**shell脚本运行的原理（source命令等）**

2017年06月28日 16:29:36 阅读数：2604 标签： [shell](http://so.csdn.net/so/search/s.do?q=shell&t=blog)[linux](http://so.csdn.net/so/search/s.do?q=linux&t=blog)[脚本](http://so.csdn.net/so/search/s.do?q=%E8%84%9A%E6%9C%AC&t=blog)[编程](http://so.csdn.net/so/search/s.do?q=%E7%BC%96%E7%A8%8B&t=blog) 更多

个人分类： [Linux/操作系统](https://blog.csdn.net/xy913741894/article/category/6796470)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/xy913741894/article/details/73849441

Shell英文意思是外壳， 用户可以通过Shell和操作系统交互，通俗的讲Shell就是一个解释器，当我们输入命令Shell就解释执行，Shell有很多版本，我们一般使用的都是bash，在Linux输入一条命令可以查看当前正在使用的Shell：

echo $SHELL

屏幕会出现：

/bin/bash

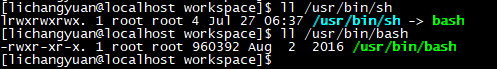
查看当前是同使用的shell解释器。

ll /usr/bin/sh

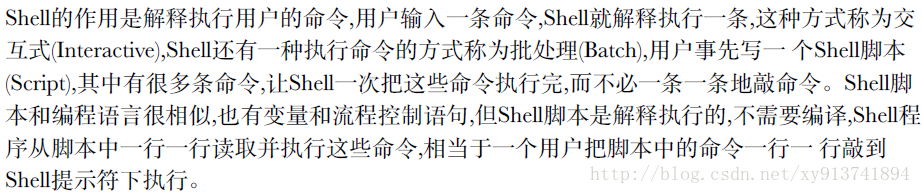
ll /usr/bin/bash

屏幕会出现：

说明默认使用的使bash



Shell有两种方式：



因此，我们主要来学习Shell脚本编程。

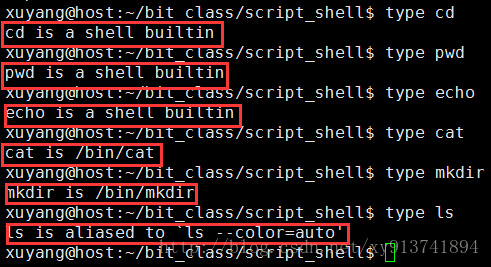
**在理解Shell脚本运行的原理之前，我们必须得先搞清楚shell运行的原理：**

当我们在Linux下输入：

ls -al 或者 cd ..

到底发生了什么？

在解释原理之前，我们必须分清楚的是：我们在Linux输入的命令是有区别的，具体来说，分为内部命令（built-in）以及外部命令，向ls，cat，mkdir这些都属于外部命令，而echo，cd，pwd这些都属于内置命令，如何区分这些命令是否是内置，外部命令，可以利用type命令来辨别



输入ls，cd这些内部或者外部的命令有什么区别？具体而言：

像cd，pwd这些内置命令是属于Shell的一部分，当Shell一运行起来就随Shell加载入内存，因此，当我们在命令行上输入这些命令就可以像调用函数一样直接使用，效率非常高。

而如ls，cat这些外部命令却不是如此，当我们在命令行输入cat，当前的Shell会fork一个子进程，然后调用exec载入这个命令的可执行文件，比如bin/cat，因此效率上稍微低了点。

理解了单个命令的运行原理，我们现在探究当编写一个Shell脚本运行的原理，首先需要的说明的是，Shell，和Python一样，都是一门解释性语言，所谓的解释性语言就是不用编译，执行一行解释一行。

当我们编写一个Shell脚本test.sh，比如：

#!/bin/bash

echo "hello"

ls -al

在Linux命令输入./ test.sh时（需要加可执行权限）时，我们在这个文本文件开头指定了bash为默认的解释器，因此当前的交互式Shell会fork一个子进程，用bash解释器的代码去替换（也就是exec），而这个文本文件被当作是命令行参数传给这个子bash，等这个子bash执行完就会到我们的交互式bash了。

为了验证我们所说的交互式bash会fork/exec一个子bash来执行命令而非交互式bash，我们做个测试：

**1.直接在交互式bash输入：**

cd ..

