shell学习

1. cron和crontab

cron系统的主要调度进程，可以在无需人工干预的情况下运行作业

crontab命令允许用户提交、编辑、或删除相应的作业，每个用户都可以有一个crontab文件来保存调度信息

2.crontab的域

格式：

分<>时<>日<>月<>星期<>要运行的命令

<>表示空格

#表注释

在每个域中：

- 表示一个时间范围

，表示或的意思

\* 表示连续的时间段，若对某个域没有特别的限定，也应该在该域填入"\*"

例：

30 21\* \* \* /apps/bin/cleanup.sh

表示每晚的21：30运行/apps/bin目录下的cleanup.sh

注意：每个命令必须给出shell脚本的绝对路径，设置相应的环境变量

3.crontab命令选项

命令的一般形式为：

crontab [-u user] -e -l -r

其中

-u 用户名

-e 编辑crontab文件

-l 列出crontab文件中的内容

-r 删除crontab文件

crotab可识别当前用户

4.创建一个新的crontab文件

在考虑向cron进程提交一个crontab文件之前，首先要做的一件事就是设置环境变量EDITOR

确定使用的编辑器编辑crontab文件，大多数unix和linux为vi

编辑$HOME目录下的.profile文件，加入下面那一行

EDITOR = vi; export EDITOR

5.at命令

at命令允许用户向cron守护进程提交作业，使其在稍后的时间运行，时间没有contab长

一旦一个作业被提交，at命令将会保留所有当前的环境变量，包括路径，不像crontab，只提供缺省的环境

根用户可以在/etc目录下的at.allow和at.deny文件来控制哪些用户使用at命令，哪些用户不行

基本形式：

at [ -f script] [-m -l -r] [time] [data]

其中

-f script 是所要提交的脚本或命令

-l 列出当前所有等待运行的作业 atq命令具有相同的作用

-r 清除作业

-m 作业完成后给用户发邮件

time 时间格式非常灵活

data 日期格式可以是月份数或日期数 可以识别today tomorrow这样的词

1. 使用at命令提交命令或脚本

(1).at 时间 回车进入at命令提示符

(2).逐条输入相应的命令 ctrl+d结束

在指定的时间执行，并把结果用邮件发回来

1. 列出所提交的作业

at -l

2 Tue Aug 1 15:00:00 2017 a root

作业标识 运行日期/时间 a表示at root用户

1. 清除一个作业

atrm [job no] 或 at -r [job no]

例：atrm job 3

1. &命令

把作业放到后台执行

1. 一般形式： 命令 &

2.向后台提交命令提交成功后，系统会给出它的进程号

3.用ps命令查看进程

ps x | grep 进程号

或ps -ef | grep 进程号

ps列出进程，没法确定进程运行在前台还是后台

1. 杀死后台进程

kill -signal [process\_number]

1. nohup命令

1. 使在退出账号之后继续运行相应的进程，即不挂起的意思

一般形式：nohup command &

2. 使用nohup命令提交作业，在缺省情况下该作业的所有输出都被重定向到一个名为nohup.out的文件中，除非另外指定了输出文件

3.一次提交几个作业

如果希望一次提交几个命令，最好能够把他们写入到一个shell脚本文件中，并用nohup命令来执行它

nohup ./shell文件名

第四章 文件名置换

1. 使用\*

使用\*可以匹配文件名中的任何字符串

1.app\* 文件名以app开头，后面可以跟随任何字符串，包括空字符串

2.\*.doc 匹配所有以.doc结尾的文件名

3. cl\*.sed 匹配所有以cl开头、后面跟任何字符串，最后以.sed结尾的文件名

1. 使用？

使用？可以匹配文件名中的任意单个字符

1. 使用[...]和[!...]

使用[...]可以用来匹配方括号[]中的任何字符，还可以使用一个横杠，来连接两个字母或数字，以此来表示一个范围

例：

ls [io]\* 列出以i或o开头的文件名

ls log.[0-9] \* 列出以log.开头 后面跟随0-9任意一个数字，然后可以是任意字符串的文件名

第五章 shell输入与输出

在shell脚本中

1. echo命令

使用echo命令可以显示文本行或变量，或者把字符串输入到文件

一般形式： echo string

1.常用功能

\c 不换行

\f 进纸

\t 跳格

\n 换行

在linux系统中 必须使用-n来禁止echo命令输出后换行

必须使用-e 来使转义符生效

例：

echo -e “what is your name:\c”

2.想把一个字符输出到文件中，使用重定向符号>

echo “ string” > 文件名

3.把字符串追加到一个文件的末尾 使用>>

echo “string” >> 文件名

1. read命令

1.可以使用read语句从键盘或文件的某一行文本读入信息，并将其赋给一个变量，如果只指定了一个变量，那么read将会把所有的输入赋给该变量，直到遇到第一个文件结束符或回车

一般形式： read varible1 varible2...

