$B"
    else
        echo "$C"
fi


read -p '请输入一个字符串:' str;
 [ "$str" = 'hello' ] && echo 'world' ||  echo '请输入hello!'
```

㈢ if...elif...else结构

箴言3：选择很多，能走的只有一条

```
if [ condition1 ];then
      command1    结束
   elif [ condition2 ];then
      command2    结束
   else
      command3
fi
注释:
如果条件1满足，执行命令1后结束；如果条件1不满足，再看条件2，如果条件2满足执行命令2后结束；如果条件1和条件2都不满足执行命令3结束.
```



㈣ 层层嵌套结构

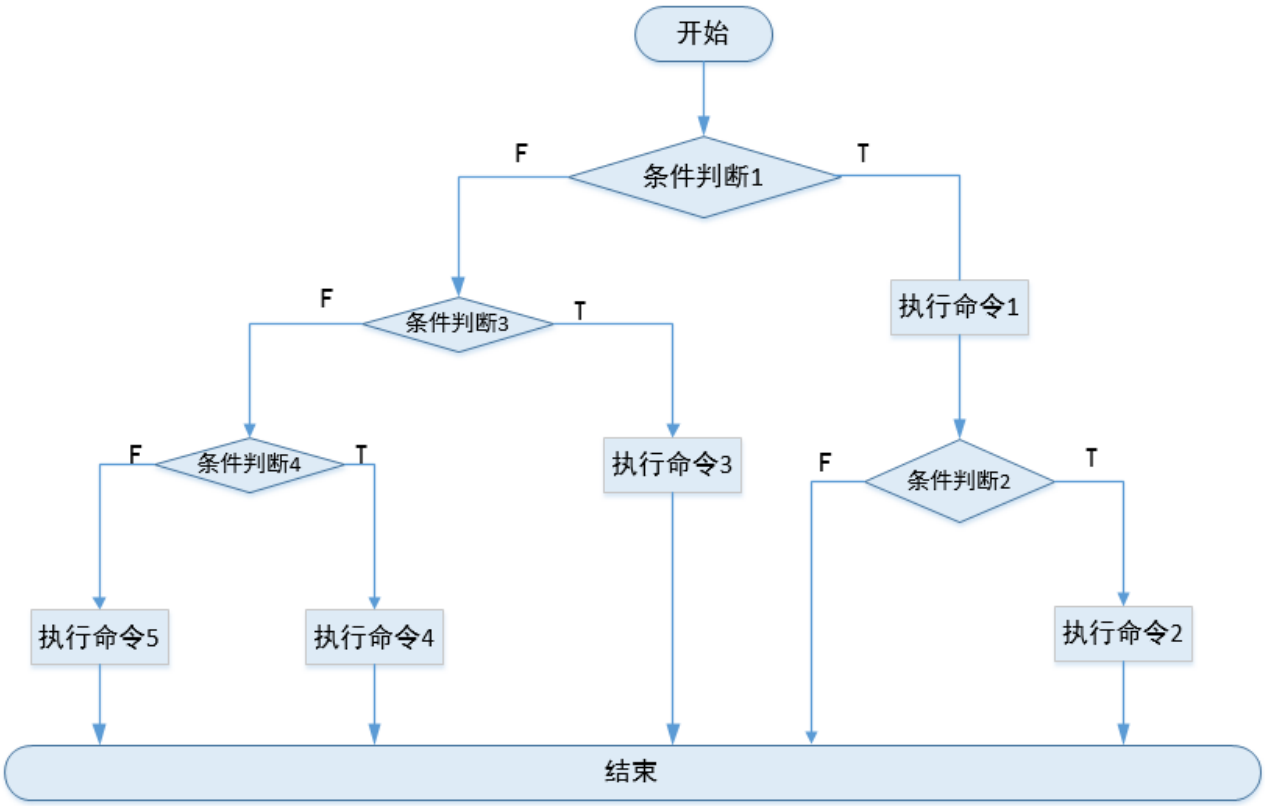
箴言4：多次判断，带你走出人生迷雾。

```
if [ condition1 ];then
      command1
      if [ condition2 ];then
         command2
      fi
 else
      if [ condition3 ];then
         command3
      elif [ condition4 ];then
         command4
      else
         command5
      fi
fi
```

注释：

如果条件1满足，执行命令1；如果条件2也满足执行命令2，如果不满足就只执行命令1结束；

如果条件1不满足，不看条件2；直接看条件3，如果条件3满足执行命令3；如果不满足则看条件4，如果条件4满足执行命令4；否则执行命令5



2. 应用案例

㈠ 判断两台主机是否ping通

需求：判断当前主机是否和远程主机是否ping通

① 思路

1. 使用哪个命令实现 `ping -c次数`
2. 根据命令的执行结果状态来判断是否通 `$?`
3. 根据逻辑和语法结构来编写脚本(条件判断或者流程控制)

② 落地实现

