第5天-Shell 流编辑器 AWK

一、AWK 简介

- AWK 是一种编程语言，用于在linux/unix下对文本和数据进行处理。数据可以来自标准输入、一个或多个文件，或其它命令的输出。它支持用户自定义函数和动态正则表达式等先进功能，是linux/unix下的一个强大编程工具。它在命令行中使用，但更多是作为脚本来使用。
- AWK 的处理文本和数据的方式是这样的，它逐行扫描文件，从第一行到最后一行，寻找匹配的特定模式的行，并在这些行上进行你想要的操作。如果没有指定处理动作，则把匹配的行显示到标准输出(屏幕)，如果没有指定模式，则所有被操作所指定的行都被处理。awk分别代表其作者姓氏的第一个字母。因为它的作者是三个人，分别是Alfred Aho、Brian Kernighan、Peter Weinberger。gawk是awk的GNU版本，它提供了Bell实验室和GNU的一些扩展。

二、AWK 语法格式

1、AWK 命令行方式

```
awk [-F  field-separator]  'commands'  input-file(s)
awk [选项参数] 'commands' var=value file(s)
```

- commands 是真正 AWK 命令，[-F域分隔符]是可选的。 input-file(s) 是待处理的文件
- AWK 中，文件的每一行中，由域分隔符分开的每一项称为一个域。通常，在不指名-F域分隔符的情况下，默认的域分隔符是空格。

2、AWK 命令插入一个单独文件调用

```
awk -f awk-script-file input-file(s)
awk [选项参数] -f scriptfile var=value file(s)
```

- -f 选项加载 awk-script-file 中的 awk 脚本，input-file(s) 是待处理的文件。

3、shell 脚本方式

- 将所有的awk命令插入一个文件，并使 awk 程序可执行，然后awk命令解释器作为脚本的首行，一遍通过键入脚本名称来调用。相当于shell脚本首行的：#!/bin/sh 可以换成：#!/bin/awk

三、AWK 工作原理

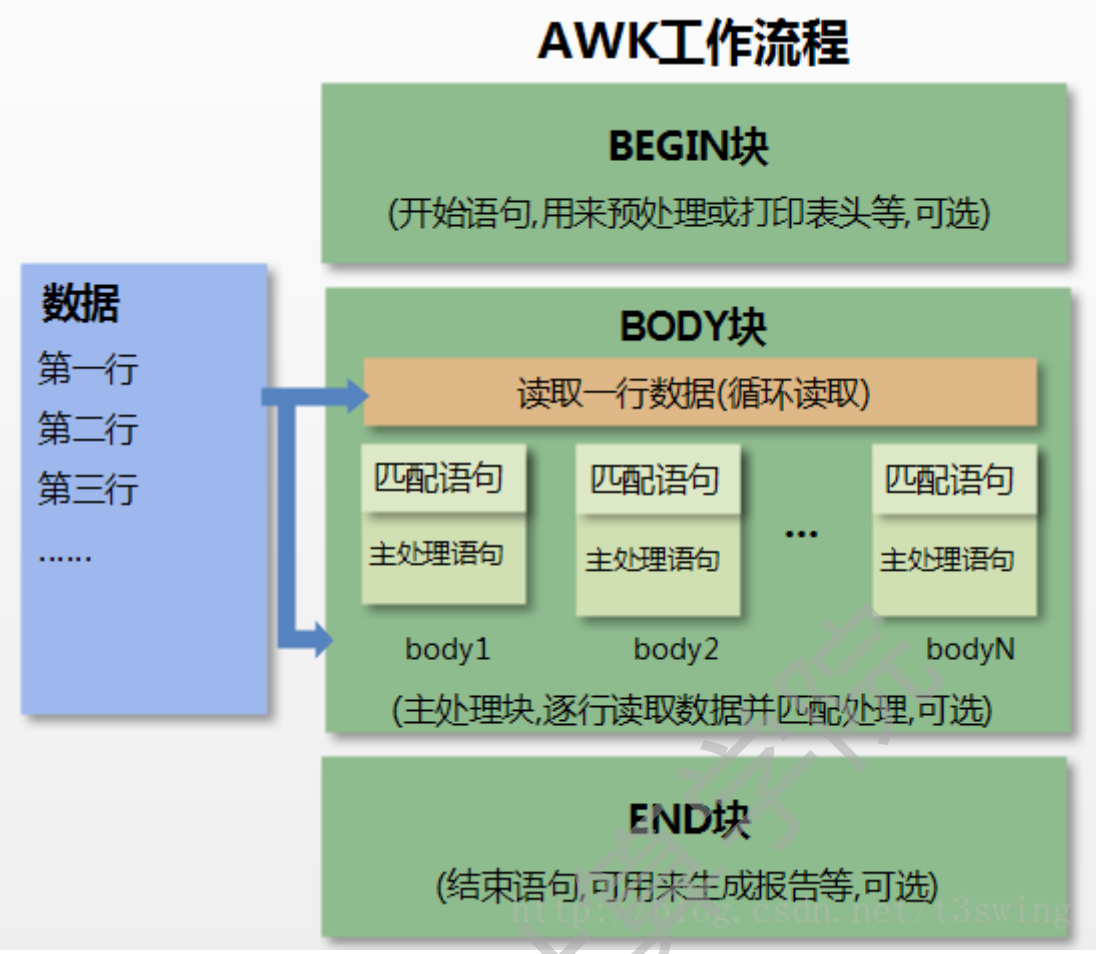
AWK 工作流程可分为三个部分：

- 读输入文件之前执行的代码段（由BEGIN关键字标识）。
- 主循环执行输入文件的代码段。
- 读输入文件之后的代码段（由END关键字标识）。

1、AWK 命令结构

```
awk 'BEGIN{ commands } pattern{ commands } END{ commands }'
```

下图是 AWK 的工作流程



- 通过关键字 BEGIN 执行 BEGIN 块的内容，即 BEGIN 后花括号 **{}** 的内容。
- 完成 BEGIN 块的执行，开始执行body块。
- 读入有 **\n** 换行符分割的记录。
- 将记录按指定的域分隔符划分域，填充域，**$0** 则表示所有域(即一行内容)，**$1** 表示第一个域，**$n** 表示第 n 个域。
- 依次执行各 BODY 块，pattern 部分匹配该行内容成功后，才会执行 awk-commands 的内容。
- 循环读取并执行各行直到文件结束，完成body块执行。
- 开始 END 块执行，END 块可以输出最终结果。

2、开始块（BEGIN）

开始块的语法格式如下：

```
BEGIN {awk-commands}
```

开始块就是在程序启动的时候执行的代码部分，并且它在整个过程中只执行一次。

一般情况下，我们可以在开始块中初始化一些变量。

BEGIN 是 AWK 的关键字，因此它必须是大写的。

注意：开始块部分是可选的，你的程序可以没有开始块部分。

3、主体块（BODY）

主体部分的语法格式如下：

```
/pattern/ {awk-commands}
```

对于每一个输入的行都会执行一次主体部分的命令。

默认情况下，对于输入的每一行，AWK 都会执行命令。但是，我们可以将其限定在指定的模式中。

注意：在主体块部分没有关键字存在。

4、结束块（END）

结束块的语法格式如下：

```
END {awk-commands}
```

结束块是在程序结束时执行的代码。 END 也是 AWK 的关键字，它也必须大写。 与开始块相似，结束块也是可选的。

5、实例

先创建一个名为 marks.txt 的文件。其中包括序列号、学生名字、课程名称与所得分数。

```
1)      张三      语文      80
2)      李四      数学      90
3)      王五      英语      87
```

接下来，我们将使用 AWK 脚本来显示输出文件中的内容，同时输出表头信息。

```
[root@qfedu.com ~]# awk 'BEGIN{printf "序号\t名字\t课程\t分数\n"} {print}'
marks.txt
```

执行以上命令，输出结果如下：

```
序号      名字      课程      分数
1)      张三      语文      80
2)      李四      数学      90
3)      王五      英语      87
```

程序开始执行时，AWK 在开始块中输出表头信息。在主体块中，AWK 每读入一行就将读入的内容输出至标准输出流中，一直到整个文件被全部读入为止。

四、AWK 选项参数说明

- -F fs or --field-separator fs
  指定输入文件折分隔符，fs是一个字符串或者是一个正则表达式，如-F:。
- -v var=value or --asign var=value
  赋值一个用户定义变量。
- -f scripfile or --file scriptfile
  从脚本文件中读取awk命令。
- -mf nnn and -mr nnn
  对nnn值设置内在限制，-mf选项限制分配给nnn的最大块数目；-mr选项限制记录的最大数目。这两个功能是Bell实验室版awk的扩展功能，在标准awk中不适用。
- -W compact or --compat, -W traditional or --traditional
  在兼容模式下运行awk。所以gawk的行为和标准的awk完全一样，所有的awk扩展都被忽略。
- -W copyleft or --copyleft, -W copyright or --copyright
  打印简短的版权信息。

- -W help or --help, -W usage or --usage
  打印全部awk选项和每个选项的简短说明。
- -W lint or --lint
  打印不能向传统unix平台移植的结构的警告。
- -W lint-old or --lint-old
  打印关于不能向传统unix平台移植的结构的警告。
- -W posix
  打开兼容模式。但有以下限制，不识别：/x、函数关键字、func、换码序列以及当fs是一个空格时，将新行作为一个域分隔符；操作符**和**=不能代替^和^=；fflush无效。
- -W re-interval or --re-inerval
  允许间隔正则表达式的使用，参考(grep中的Posix字符类)，如括号表达式[[:alpha:]]。
- -W source program-text or --source program-text
  使用program-text作为源代码，可与-f命令混用。
- -W version or --version
  打印bug报告信息的版本。