**2.编写脚本test.sh，然后运行：**

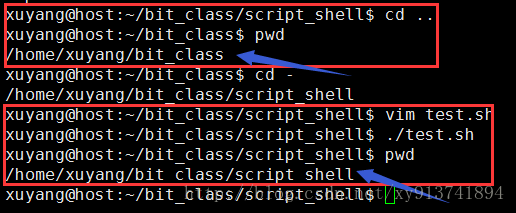
//其中test.sh内容如下：

#!/bin/bash

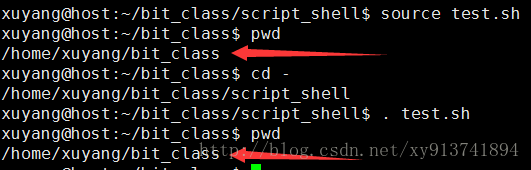
cd ..

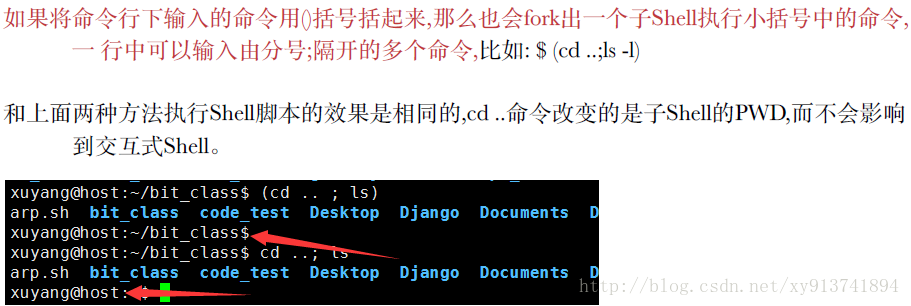
在Linux命令行输入 ./test.sh

**结果如下：**



我们发现同样都是cd ..，为什么执行完脚本当前bash目录不变，这就证明了我们所说的是由子bash执行，而非交互式bash，那么如何交互式bash执行脚呢，我们可以使用. 或者source命令，如下图：





···········牛号

！！！！！！！！！！！！！！！！！！！！！

@@@@@@@@@@@@@@

^^^^^^^^^^^^^^^^^^^^^

&&&&&&&&&&&&&&&&&&&&

(((((((((((((((((((((((((((

)))))))))))))))))))))))

++++++++++++++++++++++++++++

《《《《《《《《《《《

<<<<<<<<<<

>>>>>>>>>>>>>>

》》》》》》》》》》》》

？？？？？？？？？？？？

////////////////

；；；；；；；；；

‘’‘’‘’‘’‘’‘

：：：：：：：：：：

“”“”“”“”“”“”

、、、、、、、、、

|||||||||||||||||

|||||||||||||||

\\\\\\\\\\

解释二

注意：当指定用sh解释器去执行脚本时，脚本内第一行解释器说明就会失效(#!/bin/sh)或（#!/bin/bash）.

source filepath 或者 . filepath

使当前shell解释器读入路径为filepath和shell文件并以此执行文件中所有语句，通畅用于重新执行刚修改的初始化文件，使之生效，而不必注销并重新登陆。六日我们修改了/etc/profiile文件，并向让他立刻生效，而不用重新登陆，就可以使用source命令，如source /etc/profile。

source命令(从 C Shell 而来)是bash shell的内置命令；点命令(.)，就是个点符号(从Bourne Shell而来)是source的另一名称。这从用法中也能看出来。

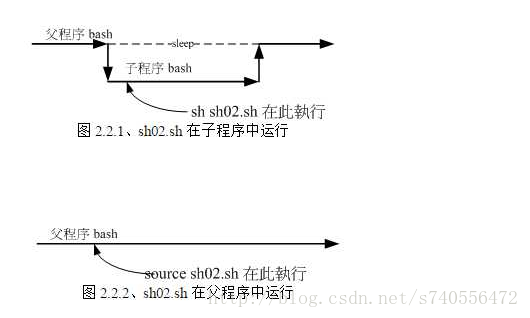
使用这个命令执行脚本即使没有执行权限也可以运行。

## sh

sh是一个shell。运行sh xxx.sh，表示我使用sh来解释这个脚本,sh和source命令

一样，没有执行权限也可以运行脚本。

但是这两者还是有些区别的，如下图所示



## 总结：

Source 和 . 命令会强制shell脚本使用交互式执行。

Sh和source 及时脚本没有执行权限可以运行。

./需要先将文件提升为可执行权限，

例;chmod +x filename。还有这种方式会查找文件第一行是否指定了解释器，如果没指定就用当前系统默认的shell（大多说linux默认使bash），如果指定了解释器，那么就将改脚本交给指定的解释器。

