

1-linux 提供的shell的解析器(minimize install)

查看# `cat /etc/shells`

`/bin/sh /bin/bash /usr/bin/sh`

`/usr/bin/bash`

2- 创建helloworld.sh文件

`vi helloworld.sh` 在helloworld.sh 文件里insert 内容

*** 注意：在文件里最好加入#!/bin/bash指定bash的path,以防出错***

`#!/bin/bash`

`echo "hello world!" echo "taylor swift!"`

`insert.`

3-excute script的四种方法

1. 方法1: `bash helloworld.sh` 方法2: `sh helloworld.sh` 下面的两个方法需要改文件有执行权限 需要修改权限 `chmod u+x helloworld.sh` 方法3: `./helloworld.sh` 方法4: `.helloworld.sh`

2. 事实上查询bash会发现, sh是bash的软连接

```
ll |grep bash -rwxr-xr-x. 1 root root 964600 8月 8 2019 bash lrwxrwxrwx. 1 root root 10 5月 6 17:43 bashbug -> bashbug-64 -rwxr-xr-x. 1 root root 6964 8月 8 2019 bashbug-64 lrwxrwxrwx. 1 root root 4 5月 6 17:43 sh -> bash
```

4-参数parameter

1. 可以脚本里传入参数 用\$1,\$2...\${10},\${11} 当参数, 注意当参数parameter -ge 10 需要加入{number} \$# 获取所有输入参数个数, 常用于循环 \$* 这个变量代表命令行中所有的参数, \$*把所有的参数看成一个整体 \$@ 这个变量也代表命令行中所有的参数, 不过\$@把每个参数区分对待

2. parameter.sh内容 `#!/bin/bash echo 'demo of parameters' echo $1 echo $2 echo $3 echo $4 echo $# echo $* echo $@`

3. 执行: `bash parameter.sh 1 2 3 a` output: demo of parameters 1 2 3 a 1 2 3 a 1 2 3 a

5-echo \$?

最后一次执行的命令的返回状态。如果这个变量的值为0, 证明上一个命令正确执行; 如果这个变量的值为非0 (具体是哪个数, 由命令自己来决定), 则证明上一个命令执行不正确了。

6-运算符(+,-,*,/)

计算(2+3)*2 格式 变量=\${表达式} [taylor@Hadoop_1 Codes]\$ a=\$((2+3)*2) [taylor@Hadoop_1 Codes]\$ echo \$a 10

7-条件判断

1. 数字判断 -lt:less than (<) -le:less equal(<=) -gt:greater than (>) -le:greater equal(>=) -eq :equal(=) -ne:not equal(!=)
2. 文件权限 -r 有读的权限(read) -w 有写的权限 (write) -x 有执行的权限 (execute)
3. 按照文件类型进行判断 -f 文件存在并且是一个常规的文件 (file) -e 文件存在 (existence) -d 文件存在并且是一个目录 (directory) -e 判断对象是否存在 -d 判断对象是否存在, 并且为目录 -f 判断对象是否存在, 并且为常规文件 -L 判断对象是否存在, 并且为符号链接 -h 判断对象是否存在, 并且为软链接 -s 判断对象是否存在, 并且长度不为0 -r 判断对象是否存在, 并且可读 -w 判断对象是否存在, 并且可写 -x 判断对象是否存在, 并且可执行 -O 判断对象是否存在, 并且属于当前用户 -G 判断对象是否存在, 并且属于当前用户组
4.

```
[taylor@Hadoop_1 Codes]$ [ -e oo.xx ] --文件不存在 [taylor@Hadoop_1 Codes]$ echo $? 1
[taylor@Hadoop_1 Codes]$ [ -e pp.sh ] --文件存在 [taylor@Hadoop_1 Codes]$ echo $? 0
[taylor@Hadoop_1 ~]$ [ -d Files ] --文件夹存在 [taylor@Hadoop_1 ~]$ echo $? 0 [taylor@Hadoop_1 ~]$ [ -f Files ] --文件夹存在但不是常规文件 [taylor@Hadoop_1 ~]$ echo $? 1
```

