1. 别名功能 alias,unalias, eg:alias tt=”echo 123” unalias tt
2. 前后台作业控制，screen

Screen -S name(控制台名) 创建控制台并切换到该控制台

Screen -list 查看所有控制后台，包括控制台PID

Screen -r PID 切换操作控制台

快捷键：ctrl+a---->悬挂控制台在后台 ctrl+d----->退出当前控制台

1. 数据流输入输出重定向：>,>>,1>,2>>,&>,<,cat,
2. 管道，| 将前者命令的标准输出交给后者命令的输入，tee,

Tee filename---->创建文件并通过键盘输入 eg:cmd1 | tee test.txt-->将cmd1的执行结果导入到test.txt

1. 命令排序 cmd1；cmd2，cmd1&&cmd2，cmd1||cmd2

cmd1；cmd2：命令执行没有逻辑关系，无论前面的命令成功与否，后面cmd2依然执行

cmd1&&cmd2：前面cmd1执行成功，后面cmd2才执行

cmd1||cmd2：前面cmd1执行不成功，后面cmd2才执行

1. Shell通配符

\*：匹配任意多个字符

？：匹配任意一个字符

[ ]：匹配括号中任意一个字符

( )：在子shell中执行（cd /root;touch file）（umask 0002 filename）

{ }：集合 eg:touch file{1..9}--->创建九个文件file1,file2....file9

\：转义 eg:echo “\$test”--->输出$test

1. Echo 字符输出的颜色和格式化
2. 自定义变量

定义：变量名=xxx

查看变量：echo+$变量名 或者set,显示所有变量，包括自定义变量，环境变量

取消变量：Unset+变量名，自定义变量作用范围:仅在当前的shell中有效

1. 系统环境变量

定义：Export 变量名=xxx

查看变量：echo+$变量名 或者env |grep name

作用范围：在当前shell和子shell中有效

变量永久有效：在/etc/profile下添加环境变量

1. 位置参数变量

脚本参数传参：$1,$2,....$9,${10} eg:sh xxx.sh $1 $2...

Read -p “ description ” 参数1 参数2 ...------->类似于scanf函数，通过键盘输入给参数赋值

1. 预先定义变量

$0:脚本名--->你执行什么它就显示什么 eg:echo $(basename $0)---->只显示文件名

$\*:所有的参数

$@:所有的参数

$#:参数的个数

$$：当前进程的PID

$!：上一个后台进程的PID

$?：上一个命令的返回值，0表示成功

12.变量赋值方式

Eg:ip1=192.168.16.22

School=”wuhan hunan”

Today1=`date +%F`(不建议使用)

Today2=$(dat +%F)---🡪显示日期

Note:` `里面的内容和$( )实现的功能相同，都是将shell命令执行结果打印出来,等价于执行命令。

Read -p “ description ” 参数1 参数2 ...

Read -t 5--->5秒之后关闭

13.双引号” ”和单引号’ ’的区别：” ”弱引用，一般要解析里面的字符如$，但是解析不了shell命令, ’ ’强引用，一般可以输出特殊字符，如$\@。

Eg:tough `date +%F`\_file.txt和touch $(date +%F)\_file.txt一样是创建相应日期文件

Note:只要是解析shell命令，都用$( )或者` `

1. 数值运算

（1）整数运算

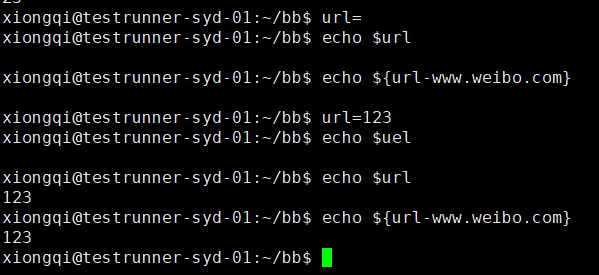
Expr $a + $b + - \\* / % $(($a+$b)) + - \\* / %

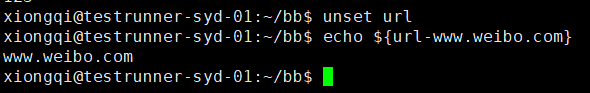
$[$a+$b] let sum=$a\*$b echo $sum

1. 变量替代

（1）${变量名-新的变量值}

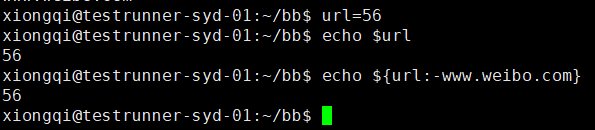
如果变量被赋值了（包括空值），则不赋予它新的变量值；如果变量没有被赋值，则赋予它新的变量值。

