Linux Learning

# 初识shell

## External → Shell → Linux

## bash : Bourne Again Shell, 简称bash

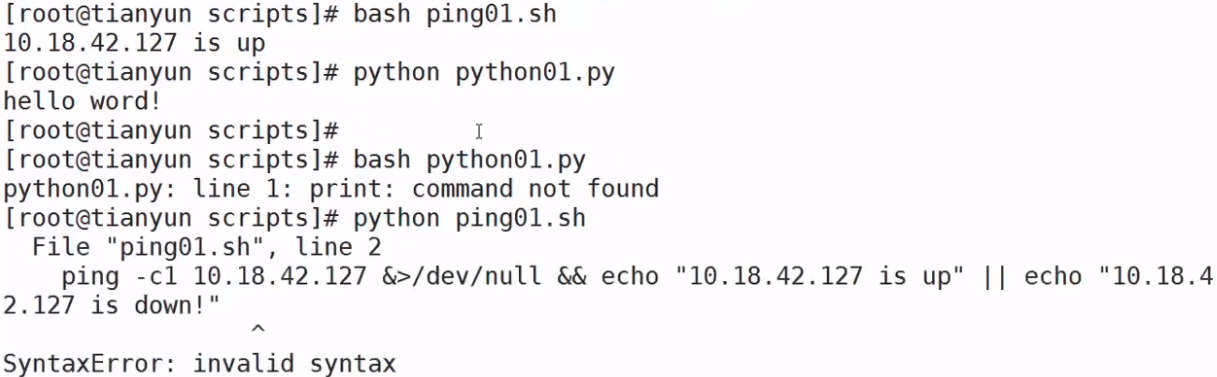
Linux 默认的shell

## 初始shell

### 创建

* + - 1. 创建以.sh为扩展名的文件
         1. vi hello.sh

### 编辑

* + - 1. 第一行以 “#!” 开头，声明所使用shell的全路径（#!：shebang符号）
         1. #!/usr/bin/bash/ ：表示该脚本用/usr/bin/bash/解释器执行）
         2. 

不可以用shell的bash解释器去执行.py文件，同理也不可以用python解释器去执行shell文件

#!/usr/bin/bash：该解释器，只有当在shell中没有输入执行命令时(e.g. pingg01.sh)，才会生效

* + - 1. 注释
         1. 使用 “#” 作为注释符号
      2. e.g.

#!/bin/bash

# print “Hello World!”

echo “Hello World!”

# 在一般的linux系统当中（如redhat），使用sh调用执行脚本相当于

# 打开了bash的POSIX标准模式

# 也就是说 /bin/sh 相当于 /bin/bash --posix

# 可移植操作系统接口（英语：Portable Operating System Interface

# 缩写为POSIX）是IEEE为要在各种UNIX操作系统上运行软件

### 增加脚本文件执行权限

* + - 1. chmod +x XXX.sh #所有用户均对该脚本有执行权限
      2. chmod 755 XXX.sh
      3. 注释

在linux终端先输入ll,可以看到如:

-rwx-r--r-- (一共10个参数) 表示文件所属组和用户的对应权限。

第一个跟参数属于管理员，跟chmod无关,先不管.

2-4参数:属于user

5-7参数:属于group

8-10参数:属于others

接下来就简单了:r==>可读 w==>可写 x==>可执行

r=4 w=2 x=1

所以755代表 rwxr-xr-x

777 代表 rwxr-rwx-rwx 所有用户都可读可写可执行。

e.g.

-rwxr-xr-x 1 jingsheng jingsheng 17 Feb 29 18:44 date\_755.sh\*

-rwxrwxrwx 1 jingsheng jingsheng 17 Feb 29 18:40 date\_777.sh\*

-rwxrwxr-x 1 jingsheng jingsheng 17 Feb 29 18:43 date\_x.sh\*

### 运行脚本方法

* + - 1. 进入脚本目录
         1. cd /home/jingsheng/script
      2. 运行脚本
         1. ./XXX.sh # 运行脚本
         2. 以/bin/sh XXX.sh 来运行脚本

e.g. /bin/sh hello.sh

* + - * 1. bash XXX.sh 可以运行脚本，无需权限
      1. Difference(source / sh / bash / ./)
         1. source a.sh

在当前shell内去读取、执行a.sh，而a.sh不需要有"执行权限"

source命令可以简写为"."

* + - * 1. sh a.sh

bash a.sh

都是打开一个subshell去读取、执行a.sh，而a.sh不需要有"执行权限"

通常在subshell里运行的脚本里设置变量，不会影响到父shell的。

* + - * 1. ./a.sh

#bash: ./a.sh: 权限不够

chmod +x a.sh

./a.sh

打开一个subshell去读取、执行a.sh，但a.sh需要有"执行权限"

可以用chmod +x添加执行权限

# 符号表达

## 通配符(元字符)

* + 1. \*：匹配任意多个字符
       1. e.g. ls \*.pdf
          1. 显示该文件中所有的PDF文件
    2. ？：匹配一个字符
       1. e.g. ls L?ve.pdf
          1. 显示LXve.pdf的文件 X代表不确定字符
    3. []：匹配括号中任意一个字符
    4. ()：在子shell中执行
    5. {}：集合touch file{}
    6. \：转义字符，让元字符回归本意

## 

# 变量

## 自定义变量(类别C,局部变量)

### 定义变量

* + - 1. 变量名 = 变量值
         1. e.g. ip1 = 192.168.1.1

### 引用变量

* + - 1. $变量名/${变量名}
         1. e.g. $ip1

### 查看变量

* + - 1. echo $变量名
         1. e.g. echo $ip1

### 作用范围

* + - 1. 仅在当前shell中生效

## 环境变量(类比C,全局变量)

### 定义环境变量

* + - 1. 直接定义
         1. export back\_dir = /home/backup
      2. 转换
         1. export back\_dir\_ZDY (将自定义变量转换为环境变量[back\_dir\_ZDY 为自定义变量])

### 引用变量

* + - 1. $变量名/${变量名}
         1. e.g. $back\_dir

### 查看变量

* + - 1. echo $变量名
         1. e.g. echo $ip1
      2. 查看全部
         1. env
         2. Env |grep back\_dir

### 作用范围

* + - 1. 在当前shell中生效和子shell均有效

# 常用命令

## 显示PATH变量

### echo $PATH

## 查看环境变量

### env (查看所有环境变量)

### env |grep SIZE (该命令表示，查看带有SIZE的环境变量。|grep： 信息过滤)

# 小提示Tip

## 可以在当前shell中调用其他shell

* + 1. . publish.sh 例如将该命令写入其他shell，则可以调用publish.sh