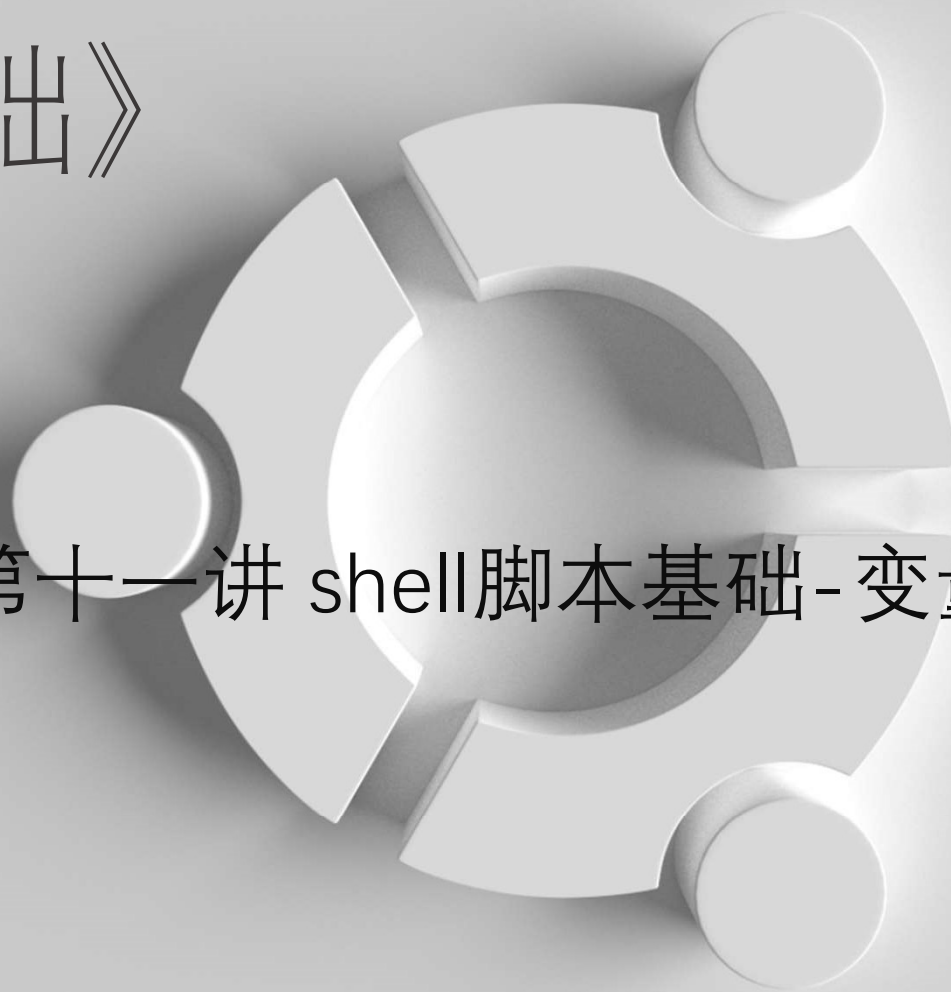


# 《Linux基础》

## 第十一讲 shell脚本基础-变量



# shell在系统的角色

---

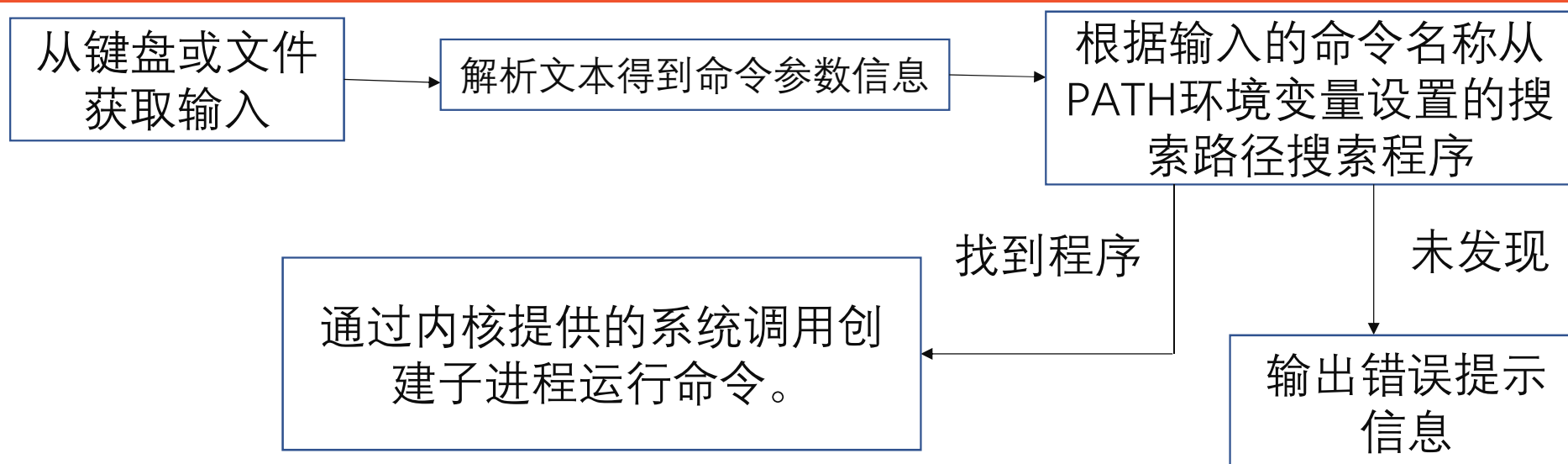
- shell是用户和系统交互的桥梁：
  - shell是Linux的一个程序，实现版本有多种。shell的主要工作就是运行命令。
- sh是shell的简写，shell的实现版本有：sh, bash, csh, tcsh, zsh。
- 多数Linux默认的shell是bash。Linux启动登录以后，会运行一个shell等待用户输入命令。
- 用户通过shell和系统交互。

# shell环境变量

---

- 当前shell运行时保存的信息，包括终端类型，当前目录，主目录，语言编码，默认命令搜索路径等信息。运行env命令可以查看当前环境变量。
- 环境变量是一个名称和值的对应列表。一种是shell启动时解析配置文件生成，还有一种临时环境变量是在shell中使用export生成。
- PATH变量记录了要查找命令的路径顺序；HOME记录当前用户主目录；PWD是当前工作目录，每次切换目录都会变化。
- 使用\$取值：echo \$PWD，而echo PWD仅仅是输出PWD字符串。

# shell运行命令的基本过程



## shell运行命令的文字描述

---

- 输入一条命令并确认后，实际shell获取的是一行字符串，shell要对字符串进行解析，并确定命令名称，参数等信息。shell支持从文件读取文本逐条解释执行。
- 然后，shell要根据配置文件的搜索路径，从每个路径寻找命令，没有找到则提示错误信息，找到就调用Linux提供的系统调用运行命令。