五、AWK 基本用法

1、创建 log.txt 文件

```
[root@qfedu.com ~]# cat log.txt
2 this is a test
3 Are you like awk
This's a test
10 There are orange,apple,mongo
```

2、行匹配语句（默认分隔符）

```
awk '{[pattern] action}' {filenames}   # 行匹配语句 awk '' 只能用单引号
```

- 实例：

```
# 每行按空格或TAB分割，输出文本中的1、4项
[root@qfedu.com ~]# awk '{print $1,$4}' log.txt
2 a
3 like
This's
10 orange,apple,mongo

# 格式化输出
[root@qfedu.com ~]# awk '{printf "%-8s %-10s\n",$1,$4}' log.txt
2        a
3        like
This's
10       orange,apple,mongo
```

3、指定分割字符

```
awk -F  #-F相当于内置变量FS, 指定分割字符
```

- 实例

```
# 使用","分割
[root@qfedu.com ~]# awk -F, '{print $1,$2}' log.txt
2 this is a test
3 Are you like awk
This's a test
10 There are orange apple

# 或者使用内建变量
[root@qfedu.com ~]# awk 'BEGIN{FS=","} {print $1,$2}' log.txt
2 this is a test
3 Are you like awk
This's a test
10 There are orange apple

# 使用多个分隔符.先使用空格分割，然后对分割结果再使用","分割
[root@qfedu.com ~]# awk -F '[ ,]'  '{print $1,$2,$5}' log.txt
2 this test
3 Are awk
This's a
10 There apple
```

3、设置变量

```
awk -v  #设置变量
```

- 实例

```
[root@qfedu.com ~]# awk -va=1 '{print $1,$1+a}' log.txt
2 3
3 4
This's 1
10 11

[root@qfedu.com ~]# awk -va=1 -vb=s '{print $1,$1+a,$1b}' log.txt
2 3 2s
3 4 3s
This's 1 This'ss
10 11 10s
```

4、正则字符串匹配

~ 表示模式开始。// 中是模式。

```
[root@qfedu.com ~]# awk '$1 ~ /bin/ {print $1,$4}' /etc/passwd
[root@qfedu.com ~]# awk -F: '$1 ~ /^alice/'  /etc/passwd

# 输出包含"root" 的行
[root@qfedu.com ~]# awk '/root/ ' /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin

# 匹配记录（整行）：
[root@qfedu.com ~]# awk '/^alice/'  /etc/passwd
[root@qfedu.com ~]# awk '$0 ~ /^alice/'  /etc/passwd
[root@qfedu.com ~]# awk '!/alice/' passwd
[root@qfedu.com ~]# awk '$0 !~ /^alice/'  /etc/passwd
```

5、忽略大小写

```
[root@qfedu.com ~]# awk 'BEGIN{IGNORECASE=1} /bin/' /etc/passwd
```

6、模式取反

```
[root@qfedu.com ~]# awk '$1 !~ /bin/ {print $1,$4}' /etc/passwd
[root@qfedu.com ~]# awk -F: '$NF !~ /bash$/'  /etc/passwd
```

7、使用 AWK 脚本

```
awk -f {awk脚本} {文件名}
```

实例

```
[root@qfedu.com ~]# awk -f cal.awk log.txt
```

六、AWK 的 print 和 printf

- print 和 printf 都是打印输出的，不过两者用法和显示上有些不同而已。

```
print 格式：print item1,item2, ...
printf格式：printf “FORMAT ”,item1,item2, ...
```

- 逗号为分隔符时，显示的是空格；
- 分隔符分隔的字段（域）标记称为域标识，用$0,$1,$2,...,$n表示，其中$0 为所有域，$1就是表示第一个字段（域），以此类推；
- 输出的各item可以字符串，也可以是数值，当前记录的字段，变量或awk 的表达式等；
- 如果省略了item ，相当于print $0
- 对于printf来说，其中格式化字符串包括两部分内容: 一部分是正常字符，这些字符将按原样输出; 另一部分是格式化规定字符， 以 **%** 开始, 后跟一个或几个规定字符，用来确定输出内容格式，必须指定 FORMAT，即必须指出后面每个itemsN 的输出格式，**printf** 时默认是不会换行的，而 **print** 函数默认会在每行后面加上 **\n** 换行符

格式符	说明
%d	十进制有符号整数
%u	十进制无符号整数
%f	浮点数
%s	字符串
%c	单个字符
%p	指针的值
%e	指数形式的浮点数
%x	%X 无符号以十六进制表示的整数
%o	无符号以八进制表示的整数
%g	自动选择合适的表示法
%%	显示%自身
#[.#]	第一个数字控制显示的宽度；第二个#表示小数点后精度，%3.1f
-	左对齐（默认右对齐）；%-15s，就是以左对齐方式显示15个字符长度
+	显示数值的正负符号 %+d

- 实例

```
# print 函数
[root@qfedu.com ~]# awk '{print "hello,awk"}'
[root@qfedu.com ~]# awk -F: '{print}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{print “wang”}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{print $1}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{print $0}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{print $1"\t"$3}' /etc/passwd
[root@qfedu.com ~]# date |awk '{print "Month: " $2 "\nYear: " $NF}'
[root@qfedu.com ~]# awk -F: '{print "username is: " $1 "\t uid is: " $3}'
/etc/passwd
[root@qfedu.com ~]# awk -F: '{print "\tusername and uid: " $1,$3 "!"}'
/etc/passwd

# printf函数
[root@qfedu.com ~]# tail -3 /etc/fstab |awk '{print $2,$4}'
[root@qfedu.com ~]# awk -F: '{printf "%-15s %-10s %-15s\n", $1,$2,$3}'
/etc/passwd
[root@qfedu.com ~]# awk -F: '{printf "|%-15s| %-10s| %-15s|\n", $1,$2,$3}'
/etc/passwd
[root@qfedu.com ~]# awk -F: '{printf "%s",$1}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{printf "%s\n",$1}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{printf "%-20s %10d\n",$1,$3}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{printf "Username: %s\n",$1}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{printf "Username: %s,UID:%d\n",$1,$3}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{printf "Username: %15s,UID:%d\n",$1,$3}'
/etc/passwd
```

```
[root@qfedu.com ~]# awk -F: '{printf "Username: %-15s,UID:%d\n",$1,$3}' /etc/passwd
[root@qfedu.com ~]# lsmod | awk -v FS=" " 'BEGIN{printf "%s %26s %10s\n","Module","Size","Used by"}{printf "%-20s %13d %5s %s\n",$1,$2,$3,$4}' /proc/modules
```

七、AWK 运算符

运算符	描述
= += -= *= /= %= ^= **=	赋值
?:	C条件表达式
||	逻辑或
&&	逻辑与
~ 和 !~	匹配正则表达式和不匹配正则表达式
< <= > >= != ==	关系运算符
空格	连接
+ -	加，减
* / %	乘，除与求余
+ - !	一元加，减和逻辑非
^ ***	求幂
++ --	增加或减少，作为前缀或后缀
$	字段引用
in	数组成员

- 实例

1、过滤第一列大于2的行

```
[root@qfedu.com ~]# awk '$1>2' log.txt
3 Are you like awk
This's a test
10 There are orange,apple,mongo
```

2、过滤第一列等于2的行

```
[root@qfedu.com ~]# awk '$1==2 {print $1,$3}' log.txt
2 is
```

3、过滤第一列大于2并且第二列等于'Are'的行

```
[root@qfedu.com ~]# awk '$1>2 && $2=="Are" {print $1,$2,$3}' log.txt
3 Are you
```

4、过滤练习

```
[root@qfedu.com ~]# awk -F: '$3 == 0' /etc/passwd
[root@qfedu.com ~]# awk -F: '$3 < 10' /etc/passwd
[root@qfedu.com ~]# awk -F: '$NF == "/bin/bash"' /etc/passwd
[root@qfedu.com ~]# awk -F: '$1 == "alice"' /etc/passwd
[root@qfedu.com ~]# awk -F: '$1 ~ /alic/ ' /etc/passwd
[root@qfedu.com ~]# awk -F: '$1 !~ /alic/ ' /etc/passwd
[root@qfedu.com ~]# df -P | grep  '/' |awk '$4 > 25000'
```

八、AWK 变量

1、AWK 内置变量

变量	描述
$n	当前记录的第n个字段，字段间由FS分隔
$0	完整的输入记录
ARGC	命令行参数的数目
ARGIND	命令行中当前文件的位置(从0开始算)
ARGV	包含命令行参数的数组
CONVFMT	数字转换格式(默认值为%.6g)ENVIRON环境变量关联数组
ERRNO	最后一个系统错误的描述
FIELDWIDTHS	字段宽度列表(用空格键分隔)
FILENAME	当前文件名
FNR	各文件分别计数的行号
FS	字段分隔符(默认是任何空格)
IGNORECASE	如果为真，则进行忽略大小写的匹配
NF	一条记录的字段的数目
NR	已经读出的记录数，就是行号，从1开始
OFMT	数字的输出格式(默认值是%.6g)
OFS	输出记录分隔符（输出换行符），输出时用指定的符号代替换行符
ORS	输出记录分隔符(默认值是一个换行符)
RLENGTH	由match函数所匹配的字符串的长度
RS	记录分隔符(默认是一个换行符)
RSTART	由match函数所匹配的字符串的第一个位置
SUBSEP	数组下标分隔符(默认值是/034)

