**第二十四章 shell中色彩处理和awk使用技巧**

**本节所讲内容：**

**24.1 shell中的色彩处理**

**24.2 awk基本应用**

**24.3 awk高级应用**

**24.4 实战-shell脚本实战**

# 24.1 Shell中的色彩处理

**shell脚本中echo显示内容带颜色显示,echo显示带颜色，需要使用参数-e**

**格式1： echo -e “\033[背景颜色;文字颜色m 要输出的字符 \033[0m”**

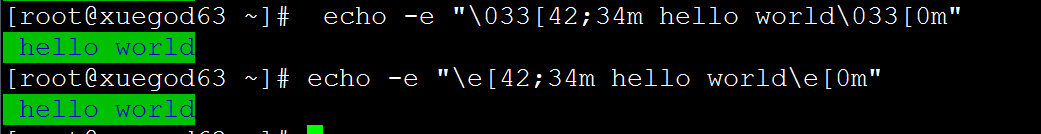
**格式2：echo -e “\e[背景颜色;文字颜色m要输出的字符\e[0m”**

**例：绿底蓝字**

**[root@xuegod63 ~]# echo -e "\033[42;34m hello world\033[0m"**

**[root@xuegod63 ~]# echo -e "\e[42;34m hello world\e[0m"**

**如图：**



**注:其中42的位置代表底色，34的位置代表的是字的颜色，0m是清除所有格式**

**1、字背景颜色和文字颜色之间是英文的分号";"**

**2、文字颜色后面有个m**

**3、字符串前后可以没有空格，如果有的话，输出也是同样有空格**

**4、echo显示带颜色，需要使用参数-e ,-e 允许对下面列出的加反斜线转义的字符进行解释.**

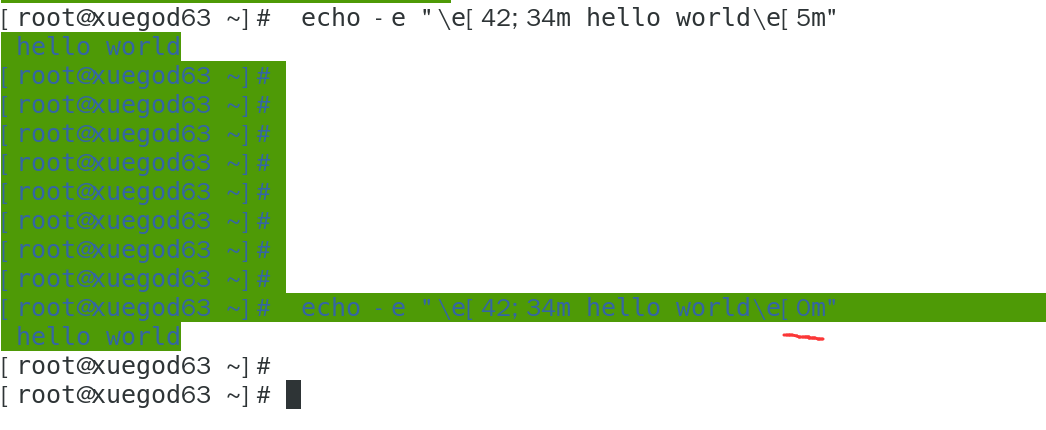
**控制选项：**

**\033[0m 关闭所有属性**

**\033[1m 设置高亮度，加粗**

**\033[5m 闪烁**

**[root@xuegod63 ~]# echo -e "\e[42;34m hello world\e[5m" #执行后，发现后期所有输出都闪烁状态，如何关闭？**



**[root@xuegod63 ~]# echo -e "\e[42;34m hello world\e[0m" #可以使用\033[0m 关闭所有属性**

**常见shell输入带颜色文字： 3x代表字的颜色，4x代表背景色**

**echo -e "\033[30m 黑色字 \033[0m"**

**echo -e "\033[31m 红色字 \033[0m"**

**echo -e "\033[32m 绿色字 \033[0m"**

**echo -e "\033[33m 黄色字 \033[0m"**

**echo -e "\033[34m 蓝色字 \033[0m"**

**echo -e "\033[35m 紫色字 \033[0m"**

**echo -e "\033[36m 天蓝字 \033[0m"**

**echo -e "\033[37m 白色字 \033[0m"**

**echo -e "\033[40;37m 黑底白字 \033[0m"**

**echo -e "\033[41;37m 红底白字 \033[0m"**

**echo -e "\033[42;37m 绿底白字 \033[0m"**

**echo -e "\033[43;37m 黄底白字 \033[0m"**

**echo -e "\033[44;37m 蓝底白字 \033[0m"**

**echo -e "\033[45;37m 紫底白字 \033[0m"**

**echo -e "\033[46;37m 天蓝底白字 \033[0m"**

**echo -e "\033[47;30m 白底黑字 \033[0m"**

# 24.2 awk基本应用

**grep和egrep：文本过滤的**

**sed：流编辑器，实现编辑的**

**awk：文本报告生成器，实现格式化文本输出**

## 24.2.1 概念

**AWK是一种优良的文本处理工具，Linux及Unix环境中现有的功能最强大的数据处理引擎之一。这种编程及数据操作语言的最大功能取决于一个人所拥有的知识。awk命名:Alfred Aho Peter 、Weinberger和brian kernighan三个人的姓的缩写。**

**awk---->gawk 即： gun awk**

**在linux上常用的是gawk,awk是gawk的链接文件**

**man gawk----》pattern scanning and processing language 模式扫描和处理语言。**

**pattern [ˈpætn] 模式 ； process [ˈprəʊses] 处理**

**任何awk语句都是由模式和动作组成，一个awk脚本可以有多个语句。模式决定动作语句的触发条件和触发时间。**

**模式：**

**正则表达式 : /root/ 匹配含有root的行 /\*.root/**

**关系表达式： < > && || + \***

**匹配表达式： ~ ！~**

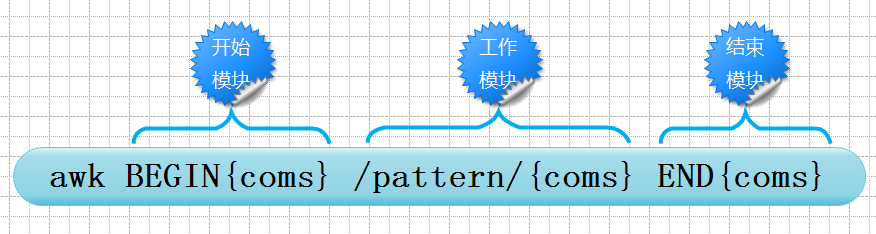
**动作：**

**变量 命令 内置函数 流控制语句**

**它的语法结构如下：**

**awk [options] 'BEGIN{ print "start" } ‘pattern{ commands }’ END{ print "end" }' file**

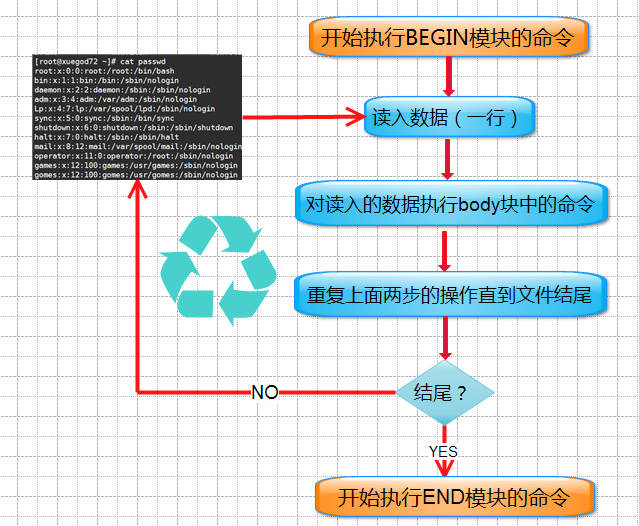
**其中：BEGIN END是AWK的关键字部，因此必须大写；这两个部分开始块和结束块是可选的**