1. shell将空格作为变量之间的分割符，如果输入的文本域过长，shell将所有的超长部分赋予最后一个变量

例：

read name surname

输入John Lemon Doe

则name 为John

surname为 Lemon Doe

1. cat命令
2. 用来显示文件内容，创建文件，还可以显示控制字符，不会在文件分页符处停下来，会显示完整个文件

一般形式：

cat [options] filename1 ... filename2 ...

cat myfile 显示名为myfile的文件

cat > myfile 从键盘输入数据ctrl+D结束 存入myfile

cat -v 文件名 显示所有的控制字符

注意：如果cat后直接回车，该命令会等待输入，并且回显输入，Ctrl+d 结束

1. 管道

1.可以通过管道把一个命令的输出传递给另一个命令作为输入

一般形式：命令1 | 命令2

例

ls | grep a.doc 表示grep命令在ls出来的文件列表中搜索a.doc

1. tee

把输出的一个副本输送到标准输出，另一个副本拷贝到相应的文件中

一般形式：tee -a files

-a表示追加到文件末尾

例：

who | tee who.out 表示who的结果输出到屏幕，且输出到who.out

六、标准输入、标准输出、标准错误

1.标准输入：文件描述符是0，是命令的输入，缺省是键盘，也可以是其他文件或者命令的输出

2.标准输出：文件描述符是1，是命令的输出，缺省是屏幕，也可以是文件

3.标准错误：文件描述符是2，是命令错误的输出，缺省是屏幕，也可以是文件

1. 文件重定向

1.在对标准错误进行重定向时，必须要使用文件描述符，但是对于标准输入和输出来说，这不是必须的。

常见的文件重定向命令

|  |  |
| --- | --- |
| command>filename | 把标准输出重定向到一个新文件 |
| command>>filename | 把标准输出重定向到一个文件中追加 |
| command 1> filename | 把标准输出重定向到一个文件中 |
| command > filename 2 >&1 | 把标准输出和标准错误一起重定向到一个文件中 |
| command 2> filename | 把标准错误重定向到一个文件中 |
| command 2>> filename | 把标准错误重定向到一个文件中 追加 |
| command >> filename 2>&1 | 把标准输出和标准错误一起重定向到一个文件中追加 |
| command < filename > filename2 | command命令以filename文件作为标准输入，以filename文件作为标准输出 |
| command<filename | command命令以filename文件作为标准输入 |
| command << delimiter | 从标准输入中读入，直到遇到delimiter分界符 |
| command < &m | 把文件描述符m作为标准输入 |
| command > &m | 把标准输出重定向到文件描述符m中 |
| command < & | 关闭标准输入 |

创建一个长度为0的空文件，可以用 ‘>filename’

1. exec命令

1. exec命令可以用来替代当前的shell，换句话说，并没有启动子shell

使用这一命令时任何现有环境都会被清除，并重新启动一个shell

一般形式：

exec command

command通常是一个shell脚本

1. exec可以通过文件描述符打开和关闭文件

exec <file 将file中的内容作为exec的标准输入

exec >file 将file中的内容作为标准写出

exec 3<file 将file读入到fd3中

exec 4>file 将写入fd4中的内容写入file中

exec 5<&4 创建fd4的拷贝fd5

exec 3<&- 关闭fd3

例:

exec 4<&0 0<stock.txt 把文件描述符4指定为标准输入，然后打开stock.txt文件

exec 0<&4 作为标准输入的文件描述符4被关闭

第六章 命令执行顺序

一、使用&&

一般形式： 命令1&&命令2

命令1返回真，命令2才会被执行

二、使用||

一般形式： 命令1||命令2

命令1未执行成功，命令2就执行

1. 用{}和()将命令结合在一起

如果希望把几个命令合在一起执行，shell提供两种方法，既可以在当前shell也可以在子shell中执行一组命令

在当前shell中执行一组命令

(命令1;命令2;...)