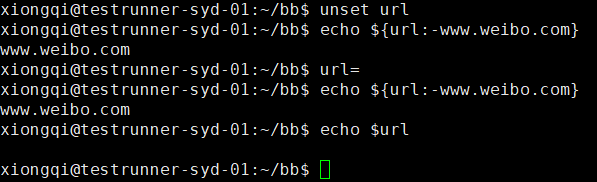




（2）${变量名：-新的变量值}

如果变量没有被赋值（包括空值），则赋予它新的变量值；如果变量被赋值了，则不赋予它新的变量值。





1. Shell条件测试格式

格式1：test 条件表达式

格式2：[ 条件表达式 ]

格式3：[[ 条件表达式 ]]

Test -d /../.(目录) ---->判断目录是否存在 $?

If [ $? -eq 0 ];then

.....

fi

[ -e dir|file ] [ 1 -gt 10 ] 大于

[ -d dir ] 文件夹目录是否存在 [ 1 -lt 10 ] 小于

[ -f file ] 文件是否存在 [ 1 -eq 10 ] 等于

[ -r file ] 当前用户对文件是否有读权限 [ 1 -ne 10 ] 不等于

[ -x file ] [ 1 -ge 10 ] 大于等于

[ -w file ] [ 1 -le 10 ] 小于等于

[ -L file ]

1. curl -s icanhazip.com---->查看外网地址

W ------>查看CPU的负载情况，或者 uptime

Echo -e “\n” -------->换行

Free -m ------->查看内存使用大小

循环流程1：for i in $(seq $变量名)；do

.......

Done

Or:

For 变量名 in [ 取值列表 ]

Do

循环体

Done

------------------------------------------------------------------------------------

For 变量名 in [ 取值列表 ]

Do

{循环体}& //并发在子shell里执行，不影响当前shell

Done

Wait //等待上面所有执行完毕之后，再执行下面

........

For 变量名 in `cat file.txt`

Do

循环体

Done

Note: 注意它是按文件file.txt的空格为分隔读入

循环流程2: while true:do

.....

Done

-----------------------------------------------------------------------------------------

While 条件测试

Do

循环体

Done

多分支流程：if [ 条件1 ];then

....

Elif [ 条件2 ];then

.....

Elif [ 条件3 ];then

......

.

.

Else

.....

Fi

流程控制语句case:

Case 变量 in

模式 1)

命令序列 1；；

模式 2)

命令序列 2；；

模式 3)

命令序列 3；；

\*)

无匹配后命令序列

Esac

1. （1）

While read line

Do

.....

Done<ip.txt

Note:把ip.txt文件逐行读入给变量$line（注意是按行读入）

（2）

Expect {

“yes/no ” { send “yes\r”;exp\_continue }

“password:”{ send “1\r” }

}

Expect eof

Note:当执行脚本出现交互，如“yes/no”,”password:”,用expect发送消息send“”。

（3）

If [ ! -f file ];then

.......

Fi

Note：如果file文件不存在

（4）

Echo $变量 >>file.txt note:把变量值导入到file.txt（追加的方式）

1. 关联数组
2. 定义关联数组

Declare -A array\_1 declare -A array\_2

1. 给关联数组赋值

数组名[索引]=变量值

Array\_1[name]=xiongqi array\_1[age]=23 array\_1[sex]=m

1. 给关联数组一次赋值多个

Array\_2=([name]=xiongqi [age]=23 [sex]=m)

Echo {Array\_2[\*]} or echo {array\_2[@]}------->一次性输出所有value

Echo {!Array\_2[\*]} or echo {!array\_2[@]}------->一次性输出所有索引

1. 遍历数组
2. 通过数组元素的个数进行遍历
3. 通过数组的索引进行遍历

Note:将统计的对象作为索引，仅针对关联数据。

1. 函数

作用：自定义一堆命令

避免大量的重复代码，加强可读性

1. 定义函数

Function hello(){

Echo “第一种定义方式”

.......

}

Hello(){

Echo “第二种定义方式”

........