- 比如：输入 `ls -l`, shell要解析成'`ls`'，'`-l`'，'`ls`'就是命令名称，并在PATH设置的路径中寻找，找到`/bin/ls`这个命令，fork一个子进程调用`execv`等系统调用传递参数运行命令。并等待结束。
- 注意：真正运行命令的不是shell，而是内核，shell去调用内核提供的接口，shell是调用`fork`创建子进程去运行命令。

## shell中的通配符

---

- shell支持通配符，使用\*表示匹配任意长度的字符，?匹配任意一个字符。
- shell在遇到通配符会进行扩展，比如输入ls ./a\*，会匹配a开头的所有文件并显示，如果存在ab.txt，ac.txt，则会扩展成ls ./ab.txt ./ac.txt。
- 注意：扩展通配符的是shell，不是命令自身，如果是命令本身实现的，那就每个命令都要实现。
- 通配符在使用时会带来很多便利，但是有些特殊情况也要注意。

# shell脚本

---

- shell可以从一个文件读取命令并逐条执行。文件一般被称为脚本。
- 大多数Linux发行版的默认shell都是bash。
- 文件第一行使用#!/bin/bash表明这是一个bash脚本，注意有些脚本程序使用#!/bin/sh表示。
- 在Ubuntu/Debian上，sh是一个符号链接指向dash，dash是一个专为执行脚本而设计的shell程序，执行速度快，语法遵循POSIX标准，但是功能比bash少很多。
- 一个简单的脚本：开头声明这是一个bash脚本，然后是主要操作代码，最后以exit 0退出。

```
1 #!/bin/bash
2 echo 'Hello world'
3 exit 0
```

