Linux命令鸟瞰

笨办法学

概述

- 内部命令和内置命令 Internal Commands and Builtins
- 外部程序和命令 External Programs and Commands
- 系统和管理命令 System and Administrative Commands

KISS=Keep It Simple, Stupid)

◆ 内部命令和内置命令 Internal Commands and Builtins

- I/O
- 文件系统 Filesystem
- 变量 Variables
- 脚本行为 Script Behavior
- 命令 Commands

I/O

- **echo**
  - -e 可以打印转义字符
  - -n 阻止新起一行
- **printf**
  - 格式化输出 **printf *format-string... parameter...***
- **read**
  - -r 取消转义符
  - -n 限制接受字符的个数
  - -p 提示符
  - -t 限制读取时等待的时间(秒)

read命令的6个示例

- 示例1：基本read实验
- 示例2：新行
- 示例3：keypress
- 示例4：方向与控制键
- 示例5：定时read
- 示例6：文件重定向read

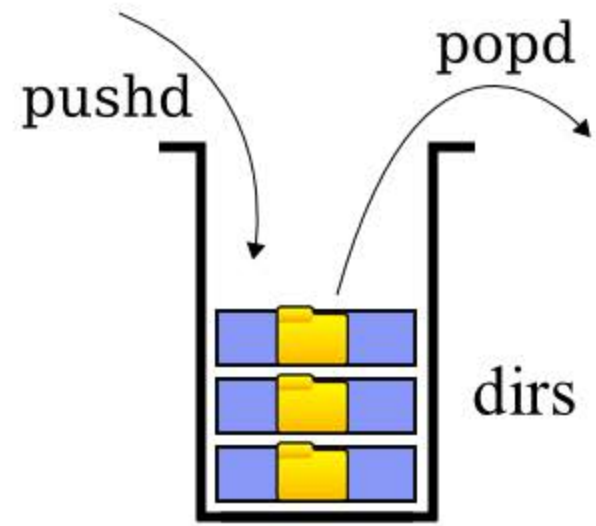
文件系统　　Filesystem

- **cd**
- **pwd**
- **dirs、pushd、popd**
  - **dirs**
    - 显示当前目录栈中的所有记录

格式：**dirs [-clpv] [+n] [-n]**

选项：

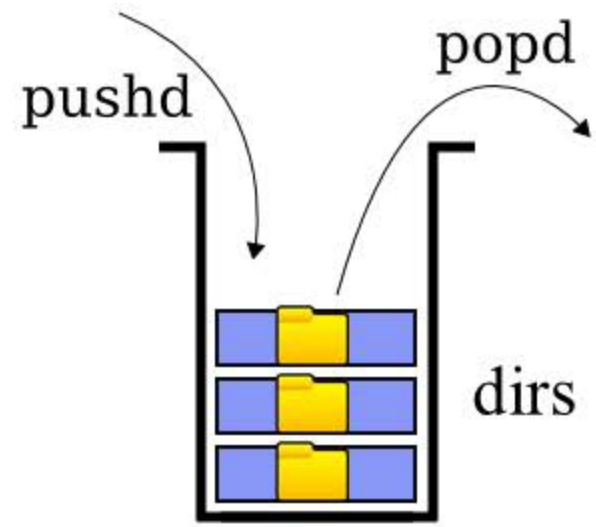
- **-c** 删除目录栈中的所有记录
- **-l** 以完整格式显示
- **-p** 一个目录一行的方式显示
- **-v** 每行一个目录来显示目录栈的内容，每个目录前加上的编号
- **+n** 显示从左到右的第n个目录，数字从0开始
- **-n** 显示从右到左的第n个日录，数字从0开始



的目录栈中的最上面的目录

文件系统　　Filesystem

- **cd**
- **pwd**
- **dirs、pushd、popd**
  - **dirs**
    - 显示当前目录栈中的所有记录
  - **pushd**
    - 带参数：将目录压入到栈中，并切换到该目录
    - 不带参数：将位于记录栈最上面的2个目录对换位置



格式：**pushd [-N | +N |** *目录* **] [-n]**

选项：

目录	将该目录加入到栈顶，并执行"cd 目录"，切换到该目录
+N	将第N个目录移至栈顶（从左边数起，数字从0开始）
-N	将第N个目录移至栈顶（从右边数起，数字从0开始）
-n	将目录入栈时，不切换目录