}

1. 调用函数

Hello

Eg:

#!/usr/bin/sh

FILE=./dis.txt

count(){

local i=0 //local 用于定义函数内部变量，函数外的属于全局变量

local j=100

while read line

do

let i++

done <$FILE

echo "count is $i"

echo $j

}

Count

1. return返回值

Return的值只能是整数，不能字符串。$?一般是0或者1，用$?作为判断时，要转存一下，test=$?,再对test进行判断。

1. 指定位置参数

使用内置命令set给脚本指定位置的参数传值（又叫重置）。一旦使用set设置了传入的参数，脚本将忽略运行时传入的位置参数。

1. 正则

大量的字符串文件需要配置，而且是非交互式的。

过滤相关的字符串，得到字符串，打印字符串

正则表达式和通配符特殊字符是有区别的。

（1）通配符

>\\* 匹配零个或多个任意字符

？ 代表匹配任意单个字符

[ ] 代表指定范围的单个字符

；； 前面命令是否执行完成都会执行后面的命令

&& 前面命令执行成功才会执行后面的命令

|| 前面命令执行不成功才会执行后面的命令

1. 特殊字符

[[ :space: ]] 空格

[[ :digit: ]] [0-9]

[[ :lower: ]] [a-z]

[[ :upper: ]] [A-Z]

[[ :alpha: ]] [a-Z]

1. 基础的正则表达式

^word //匹配以word开头的内容

Word$ //匹配以word结尾的内容

^$ //表示空行

. //代表且只能代表任意一个字符

\ //让原有的字符没有含义

\* //重复前面一个字符0次或者多次

.\* //匹配所有字符

[abc] //匹配中括号中的任意字符

[^abc] //匹配任意不含集合中字符的

^[abc] //匹配任意以集合中字符开头的

A\{n,m\} //重复前一个字符’A’，n-m次的

a\{n,\} //重复前一个字符’a’至少n次的

a\{n\} //重复前一个字符’a’ n次的

? //匹配之前的项1次或者0次 （需egrep模式匹配或者grep -E）

+ //匹配之前的项1次或者多次 （需egrep模式匹配或者grep -E）

( ) //匹配表达式，创建一个用于匹配的字符串（需egrep模式匹配或者grep -E）

| //交替匹配|两边的任意一项，eg：ab(c|d),它匹配abc or abd

Grep “^$” file/path //过滤出空行

Grep -v “^$” file/path //过滤出除空行之外的（取反的意思）

Grep -vn “^$” file/path //过滤出除空行之外的,并打印行号

Grep “.” file/path //匹配所有内容，除了空行

Grep “.\*” file/path //匹配所有内容，包括空行

Grep “sbcdhf.i” file/path //匹配单个字符串，.代表任意单个字符，前面一样

Grep “sbcdhf..i” file/path

Grep “sbcdhf...i” file/path

Grep “.$” file/path //匹配任意一个字符结尾的

Grep “\.$” file/path //匹配以.结尾的（转义.）.代表任意一个字符

Grep “8\*” file/path //匹配’8’字符重复出现0次或者任意多次的

Grep -o “8\*” file/path //匹配’8’字符重复出现0次或者任意多次的结果

Grep “[abc]” file/path //匹配集合中任意字符的

Grep “[0-9]” file/path //匹配集合中任意字符的

Grep “8\{3\}” file/path //匹配’8’字符重复3次的

Grep -E “8{3}” file/path //匹配’8’字符重复3次的

Grep “8\{3,\}” file/path //匹配’8’字符重复至少3次的

Grep -E “8{3,}” file/path //匹配’8’字符重复至少3次的

Grep “8\{3,5\}” file/path //匹配’8’字符重复3-5次的

Grep -E “8{3,5}” file/path //匹配’8’字符重复3-5次的

1. Sed流编辑器-----非交互式的编辑器

它一次就只处理一行内容，sed处理内容时，先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命令处理缓冲区里面的内容，处理完之后再把它送往终端显示。然后处理下一行。

（1）Sed命令格式：https://www.linuxprobe.com/linux-sed-command.html

sed的命令格式：sed [options] 'command' file(s);

sed的脚本格式：sed [options] -f scriptfile file(s);

1. 选项

-e ：直接在命令行模式上进行sed动作编辑，此为默认选项;

-f ：将sed的动作写在一个文件内，用–f filename 执行filename内的sed动作;

-i ：直接修改文件内容;

-n ：只打印模式匹配的行；

-r ：支持扩展表达式;

-h或--help：显示帮助；

-V或--version：显示版本信息。

（3）

Sed -n ‘/halt/p’ passwd //打印匹配‘halt’的行

Sed -n ‘2p’ passwd //打印第二行

Sed -n ‘$p’ passwd //打印最后一行

### 二 ,日志文件

1. Ubantu 系统的日志配置文件位置：/etc/rsyslog.d/50-default.conf
2. 重启syslog服务：service syslog restart

### 三 ,

1. Set命令

<http://www.ruanyifeng.com/blog/2017/11/bash-set.html>