- 实例

```
[root@qfedu.com ~]# awk -F: '{print $0}' /etc/passwd                          # $0
[root@qfedu.com ~]# awk -F: '{print NR, $0}' /etc/passwd /etc/hosts           # NR
[root@qfedu.com ~]# awk -F: '{print FNR, $0}' /etc/passwd /etc/hosts          #
FNR
[root@qfedu.com ~]# awk -F: '{print $0,NF}' /etc/passwd                       # NF
[root@qfedu.com ~]# awk -F: '/alice/{print $1, $3}' /etc/passwd               # FS
[root@qfedu.com ~]# awk -F'[ :\t]' '{print $1,$2,$3}' /etc/passwd
[root@qfedu.com ~]# awk 'BEGIN{FS=":"} {print $1,$3}' /etc/passwd
[root@qfedu.com ~]# awk -F: '/alice/{print $1,$2,$3,$4}' /etc/passwd          #
OFS
[root@qfedu.com ~]# awk 'BEGIN{FS=":"; OFS="+++"} /^root/{print $1,$2,$3,$4}'
passwd
[root@qfedu.com ~]# awk -F: 'BEGIN{RS=" "} {print $0}' a.txt                  # RS
[root@qfedu.com ~]# awk -F: 'BEGIN{ORS=""} {print $0}' passwd                 #
ORS
```

- 字段分隔符: FS OFS 默认空格或制表符
- 记录分隔符: RS ORS 默认换行符

```
# ORS 默认输出一条记录应该回车，加了一个空格
[root@qfedu.com ~]# awk 'BEGIN{ORS=" "} {print $0}' /etc/passwd
# 将文件每一行合并为一行

[root@qfedu.com ~]# head -1 /etc/passwd > passwd1
[root@qfedu.com ~]# cat passwd1
root:x:0:0:root:/root:/bin/bash
[root@qfedu.com ~]# awk 'BEGIN{RS=":"} {print $0}' passwd1
root
x
0
0
root
/root
/bin/bash

[root@qfedu.com ~]# awk 'BEGIN{RS=":"} {print $0}' passwd1 |grep -v '^$' >
passwd2

# 输出顺序号 NR, 匹配文本行号
[root@localhost ~]# awk '{print NR,FNR,$1,$2,$3}' /etc/passwd
1 1 root:x:0:0:root:/root:/bin/bash
2 2 bin:x:1:1:bin:/bin:/sbin/nologin
3 3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

# 指定输出分割符
[root@localhost ~]# awk '{print $1,$2,$5}' OFS=" $ "  /etc/passwd
root:x:0:0:root:/root:/bin/bash $  $
bin:x:1:1:bin:/bin:/sbin/nologin $  $
daemon:x:2:2:daemon:/sbin:/sbin/nologin $  $
adm:x:3:4:adm:/var/adm:/sbin/nologin $  $
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin $  $
```

2、AWK 自定义变量(区分字符大小写)

```
在'{...}'前，需要用-v var=value：awk -v var=value '{...}'
在program 中直接定义：awk '{var=vlue}'
```

3、AWK 使用外部变量

1、在双引号的情况下使用

```
[root@qfedu.com ~]# var="bash"
[root@qfedu.com ~]# echo "unix script" |awk "gsub(/unix/,\"$var\")"
bash script
```

2、在单引号的情况下使用

```
[root@qfedu.com ~]# var="bash"
[root@qfedu.com ~]# echo "unix script" |awk 'gsub(/unix/,"'"$var"'")'
bash script

[root@qfedu.com ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/cl-root  2.8T  246G  2.5T   9% /
tmpfs                 24G   20K   24G   1% /dev/shm
/dev/sda2           1014M  194M  821M  20% /boot

[root@qfedu.com ~]# df -h |awk '{ if(int($5)>5){print $6":"$5} }'
/:9%
/boot:20%

[root@qfedu.com ~]# i=10
[root@qfedu.com ~]# df -h |awk '{ if(int($5)>'''$i'''){print $6":"$5} }'
/boot:20%
```

3、AWK 参数-v（建议）

```
[root@qfedu.com ~]# echo "unix script" |awk -v var="bash" 'gsub(/unix/,var)'
bash script

[root@qfedu.com ~]# awk -v user=root  -F: '$1 == user' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

4、AWK 使用练习

```
[root@qfedu.com ~]# awk -F ':' '$3=="0"' /etc/passwd                    # 以冒号为分隔符，打印第三段是文本0的行，双引号代表字符，没有双引号代表数字
[root@qfedu.com ~]# awk -F ':' '$3>="500"' /etc/passwd                  # 以冒号为分隔符，打印第三段大于等于字符串500的行
[root@qfedu.com ~]# awk -F ':' '$3>=500' /etc/passwd                    # 以冒号为分隔符，打印第三段大于等于数字500的行
[root@qfedu.com ~]# awk -F ':' '$7!="/sbin/nologin"' /etc/passwd        # 以冒号为分隔符，打印第七段不为/sbin/nologin的行
[root@qfedu.com ~]# awk -F ':' '$3<$4' /etc/passwd                      # 以冒号为分隔符，打印第三段小于第四段的行
[root@qfedu.com ~]# awk -F ':' '$3>"5" && $3<"7"' /etc/passwd           # 以冒号为分隔符，打印第三段大于字符5且第三段小于字符7的行
```

```
[root@qfedu.com ~]# awk -F ':' '$3>1000 || $7=="/bin/bash"' /etc/passwd # 以冒号为分隔符，打印第三段大于1000或者第七段等于/bin/bash的行
[root@qfedu.com ~]# head -5 /etc/passwd |awk -F ':' '{OFS="#"} {print $1,$3,$4}‘ # 以冒号为分隔符，打印每行第1、3、4段，并以井号间隔
[root@qfedu.com ~]# awk -F ':' '{OFS="#"} {if ($3>1000) {print $1,$2,$3,$4}}' /etc/passwd # 以冒号为分隔符，如果第三段大于1000则打印第1、3、4段，并以井号间隔
[root@qfedu.com ~]# head -n3 /etc/passwd | awk -F ':' '{print NF}‘      # 以冒号为分隔符，逐行打印该行列数
[root@qfedu.com ~]# head -n3 /etc/passwd | awk -F ':' '{print NR}‘      # 以冒号为分隔符，逐行打印该行行数
[root@qfedu.com ~]# awk 'NR>40' /etc/passwd                             # 打印行数大于40的行
[root@qfedu.com ~]# awk -F ':' 'NR<20 && $1 ~ /roo/' /etc/passwd        # 打印行数小于20并且第一段包含roo的行
[root@qfedu.com ~]# head -n 3 /etc/passwd |awk -F ':' '$1="root"‘       # 以冒号为分隔符，给第一段赋值root，然后打印每一行
[root@qfedu.com ~]# awk -F ':' '{(tot=tot+$3)}; END {print tot}' /etc/passwd # 逐行做完tot=tot+3的运算，最后打印出tot的值
[root@qfedu.com ~]# awk -F ':' '{if ($1=="root") {print $0}}' /etc/passwd   # 如果第一段是root，打印该行

[root@qfedu.com ~]# awk -v FS=':' '{print $1,FS,$3}' /etc/passwd
[root@qfedu.com ~]# awk -F: '{print $1,$3,$7}' /etc/passwd
[root@qfedu.com ~]# awk -v FS=‘:’ -v OFS=‘:’ '{print $1,$3,$7}' /etc/passwd
[root@qfedu.com ~]# awk -v RS=' ' ‘{print }’ /etc/passwd
[root@qfedu.com ~]# awk -v RS="[[:space:]/=]" '{print }' /etc/fstab |sort
[root@qfedu.com ~]# awk -v RS=' ' -v ORS='###'‘{print }’ /etc/passwd
[root@qfedu.com ~]# awk -F:  ‘{print NF}’ /etc/fstab, 引用内置变量不用$
[root@qfedu.com ~]# awk -F: '{print $(NF-1)}' /etc/passwd
[root@qfedu.com ~]# awk '{print NR}' /etc/fstab ; awk 'END{print NR}' /etc/fstab
[root@qfedu.com ~]# awk '{print FNR}' /etc/fstab /etc/inittab
[root@qfedu.com ~]# awk '{print FNR}' /etc/fstab /etc/inittab
[root@qfedu.com ~]# awk '{print FILENAME}' /etc/fstab
[root@qfedu.com ~]# awk '{print ARGC}' /etc/fstab /etc/inittab
[root@qfedu.com ~]# awk ‘BEGIN {print ARGC}’ /etc/fstab /etc/inittab
[root@qfedu.com ~]# awk ‘BEGIN {print ARGV[0]}’ /etc/fstab   /etc/inittab
[root@qfedu.com ~]# awk ‘BEGIN {print ARGV[1]}’ /etc/fstab  /etc/inittab
[root@qfedu.com ~]# awk -v test='hello gawk' '{print test}' /etc/fstab
[root@qfedu.com ~]# awk -v test='hello gawk' 'BEGIN{print test}'
[root@qfedu.com ~]# awk 'BEGIN{test="hello,gawk";print test}'
[root@qfedu.com ~]# awk -F:‘{sex=“male”;print $1,sex,age;age=18}’ /etc/passwd
[root@qfedu.com ~]# awk -F: '{sex="male";age=18;print $1,sex,age}' /etc/passwd
[root@qfedu.com ~]# echo "{print script,\$1,\$2}"  > awkscript
[root@qfedu.com ~]# awk -F: -f awkscript script=“awk” /etc/passwd
```

九、AWK 脚本

1、AWK 脚本定义格式

```
格式1：
BEGIN{} pattern{} END{}

格式2：
#!/bin/awk  -f
#add 'x'  right
BEGIN{} pattern{} END{}
```

关于awk 脚本，需要注意两个关键词BEGIN和END。

- BEGIN{ 这里面放的是执行前的语句 }
- END {这里面放的是处理完所有的行后要执行的语句 }
- {这里面放的是处理每一行时要执行的语句}
- 格式1假设为f1.awk文件，格式2假设为f2.awk文件