的目录栈中的最上面的目录

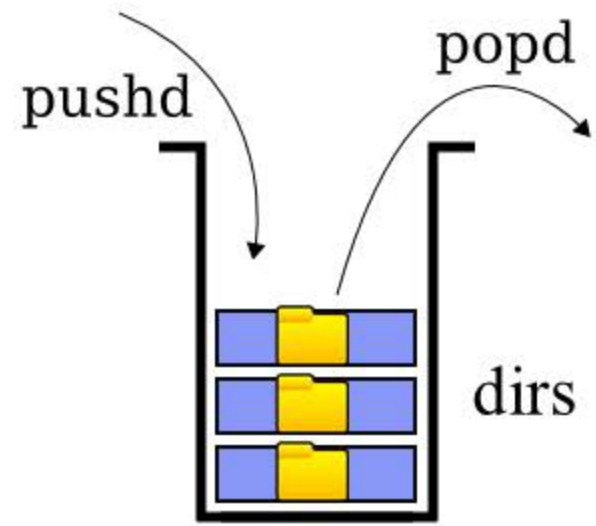
文件系统　　Filesystem

- **cd**
- **pwd**
- **dirs、pushd、popd**

格式：**popd [-N | +N] [-n]**

选项：

+N	将第N个目录删除（从左边数起，数字从0开始）
-N	将第N个目录删除（从右边数起，数字从0开始）
-n	将目录入栈时，不切换目录



- **popd**
  - 带参数：删除目录栈中的记录
  - 不带参数：先删除栈最上面的记录，然后切换到删除后的目录栈中的最上面的目录

变量 Variables

- let
- eval
- set
- unset
- export
- declare, typeset
- readonly
- getopts

root@tomlab1:~

```bash
#!/bin/bash

echo
let a=11                # 与 a=11 赋值相同
let a=a+5               # 等价于 let "a = a + 5"

echo "11 + 5 = $a "     # 16

let "a-=5"              # 等价于 let "a = a - 5"

exit 0
~
~
                                        1,1           All
```

变量 Variables

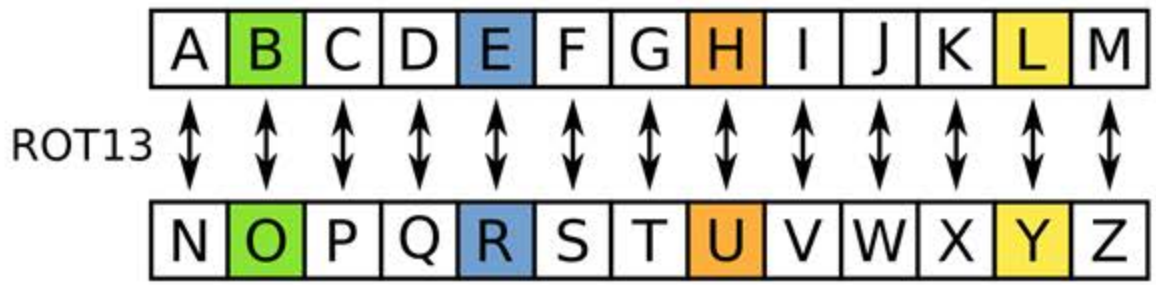
- ~~let~~
- eval
- set
- unset
- export
- declare, typeset
- readonly
- getopts

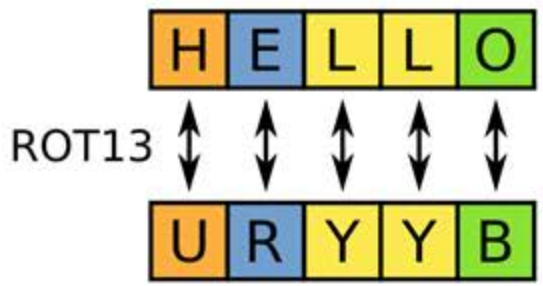
```
1 $ process=sshd
2 $ show_process="eval ps ax | grep $process"

3 $ $show_process
  922 ?        Ss     0:00 /usr/sbin/sshd -D
16664 ?        Ss     0:00 sshd: root@pts/0
17025 pts/0    R+     0:00 grep --color=auto sshd
```

示例：ROT13加密算法

- ROT13：回转13位，rotate by 13 places
- ROT13 是凯撒密码的一种变体





通过bash中的tr命令来实现：

```
tr a-z n-za-m
```

变量 Variables

- ~~let~~
- ~~eval~~
- **set**
  - 通过修改**Bash**选项标志位，从而达到修改定脚本的行为的目的
  - 重新设置脚本的位置参数
- **unset**
- **export**
- **declare, typeset**
- **readonly**
- **getopts**

```
set [--abefhkmnptuvxBCEHPT] [-o option-name] [arg ...]

set [+abefhkmnptuvxBCEHPT] [+o option-name] [arg ...]
```

