Linux Shell笔记要点

1. shell简介 （了解）
2. bash是当前主流shell，即命令解释器。（了解）
3. 存取权限与安全（重要）
4. 掩码相关的思想（理解并可应用）
5. 脚本调用的机制（理解）



1. 脚本的基本格式：
2. 脚本文件头，格式#! 注意：这不是注释！
3. 命令替换，即别名alias（掌握）
4. 管道：即| （掌握）
5. 后台命令： nohup作用，&作用 （掌握）
6. 模式匹配（重要，见后面正则表达式）
7. 输出重定向 注意：输入输出到文件（掌握）
8. 特殊符号：
   1. 双引号：
   2. 单引号
   3. 反引号 （掌握）
   4. &（注意此符号的执行机制） 配合nohup命令的效果。
9. 变量
   1. 环境变量 注意常用的环境变量名称全部为大写，PATH、HOME（注意随登录用户改变）
   2. 给PATH变量追加内容： 格式 export PATH=$PATH:(增加的内容)

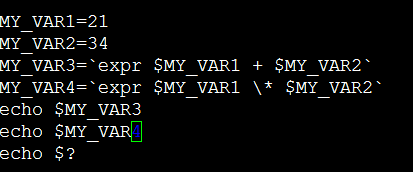


* 1. Env命令 （非常用用）
  2. 本地变量与环境变量的生存周期：本地变量随着当前终端的关闭而消失。环境变量：如果不去改变环境变量，那么环境变量一直是默认值。如果改变了环境变量，当前终端结束后，再重新开启一个终端，那么会恢复成为原来的样子。（重要）
  3. 延伸：环境变量在shell嵌套调用时的变化
  4. 使用变量的方法：两种，最好用 ${VAR} 的形式
  5. Set命令用于查看当前环境下的变量名和值
  6. 取消变量unset （注意用法，重要）
  7. 位置变量 （非常重要） 传参规则
  8. 标准变量：重要的变量：HOME，PATH，TZ，PWD（含义和用法）
  9. **$#变量** **$0不计入参数个数的数量之内** （重要） ；**$\*变量**（），**前两者管辖范围是$1~$9** (重要)； $?（重要）

1. Shell变量的最基本规则：变量只有字符串和整数两种类型。在shell运算中都是整数运算或字符串操作运算。
2. 三种引号之间的区别（重要）
3. 表达式运算，格式： $[表达式] （重要） **取值要加$， 赋值不要加$**
4. 运算符，对初学者来说运算时不要忘记参与运算的变量前要加上$符号，即取值时要$符号参与。
5. 表达式替换形式两种：$[表达式], $((表达式 )) ，两种形式都要知道。
6. 输出命令： echo 选项-e （常用）
7. 输入命令：read 参数的使用规则 （常用）
8. 输出命令：tee 用在管道相关的情况，调试时非常有用
9. 标准输入输出，内容同C语言文件处理相关部分。
10. 输入输出重定向，基本操作： >, >>, <, 2>&1,等形式 （常用）

流程控制

1. test命令 test [条件]。Test的两种使用形式，使用中括号形式表示时，注意空格规则
2. test 逻辑运算的是逻辑值，而不是数值，
3. test 字符串判断操作 注意=，!=都是针对字符串操作，而不是数值 –z， -n
4. **学习小技巧，如何方便的去记忆判断选项，对照英文单词的首字母或字母组合。**
5. **Test 判断的结果如何正确的判断？不要直接用直觉，注意test后面的表达式是一个整体，注意这个整体的结果是成功还是失败，成功返回0，失败返回1**
6. 只有一个变量的情况下做测试，格式为： -选项 变量
7. Expr 算术运算命令要点：如何得到运算的值，乘法运算需要用到转义，注意操作符左右要有空格。



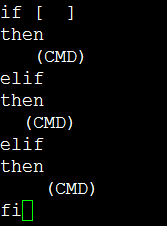
1. If then fi 条件判断语句：**注意表达式的值如何判断！**
   1. 格式1

（推荐使用）

* 1. 格式2



* 1. 多重判断格式：



1. case 条件分支语句：要点
   1. 条件处理的是字符串
   2. 每一个分支以“;;”作为结束
   3. 条件分支一esac结束
   4. 通配符\*,?,[]

练习：通过read从键盘读入一个字符串，如果输入的是数字，如果范围在1～3，进入相应的处理分支，如果输入E或e ,屏幕输出一行：“EXIT！”，如果是其他字符，那么输出：“非法输入+输入的字符串”

10、for循环 四种使用情况

1）For VAR in 列表

2）For VAR in `命令执行之后的一个结果集`

3）For VAR in `cat 文本文件的内容` (分隔的依据，空格和换行)