# 脚本的可执行权限

---

- 执行脚本可以使用 `bash [SCRIPT NAME]`，此时bash读取脚本文件并执行，`#!/bin/bash`是被解释为注释。
- 另一种方式就是给脚本添加可执行权限：`chmod +x [SCRIPT NAME]`
- 给脚本添加执行权限，脚本开头的`#!/bin/bash`声明这是一个脚本文件，要用`/bin/bash`执行。



# 变量

---

- shell运行`a=123`就定义了a变量。shell中的变量就是为某些需要保存的数据用一个名称标记，方便以后使用。变量的名称以字母或是下划线符号开头，后可跟任意长度的字母、数字、下划线。
- `=`左右不能有空格，否则会按照运行命令的方式去执行。
- `a=`ls`` 会把ls运行的结果赋值给a。注意ls不是被单引号包含，而是反引号。
- 获取变量的值要用`$`，`echo $a`可以输出变量的值。
- shell中的变量就是键值对 (key-value) 的列表，都是以文本的形式存储的。`a=1+2`不会进行计算把3赋值给a，而是a的值就是‘1+2’这段文本。变量的值可以用双引号/单引号括起来，包含空格的变量就必须这么做。

## 变量查看与清除

---

- 用set命令可以查看所有的变量。
- 可以使用 `set | grep linux` 进行搜索
- `unset VAR`命令可以清除变量VAR，相当于没有定义过。使用空格分隔多个变量。

# 只读变量

---

- 变量设置后，是可以修改值的：a=12; a=13，此时a的值就是13
- readonly把变量设置为只读：readonly a
- 但是设置之后，只读变量就无法更改和清除。除非重置shell环境。

# 算数运算

---

- shell支持算术运算，并且shell会对 $\$(\dots)$ 里的算数表达式进行运算。

```
a=12;b=14
```

```
x=$(($a+$b))
```

```
echo $x
```

如果 $b=12a$ ，此时会报错，但是如果以字母开头的文本，比如 $b=a12$ ，则 $x=$(($a+$b))$ 则直接就计算为 $a$ 的数值， $b$ 转成数字为0。

# 逻辑运算

---

- 逻辑运算：&&, ||, !。分别是AND, OR, NOT。
- 对逻辑运算来说，任何非0值都是真。
- 示例：echo \$((1&&0)) ; echo \$(( 2 || 0))
- 非数字格式逻辑运算：  
b=abc  
echo \$(( 1 && \$b )) //输出是0  
/\*\*\*\*\*/  
b=12a  
echo \$(( 1 && \$b )) //提示错误

## 放进环境变量

---

- 环境变量是全局存在的，在任何shell脚本中都可以直接使用。
- 使用env查看环境变量。
- export a: 把变量放到环境变量，环境变量是一个名称与值的简单列表。
- 使用示例，shell中执行：

```
$ env | grep linux  
$ linux=1  
$ export linux  
$ env | grep linux
```

# 变量的引用

---

- `${#var}` 返回变量值（字符串）的长度
- `${var:start_index}` 返回从start\_index开始一直到字符串结尾的字符串
- `${var:start_index:length}` 返回从start\_index开始的length个字符
- `${var:-newstring}` 如果var未定义或为空值，则返回newstring；否则返回var的值
- `${var:=newstring}` 如果var未定义或为空值，则返回newstring，并把newstring赋给var；否则返回var的值
- `${var:+newstring}` 如果var不为空，则返回newstring；否则返回空值（其实也是var的值）
- `${var:?newstring}` 如果var未定义或为空值，则将newstring写入标准错误，本语句失败；否则返回var的值

# 脚本的特殊变量

---

- \$0: 当前脚本的文件名/当前执行的进程/程序名
- \$n: n为从1开始的数字, \$1是第一个参数, \$2是第二个参数, \${10}是第十个参数 (从\${10}开始参数号需要用花括号括起来)
- \$#: 传入脚本的参数的个数
- \$\*: 所有的位置参数(作为单个字符串)
- \$@: 所有的位置参数(每个都作为独立的字符串)。
- \$?: 当前shell进程中, 上一个命令的返回值, 如果上一个命令成功执行则\$?的值为0, 否则为其他非零值, 常用做if语句条件
- \$\$: 当前shell进程的pid
- \$!: 后台运行的最后一个进程的pid



## 本节课任务

---

- 创建脚本文件: `varstudy.sh`
- 赋予可执行权限
- 脚本运行时输出参数个数
- 并输出参数个数, 以及程序的名称和参数字符串
- 显示当前工作目录以及用户主目录