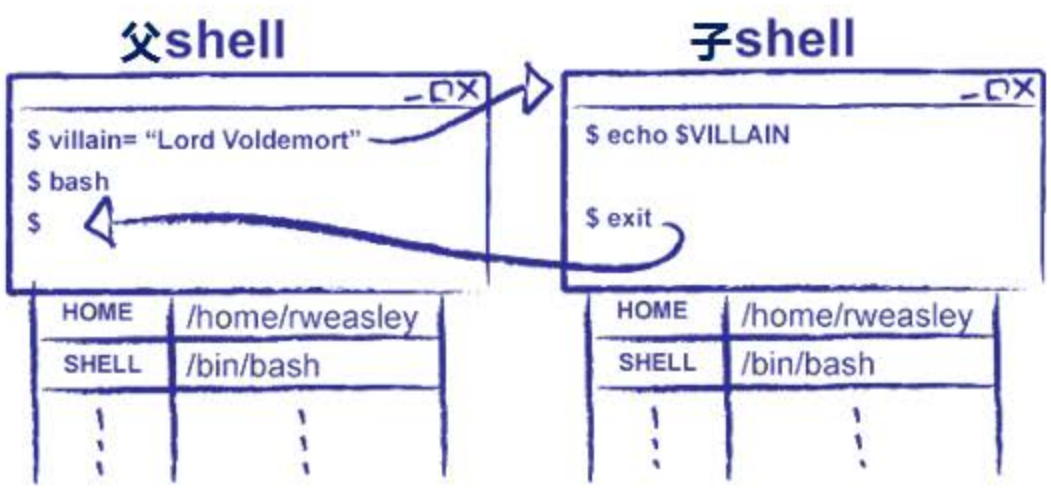
变量 Variables

- ~~let~~
- ~~eval~~
- ~~set~~
- **unset**
  - 删除一个**shell**变量
  - 但对位置参数无效
- **export**
- **declare, typeset**
- **readonly**
- **getopts**

变量 Variables

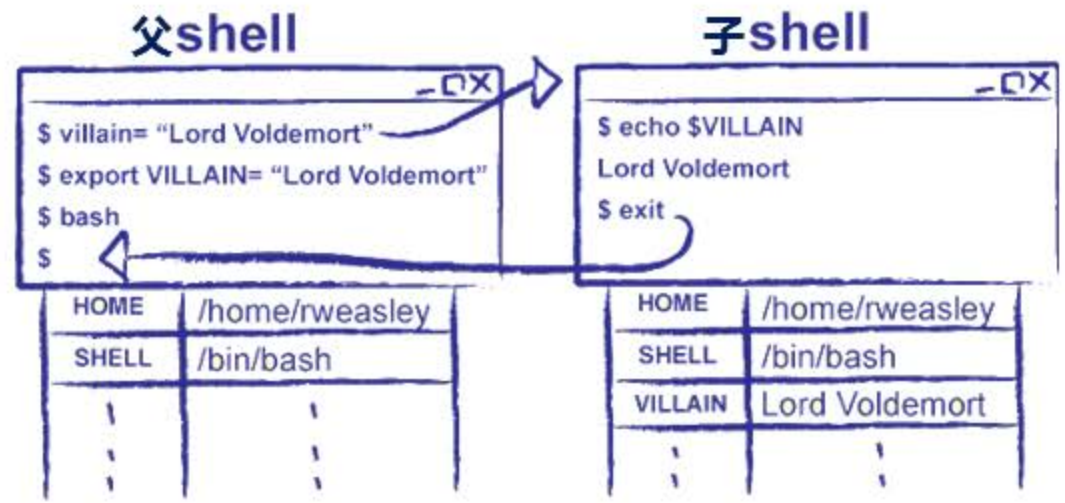
- ~~let~~
- ~~eval~~
- ~~set~~
- ~~unset~~
- **export**
  - 使得被**export**的变量在所运行脚本(或shell)的所有子进程中都可用
- **declare, typeset**
- **readonly**
- **getopts**

export示例




```
1  $ VILLAIN="Lord Voldemort"
2  $ bash
3  $ echo $VILLAIN

4  $ exit
```




```
1  $ export VILLAIN="Lord Voldemort"
2  $ bash
3  $ echo $VILLAIN
   Lord Voldemort
4  $ exit
```

变量 Variables

- ~~let~~
- ~~eval~~
- ~~set~~
- ~~unset~~
- ~~export~~
- **declare, typeset**
  - 指定或限制变量的属性
- **readonly**
- **getopts**

变量 Variables

- ~~let~~
- ~~eval~~
- ~~set~~
- ~~unset~~
- ~~export~~
- ~~declare, typeset~~
- **readonly**
  - 与**declare -r**作用相同，设置变量的只读属性
- **getopts**