在子shell中执行一组命令

{命令1;命令2;...}

例：

comet month\_end || (echo “Hello,guess what! Comet did not work” | mail dve;exit)

如果脚本comet失败，则发送邮件后退出，如果没有(),就只执行exit

第七章 正则表达式

当从一个文件或命令输出中抽取或过滤文本时，可以使用正则表达式，正则表达式是一些特殊或不很特殊的字符串模式的集合

为了抽取或获得信息，我们给出抽取操作应遵守的一些规则，这些规则由一些特殊字符或进行模式匹配操作时使用的元字符组成

基本元字符集及其含义

|  |  |
| --- | --- |
| ^ | 只匹配行首 |
| $ | 只匹配行尾 |
| \* | 一个单字符后紧跟\*，匹配0个或多个此单字符 |
| [] | 匹配[]内字符，可以是一个单字符，也可以是一个字符序列，可以使用 - 表示[]内字符序列的范围 [1-5] |
| \ | 用来屏蔽一个元字符的特殊含义，因为有时在shell中一些元字符有特殊含义 |
| . | 匹配任意单字符 |
| pattern\{n\} | 用来匹配前面pattern出现的次数，n为次数 |
| pattern\{n,\}m | 含义同上，但次数最少为n，并以m结尾 |
| pattern\{n,m\} | 含义同上，但pattern出现的次数在n与m之间 |

匹配所有的空行 ^$

[^0-9] 匹配任一非数字型字符

第八章 grep家族

一、grep

grep（全局正则表达式版本）允许对文本文件进行模式查找，如果找到匹配模式，grep打印包含模式的所有行。

grep支持基本的正则表达式，也支持其扩展集，grep有三种变形

Grep：标准grep命令

Egrep：扩展grep 支持基本及扩展的正则表达式

Fgrep：快速grep 允许查找字符串而不是一个模式

二grep的使用

1.一般格式：

grep [选项] 基本正则表达式[文件]

这里的基本正则表达式可以为字符串

2.双引号的使用

调用字符串和变量名时加“”

调用模式匹配时用‘’

3.grep选项

-c 只输出匹配行的计数

-i 不区分大小写（只适用于单字符）

-h 查询多个文件时不显示文件名

-l 查询多文件时，只输出包含匹配字符的文件名

-n 显示匹配行及行号

-s 不显示不存在或无法匹配文本的错误信息

-v 显示不包含匹配文本的所有行

例：

grep “sort” \*.doc 查询当前目录下所有.doc文件中查找字符串“sort”

4.系统grep命令

用管道来获取grep命令得到的系统信息

ps ax | grep “named”

查看dns服务器是否正在运行

5.egrep 接受所有的正则表达式，显著特征是可以以一个文件作为保存的字符串，然后将之传给egrep作为参数，，可使用-f开关创建一个文件

第九章 awk介绍

一、.awk格式化报文或从一个大的文本文件中抽取数据包

二、.调用awk

三种方式

1.命令行方式

awk [-F field-separator] ‘commands’ input-file(s)

这里： cmmands是真正的awk命令

[-F 域间分隔符] 是可选的，因为awk使用空格作为缺省的域分隔符，因此如果要浏览域间有空格的文本，不必指定这个选项

如果要浏览诸如passwd文件，此文件域间以冒号作为分隔符，则必须指明-F选项

2.将所有的awk命令插入一个文件，并使awk程序可执行，然后用awk命令解释器作为脚本的首行，以便通过键入脚本的名称来调用它

1. 将所有的awk命令插入一个单独的文件，然后调用它

awk -f awk-script-file input-files

-f 指明在文件awk-script-file中的awk脚本，input-file(s)是使用awk进行浏览的文件

1. awk脚本

1. 在命令中调用awk时，awk脚本由各种操作和模式组成

如果设置了-F选项，则awk每次读一条记录或一行，并使用指定的分割符分割指定域，未设置，则假定空格为分隔符，并保持这个设置直到发现一新行，新行出现，awk命令获悉已读完整条记录，然后在下一记录启动读命令

