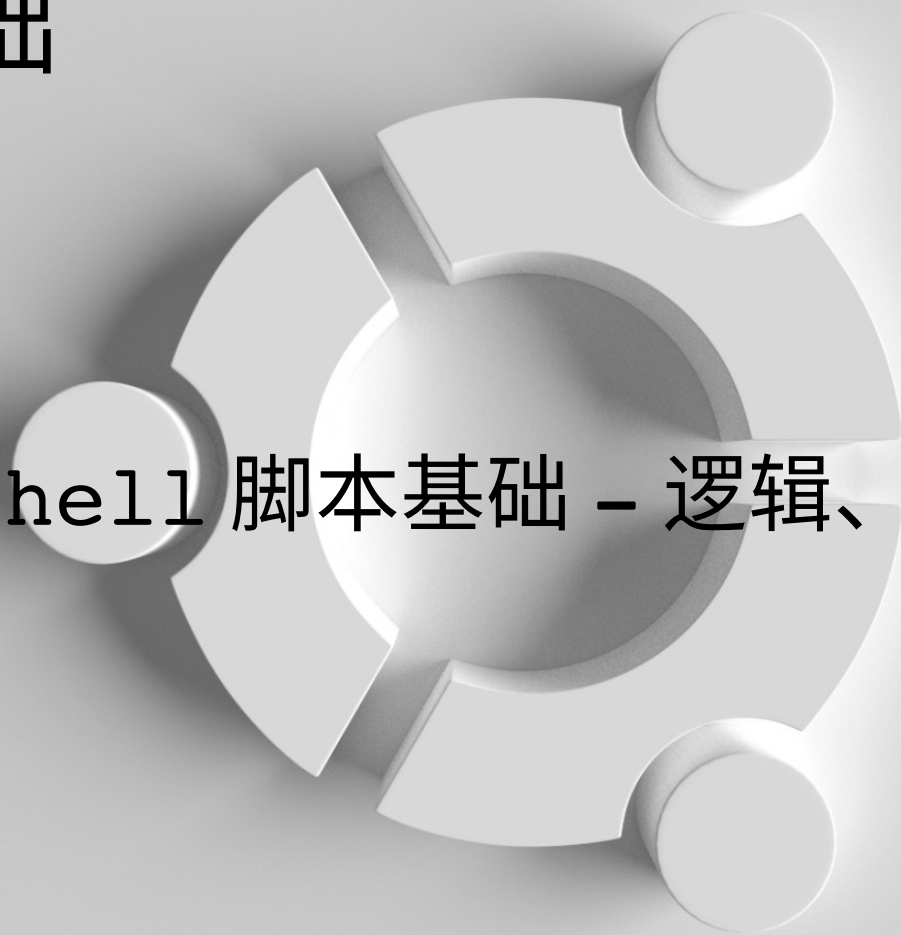


Linux 基础

第 15 讲 shell 脚本基础 - 逻辑、循环、函数



test 命令

- `test` 是 `shell` 的内建命令，经常用于各类的检测工作，产生的不是一般形式的输出，而是可用的退出状态。
- `test` 经常和 `if`，`while` 等用于条件的关键字配合使用。

test 命令

- `test` 返回的结果在 `shell` 中运行不会有输出。
- 查看 `test` 帮助文档: `help test`。
- `test` 返回 `true` 或 `false`。

shell 中的 true 和 false

- shell 中通过程序的最终的返回值判断程序的运行状态。
- 但是和通常的编程语言不同的是，在 shell 中 true 是 0，非 0 值为 false_[1]。

[1] 当初 Ken·Thompson 和 Dennis·Ritchie 设计 Unix 和 C 语言时，把返回值 0 作为程序正确运行的状态，表示 0 错误，其他任何非 0 值都表示出错了。所以在 C 语言中 main 函数最后都会有 return 0;

test 示例

- `test "abc"="abc"` # 检测字符串是否相等
- `test -f c/a.c` # 检测 `c/a.c` 文件是否存在并且为普通文件
- `test -d bin` # 检测 `bin` 是否为目录
- `test -z $A` # 如果字符串为空则返回 `true`
- `test -n $A` # 如果字符串不为空则返回 `true`

if elif else

```
if [COMMAND]
then
    [COMMAND]
fi
```

```
if [COMMAND]; then
    [COMMAND]
else
    [COMMAND]
fi
```

```
if [COMMAND]; then
    [COMMAND]
elif [COMMAND] ;
then
    [COMMAND]
else
    [COMMAND]
fi
```

写在一行要使用分号分隔：

```
if [COMMAND] ; then [COMMAND] ; fi
```

case

- case 相当于其他编程语言的 switch， case 结构如下：

```
case WORD in
    VALUE1)
        [COMMANDS]
        ;;
    VALUE2)
        [COMMANDS]
        ;;
    *)
        [COMMANDS]
        ;; //esac之前的;;可以省略
esac
```

for 循环

- for 循环用于遍历整个列表。
- `for NAME in WORDS; do COMMANDS; done`
- `for NAME in WORDS ; do
 COMMANDS
done`

while 循环

- while CONDITION ; do

COMMANDS

done

- while 循环的条件是命令的执行状态。

示例

- 实时显示时间 `lt.sh` :

```
1  #!/bin/bash
2
3  while date ; do
4      sleep 1
5      clear
6  done
```

shell 脚本的一些特殊变量

- `$0` : 当前 shell 脚本的名称。
- `$N` : 参数, 使用 `$1, $2...`, 10 以上要使用 `${10}` 的形式。
- `$@` : 所有参数, 可以使用 `for` 循环遍历。
- `$#` : 参数个数, 不包括脚本名称 (`$0`)。
- `$?` : 上一个命令的返回值, 通常在 `if` 中使用。
- `$$` : 当前 shell 的 PID。

test 等效形式

- test 等效于 [...] ，注意括号中要有空格
分开。

```
if test -f bin/a ; then
    echo 'bin/a exists'
fi
```

等效

```
if [ -f bin/a ] ; then
    echo 'bin/a exists'
fi
```

函数

- shell 中的函数编写很简单。
- 直接写函数结构，并在其中编写命令。
- 函数的名称就像是命令直接调用。

```
3 #定义函数
4 time_say() {
5     while date ; do
6         sleep 1
7         clear
8     done
9 }
10
11 #调用
12 time_say
```

函数的参数

- 函数的参数传递和使用命令传递参数形式相同。
- 在函数中，`$@` 获取的是传递给函数的参数，`shell` 的参数此时被隐藏，或者说被掩盖。

```
3 funca() {  
4     for a in $@ ; do  
5         echo $a  
6     done  
7 }  
8  
9 funca 1 2 3  
10  
11 for a in $@ ; do  
12     echo $a  
13 done
```