#!/bin/bash

Bash 简介

2013.4.5

Roy Zhang
pudh4418@gmail.com

历史

在遥远的 Unix 时代，那是还没有图形界面这种东西……

Shell 是人们与电脑交互的唯一方式……


TWX
#109

在图像界面下，打开终端模拟器，绝大多数情况下启动的将是 Bash 。

基础的 bash 几乎开箱即用，非常简单。

按上下键查看历史，按 TAB 键自动补全。

这，够用了么？

从简单的开始　重定向

通常命令行程序运行时，默认打开了三个文件：

标准输入（ stdin ）
标准输出（ stdout ）
标准错误（ stderr ）

使用 < 与 > 重定向输入输出流。

例如:

```
ls > file_list
ls >> file_list
sort < file_list
ls non-exist > file_list 2>&1
ls non-exist &> file_list
ls non-exist > file_list 2>err
```

管道 | : 拼接多个程序的输入输出

Unix 哲学的具体体现之一

```
find ~ -name '*.gz'|grep abs

uniq error.log|less

gpg --list-sigs|sig2dot > t.gv
```

创建文件:

```
touch newfile
> newfile

echo “Something” > file
```

多行文件怎么办？

Here Document :

```
$ cat > file <<EOF
> Some lines
> In the file.
> HERE!!!
> EOF
```

简单的作业管理:

```
./prog &          # 在后台执行

jobs              # 显示当前作业

Ctrl+Z            # 暂停当前前台作业

bg %n              # 后台执行
fg %n              # 前台执行
```

括号展开:　　非常实用

```
echo a{b,c}d

echo a{b,c}{d,e}d

echo a{{b,c}{d,e},f}d

cp ~/file{,.bak}
```

基础的命令就这么多，

但这足够了么？

有没有提高效率的办法？

当然有！！！

例如：
在 Archlinux，要启动 GoAgent，需要在终端下输入以下命令：

```
sudo systemctl start goagent.service
```

每次有需求的时候都要敲上一长串字符，很烦哪……

使用 Ctrl+R 向前增量搜索历史命令！

非常好用！

连续按 Ctrl+R 继续向前搜索

诸如此类的快捷键还有很多，
且听我细细道来

光标移动

Ctrl+B	向后 (Backward) 移动一个字符
Ctrl+F	向前 (Forward) 移动一个字符
Meta+B	向后移动一个单词
Meta+F	向前移动一个单词

等等

Ctrl+A	光标移至行首
Ctrl+K	删除光标后的所有内容
Meta+Backspace	删除一个单词
Meta+.	插入上一条命令的最后一个单词

这都是 readline 库的功能，
其他使用 readline 库的程序例子有：

gdb 、 python 、 mysql 等

Emacs 会觉得上述快捷键十分亲切。
因为 readline 和 Emacs 都是 Richard Stallman 教主做出来的，默认键绑定自然是一样的。

Bash 不仅仅是我们与计算机交互的接口，它更是一门编程语言。

bash 脚本为我们完成了许多日常繁琐的小事，使我们能够把有限的精力放在真正重要的事情上。

```
#!/bin/bash

exit 0
```

定义变量

同多数脚本语言一样， bash 的变量是弱类型的。

```
VAR=”Some text”
echo $VAR
echo ${VAR}
NUM=23333
echo $NUM
```

注意：
等号两侧不能有空格！
变量名称全部用大写是编程惯例

（可以通过 declare 内部命令进行细致的定义）

数组

```
a=(1 2 3 4 5 a b c d)
echo ${a[1]}
echo ${a[@]}

b[5]=”T1”
b[“sth”]=”T2”
echo ${b[“sth”]}
```

条件判断

```
[[ expression ]]   (bash 语法)
[ expression ]     (sh 语法)
test expression    (sh 语法)

常用:
[[ -e file ]]      (file 存在)
[[ -d file ]]      (file 是文件夹)
```

[[str1 == str2]]（字符串相等）
[[str1 > str2]] （字典序比较）

[[num1 OP num2]]（数字比较）

OP：

-eq	相等	-le	小于等于
-ne	不等	-gt	大于
-lt	小于	-ge	大于等于

注意括号两边有空格！

算术运算

((*expression*))
$((*expression*))（获取返回值）

```
n=1
((n++))
m=$((n+1))
echo $n
echo $m
```

子 shell

```
(cmd)       ( 在子 shell 中执行 )
$(cmd)      ( 获取子 shell 输出 )
`cmd`       (sh 语法 )
```

控制流

```
if cmda; then
    cmd1
elif cmdb; then
    cmd2;
else cmdc; then
    Cmd3;
fi
```

```
while cmd1; do
    cmd2;
done

until cmd1; do
    cmd2;
done
```

```
for name in words; do
    cmd;
done

for ((exp1; exp2; exp3)); do
    cmd;
done
```

```
case word in
    pat1) cmd1;;
    pat2) cmd2;;
    pat3) cmd3;;
esac
```

```
a=”1 2 3 4 5 a b c”

for i in $a; do
    echo $i
done

a=(1 2 3 4 5 a b c)

for ((i=0;i<10;++i)); do
    echo $i
    echo ${a[i]}
done
```

魔术变量

$#	argc
$0	argv[0]
$1	argv[1]
${11}	argv[11]
$?	上一语句的返回值
$$	当前 bash 进程的 PID
$-	当前的 bash 参数
$*	作为一个整体的参数
$@	分词后的参数

函数

```
function fname() {
    cmds;
}
```

其中 function 与小括号可以选择省略一个

函数中位置变量为函数的参数

函数中可以使用 local 定义局部变量

```
local n=4
```

可以使用 return 语句返回

使用 shift 可以平移位置变量

今天的内容就这么多了

Thanks all.

Any Questions?