```
awk [-v var=value] f1.awk [file]
f2.awk [-v var=value] [var1=value1] [file]
```

- awk [-v var=value] f1.awk [file]，把处理阶段放到一个文件而已，展开后就是普通的awk语句。
- f2.awk [-v var=value] [var1=value1] [file] 中 [-v var=value] 是在BEGIN之前设置的变量值，[var1=value1]是在BEGIN过程之后进行的，也就是说直到首行输入完成后，这个变量才可用。

2、AWK 脚本练习

1、AWK 脚本实例1

1、创建一个文件（学生成绩表）

```
[root@qfedu.com ~]# cat score.txt
Marry   2143 78 84 77
Jack    2321 66 78 45
Tom     2122 48 77 71
Mike    2537 87 97 95
Bob     2415 40 57 62
```

2、定义 awk 脚本

```
[root@qfedu.com ~]# cat cal.awk
#!/bin/awk -f
#运行前
BEGIN {
    math = 0
    english = 0
    computer = 0

    printf "NAME    NO.   MATH  ENGLISH  COMPUTER   TOTAL\n"
    printf "---------------------------------------------\n"
}
#运行中
{
    math+=$3
    english+=$4
    computer+=$5
    printf "%-6s %-6s %4d %8d %8d %8d\n", $1, $2, $3,$4,$5, $3+$4+$5
}
#运行后
END {
    printf "---------------------------------------------\n"
    printf "  TOTAL:%10d %8d %8d \n", math, english, computer
    printf "AVERAGE:%10.2f %8.2f %8.2f\n", math/NR, english/NR, computer/NR
}

[root@qfedu.com ~]# awk -f cal.awk score.txt
NAME    NO.   MATH  ENGLISH  COMPUTER   TOTAL
```

```
---------------------------------------------
Marry   2143     78       84       77       239
Jack    2321     66       78       45       189
Tom     2122     48       77       71       196
Mike    2537     87       97       95       279
Bob     2415     40       57       62       159
---------------------------------------------
  TOTAL:        319      393      350
AVERAGE:      63.80    78.60    70.00
```

2、AWK 脚本实例2

```
cat f1.awk
    {if($3>=1000)print $1,$3}
awk -F: -f f1.awk /etc/passwd

cat f2.awk
    #!/bin/awk -f
    # this is a awk script
    {if($3>=1000)print $1,$3}
    # chmod +x f2.awk
f2.awk -F: /etc/passwd

cat test.awk
    #!/bin/awk -f
    {if($3 >=min && $3<=max)print $1,$3}
    #chmod +x test.awk
test.awk -F: min=100 max=200 /etc/passwd
```

十、AWK 条件语句与循环

1、AWK 条件语句

1、IF 语句

1、IF 条件语句语法格式

```
if (condition)
    action
```

- 使用花括号语法格式

```
if (condition)
{
    action1;
    action2;
    ...
}

{if(表达式){语句1;语句2;...}}
```

2、IF 语句实例

1、判断数字是奇数还是偶数

```
[root@qfedu.com ~]# awk 'BEGIN {num = 10; if (num % 2 == 0) printf "%d 是偶数\n", num }'
10 是偶数
```

2、判断 root 是不是 administrator

```
[root@qfedu.com ~]# awk -F: '{if($3==0) {print $1 " is administrator."}}' /etc/passwd
root is administrator.
```

3、 统计系统用户数

```
[root@qfedu.com ~]# awk -F: '{if($3>0 && $3<1000){count++;}}  END{print count}' /etc/passwd
20
```

2、IF - ELSE 语句

1、IF - ELSE 条件语句语法格式

```
if (condition)
    action1
else
    action2
```

- 使用花括号语法格式

```
{if (condition)
{
    action1;
    action2;
    ...
}
else
{
    action1;
    action2;
    ...
}}

{if(表达式){语句1;语句2;...}else{语句1;语句2;...}}
```

2、IF - ELSE 语句实例

1、判断数字是奇数还是偶数

```
[root@qfedu.com ~]# awk 'BEGIN {
    num = 11;
    if (num % 2 == 0) printf "%d 是偶数\n", num;
    else printf "%d 是奇数\n", num
}'

11 是奇数
```

2、判断用户为root就打印用户名否则打印shell类型

```
[root@qfedu.com ~]# awk -F: '{if($3==0){print $1} else {print $7}}' /etc/passwd
root
/sbin/nologin
/sbin/nologin
/sbin/nologin
/sbin/nologin
/bin/sync
/sbin/shutdown
```

3、统计管理员数量和系统用户数量

```
[root@qfedu.com ~]# awk -F: '{if($3==0){count++} else{i++}} END{print "管理员个数: "count ; print "系统用户数: "i}' /etc/passwd
管理员个数: 1
系统用户数: 20
```

3、IF - ELSE - IF 语句

1、IF - ELSE - IF条件语句语法格式

```
{if (condition1)
{
    action1;
    action2;
    ...
}
else if (condition2)
{
    action1;
    action2;
    ...
}
else if (condition3)
{
    action1;
    action2;
    ...
}
else
{
    action1;
    action2;
    ...
}}

{if(表达式1){语句1;语句2; ...} else if(表达式2){语句1;语句2; ...} else if(表达式3){语句1;语句2; ...} else{语句1;语句2; ...} }
```

2、IF - ELSE - IF 语句实例

1、多级判断结果

```
[root@qfedu.com ~]# awk 'BEGIN {
a=30;
if (a==10)
  print "a = 10";
else if (a == 20)
  print "a = 20";
else if (a == 30)
  print "a = 30";
}'

a = 30
```

2、统计管理员，系统，普通用户数量

```
[root@qfedu.com ~]# awk -F: '{if($3==0){i++} else if($3>999){k++} else{j++}}
END{print i; print k; print j}' /etc/passwd
1

20

[root@qfedu.com ~]# awk -F: '{if($3==0){i++} else if($3>999){k++} else{j++}}
END{print "管理员个数: "i; print "普通用个数: "k; print "系统用户: "j}' /etc/passwd
管理员个数: 1
普通用个数:
系统用户: 20
```

2、AWK 循环

1、For 循环

1、For 循环的语法

```
for(variable addignment; condition; iteration peocess)
{
    statement1
    statement2
    ...
}
```

- for 语句首先执行初始化动作(initialisation)，然后再检查条件(condition)。如果条件为真，则执行动作(action)，然后执行递增(increment)或者递减(decrement)操作。只要条件为 true 循环就会一直执行。每次循环结束都会进条件检查，若条件为 false 则结束循环。

2、For 循环实例

1、For 循环输出数字 1 至 5

```
[root@qfedu.com ~]# awk 'BEGIN { for (i = 1; i <= 5; ++i) print i }'
1
2
3
4
5
```

2、For 循环输出数字 1 至 9 的奇数之和

```
[root@qfedu.com ~]# echo "hello" | awk '
{
    total = 0
    for(i=1; i<10; i++)
    {
        if(i % 2 == 0)
        {
            continue
        }
        total = total + i
    }
    print "total=", total
}'
```

3、将每行打印10次

```
[root@qfedu.com ~]# awk -F: '{ for(i=1;i<=10;i++) {print $0} }' /etc/passwd
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

4、分别打印每行的每列

```
[root@qfedu.com ~]# awk -F: '{ for(i=1;i<=NF;i++) {print $i} }' /etc/passwd
root
x
0
0
root
/root
/bin/bash
bin
x
1
1
bin
/bin
/sbin/nologin
```

2、While 循环

1、While 循环的语法

```
while(condition)
{
    statement1
    statement2
    ...
}
```

While 循环首先检查条件 condition 是否为 true ，若条件为 true 则执行动作 action。此过程一直重复直到条件 condition 为 flase 才停止。

2、While 循环实例

1、While 循环输出数字 1 到 5

```
[root@qfedu.com ~]# awk 'BEGIN {i = 1; while (i < 6) { print i; ++i } }'
1
2
3
4
5

[root@qfedu.com ~]# awk 'BEGIN{ i=1; while(i<=5){print i; i++}  }'
1
2
3
4
5
```

2、While 循环输出数字 1 至 9 的奇数之和

```
[root@qfedu.com ~]# echo "hello" | awk '
{
    total = 0
    i = 1
    while(i < 10)
    {
        if(i % 2 == 0)
        {
            i++
            continue
        }
        total = total + i
        i++
    }
    print "total=", total
}'
```

3、打印第一行的前七列

```
[root@qfedu.com ~]# awk -F: '/^root/{i=1; while(i<=7){print $i; i++}}'
/etc/passwd
root
x
0
0
root
/root
/bin/bash
```

4、分别打印每行的每列

```
[root@qfedu.com ~]# awk  '{i=1; while(i<=NF){print $i; i++}}' /etc/hosts
127.0.0.1
localhost
localhost.localdomain
localhost4
localhost4.localdomain4
::1
localhost
localhost.localdomain
localhost6
localhost6.localdomain6
```

5、将每行打印10次

```
[root@qfedu.com ~]# awk -F: '{i=1; while(i<=10) {print $0;  i++}}' /etc/passwd
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
```


3、Break 结束循环

1、Break 结束循环实例

- break[n]：当第n次循环到来后，结束整个循环，n=0就是指本次循环
- 当计算的和大于 50 的时候使用 break 结束循环：

```
[root@qfedu.com ~]# awk 'BEGIN {
   sum = 0; for (i = 0; i < 20; ++i) {
      sum += i; if (sum > 50) break; else print "Sum =", sum
   }
}'

Sum = 0
Sum = 1
Sum = 3
Sum = 6
Sum = 10
```

```
Sum = 15
Sum = 21
Sum = 28
Sum = 36
Sum = 45
```

