Classic Shell Scripting Notes

Lu, Phil

1. 背景知识

- POSIX

  POSIX，Portable Operating System Interface。
  是 UNIX 系统的一个设计标准，很多类 UNIX 系统也在支持兼容这个标准，如 Linux。
  遵循这个标准的好处是软件可以跨平台。所以 windows 也支持就很容易理解了，那么多优秀的开源软件，支持了这个这些软件就可能有 windows 版本，就可以完善丰富 windows 下的软件。

2. 入门

2.1. #!

```
cat > nursers
#! /bin/bash -

who | wc -l
```

之后每条命令都会用 bash 运行所以，如果是 py 文件可以在第一行加上 #! /bin/python, 这样就可以使用./来运行它

2.2. printf

- printf 命令用法与 C 语言非常接近

2.3. 重定向与管道

2.3.1. < 改变标准输入

tr 命令
translate charactors

```
tr -d '0-9'
```

可以将标准输入中的数字全都删除

```
tr -d '0-9' < mytext.txt
```

这就改变了标准输入而是将 txt 文件中的内容进行修改

2.3.2. > 改变标准输出

2.3.3. >> 输出到文件时不覆盖文件而是附加

2.3.4. | 建立管道

```
program1 | program2
```

将 program1 的标准输出修改为 program2 的标准输入

```
tr -d '\r' < mytext.txt | sort > mytext2.txt
```

过程:

1. tr 的标准输入改成 mytext.txt
2. tr 的标准输出变为 sort 的标准输入
3. sort 的标准输出重定向到 mytext2.txt

2.3.5. /dev/null 和/dev/tty

/dev/null 是一个垃圾桶，所有输出到这里的数据都会被扔掉

```
echo 12321 > /dev/null
```

/dev/tty 将重定向一个终端，键盘输入 maybe，所以这个是用来作为输入的

```
read password < /dev/tty
```

将会强制从/dev/tty 中读取数据，一般情况下不写问题也不大貌似

stty -echo
可以将输入不显示在屏幕上
stty echo
重新显示

2.4. Shell 脚本的参数

$1 代表第一个参数 $2 代表第二个参数

```
cat > finduser
#! /bin/sh

who | grep $1
^D
```

使用的时候就可以./finduser lxc

2.5. 简单的执行跟踪

set -x 可以设置是否跟踪命令，跟踪命令是指每条命令执行时候在前面价一个 + 并显示出来

2.6. .profile .bashrc 区别

.profile 是每次登陆时运行
.bashrc 是每次运行 bash 时运行