变量 Variables

- ~~let~~
- ~~eval~~
- ~~set~~
- ~~unset~~
- ~~export~~
- ~~declare, typeset~~
- ~~readonly~~
- **getopts**
  - 处理命令行参数，并校验有效选项

```
$ myscript.sh -e -d bar -c tom -b man -a foo file1.txt file2.txt
```

getopts语法示例

```
$ script1.sh -c chen -a tom -b
```

- 示例1：规定 a、b 和 c 为有效选项，并且选项 a 和 c 带有参数

```
getopts a:bc: OPT
```

- 示例2：在示例1的基础上，如遇到未定义的选项时，将OPT的值设置为？

```
getopts :a:bc: OPT
```

变量 Variables

- ~~let~~
- ~~eval~~
- ~~set~~
- ~~unset~~
- ~~export~~
- ~~declare, typeset~~
- ~~readonly~~
- ~~getopts~~

变量 Variables

- ~~let~~
- ~~eval~~
- ~~set~~
- ~~unset~~
- ~~export~~
- ~~declare, typeset~~
- ~~readonly~~
- ~~getopts~~

脚本行为　　Script Behavior

- source, . (点命令)
- exit
- exec
- shopt
- caller

脚本行为　　Script Behavior

- **source，. （点命令）**
  - 在脚本中引入其它文件中代码
  - 类似C语言的**include**
- **exit**
- **exec**
- **shopt**
- **caller**

```
$ vim SourceFile1.txt
1 D1=111
2 D2=222
3 D3=333
```

```
$ vim TestScript.sh
1 #!/bin/bash
2
3 # 加载一个数据文件
4 . SourceFile1.txt
5 echo D1
6
7 exit 0
```

脚本行为　　Script Behavior

- source, . (点命令)
- exit
- exec
- shopt
- caller

脚本行为　　Script Behavior

- ~~source, . (点命令)~~
- **exit**
  - 调用并执行其它指定的命令
  - 指定的命令将取代当前进程

- **exec**
- **shopt**
- **caller**

脚本行为　　Script Behavior

- ~~source, . (点命令)~~
- ~~exit~~
- **exec**
  - 用于调用并执行的其它命令
  - 新命令的执行将会替换掉当前进程（**Shell**或脚本）
- **shopt**
- **caller**

脚本行为　　Script Behavior

- ~~source, . (点命令)~~
- ~~exit~~
- ~~exec~~
- **shopt**
  - **shopt = Shell Option**
  - 显示和设置shell中的行为选项
  - 语法： **shopt** *选项 参数*
- **caller**

脚本行为　　Script Behavior

- ~~source, . (点命令)~~
- ~~exit~~
- ~~exec~~
- ~~shopt~~
- **caller**
  - 返回当前活动的子程序调用的上下文
  - 主要用于为排错工作提供更多的信息

脚本行为　　Script Behavior

- ~~source, . (点命令)~~
- **exit**
  - 调用并执行其它指定的命令
  - 指定的命令将取代当前进程
- **exec**
- **shopt**
- **caller**

其它的内置命令

- true
- false
- type
- hash
- bind
- help

其它的内置命令

- true
- false
- type
- hash
- bind
- help

```
# 死循环
while true  # 也可以将true替换为冒号:
do
   command-1
   comnnad-2
   ......
   commnad-n
   # 需要有一种手段从循环中跳出来，或挂起
done
```

其它的内置命令

- ~~true~~
- false
- type
- hash
- bind
- help

```
# 测试false
if false ; then
  echo "false evaluates \"true\""
else
  echo "false evaluates \"flase\""
fi


# 空循环
while false      # 不会进入循环体
do
  command-1
  comnnad-2
  ......
  commnad-n
  # 什么事情都没有发生 :)
done
```

其它的内置命令

- ~~true~~
- ~~false~~
- type
- hash
- bind
- help

```
[tom@tomlab1 ~]$ type '['
[ is a shell builtin

[tom@tomlab1 ~]$ which '['
/bin/[

[tom@tomlab1 ~]$ type -a '['
[ is a shell builtin
[ is /bin/[
[ is /usr/bin/[
```

其它的内置命令

- ~~true~~
- ~~false~~
- ~~type~~
- **hash**
  - **hash** 查看**hash**表
  - **hash -d *command*** 删除表中的***command***命令
  - **hash -r** 清空**hash**表
- **bind**
- **help**

其它的内置命令

- ~~true~~
- ~~false~~
- ~~type~~
- ~~hash~~
- **bind**
  - 显示和设置命令行的键盘序列绑定功能
  - 示例： `bind -x '"\C-l":ls -l'` # *直接按* CTRL+L *就列出目录*
- **help**

其它的内置命令

- ~~true~~
- ~~false~~
- ~~type~~
- ~~hash~~
- ~~bind~~
- **help**
  - 输出shell内置命令的简化版的帮助信息
  - 示例： **help exit**

```
exit: exit [n]
  Exit the shell.

  Exits the shell with a status of N.  If N is omitted, the exit status
  is that of the last command executed.
```