4、Continue 跳出本次循环

- Continue 语句用于在循环体内部结束本次循环，从而直接进入下一次循环迭代。
- Continue[n]：满足条件后，直接进行第n次循环，本次循环不在进行，n=0也就是提前结束本次循环而直接进入下一轮
- 输出 1 到 20 之间的偶数：

```
[root@qfedu.com ~]# awk 'BEGIN {for (i = 1; i <= 20; ++i) {if (i % 2 == 0) print i ; else continue} }'
2
4
6
8
10
12
14
16
18
20
```

5、Exit 结束脚本程序

- Exit 用于结束脚本程序的执行。
- 该函数接受一个整数作为参数表示 AWK 进程结束状态。 如果没有提供该参数，其默认状态为 0。
- 当和大于 50 时结束 AWK 程序。

```
[root@qfedu.com ~]# awk 'BEGIN {
   sum = 0; for (i = 0; i < 20; ++i) {
      sum += i; if (sum > 50) exit(10); else print "Sum =", sum
   }
}'

Sum = 0
Sum = 1
Sum = 3
Sum = 6
Sum = 10
Sum = 15
Sum = 21
Sum = 28
Sum = 36
Sum = 45
```

- 检查脚本执行后的返回状态

```
[root@qfedu.com ~]# echo $?
10
```

6、Next 停止处理

- next：提前结束对本行的处理动作而直接进入下一行处理

```
[root@qfedu.com ~]# awk -F: '{if($3%2!=0) next; print $1,$3}' /etc/passwd
```

十一、AWK 数组

- AWK 可以使用关联数组这种数据结构，索引可以是数字或字符串。
- AWK关联数 组也不需要提前声明其大小，因为它在运行时可以自动的增大或减小。

1、数组语法格式

```
array_name[index]=value
```

- array_name：数组的名称
- index：数组索引
- value：数组中元素所赋予的值

2、创建数组

- 定义了一个站点(sites)数组，该数组的索引为网站英文简称，值为网站访问地址。

```
[root@qfedu.com ~]# awk 'BEGIN {
sites["qfedu"]="www.qfedu.com";
sites["google"]="www.google.com"
print sites["qfedu"] "\n" sites["google"]
}'
www.qfedu.com
www.google.com
```

3、访问数组元素

1、访问数组元素语法格式

```
array_name[index]
```

2、访问数组元素

```
[root@qfedu.com ~]# awk -F: '{username[++i]=$1} END{print username[1]}'
/etc/passwd
root
[root@qfedu.com ~]# awk -F: '{username[i++]=$1} END{print username[1]}'
/etc/passwd
bin
[root@qfedu.com ~]# awk -F: '{username[i++]=$1} END{print username[0]}'
/etc/passwd
root
```

3、按元数个数遍历

```
[root@qfedu.com ~]# awk -F: '{username[x++]=$1} END{for(i=0;i<x;i++) print
i,username[i]}' /etc/passwd
0 root
```

```
1 bin
2 daemon
3 adm
4 lp
5 sync
6 shutdown
7 halt

[root@qfedu.com ~]# awk -F: '{username[++x]=$1} END{for(i=1;i<=x;i++) print
i,username[i]}' /etc/passwd
1 root
2 bin
3 daemon
4 adm
5 lp
6 sync
7 shutdown
8 halt
9 mail
10 operator
```

4、按索引遍历

```
[root@qfedu.com ~]# awk -F: '{username[x++]=$1} END{for(i in username) {print
i,username[i]} }' /etc/passwd
17 sshd
4 lp
18 postfix
5 sync
19 mysql
6 shutdown
7 halt
8 mail
9 operator
10 games
20 nginx

[root@qfedu.com ~]# awk -F: '{username[++x]=$1} END{for(i in username) {print
i,username[i]} }' /etc/passwd
17 tss
4 adm
18 sshd
5 lp
19 postfix
6 sync
7 shutdown
8 halt
9 mail
10 operator
20 mysql
11 games

# 注：变量i是索引
```

4、删除数组元素

1、删除数组元素语法格式

- 使用 delete 语句来删除数组元素，语法格式如下：

```
delete array_name[index]
```

2、删除数组元素

- 删除数组中的 google 元素（删除命令没有输出）：

```
[root@qfedu.com ~]# awk 'BEGIN {
sites["qfedu"]="www.qfedu.com";
sites["google"]="www.google.com"
delete sites["google"];
print fruits["google"]
}'
```

5、多维数组

- AWK 本身不支持多维数组，不过我们可以很容易地使用一维数组模拟实现多维数组。
- 如下示例为一个 3x3 的三维数组：

```
100 200 300
400 500 600
700 800 900
```

实例中，array0 存储 100，array0 存储 200 ，依次类推。

要在 array0 处存储 100, 使用： array["0,0"] = 100。

使用 0,0 作为索引，这并不是两个索引值，而是一个字符串索引 0,0。

1、模拟二维数组

```
[root@qfedu.com ~]# awk 'BEGIN {
array["0,0"] = 100;
array["0,1"] = 200;
array["0,2"] = 300;
array["1,0"] = 400;
array["1,1"] = 500;
array["1,2"] = 600;
# 输出数组元素
print "array[0,0] = " array["0,0"];
print "array[0,1] = " array["0,1"];
print "array[0,2] = " array["0,2"];
print "array[1,0] = " array["1,0"];
print "array[1,1] = " array["1,1"];
print "array[1,2] = " array["1,2"];
}'

array[0,0] = 100
array[0,1] = 200
array[0,2] = 300
array[1,0] = 400
array[1,1] = 500
array[1,2] = 600
```

- 在数组上可以执行很多操作，比如，使用 asort 完成数组元素的排序，或者使用 asorti 实现数组索引的排序等等。

十二、AWK 的函数

- awk的函数有许多，除了系统自带的内建函数还有就是用户自定义的函数，
- AWK 常用的函数

```
rand()             # 返回0 和1 之间一个随机数
srand()            # 生成随机数种子
int()              # 取整数
length([s])        # 返回指定字符串的长度
sub(r,s,[t])       # 对t字符串进行搜索，r表示的模式匹配的内容，并将第一个匹配的内容替换为s
gsub(r,s,[t])      # 对t字符串进行搜索，r表示的模式匹配的内容，并全部替换为s所表示的内容
split(s,array,[r])# 以r为分隔符，切割字符串s，并将切割后的结果保存至array 所表示的数组中，第一个索引值为1，第二个索引值为2,…也就是说awk的数组下标是从1开始编的
substr(s,i,[n]):  # 从s所表示的字符串中取子串，取法：从i表示的位置开始，取n个字符。
systime()          # 取当前系统时间，结果形式为时间戳。
system()           # 调用shell中的命令。空格是awk中的字符串连接符，如果system中需要使用awk中的变量可以使用空格分隔，或者说除了awk的变量外其他一律用""引用起来。
```

- 自定义函数语法格式

```
function fname ( arg1,arg2 , ... ) {
statements
return expr
}
```

- fname为函数名，
- arg1...为函数的参数，
- statements是动作语言，
- return expr为由 statements 的结果从而决定最终函数所显示的内容。
- 自定义函数示例

```
[root@qfedu.com ~]# cat fun.awk
    function max(v1,v2) {
        v1>v2?var=v1:var=v2
        return var
    }
      BEGIN{a=3;b=2;print max(a,b)}
awk -f fun.awk
```

