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

3. 查找与替换

3.1. 正则表达式与 BRE，ERE

BRE 是基本正则表达式 (Basic Regular Expression)
ERE 是扩展正则表达式子 (Extended Regular Expression)
BRE 是 grep 的默认，而程序中一般使用 ERE，如 python 神马的，+ 就属于 ERE 的 meta 符号

grep -E 表示 ERE
grep 则表示 BRE

3.2. 一些小笔记

- . 用来匹配任意一个字符
- + 是之前的字符一个或多个 **ERE**
- * 是之前的字符 0 个或多个
- ？是之前的字符有或没有 **ERE**
- [abc] abc 中的一个
- [a-f0-9] 16 进制字符
- (abc) 连续的 abc, 代表一个整体，后面可以以加入 *+ 之类的 **ERE**
- {a,b} 表示前面的字符出现 a,b 次，或者可以不写 b 精确 a 次，有逗号表示没有上限
- 一些字符集 [:alnum:], [:alpha:]

3.3. sed 命令

sed 用来处理文本里的每一行，可以查找，替换，删除，显示等等等等，其功能完全包含 grep，tr 等命令

- 它的最为常用的功能就是 s:

```
sed 's/要匹配的字符/要替换成的字符/g' file
```

- 其中/代表分界符，可以用其他符号代替；g 代表搜索全文件，不写则找到第一个就结束了
- sed 也可以接受一个脚本作为输入的一部分，把其中要执行的所有 sed 操作全部写入这个脚本里面
  举个例子:

```
find /home/tolstoy -type d -print |    -type d 表示类型为
directory
sed 's;/home/tolstoy/;/home/lt/;' |
sed 's/^/mkdir /' |                    插入 mkdir 指令
sh -x                                  -x 表示指令追踪
```

3.4. cut 切割字段

cut 命令从文件的每一行剪切字节、字符和字段并将这些字节、字符和字段写至标准输出。
如果不指定 File 参数，cut 命令将读取标准输入。必须指定 -b、-c 或 -f 标志之一。

主要参数
-b ：以字节为单位进行分割。这些字节位置将忽略多字节字符边界，除非也指定了 -n 标志。
-c ：以字符为单位进行分割。
-d ：自定义分隔符，默认为制表符。
-f ：与-d 一起使用，指定显示哪个区域。
-n ：取消分割多字节字符。仅和 -b 标志一起使用。如果字符的最后一个字节落在由 -b 标志的 List 参数指示的范围之内，该字符将被写出；否则，该字符将被排除。

举 2 个例子：

```
cut -d : -f 1,5 /etc/passwd
```
该命令以：分割，取每行的第 1,5 列

```
ls -l | cut -c 1-10
```
该命令取第 1~10 个字符，正好是文件的读写执行权限

3.5. join 命令

将两个文件中，指定栏位内容相同的行连接起来。

3.6. awk

awk 是一种“单命令行程序”
它的使用模式为:

```
awk 'program' [ file ]
```

和 sed 命令很像
其中 program 的基本格式为 pattern { action }

举个例子：

`awk -F : '{print $1, $5 }' /etc/passwd` 以：作为分隔符，输出第一字段第五字段