4） 通过循环得到命令行参数的方法

VAR\_I 循环变量

For VAR （后面不要有in 和后续内容）

Do

VAR 输入的参数内容

VAR\_I=$[$VAR\_I+1]

Done

For( ; ; )

11、shell循环分为当型循环和直到型循环

While

Do

CMD

Done

对应C语言

While()

{

}

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

Until

Do

CMD

Done

对应C语言中

Do{

}while()

12、循环控制关键字：continue和break

注意：break可以控制我们跳出循环的层数

Break 3 （跳出3层循环）

默认值是1

13、通过while循环得到文件内容

while read LINE

do

echo $LINE

done < shell01.sh

注意：使用重定向输入的位置，使用read得到文件内容

While与for处理文件内容的分别（重要）

练习：

1、从当前目录得到所有普通文件的列表，逐个输出文件的内容

2、从键盘得到得到输入的字符串，重定向输出到一个文本文件中，文本文件的文件名从命令行输入，当输入exit时退出shell程序，注意exit字符串不要输出到文本文件中。

正则表达式与文本处理

1. 做什么用的（了解）
2. 正则表达式符号 “.” 对应于”?”
3. “^” “$”



1. []表示匹配内容的范围 [1234567890] [0-9][a-z0-9]

[0-9][0-9][0-9]. [0-9][0-9][0-9]. [0-9][0-9][0-9]. [0-9][0-9][0-9]

[0-9]\{3\}.[0-9]\{3\}.[0-9]\{3\}.[0-9]\{3\}

[a-z]

1. Grep常用参数 (重要)

练习：统计当前目录下所有脚本文件中的有效代码行数（即不包括注释，也不包含空行）

1. find命令用于查找，注意，如果处理大量文件的话应该将命令切换到后台处理

find基本命令格式： find 路径 选项

注意一个细节，find命令查找的输出中，路径是怎样显示的？

注意-print选项存在的意义，强制输出到标准输出

atime、ctime与mtime之间的规则

|  |  |  |
| --- | --- | --- |
| Atime（amin） | 访问时间 | 对文件做读取操作、或touch |
| Ctime | 文件状态 | 改变的与文件内容无关，只与文件描述相关，如权限、用户名等 |
| Mtime | 内容状态 | 对文件进行写操作时才会改变这个时间 |

1. xargs的处理机制



处理大量的参数传递问题。

1. 常用的find选项：

-name

-type

与时间相关的选项：atime,ctime,mtimes

-perm

-type

-depth

Find相关的练习：

1. 环境配置步骤：创建10个目录，名称从a到i，每个目录中创建10个后缀为.sh的文件，10个后缀为.c的文件，文件名的规则：目录名称+数字+后缀
2. 查找
   1. 查找这10个目录中编号为9的文件
   2. 查找10个目录中编号为7，且后缀为.c的文件
   3. 把编号为9的文件复制到一个新目录下，名称为new
   4. 把所有编号为7且后缀为.C的文件复制到名为cfile的路径下
3. 查找/etc/init.d目录下 所有脚本文件，并筛选出10天前被修改的文件
4. awk

awk是对列作操作的，每一列叫“域”列之间做区分的符号叫分隔符

awk列控制变量： $0~$n。注意：第一列是$1，而$0代表所有的列

awk命令脚本中引号的用法（注意） 引号的匹配是就近匹配，注意使用转义

awk命令支持对数字的运算

注意多个脚本表达式之间用分号进行”;”分割

1. sed命令

扩展知识：为什么awk和sed允许从文件中读取控制脚本，体现了对代码复用的思想

在使用 awk、sed等命令时要注意开源软件众多造成的**版本差异**问题

脚本对程序员的意义：

1. 解决重复性工作
2. 控制linux服务
3. 对重要文本信息进行处理（日志等）

文本处理相关：

如果存在多个筛选条件使用管道先进行第一次筛选，再第二次筛选

wc命令配合选项 –l可以统计行数

排序命令 sort

对于比较复杂的操作

使用分治法，即把大问题拆分成小问题，分步骤解决。

文本处理的技巧总结

find命令，按照文件名，在某一目录下，查找符合条件的目录或者文件

find -name "\*.sh" ./

grep命令，对文本模式匹配，按照行方式

-v取反 -n显示匹配行及行号 -c只输出匹配计数

awk命令：对文本按照列进行处理或打印

sort命令，对文本按照行进行排序处理，默认从小到大，

-n代表按照数字处理 -k7代表第7列 -r逆序

sed命令 按照行或者处理文本

sed -n '1p' -n按照行定为打印

sed -n '1,3p' 范围

sed -n '$p' 代表最后一行

sed文本替换

sed 's/old/new/g'

sed p指令参数规则，参数在指令前面