2.模式和动作

任何awk语句都是由模式和动作组成，模式部分决定动作语句何时触发及触发事件，处理即对数据进行操作，如果省略模式部分，动作将时刻保持执行状态

1. 域和记录

awk执行时，其浏览域标记为$1,$2...,$n,这种方法称为域标识

$0表示所有域

awk ‘{print $0}’ grade.txt >wow 打印所有域到wow文件

awk ‘{print $0}’ grade.txt | tee wow 使用tee命令将输出到wow文件

awk ‘BEGIN {print “Name Belt\n------”}

{print $1 “\t” $4}’ grade.txt

BEGIN设置计数或者打印头

awk ‘BEGIN {print “Name Belt\n------”}

{print $1 “\t” $4} END {“end-of-report”}’ grade.txt

END在文本处理动作执行之后才执行

1. awk条件操作符

< 小于 >=大于等于

<= 小于等于 ~匹配正则表达式

== 等于 !~不匹配正则表达式

!= 不等于

例：awk ‘{if($4~/Brown/) print $0}’ grade.txt

如果field-4包含brown，打印

4.awk内置变量

|  |  |
| --- | --- |
| ARGC | 命令行参数个数 |
| ARGV | 命令行参数排列 |
| ENVIRON | 支持队列中系统环境变量的使用 |
| FILENAME | awk浏览的文件名 |
| FNR | 浏览文件的记录数 |
| FS | 设置输入域分隔符，等价于命令行-F选项 |
| NF | 浏览记录的域个数 |
| BR | 已读的记录数 |
| OFS | 输出域分隔符 |
| ORS | 输出记录分隔符 |
| RS | 控制记录分隔符 |

例：

awk ‘{print NF,NR,$0}END{print FILENAME}’ grade.txt

显示记录的域的个数，每条记录当前已读的记录数，记录 最后显示浏览的文件名

5.awk操作符

|  |  |
| --- | --- |
| = += \*= /= %= ^= | 赋值操作符 |
| ? | 条件表达操作符 |
| || && ! | 并、与、非 |
| ~ !~ | 匹配 不匹配 |
| < <= == != >> | 关系操作符 |
| + - \* / % ^ | 算术操作符 |
| ++ -- | 前缀和后缀 |

1. 内置的字符串函数

|  |  |
| --- | --- |
| gsub(r,s) | 在整个$0中用s替代r |
| gsub(r,s,t) | 在整个t中用s替代r |
| index(s,t) | 返回s中字符串t的第一位置 |
| length(s) | 返回s的长度 |
| match(s,r) | 测试s是否包含匹配r的字符串 |
| split(s,a,fs) | 用fs分隔符将s分成序列a |
| sprint(fmt,exp) | 返回经fmt格式化后的exp |
| sub(r,s) | 用s代替$0中第一次出现的r |
| substr(s,p) | 返回字符串s中从p开始的后缀部分 |
| substr(s,p,n) | 返回字符串s中从p开始长度为n的后缀部分 |

1. 字符串屏蔽序列

\b 退格键 \t tab键

\f 走纸换页 \ddd 八进制值

\n 新行 \c 任意其他特殊字符，例如\\为反斜线符号

\r 回车键

1. awk输出函数printf

awk printf修饰符

|  |  |
| --- | --- |
| - | 左对齐 |
| Width | 域的步长，用0表示0步长 |
| .prec | 最大字符串长度，或小数点右边的位数 |

awk printf格式

|  |  |
| --- | --- |
| %c | ASCII字符 |
| %d | 整数 |
| %e | 浮点数 科学计数法 |
| %f | 浮点数 如123.44 |
| %g | awk自定用f或e |
| %o | 八进制数 |
| %s | 字符串 |
| %x | 十六进制数 |

例：

echo “65” | awk ‘{printf “%c\n” ,$0}’

A

第十章 sed用法

1. sed

sed从文件的一个文本行或从标准输入的几种格式中读取数据，将之拷贝到一个编辑缓冲区，然后读命令或脚本的第一条命令。并使用这些命令查找模式或定位行号编辑它，重复此过程，直至命令结束