8-if -else

1. 编写if-else 传入参数 `if [] then echo " " elif [] then echo " " else echo " " fi`
2. 注意if与[]有空格, []括号里两边也要有空格

```
#!/bin/bash if [ $1 -eq 1 ] then echo "1-1" elif [ $1 -eq 2 ]
then echo "2-2" else echo "3-3" fi
```

9- case

1. case行尾必须为单词“in”, 每一个模式匹配必须以右括号”) ”结束。双分号“;;”表示命令序列结束, 相当于java中的break。最后的“*) ”表示默认模式, 相当于java中的default。
2.

```
#!/bin/bash case $1 in "1") echo "1-1" ;; "2") echo "2-2" ;; *) echo "default" esac
```

10-for

1. 基本语法

```
for (( 初始值;循环控制条件;变量变化 ))
do
程序
done
```
2.

```
#!/bin/bash
sum=0 for((i=1;i<=100;i++)) do sum=$((i+$sum)) done
echo "sum=$sum" echo "sum=$sum"
```

11-while

1. while的语法 `while [条件判断式] do 程序 done`
2.

```
#!/bin/bash sum=0 i=1 while [ $i -le 100 ] do sum=$((sum+$i)) i=$((i+1)) done echo
"sum=$sum"
```

12-read读取控制台输入

从控制台读取 -t 代表读取等待的时间，-p 代表输入的提示信息

```
#!/bin/bash read -t 10 -p "enter a number " number echo "your number is "$number
```

13- basename and dirname

1. basename命令会删掉所有的前缀包括最后一个（/）字符，然后将字符串显示出来。

```
basename root/taylor/t.java t.java
```

```
basename root/taylor/t.java .java t
```

2. dirname 文件绝对路径:功能描述：从给定的包含绝对路径的文件名中去除文件名（非目录的部分），然后返回剩下的路径（目录的部分）

```
dirname root/taylor/t.java root/taylor
```

```
dirname root/taylor/t.java taylor swift root/taylor ..
```

14 -function自定义函数

函数返回值，只能通过\$?系统变量获得，可以显示加：return返回，如果不加，将以最后一条命令运行结果，作为返回值。return后跟数值n(0-255)

```
#!/bin/bash function add1(){ return $[${1}+${2}] } add1 $1 $2
```

无返回值

```
#!/bin/bash total=0 function add(){ total=$(( ${1}+${2} ) } read -p "enter number" n1 read -p "enter number" n2 add $n1 $n2 echo $total
```

15-shell工具cut

1. cut

存在一个文件cut.txt文件内容

```
taylor swift taylor swift hello world taylorswift
```

-f 列号，提取第几列 -d 分隔符，按照指定分隔符分割列 -c 指定具体的字符

解释：cut -d " " -f 1 cut.txt 取分割后的第一列

解释：cut -d " " -f 1,3 cut.txt 取分割后的第1,3列

单词中间有一个空格分成2列，2个空格 分成3列 其中第二列为空格

cut 出来 world

```
cat cut.txt | grep hello | cut -d " " -f 2
world
#切割ip
ifconfig |grep netmask|cut -d "t" -f 2 |cut -d " " -f 1
```

16-shell工具sed

sed是一种流编辑器，它一次处理一行内容。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”，接着用sed命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有改变，除非你使用重定向存储输出。

选项参数	功能
-e	直接在指令列模式上进行sed的动作编辑。
-i	直接编辑文件

命令	功能描述
a	新增，a的后面可以接字符串，在下一行出现
d	删除
s	查找并替换

```
sed "2a taylor swift" cut.txt      在第二行后插入taylor swift,即在插入在第三行并打印，原文件不变
sed "/hh/d" cut.txt  删除带hh字符的行 原文件不变
sed 's/taylor/kk/g' cut.txt  将taylor全部被kk替换 g=globle全部代替
sed '2d' cut.txt      删除第二行
```