案例分析：

总结生产环境启动control启动失败问题。

shell 脚本查看笔记

if [ -z "$JAVA\_HOME" ]; then

JAVA\_cmd=$(which java)

if [-x "JAVA\_CMD"]; then

JAVA\_HOME=$(dirname $(dirname $JAVA\_CMD))

else

echo "No JAVA\_HOME specified and executable java command"

exit 1

fi

fi

############################################################################

whift

while [! $# -eq 0 ]; do

CASE "$1" IN

-i)

INTERACTIVE=yes

shift

;;

-system-id)

sheift

if [ ! -z "$1" ]; then

SYSTEM\_ID=$1

fi

shift

;;

-server-id)

shift

if [ ! -z "$1"]; then

SERVER\_ID=$1

fi

PIDfILE=$(pwd)/${SERVER\_ID}.pid

STOPSINGLE=1

shift

;;

-machine-id)

shift

if [ ! -z "$1" ]; then

MACHINE\_ID=$1

fi

shift

;;

--help|-h)

shift

usage

;;

\*)

export EXTRA\_ARGS="$EXTRA\_ATGS $1"

shift

;;

esac

done

LOGS\_DIR=$(pwd)/logs

LIBS=$(find `pwd` -name '\*.jar' -type f -o -type l | tr '\n' ':')

CLASSPATH=$LIBS:$CLASSPATH

SERVER\_ID=$(basename $(pwd))

MAIN\_CLASS=org.springframework.boot.loader.JarLauncher

JAVA\_CMD=$JAVA\_HOME/bin/java

JDB=$JAVA\_HOME/bin/jdb

STOPSIG="itframe${SERVER\_ID}\_`date +%Y%m%d%H%M%S`"

JAVA\_OPTS="$JAVA\_OPTS -Dfile.encoding=UTF-8"

JAVA\_OPTS="$JAVA\_OPTS -Dcmos.system.id=${SYSTEM\_ID}"

JAVA\_OPTS="$JAVA\_OPTS -Dmachine.id=${MACHINE\_ID}"

JAVA\_OPTS="$JAVA\_OPTS -Dtask.txt.path=${TASK\_TXT\_PATH}"

if [ "$COMMAND" = "start" ]; then

nohup $JAVA\_CMD -cp $CLASSPATH $JAVA\_OPTS $MAIN\_CLASS $EXTRA\_ARGS "$STOPSIG" &>/dev/null &

sleep 1

PID=$(pgrep -f $STOPSIG)

echo "Server started: $PID"

echo $PID > $PIDFILE

fi

shift 参数左移，按照空格符认为一个参数。

$HOME同${HOME}一样都是引用变量，加花括号可以避免歧义，引用变量时推荐加或花括号

shell变量的种类：

最大区别 ：本地变量执行export后，会成为环境变量，用unset命令可以删除已定义的环境变量或本地变量。

环境变量：shell进程的环境变量可以从当前shell进程传给fork出来的子进程

本地变量：只存在与当前shell进程。

根据时间区分为临时和永久；只有添加到path中的才会成为永久变量

命令代换：

反引号 `` 或 $()

shell 会先执行反引号或$()中的命令，将结果带换到当前命令中！

反引号和$()的区别：

单引号和双引号

Shell脚本中的单引号和双引号一样都是字符串的界定符，单引号用于保持引号内所有字符的字面值，而双引号有些情况则特殊。如果字符串中有特殊字符需要处理是就用双引号。

注：单引号标注的字符中不能再出现单引号

例子：

echo '\\'

echo "\\"

echo '`date`'

echo "`date`"

输出结果:

\\

\

`date`

Fri Aug 12 00:39:44 CST 2016