1. 调用sed

1.命令行键入命令

sed [选项] sed命令 输入文件

实际命令要加单引号

2.将sed命令插入脚本文件

sed [选项] -f sed脚本 输入文件

3.使用第一行具有sed命令解释器的sed脚本文件

sed脚本文件 [选项] 输入文件

如果没有指定输入文件，默认标准输入

sed选项：

n 不打印；sed不写编辑行到标准输出，缺省为打印所有行（编辑和未编辑）

p 打印编辑行

c 下一命令是编辑命令

f 如果正在调用sed脚本文件，使用此选项，此选项通知sed一个脚本文件支持所有sed命令

4.保存sed输出

sed ‘some-sed-commands’ input-file > myoutfile

5.使用sed在文件中查询文本的方式

1. .使用行号，可以是一个简单的数字，或是一个行号范围
2. .使用正则表达式

|  |  |
| --- | --- |
| x | x为一行号，如1 |
| x,y | 表示行号范围从x到y |
| /pattern/ | 查询包含模式的行 |
| /pattern/pattern/ | 查询包含两个模式的行 |
| /pattern/,x | 在给定行号上查询包含模式的行 |
| x,/pattern/ | 通过行号和模式查询匹配行 |
| x,y! | 查询不包含指定行号x和y的行 |

1. 基本的sed编辑命令

p 打印匹配行

= 显示文本行号

a/ 在定位行后附加新文本信息

i/ 在定位行后插入新文本信息

d 删除定位行

c/ 用文本替换定位文本

s 使用替换模式替换相应模式

r 从另一个文本中读文本

w 写文本到一个文件

q 第一个模式匹配完成后推出或立即推出

l 显示与八进制ASCII代码等价的控制字符

{} 在定位行执行的命令组

n 从另一个文本中读文本下一行，并附加下一行

g 将模式2粘贴到/pattern n/

y 传送字符

1. 基本的sed编程举例

1.print命令格式[address[,address]]P

sed -n ‘2p’ quote.txt 打印quote.txt前两行

sed -n ‘1,,3p’ quote.txt 打印quote.txt的1-3行

$表示最后一行

2.创建sed脚本文件

vi append.sed

#!/bin/sed -f sed命令解释行

/company/ a\ /company/附加操作起始位置 a\通知sed这是一个附加操作

Then suddenly it happened 附加操作要加入到拷贝的实际文本

3. 插入文本

在指定文件插入一行文本，用[address]i/

例如insert.sed

#!/bin/sed -f

/attendance/ i/

Utter confusion followed.

运行insert.sed quote.txt即可

4.修改文本，用[address]c/

例如change.sed

#!/bin/sed -f

/honeysuckle/ c/

The office Dibble band played well.

运行change.sed quote.txt

1. 删除文本

格式[address[,address]]d

sed '1d' quote.txt 删除第一行

sed '1,3d' quote.txt 删除1到3行

sed '$d' quote.txt 删除最后一行

1. 替换文本

格式：

[address[,address]] s/ pattern-to-find /replacement-pattern/[g p w n]

s选项通知sed这是个替换操作，并查询pattern-to-find 成功后用replacement-pattern来替换

g 缺省情况下值替换第一次出现模式，使用g替换全局所有的出现模式

p打印编辑行

w 文件名，将输出定向到一个文件

7.使用替换修改字符串

使用&命令，在一个字符串前添加一个字符串

sed -n ‘s/nurse/”Hello” &/p’ quote.txt

在文件中nurse前添加”Hello”

1. 将sed结果重定向到一个文件

[address[,address]]w filename

9.从文件中读文本

address r filename

sed ‘/company./r sedex.txt’ quote.txt

从seddex中读出文本附加到quote.txt的company.的后面

1. 匹配后退出

address q

1. 显示文件中的控制字符

[address[,address]] l

十二章 tr用法

一、tr

tr用来从标准输入中通过替换或删除操作进行字符转换，tr主要用于删除文件中控制字符或进行字符转换

格式

tr -c-d-s [“string1\_to\_translate\_from”] [“string2\_to\_triampsultate\_to”] file

-c 用字符串1中字符集的补集替换此字符集，要求字符集为ASCII

-d 删除字符串1所有输入字符

-s 删除所有重复出现的字符序列，只保留1个

file 转换文件名

第十八章 控制流结构

1. if then else语句

if 条件1

then 命令1

elif 条件2

then 命令2

else 命令3

fi

最简单的if语句

if 条件

then 命令

fi

1. case语句

case 值 in

模式1)

命令1

...

;;

模式2)

命令2

...

;;

esac

1. for循环

for 变量名 in 列表

do

命令1

命令2...

done

如果不加in列表，它将接受命令行位置参数作为参数

1. until循环

until [条件]

do

命令

done

1. while循环

while 命令

do

命令1

命令2

done

第19章 shell函数

1. 定义函数的格式

函数名()

{

命令

}