17-shell工具awk

一个强大的文本分析工具，把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行分析处理。

-F 指定输入文件折分隔符

#操作的文件为passwd

```
1-
awk -F : '/^root/{print $1,$2,$3}' passwd 匹配以root开头 并输出1-3的列

root x 0
```

```
2-在输入之前输入12结尾输入13
awk -F : 'BEGIN{print "12"}/^root/{print $7}END{print 13}' passwd
12
/bin/bash
13
```

```
3- awk -F: '{print "Filename:" FILENAME, "row:"NR,"col:" NF }' passwd
Filename:passwd row:1 col:7
Filename:passwd row:2 col:7
```

```
Filename:passwd row:3 col:7
Filename:passwd row:4 col:7
Filename:passwd row:5 col:7
Filename:passwd row:6 col:7
Filename:passwd row:7 col:7
Filename:passwd row:8 col:7
Filename:passwd row:9 col:7
Filename:passwd row:10 col:7
Filename:passwd row:11 col:7
Filename:passwd row:12 col:7
Filename:passwd row:13 col:7
Filename:passwd row:14 col:7
Filename:passwd row:15 col:7
Filename:passwd row:16 col:7
Filename:passwd row:17 col:7
Filename:passwd row:18 col:7
Filename:passwd row:19 col:7
Filename:passwd row:20 col:7
```

解释：以：分割，FILENAME是读取的文件 NR第几行，NF分了多少列

4-查询ip

```
ifconfig|grep "mask" |awk -F"inet" '{print $2}'|awk -F"net" '{print $1}'
```

5-查询所有的空行

```
awk '/^$/ {print NR}' cut.txt
2
```

6-统计成绩

```
cat score.txt
tay 40
kkk 50
gkd 60
```

```
[root@Hadoop_1 Codes]# awk -F " " '{sum=sum+$2; print $2}END{print "sum="sum}' score.txt
40
50
60
sum=150
```

7-打印当前目录home下的相对路径路径

```
grep -r "hello" /home |awk -F: '{print $1}'
/home/karlieswift/Codes/helloworld.sh
/home/karlieswift/Codes/cut.txt
```

18-shell工具sort

sort命令是在Linux里非常有用，它将文件进行排序，并将排序结果标准输出。

sort(选项)(参数)

选项	说明
-n	依照数值的大小排序
-r	以相反的顺序来排序
-t	设置排序时所用的分隔字符
-k	指定需要排序的列

参数：指定待排序的文件列表

```
sort.txt文件内容
aa:40:5.4
bb:20:4.2
dd:10:3.5
cc:50:2.3
sort -t : -nrk 2 sort.txt #反序 按第二列
cc:50:2.3
aa:40:5.4
bb:20:4.2
dd:10:3.5
sort -t : -nk 2 sort.txt #升序 按第二列
dd:10:3.5
bb:20:4.2
aa:40:5.4
cc:50:2.3

cat sort1.txt
1
23
2
sort -nr sort1.txt
23
2
1
sort -n sort1.txt
1
2
23
```

19-shell工具 wc

wc命令用来计算数字。利用wc指令我们可以计算文件的Byte数、字数或是列数。

wc [选项参数] filename

选项参数	功能
-l	统计文件行数
-w	统计文件的单词数
-m	统计文件的字符数
-c	统计文件的字节数

```
wc -l tt.txt 统计行数 包括空行
wc -w tt.txt 统计文件的单词数 包括重复单词
wc -m tt.txt 统计文件的字符数
wc -c tt.txt 统计文件的字节数
```

20-正则表达式

1. `cat passwd | grep ^root` 查找以root开头的一行
2. `cat passwd | grep root$` 查找以root结尾的一行
3. `cat cut.txt | grep ^$` 返回空行 ^\$代表空行
4. `cat passwd | grep r..t` 查找有r..t的一行,例如root, raat,
5. *与一个字符连用, 表示匹配上一个字符0次或多次

`cat passwd | grep ro*t` 匹配rt, rot, root等

6. [] 表示匹配某个范围内的一个字符, 例如

`[6,8]`-----匹配6或者8

`[a-z]`-----匹配一个a-z之间的字符

`[a-z]*`-----匹配任意字母字符串

`[a-c, e-f]`-匹配a-c或者e-f之间的任意字符

`cat passwd | grep r[a,b,c]*t` 匹配所有abc组合的字符穿

7. 转义字符 \

\表示转义, 由于所有特殊字符都有其特定匹配模式, 当我们想匹配某一特殊字符本身时 (例如, 我想找出所有包含 '\$' 的行) 此时就要将转义字符和特殊字符连用, 来表示特殊字符本身

``$ cat /etc/passwd | grep t\&ay`

就会匹配所有包含 t\$ay 的行。