十三、AWK 应用实战

1、AWK 日志分析统计

```
# 统计一个时间范围内访(pv)问量
[root@qfedu.com ~]# grep '01/Sep/2017' sz.mobiletrain.org.log | wc -l
[root@qfedu.com ~]# awk '$4>="[05/Sep/2017:08:00:00" && $4<="
[05/Sep/2017:09:00:00" {print $0}' sz.mobiletrain.org.log | wc -l

# 统计一个时间范围内访问量前10的ip
```

```
[root@qfedu.com ~]# grep '05/Sep/2017' cd.mobiletrain.org.log | awk '{ips[$1]++}
END{ for(i in ips){print i ,ips[i]}}' | sort -k2 -rn | head

[root@qfedu.com ~]# awk '/05\/Sep\/2017/ {ips[$1]++} END{ for(i in ips){print i
,ips[i]}}' cd.mobiletrain.org.log | sort -k2 -rn | head

# 统计一个时间范围内访问前10的页面
[root@qfedu.com ~]# grep '05/Sep/2017' cd.mobiletrain.org.log | awk '{
urls[$7]++} END{for(i in urls){print urls[i],i}}'|sort -k1 -rn |head

[root@qfedu.com ~]# awk '/05\/Sep\/2017/ { urls[$7]++} END{for(i in urls){print
urls[i],i}}' cd.mobiletrain.org.log |sort -k1 -rn |head

# 统计一个时间范围内访问大于100次的ip
[root@qfedu.com ~]# grep '05/Sep/2017' sz.mobiletrain.org.log | awk '{ ips[$1]++
} END{for (i in ips) {if(ips[i]>100) {print i,ips[i]}}}'
[root@qfedu.com ~]# awk '/05\/Sep\/2017/ { ips[$1]++ } END{for (i in ips)
{if(ips[i]>100) {print i,ips[i]}}}'  sz.mobiletrain.org.log

# 统计一个时间范围访问前10的url
[root@qfedu.com ~]# awk '/05\/Sep\/2017/{urls[$7]++} END{for(i in urls){print
i,urls[i]}}' sz.mobiletrain.org.log |sort -k2rn |head

# 统计一个时间范围内url访问总大小前10的url
[root@qfedu.com ~]# grep '05/Sep/2017' sz.mobiletrain.org.log | awk '{
urls[$7]++; size[$7]+=$10} END { for(i in urls ) { print urls[i],size[i],i}}'|
sort -k1 -rn |head

[root@qfedu.com ~]# awk '/05\/Sep\/2017/{size[$7]+=$10} END{for(i in size){print
i,size[i]}}' sz.mobiletrain.org.log |sort -k2rn |head

# 统计一个时间范围内每个访问ip的状态码数量
[root@qfedu.com ~]# grep '05/Sep/2017' sz.mobiletrain.org.log | awk '{ ip_code[
$1" "$9]++} END{ for(i in ip_code) {print i,ip_code[i]}}'| sort -k1 -rn | head

[root@qfedu.com ~]# awk '/05\/Sep\/2017/{ip_code[$1" "$9]++} END{for(i in
ip_code){print i,ip_code[i]}}' sz.mobiletrain.org.log |sort -k1rn |head

# 统计一个时间范围内访问状态码时404的ip
[root@qfedu.com ~]# grep '05/Sep/2017' sz.mobiletrain.org.log | awk
'{if($9="404"){ip_code[$1" "$9]++}} END{for(i in ip_code){print
i,ip_code[i]}}'|sort -k3 -rn

[root@qfedu.com ~]# awk '$4>="[05/Sep/2017:08:00:00" && $4<="
[05/Sep/2017:09:00:00" {if($9="404"){ip_code[$1" "$9]++}} END{for(i in ip_code)
{print i,ip_code[i]}}' sz.mobiletrain.org.log |sort -k3 -rn

[root@qfedu.com ~]# awk '/05\/Sep\/2017/{if($9=="404"){ip_code[$1" "$9]++}}
END{for(i in ip_code){print i,ip_code[i]}}' sz.mobiletrain.org.log |sort -k3rn
|head

# 统计前一分钟的访问量
[root@qfedu.com ~]# date=$(date -d '1 minute' +%d/%b/%Y:%H:%M); awk -v
date=$date '$0 ~ date {i++} END{print i}' sz.mobiletrain.org.log

# 统计一个时间范围内出现的各种状态码
[root@qfedu.com ~]# grep '05/Sep/2017' sz.mobiletrain.org.log | awk
'{code[$9]++} END{for(i in code){print i, code[i]}}'
```

```
[root@qfedu.com ~]# awk '/05\/Sep\/2017/ {code[$9]++}END{for(i in code){print i
,code[i]}}' sz.mobiletrain.org.log
[root@qfedu.com ~]# grep '05/Sep/2017' sz.mobiletrain.org.log | awk
'{code[$9]++; total++} END{for(i in code){printf i" "; printf
code[i]"\t";printf"%.2f", code[i]/total*100;print"%"}}'

[root@qfedu.com ~]# awk '/05\/Sep\/2017/{code[$9]++; total++} END{for(i in code)
{printf i " "; printf code[i]"\t"; printf"%.2f",code[i]/total*100;print"%"}}'
sz.mobiletrain.org.log
```

2、AWK 统计 tcp 连接

```
[root@qfedu.com ~]# ss -an | grep ^tcp|awk '{tcp_connect_status[$2]++}END{for(i
in tcp_connect_status) {print i, tcp_connect_status[i]}}'
```

3、AWK 获取硬件信息

```
#!/bin/bash
yum install hdparm dmidecode pciutils -y
echo
echo "###### CPU #######"
echo
cat /proc/cpuinfo | grep "model name"| awk -F ":" '{print $2}'| uniq -f 1
cat /proc/cpuinfo | grep "cpu cores"| awk -F ":" '{print " CPU ="$2}'| uniq -f 1
echo
echo "###### Hard Disk ######"
echo
hdparm -i /dev/sda | grep -i "model"| awk -F"-" '{print $1}'| awk -F "=" '{print
$2}'
fdisk -l | grep "/dev/sda"| awk -F "," 'NR==1{print $1}'
echo
df -h
echo
echo "###### Memory ######"
echo
dmidecode -t memory | grep -i "maximum capacity"
dmidecode -t memory | grep -i "number of devices"
echo
dmidecode -t memory | grep -i "size"| awk -F ":" 'NR==1{print "        Capacity
1" $1":",$2}'
dmidecode -t memory | grep -i "speed"| awk 'NR==1'
dmidecode -t memory | grep -i "type:"| uniq -f 1
dmidecode -t memory | grep -i "size"| awk -F ":" 'NR==2{print "        Capacity
2" $1":",$2}'
dmidecode -t memory | grep -i "speed"| awk 'NR==2'
dmidecode -t memory | grep -i "type:"| uniq -f 1
dmidecode -t memory | grep -i "size"| awk -F ":" 'NR==3{print "        Capacity
3" $1":",$2}'
dmidecode -t memory | grep -i "speed"| awk 'NR==3'
dmidecode -t memory | grep -i "type:"| uniq -f 1
dmidecode -t memory | grep -i "size"| awk -F ":" 'NR==4{print "        Capacity
4" $1":",$2}'
dmidecode -t memory | grep -i "speed"| awk 'NR==4'
dmidecode -t memory | grep -i "type:"| uniq -f 1
echo
free -m
```

```
echo
echo "###### Mianboard ######"
echo
dmidecode -q | grep -i "product name"| awk -F":" 'NR==1{print "Server Model"
":",$2}'
dmidecode -q | grep -i "Manufacturer"| awk -F":" 'NR==1{print "Brand" ":",$2}'
dmidecode -q | grep -i "product name"| awk -F":" 'NR==2{print "Mainboard Model"
":",$2}'
echo
echo "###### Network Card ######"
echo
lspci | grep -i eth | awk 'NR==1'| awk -F ":" '{print $3}'
echo
echo "###### Operating System ######"
echo
cat /etc/issue | awk 'NR==1'
uname -r | awk '{print "kernel: "$1}'
echo
```