```
#!/bin/env bash
# 该脚本用于判断当前主机是否和远程指定主机互通

# 交互式定义变量，让用户自己决定ping哪个主机
read -p "请输入你要ping的主机的IP:" ip

# 使用ping程序判断主机是否互通
ping -c1 $ip &>/dev/null

if [ $? -eq 0 ];then
   echo "当前主机和远程主机$ip是互通的"
 else
   echo "当前主机和远程主机$ip不通的"
```

```
fi

逻辑运算符
test $? -eq 0 &&  echo "当前主机和远程主机$ip是互通的" || echo "当前主机和远程主机$ip不通的"
```

㈡ 判断一个进程是否存在

需求：判断web服务器中httpd进程是否存在

① 思路

1. 查看进程的相关命令 ps pgrep
2. 根据命令的返回状态值来判断进程是否存在
3. 根据逻辑用脚本语言实现

② 落地实现

```
#!/bin/env bash
# 判断一个程序(httpd)的进程是否存在
pgrep httpd &>/dev/null
if [ $? -ne 0 ];then
   echo "当前httpd进程不存在"
else
   echo "当前httpd进程存在"
fi

或者
test $? -eq 0 && echo "当前httpd进程存在" || echo "当前httpd进程不存在"
```

③ 补充命令

```
pgrep命令：以名称为依据从运行进程队列中查找进程，并显示查找到的进程id
选项
-o：仅显示找到的最小（起始）进程号；
-n：仅显示找到的最大（结束）进程号；
-l：显示进程名称；
-P：指定父进程号；pgrep -p 4764  查看父进程下的子进程id
-g：指定进程组；
-t：指定开启进程的终端；
-u：指定进程的有效用户ID。
```

㈢ 判断一个服务是否正常

需求：判断门户网站是否能够正常访问

① 思路

1. 可以判断进程是否存在，用/etc/init.d/httpd status判断状态等方法
2. 最好的方法是直接去访问一下，通过访问成功和失败的返回值来判断
    - Linux环境，wget curl elinks -dump

② 落地实现

```
#!/bin/env bash
# 判断门户网站是否能够正常提供服务

#定义变量
web_server=www.itcast.cn
#访问网站
wget -P /shell/ $web_server &>/dev/null
[ $? -eq 0 ] && echo "当前网站服务是ok" && rm -f /shell/index.* || echo "当前网站服务不ok，请立刻处理"
```

3. 课堂练习

㈠ 判断用户是否存在

需求1：输入一个用户，用脚本判断该用户是否存在

```
 #!/bin/env bash
  2 read -p "请输入一个用户名: " user_name
  3 id $user_name &>/dev/null
  4 if [ $? -eq 0 ];then
  6     echo "该用户存在! "
  7 else
  8     echo "用户不存在! "
  9 fi


#!/bin/bash
# 判断 用户 (id)  是否存在
read -p "输入壹个用户: " id
id $id &> /dev/null
if [ $? -eq 0 ];then
        echo "该用户存在"
else
        echo "该用户不存在"
fi

#!/bin/env bash
read -p "请输入你要查询的用户名:" username
grep -w $username /etc/passwd &>/dev/null
if [ $? -eq 0 ]
then
    echo "该用户已存在"
else
    echo "该用户不存在"
fi

#!/bin/bash
read -p "请输入你要检查的用户名: " name
```

```
id $name &>/dev/null
if [ $? -eq 0 ]
then
echo 用户"$name"已经存在
else
echo 用户"$name"不存在
fi

#!/bin/env bash
#判断用户是否存在
read -p "请写出用户名" id
id $id
if [ $? -eq 0 ];then
        echo "用户存在"
else
        echo "用户不存在"
fi

#!/bin/env bash
read -p '请输入用户名:' username
id $username &>/dev/null
[ $? -eq 0 ] && echo '用户存在' || echo '不存在'
```

㈡ 判断软件包是否安装

需求2：用脚本判断一个软件包是否安装，如果没安装则安装它（假设本地yum已配合）

```
```

㈢ 判断当前主机的内核版本

需求3：判断当前内核主版本是否为2，且次版本是否大于等于6；如果都满足则输出当前内核版本

```
思路：
1. 先查看内核的版本号    uname -r
2. 先将内核的版本号保存到一个变量里，然后再根据需求截取出该变量的一部分：主版本和次版本
3. 根据需求进步判断


#!/bin/bash
kernel=`uname -r`
var1=`echo $kernel|cut -d. -f1`
var2=`echo $kernel|cut -d. -f2`
test $var1 -eq 2 -a $var2 -ge 6 && echo $kernel || echo "当前内核版本不符合要求"
或者
[ $var1 -eq 2 -a $var2 -ge 6 ] && echo $kernel || echo "当前内核版本不符合要求"
或者
[[ $var1 -eq 2 && $var2 -ge 6 ]] && echo $kernel || echo "当前内核版本不符合要求"
```

或者

```
#!/bin/bash
kernel=`uname -r`
test ${kernel:0:1} -eq 2 -a ${kernel:2:1} -ge 6 && echo $kernel || echo '不符合要求'
```

其他命令参考:

```
uname -r|grep ^2.[6-9] || echo '不符合要求'
```