# 自定义函数

解释执行

函数代表着一个或一组命令的集合，表示一个功能模块，常用于模块化编程。

以下是关于函数的一些重要说明：

* 在shell中，函数必须先定义，再调用
* 使用return value来获取函数的返回值
* 函数在当前shell中执行，可以使用脚本中的变量。

函数的格式如下：

函数名()

{

命令1…..

命令2….

return 返回值变量

}

[ function ] funname [()]

{

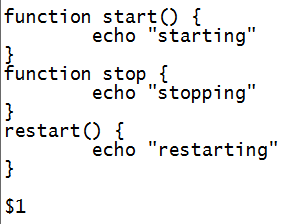
  action;

  [return int;]

}

function start() / function start / start()

eg:



注意：

函数返回值，只能通过$? 系统变量获得，可以显示加：return 返回值，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)

自定义函数，需要传递参数的时候，那么必须在函数调用的时候传值

stop a b c

自定义函数stop里面，可以使用$n来接收传递的参数

如果我们传递的参数，不是在shell中写死的，而是在调用shell脚本的时候传递进来的。

那么在调用函数的时候，就可以使用$n来接收传递的参数

stop $1 $2 $3 $4 ….

函数的名称也可以不在shell中写死，而是通过传递的参数调用。

# 脚本调试

# bash -x script  
这将执行该脚本并显示所有变量的值。

set -x  对部分脚本调试  
# bash -n script  
不执行脚本只是检查语法的模式，将返回所有语法错误。

#bash –v script

执行并显示脚本内容

# cut

**cut [选项]  文件名        默认分割符是制表符**

**作用：文件切分**

选项：

    -f 列号：    提取第几列

    -d 分隔符：    按照指定分隔符分割列

eg:#cut -f  2  aa.txt   提取第二列

eg:#cut -d ":" -f 1,3 /etc/passwd  以:分割，提取第1和第3列

eg:#cat /etc/passwd | grep /bin/bash | grep -v root | cut -d ":" -f 1    获取所有可登陆的普通用户用户名

cut的局限性    不能分割空格   df -h  不能使用cut分割

df -h | grep sda1 | cut -f 5

# 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，切开的部分再进行各种分析处理。

语法：awk ‘条件1{动作1}条件2{动作2}...’文件名

条件（Pattern）:

一般使用关系表达式作为条件： >   >=  <=等

动作（Action）：

格式化输出

流程控制语句

eg:#df -h | awk '{print $1 "\t" $3}'      显示第一列和第三列

**FS内置变量**

eg:# cat /etc/passwd | grep "/bin/bash" | awk 'BEGIN {FS=":"} {print $1 "\t"$3 }'    输出可登陆用户的用户名和UID,这里使用FS内置变量指定分隔符为：,而且使用BEGIN保证第一行也操作，因为awk命令会在读取第一行后再执行条件

**指定分隔符还可以用-F更简单**

eg:# cat /etc/passwd | grep "/bin/bash" | awk -F:  '{print $1 "\t"$3 }'   效果同上

eg:判断一下根目录的使用情况

#df -h |grep sda2 | awk '{print $5}' | awk -F% '{print $1} $1<80{print "info"}$1>80{print "warning"}'

**BEGIN 在所有数据读取之前执行**

eg：#awk 'BEGIN {printf "first Line \n"} {printf $2 }' aa.txt     在输出之前使用BEGIN输出内容

**END 在所有数据执行之后执行**

eg：#awk 'END {printf "The End \n"} {print $2}' aa.txt   所有命令执行完后，输出一句"The End"

# df -h | grep sda2 | awk '{print $5}' | awk -F% '{print $1}'

# df -h | grep sda2 | awk '{print $5}' | cut -d"%" -f 1

获取所有用户信息里的用户名：

cat /etc/passwd | awk -F: '{print $1}'

awk -F: '{print $1}' /etc/passwd

获取当前机器的ip地址：

# ifconfig eth0 | grep 'inet addr' | awk -F: '{print $2}' | awk '{print $1}'

# sed

stream editor

s e d是一个非交互性文本**流**编辑器。它编辑文件或标准输入导出的文本拷贝（副本）。标准输入可 能是来自键盘、文件重定向、字符串或变量，或者是一个管道的文本。

注意： s e d 并不与初始化文件打交道， 它操作的只是一个拷贝，然后所有的改动如果没有重定向到一个文件，将输出到屏幕（控制台输出）。

语法：sed [选项]’[动作]’ 文件名

常用选项:

-n　　使用安静（silent）模式。显示经过sed特殊处理的数据。

-e　　允许多点编辑。

-i　　直接修改读取的档案内容，而不是由屏幕输出。

|  |  |
| --- | --- |
| 命令 | 功能描述 |
| a\ | 新增， a 的后面可以接字串，在下一行出现 |
| c\ | 替换 |
| d | 删除 |
| i\ | 插入， i 的后面可以接字串 |
| p | 打印 |
| s | 查找并替换，例如 1,20s/old/new/g |

eg:

sed '2p' sed.txt 显示第二行和所有数据

sed -n '2,3p' sed.txt 显示第二和第三行

df -h | sed -n '1p' 接收命令结果数据

sed ‘2a liuyifei’sed.txt 在第二行后面添加数据

sed ‘4i fengjie \

canglaoshi’sed.txt 在第4行之前添加两行数据

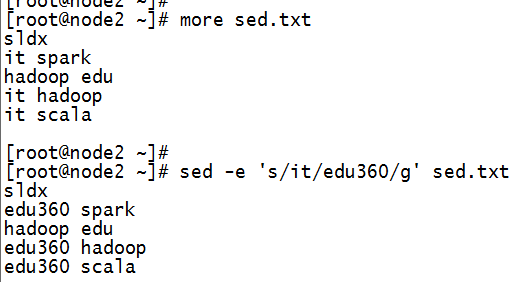
sed ‘2c this is replace’sed.txt 替换第二行数据

sed ‘s/it/edu360/g’ sed.txt 把sed.txt文件中的it替换为edu360,并输出

sed -e '1s/1/34/g;3s/yangmi//g' sed.txt 同时进行多个替换

sed –i ‘s/it/edu360/g’ sed.txt 要想真正替换，需要使用-i参数

^ :shift +6



使用sed获取机器的ip地址

ifconfig eth0 | grep 'inet addr'| sed 's/^.\*addr://g' | sed 's/ Bcast.\*$//g'

# 定时器

crontab 命令允许用户提交、编辑或删除相应的作业。每一个用户都可以有一个crontab 文件来保存调度信息。可以使用它运行任意一个 s h e l l 脚本或某个命令。

crontab命令格式

作用：用于生成cron进程所需要的crontab文件  
crontab的命令格式  
      crontab  [-u  user] {-l|-r|-e}

    -u   用户名   
    -l    显示当前的crontab

    -r    删除当前的crontab  
    -e   使用编辑器编辑当前的crontab文件。

  crontab文件格式  
minute hour  day-of-month  month-of-year  day-of-week  commands

分< >时< >日< >月< >星期< >要运行的命令     < >表示空格

其中  
    Minute              一小时中的哪一分钟     [0～59]   
    hour                  一天中的哪个小时        [0～23]   
    day-of-month    一月中的哪一天           [1～31]   
    month-of-year   一年中的哪一月           [1～12]   
    day-of-week      一周中的哪一天           [0～6]     0表示星期天  
    commands         执行的命令

书写注意事项  
1,全都不能为空，必须填入，不知道的值使用统配符\*表示任何时间  
2,每个时间字段都可以指定多个值，不连续的值用,间隔，连续的值用-间隔。  
3,命令应该给出绝对路径  
4,用户必须具有运行所对应的命令或程序的权限

每隔3分钟,第一个\*号

\*/3

3个任务

1. 需要检测，执行任务的数据，是否已经准备好了（判断一下某个文件是否存在） check1.sh
2. 运行报表任务 report.sh
3. 把任务执行完成的结果数据发送 send.sh

总的调度任务：sche.sh

check1.sh

report.sh

if [ ];then

send.sh

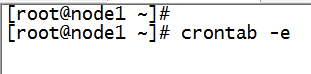
fi

如何使用crontab 运行多个任务:

方法1：在crontab -e 里 写多个

输入命令 crontab –e

敲回车



开始编写任务：



方法2：把所有的任务，写入到一个可执行的文件

再在crontab -e里面配置执行任务



分钟   小时   天   月   星期   命令/脚本

示例：

eg:4点备份

0      4        \*      \*     \*

eg:每周二，周五，下午6点  的计划任务

0     18       \*      \*     2,5

eg:1到3月份，每周二周五，下午6点的计划任务

0     18       \*     1-3   2,5

eg:周一到周五下午，5点半提醒学生15分钟后关机

30   17      \*       \*     1-5   /usr/bin/wall < /etc/issue

45   17      \*       \*     1-5  /sbin/shutdown -h now

eg:学校的计划任务， 12点14点，检查apache服务是否启动

\*/2  12-14  \*   3-6,9-12   1-5

eg:再添加一个备份，把/etc目录备份到/backup下，然后把错误的情况也记录下来，正确的文件都丢到/dev/null下，看不见（相当于一个黑洞）

\*/2  12-14  \*   3-6,9-12   1-5   /bin/cp -r /etc  /backup/etc.20170407 2> /backup/etc.bak.err

/dev/null

eg:每月 1、1 0、2 2日的4:45运行/apps/bin目录下的backup.sh

45  4  1,10,22  \*  \* /apps/bin/backup.sh

eg:每周六、周日的 1 : 10运行一个find命令

10  1  \*  \*  6,0  /bin/find -name "core" -exec rm {} \;