4、AWK 获取系统状态信息

```
#!/bin/bash
#show system information
PS3="Your choice is: "
os_check() {
        if [ -e /etc/redhat-release ]; then
                REDHAT=`cat /etc/redhat-release |cut -d' '  -f1`
        else
                DEBIAN=`cat /etc/issue |cut -d' ' -f1`
        fi

        if [ "$REDHAT" == "CentOS" -o "$REDHAT" == "Red" ]; then
                P_M=yum
        elif [ "$DEBIAN" == "Ubuntu" -o "$DEBIAN" == "ubutnu" ]; then
                P_M=apt-get
        else
                Operating system does not support.
                exit 1
        fi
}

if [ $LOGNAME != root ]; then
    echo "Please use the root account operation."
    exit 1
fi

if ! which vmstat &>/dev/null; then
        echo "vmstat command not found, now the install."
        sleep 1
        os_check
        $P_M install procps -y
        echo "-----------------------------------------------------------------
-----"
fi

which iostat &>/dev/null
if [ $? -ne 0 ]; then
```

```
        echo "iostat command not found, now the install."
        sleep 1
        os_check
        $P_M install sysstat -y
        echo "--------------------------------------------------------------------
-----"
fi

while true; do
    select input in cpu_load disk_load disk_use disk_inode mem_use tcp_status
cpu_top10 mem_top10 traffic quit; do
        case $input in
            cpu_load)
                #CPU利用率与负载
                echo "--------------------------------------"
                i=1
                while [[ $i -le 3 ]]; do
                    echo -e "\033[32m  参考值${i}\033[0m"
                    UTIL=`vmstat |awk '{if(NR==3)print 100-$15"%"}'`
                    USER=`vmstat |awk '{if(NR==3)print $13"%"}'`
                    SYS=`vmstat |awk '{if(NR==3)print $14"%"}'`
                    IOWAIT=`vmstat |awk '{if(NR==3)print $16"%"}'`
                    echo "Util: $UTIL"
                    echo "User use: $USER"
                    echo "System use: $SYS"
                    echo "I/O wait: $IOWAIT"
            let i++
                    sleep 1
                done
                echo "--------------------------------------"
                break
                ;;
            disk_load)
                #硬盘I/O负载
                echo "--------------------------------------"
                i=1
                while [[ $i -le 3 ]]; do
                    echo -e "\033[32m  参考值${i}\033[0m"
                    UTIL=`iostat -x -k |awk '/^[v|s]/{OFS=": ";print
$1,$NF"%"}'`
                    READ=`iostat -x -k |awk '/^[v|s]/{OFS=": ";print
$1,$6"KB"}'`
                    WRITE=`iostat -x -k |awk '/^[v|s]/{OFS=": ";print
$1,$7"KB"}'`
                    IOWAIT=`vmstat |awk '{if(NR==3)print $16"%"}'`
                    echo -e "Util:"
                    echo -e "${UTIL}"
                    echo -e "I/O Wait: $IOWAIT"
                    echo -e "Read/s:\n$READ"
                    echo -e "Write/s:\n$WRITE"
                    i=$(($i+1))
                    sleep 1
                done
                echo "--------------------------------------"
                break
                ;;
            disk_use)
                #硬盘利用率
```

```
                DISK_LOG=/tmp/disk_use.tmp
                DISK_TOTAL=`fdisk -l |awk '/^Disk.*bytes/ && /\/dev/{printf $2"
";printf "%d",$3;print "GB"}'`
                USE_RATE=`df -h |awk '/^\/dev/{print int($5)}'`
                for i in $USE_RATE; do
                    if [ $i -gt 90 ];then
                        PART=`df -h |awk '{if(int($5)=='''$i''') print $6}'`
                        echo "$PART = ${i}%" >> $DISK_LOG
                    fi
                done
                echo "---------------------------------------"
                echo -e "Disk total:\n${DISK_TOTAL}"
                if [ -f $DISK_LOG ]; then
                    echo "---------------------------------------"
                    cat $DISK_LOG
                    echo "---------------------------------------"
                    rm -f $DISK_LOG
                else
                    echo "---------------------------------------"
                    echo "Disk use rate no than 90% of the partition."
                    echo "---------------------------------------"
                fi
                break
                ;;
            disk_inode)
                #硬盘inode利用率
                INODE_LOG=/tmp/inode_use.tmp
                INODE_USE=`df -i |awk '/^\/dev/{print int($5)}'`
                for i in $INODE_USE; do
                    if [ $i -gt 90 ]; then
                        PART=`df -h |awk '{if(int($5)=='''$i''') print $6}'`
                        echo "$PART = ${i}%" >> $INODE_LOG
                    fi
                done
                if [ -f $INODE_LOG ]; then
                    echo "---------------------------------------"
            cat $INODE_LOG
                    rm -f $INODE_LOG
                else
                    echo "---------------------------------------"
                    echo "Inode use rate no than 90% of the partition."
                    echo "---------------------------------------"
                fi
                break
                ;;
            mem_use)
                #内存利用率
                echo "---------------------------------------"
                MEM_TOTAL=`free -m |awk '{if(NR==2)printf
"%.1f",$2/1024}END{print "G"}'`
                USE=`free -m |awk '{if(NR==2) printf "%.1f",$3/1024}END{print
"G"}'`
                FREE=`free -m |awk '{if(NR==2) printf "%.1f",$4/1024}END{print
"G"}'`
                CACHE=`free -m |awk '{if(NR==2) printf "%.1f",$6/1024}END{print
"G"}'`
                echo -e "Total: $MEM_TOTAL"
                echo -e "Use: $USE"
```

```
                echo -e "Free: $FREE"
                echo -e "Cache: $CACHE"
                echo "----------------------------------------"
                break
                ;;
            tcp_status)
                #网络连接状态
                echo "----------------------------------------"
                COUNT=`ss -ant |awk '!/State/{status[$1]++}END{for(i in status)
print i,status[i]}'`
                echo -e "TCP connection status:\n$COUNT"
                echo "----------------------------------------"
                ;;
            cpu_top10)
                #占用CPU高的前10个进程
                echo "----------------------------------------"
                CPU_LOG=/tmp/cpu_top.tmp
                i=1
                while [[ $i -le 3 ]]; do
                    #ps aux |awk '{if($3>0.1)print "CPU: "$3"% --
>",$11,$12,$13,$14,$15,$16,"(PID:"$2")" |"sort -k2 -nr |head -n 10"}' > $CPU_LOG
                    ps aux |awk '{if($3>0.1){{printf "PID: "$2" CPU: "$3"% -->
"}for(i=11;i<=NF;i++)if(i==NF)printf $i"\n";else printf $i}}' |sort -k4 -nr
|head -10 > $CPU_LOG
                    #循环从11列（进程名）开始打印，如果i等于最后一行，就打印i的列并换行，
否则就打印i的列
                    if [[ -n `cat $CPU_LOG` ]]; then
                       echo -e "\033[32m  参考值${i}\033[0m"
                       cat $CPU_LOG
                       > $CPU_LOG
                    else
                        echo "No process using the CPU."
                        break
                    fi
                    let i++
                    sleep 1
                done
                echo "----------------------------------------"
                break
                ;;
            mem_top10)
                #占用内存高的前10个进程
                echo "----------------------------------------"
                MEM_LOG=/tmp/mem_top.tmp
                i=1
                while [[ $i -le 3 ]]; do
                    #ps aux |awk '{if($4>0.1)print "Memory: "$4"% --
>",$11,$12,$13,$14,$15,$16,"(PID:"$2")" |"sort -k2 -nr |head -n 10"}' > $MEM_LOG
                    ps aux |awk '{if($4>0.1){{printf "PID: "$2" Memory: "$4"% --
> "}for(i=11;i<=NF;i++)if(i==NF)printf $i"\n";else printf $i}}' |sort -k4 -nr
|head -10 > $MEM_LOG
                    if [[ -n `cat $MEM_LOG` ]]; then
                        echo -e "\033[32m  参考值${i}\033[0m"
                        cat $MEM_LOG
                        > $MEM_LOG
                    else
                        echo "No process using the Memory."
                        break
```

```
                    fi
                    i=$(($i+1))
                    sleep 1
                done
                echo "--------------------------------------"
                break
                ;;
            traffic)
                #查看网络流量
                while true; do
                    read -p "Please enter the network card name(eth[0-9] or
em[0-9] or team[0-9]): " eth
                    if [ `ifconfig |grep -c "\<$eth\>"` -eq 1 ]; then
                            break
                    else
                        echo "Input format error or Don't have the card name,
please input again."
                    fi
                done
                echo "--------------------------------------"
                echo -e " In ------ Out"
                i=1
                while [[ $i -le 3 ]]; do
                    #CentOS6和CentOS7 ifconfig输出进出流量信息位置不同:
            #CentOS6中RX与TX行号等于8
            #CentOS7中RX行号是5，TX行号是7

                    OLD_IN=`ifconfig $eth |awk -F'[: ]+' '/bytes/{if(NR==8)print
$4;else if(NR==5)print $6}'`
                    OLD_OUT=`ifconfig $eth |awk -F'[: ]+'
'/bytes/{if(NR==8)print $9;else if(NR==7)print $6}'`
                    sleep 1
                    NEW_IN=`ifconfig $eth |awk -F'[: ]+' '/bytes/{if(NR==8)print
$4;else if(NR==5)print $6}'`
                    NEW_OUT=`ifconfig $eth |awk -F'[: ]+'
'/bytes/{if(NR==8)print $9;else if(NR==7)print $6}'`

                    IN=`awk 'BEGIN{printf
"%.1f\n",'$((${NEW_IN}-${OLD_IN}))'/1024/128}'`
                    OUT=`awk 'BEGIN{printf
"%.1f\n",'$((${NEW_OUT}-${OLD_OUT}))'/1024/128}'`
                    echo "${IN}MB/s ${OUT}MB/s"

                    i=$(($i+1))
                    sleep 1
                done
                echo "--------------------------------------"
                break
                ;;
            quit)
                            exit 0
                            ;;
            *)
                    echo "--------------------------------------"
                    echo "Please enter the number."
                    echo "--------------------------------------"
                    break
                    ;;
```

```
        esac
    done
done
```

5、Linux 系统初始化脚本

```
#!/bin/bash

# get os version
RELEASEVER=$(rpm -q --qf "%{VERSION}" $(rpm -q --whatprovides redhat-release))

# configure yum source
cd /etc/yum.repos.d/
mkdir /etc/yum.repos.d/bak
mv /etc/yum.repos.d/*.repo /etc/yum.repos.d/bak
if [ $RELEASEVER == 6 ];then
    curl http://mirrors.163.com/.help/CentOS6-Base-163.repo > qf.repo
fi
if [ $RELEASEVER == 7 ];then
    curl http://mirrors.163.com/.help/CentOS7-Base-163.repo > qf.repo
fi
yum clean all
yum check-update

# install base rpm package
yum -y insall epel-release
yum -y install nc vim iftop iotop dstat tcpdump
yum -y install ipmitool bind-libs bind-utils
yum -y install libselinux-python ntpdate

# update rpm package include kernel
yum -y update
rm -rf /etc/yum.repos.d/CentOS*

# update ulimit configure
if [ $RELEASEVER == 6 ];then
    test -f /etc/security/limits.d/90-nproc.conf && rm -rf
/etc/security/limits.d/90-nproc.conf && touch /etc/security/limits.d/90-
nproc.conf
fi
if [ $RELEASEVER == 7 ];then
    test -f /etc/security/limits.d/20-nproc.conf && rm -rf
/etc/security/limits.d/20-nproc.conf && touch /etc/security/limits.d/20-
nproc.conf
fi

> /etc/security/limits.conf
cat >> /etc/security/limits.conf <<EOF
* soft nproc 65535
* hard nproc 65535
* soft nofile 65535
* hard nofile 65535
EOF

# set timezone
test -f /etc/localtime && rm -rf /etc/localtime
```

```
ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

# set LANG
if [ $RELEASEVER == 6 ];then
    sed -i 's@LANG=.*$@LANG="en_US.UTF-8"@g' /etc/sysconfig/i18n
fi
if [ $RELEASEVER == 7 ];then
    sed -i 's@LANG=.*$@LANG="en_US.UTF-8"@g' /etc/locale.conf
fi

# update time
if [ $RELEASEVER == 6 ];then
    /usr/sbin/ntpdate -b pool.ntp.org
    grep -q ntpdate /var/spool/cron/root
    if [ $? -ne 0 ]; then
        echo '* * * * * /usr/sbin/ntpdate pool.ntp.org > /dev/null 2>&1' >
/var/spool/cron/root;chmod 600 /var/spool/cron/root
    fi
    /etc/init.d/crond restart
fi

if [ $RELEASEVER == 7 ];then
    yum -y install chrony
    > /etc/chrony.conf
cat > /etc/chrony.conf << EOF
server pool.ntp.org iburst
stratumweight 0
driftfile /var/lib/chrony/drift
rtcsync
makestep 10 3
bindcmdaddress 127.0.0.1
bindcmdaddress ::1
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
noclientlog
logchange 0.5
logdir /var/log/chrony
EOF

    systemctl restart chronyd
    systemctl enable chronyd
fi

# clean iptables default rules
if [ $RELEASEVER == 6 ];then
    /sbin/iptables -F
    service iptables save
    chkconfig iptables off
fi
if [ $RELEASEVER == 7 ];then
    systemctl disable firewalld
fi

# disable unused service
chkconfig auditd off

# disable ipv6
```

```
cd /etc/modprobe.d/ && touch ipv6.conf
> /etc/modprobe.d/ipv6.conf
cat >> /etc/modprobe.d/ipv6.conf << EOF
alias net-pf-10 off
alias ipv6 off
EOF

# disable iptable nat moudule
cd /etc/modprobe.d/ && touch connectiontracking.conf
> /etc/modprobe.d/connectiontracking.conf
cat >> /etc/modprobe.d/connectiontracking.conf <<EOF
install nf_nat /bin/true
install xt_state  /bin/true
install iptable_nat /bin/true
install nf_conntrack /bin/true
install nf_defrag_ipv4   /bin/true
install nf_conntrack_ipv4 /bin/true
install nf_conntrack_ipv6  /bin/true
EOF

# disable SELINUX
setenforce 0
sed -i 's/^SELINUX=.*$/SELINUX=disabled/' /etc/selinux/config

# update record command
sed -i 's/^HISTSIZE=.*$/HISTSIZE=100000/' /etc/profile
grep -q 'HISTTIMEFORMAT' /etc/profile
if [[ $? -eq 0 ]]; then
    sed -i 's/^HISTTIMEFORMAT=.*$/HISTTIMEFORMAT="%F %T "/' /etc/profile
else
    echo 'HISTTIMEFORMAT="%F %T "' >> /etc/profile
fi

# install dsnmasq and update configure
yum -y install dnsmasq
> /etc/dnsmasq.conf
cat >> /etc/dnsmasq.conf<< EOF
listen-address=127.0.0.1
no-dhcp-interface=lo
log-queries
log-facility=/var/log/dnsmasq.log
all-servers
no-negcache
cache-size=1024
dns-forward-max=512
EOF

if [ $RELEASEVER == 6 ];then
    /etc/init.d/dnsmasq restart
fi

if [ $RELEASEVER == 7 ];then
    systemctl restart dnsmasq
    systemctl enable dnsmasq
fi

# update /etc/resolv.conf
> /etc/resolv.conf
```

```
cat >> /etc/resolv.conf<< EOF
options timeout:1
nameserver 127.0.0.1
nameserver 114.114.114.114
EOF

# update /etc/sysctl.conf
cat >> /etc/sysctl.conf<< EOF
net.ipv4.tcp_syncookies = 1
kernel.core_uses_pid=1
kernel.core_pattern=/tmp/core-%e-%p
fs.suid_dumpable=2
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_tw_recycle=0
net.ipv4.tcp_timestamps=1
EOF
sysctl -p
```

6、LAMP 终级部署

```
#!/bin/bash
# LAMP终极部署
cat <<-EOF
+-------------------------------------------------------------------+
| LAMP终极部署 V1.0 |
+-------------------------------------------------------------------+
| a. 部署Apache服务 |
| b. 部署php服务 |
| c. 部署Mysql服务 |
| d. 一键部署LAMP |
| q. 按q键退出程序 |
+-------------------------------------------------------------------+
EOF
# 安装Apache
install_Apache()
{
systemctl stop firewalld
systemctl disable firewalld
setenforce 0
sed -i '/^\bSELINUX\b/c SELINUX=disabled' /etc/selinux/config
mkdir /usr/local/apr &> /dev/null
mkdir /usr/local/apr-util &> /dev/null
mkdir /usr/local/apache &> /dev/null
cd /usr/local/src
echo "正在下载Apache服务，请稍等！！！"
wget http://archive.apache.org/dist/apr/apr-1.6.5.tar.gz &> /dev/null
wget http://archive.apache.org/dist/apr/apr-util-1.6.1.tar.gz &> /dev/null
wget http://mirrors.hust.edu.cn/apache/httpd/httpd-2.4.37.tar.gz &> /dev/null
if [ $? -eq 0 ]
then
echo "download success"
else
echo "download failed"
exit
fi
tar xf apr-1.6.5.tar.gz
tar xf apr-util-1.6.1.tar.gz
```

```
tar xf httpd-2.4.37.tar.gz
echo "正在安装所需的依赖包"
yum -y install gcc gcc-c++ openssl openssl-devel expat-devel &> /dev/null
if [ $? -eq 0 ]
then
echo "依赖包安装成功"
else
echo "依赖包安装失败"
exit
fi
cd /usr/local/src/apr-1.6.5/
echo "正在配置和编译安装apr，请喝口水稍等！！！"
./configure --prefix=/usr/local/apr/ &> /dev/null
make &> /dev/null && make install &>/dev/null
if [ $? -eq 0 ]
then
echo "apr installed"
else
echo "apr installed failed"
exit
fi
cd /usr/local/src/apr-util-1.6.1/
./configure --prefix=/usr/local/apr-util --with-apr=/usr/local/apr/ &> /dev/null
make &> /dev/null && make install &> /dev/null
if [ $? -eq 0 ]
then
echo "apr-util installed"
else
echo "apr-util installed failed"
exit
fi
cd /usr/local/src/httpd-2.4.37/
echo "正在配置Apache"
./configure --prefix=/usr/local/apache/ --with-apr=/usr/local/apr/ --with-apr-
util=/usr/local/apr-util/ --enable-so --enable-ssl--enable-deflate=shared --
enable-expires=shared --enable-rewrite=shared --enable-static-support &>
/dev/null
make &> /dev/null && make install &> /dev/null
if [ $? -eq 0 ]
then
echo "Apache installed"
else
echo "Apache installed failed"
exit
fi
cd /usr/local/apache/bin/
echo ServerName www.fangxi.com >> /usr/local/apache/conf/httpd.conf
./apachectl start
if [ $? -eq 0 ]
then
echo "Apache安装成功并启动"
else
echo "Apache启动失败"
exit
fi
}

#安装php
```

```
install_php()
{
echo "正在安装php服务"
yum -y install php php-cli php-curl php-fpm php-intl php-mcryp php-mysql php-gd
php-mbstring php-xml php-dom &> /dev/null
if [ $? -eq 0 ]
then
echo "php安装成功"
else
echo "php安装失败"
exit
fi
systemctl start php-fpm &> /dev/null
if [ $? -eq 0 ]
then
echo "php安装成功"
else
echo "php安装失败"
exit
fi

}
#编译安装Mysql
install_mysql()
{
echo "开始安装mysql"
echo "正在准备编译环境，wait a minute"
yum -y install ncurses ncurses-devel openssl-devel bison gcc gcc-c++ make cmake
&> /dev/null
if [ $? -eq 0 ]
then
echo "编译环境已准备好"
else
echo "编译环境准备失败"
exit
fi
echo "正在下载源码包----请稍稍等一下"
wget http://ftp.ntu.edu.tw/MySQL/Downloads/MySQL-5.7/mysql-boost-5.7.26.tar.gz
groupadd mysql
useradd -r -g mysql -s /bin/nologin mysql
tar xf mysql-boost-5.7.26.tar.gz
cd mysql-5.7.26/
echo "正在配置中，请再喝口水，小憩一下"
cmake . \
-DWITH_BOOST=boost/boost_1_59_0/ \
-DCMAKE_INSTALL_PREFIX=/usr/local/mysql \
-DSYSCONFDIR=/etc \
-DMYSQL_DATADIR=/usr/local/mysql/data \
-DINSTALL_MANDIR=/usr/share/man \
-DMYSQL_TCP_PORT=3306 \
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \
-DDEFAULT_CHARSET=utf8 \
-DEXTRA_CHARSETS=all \
-DDEFAULT_COLLATION=utf8_general_ci \
-DWITH_READLINE=1 \
-DWITH_SSL=system \
-DWITH_EMBEDDED_SERVER=1 \
-DENABLED_LOCAL_INFILE=1 \
```

```
-DWITH_INNOBASE_STORAGE_ENGINE=1 &> /dev/null
if [ $? -eq 0 ]
then
echo "mysql配置成功"
else
echo "mysql配置失败"
exit
fi
echo "----------正在安装编译安装Mysql请稍等----------"
make &> /dev/null && make install /dev/null
if [ $? -eq 0 ]
then
echo "mysql编译安装成功"
else
echo "mysql编译安装失败"
exit
fi
echo [mysqld] > /etc/my.cnf
echo basedir=/usr/local/mysql >> /etc/my.cnf
echo datadir=/usr/local/mysql/data >> /etc/my.cnf
echo "mysql配置文件successed"
cd /usr/local/mysql/
mkdir mysql-files
chown -R mysql.mysql /usr/local/mysql
echo "-------------正在初始化Mysql请稍等--------------"
/usr/local/mysql/bin/mysqld --initialize --user=mysql --basedir=/usr/local/mysql
--datadir=/usr/local/mysql/data &> mima.txt
mima=awk '/password/ {print $NF}' mima.txt
echo "初始密码为：$mima"
if [ $? -eq 0 ]
then
echo "mysql初始化成功"
else
echo "mysql初始化失败"
exit
fi
bin/mysql_ssl_rsa_setup --datadir=/usr/local/mysql/data
#给数据库加密
cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld
chkconfig --add mysqld
chkconfig mysqld on
#添加到开机启动项
systemctl start mysqld
if [ $? -eq 0 ]
then
echo "mysql启动成功"
else
echo "mysql启动失败"
exit
fi
echo "export PATH=$PATH:/usr/local/mysql/bin" >> /etc/profile
source /etc/profile
echo "----------修改数据库初始密码----------"
read -p "请输入你要设置的数据库密码" new_mima
mysqladmin -uroot -p${mima} password "$new_mima"
if [ $? -eq 0 ]
then
echo "mysql初始密码修改成功，mysql部署完成"
```

```
else
echo "mysql初始密码修改失败"
exit
fi

}
while :
do
read -p "请输入你要选择的参数: " var
case $var in
a)
install_Apache
;;
b)
install_php
;;
c)
install_mysql
;;
d)
install_Apache
install_php
install_mysql
;;
q)
exit
;;
*)
printf "请按照上方提供的选项输入!!!\n"
;;
esac
done
```