****

**特殊模块：**

**BEGIN语句设置计数和打印头部信息，在任何动作之前进行**

**END语句输出统计结果，在完成动作之后执行**

****

**通过上面我们可以知道;AWK它工作通过三个步骤**

**1、读：从文件、管道或标准输入中读入一行然后把它存放到内存中**

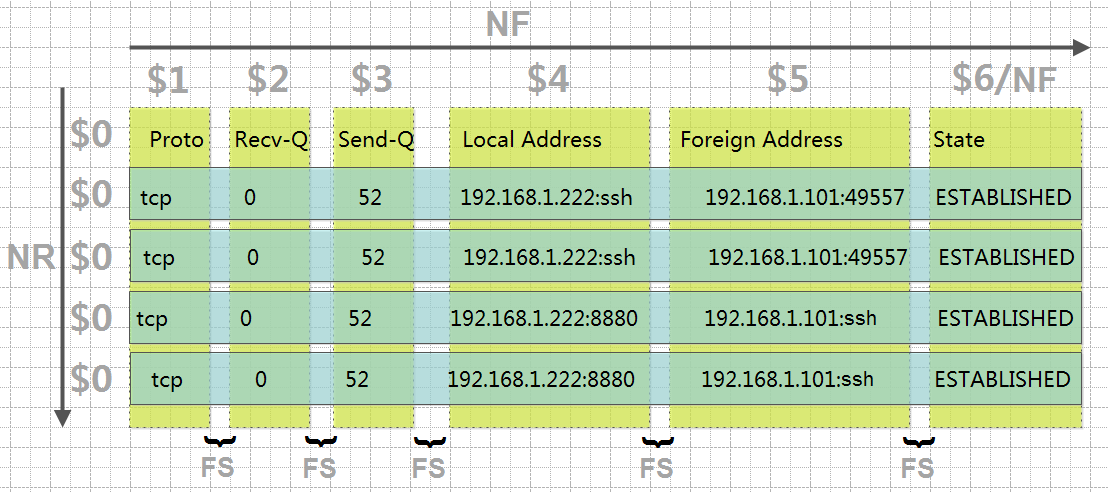
**2、执行：对每一行数据，根据AWK命令按顺序执行。默认情况是处理每一行数据，也可以指定模式**

**3、重复：一直重复上述两个过程直到文件结束**

**AWK支持两种不同类型的变量：内建变量，自定义变量**

**awk内置变量（预定义变量）**

* **$n 当前记录的第n个字段，比如: $1表示第一个字段，$2表示第二个字段**
* **$0 这个变量包含执行过程中当前行的文本内容**
* **FILENAME 当前输入文件的名**
* **FS 字段分隔符（默认是空格）**
* **NF 表示字段数，在执行过程中对应于当前的字段数，NF：列的个数**
* **FNR 各文件分别计数的行号**
* **NR 表示记录数，在执行过程中对应于当前的行号**
* **OFS 输出字段分隔符（默认值是一个空格）**
* **ORS 输出记录分隔符（默认值是一个换行符）**
* **RS 记录分隔符（默认是一个换行符）**

****

## 24.2.2 实例演示

**常用的命令选项：**

**-F fs指定分隔符**

**-v 赋值一个用户自定义变量**

**-f 指定脚本文件,从脚本中读取awk命令**

### （1）分隔符的使用

**用法：-Ffs 其中fs是指定输入分隔符，fs可以是字符串或正则表达式;分隔符默认是空格**

**常见写法：-F: -F, -F[Aa]**

**例1：**

**[root@xuegod63 ~]# echo "AA BB CC DD"|awk '{print $2}'**

**BB**

**[root@xuegod63 ~]# echo "AA|BB|CC|DD"|awk -F"|" '{print $2}'**

**BB**

**[root@xuegod63 ~]# echo "AA,BB,CC,DD"|awk -F"," '{print $2}'**

**BB**

**[root@xuegod63 ~]# echo "AA,BB,CC,DD"|awk -F, '{print $2}'**

**BB**

**[root@xuegod63 ~]# awk -F: '{print $1}' /etc/passwd #以：分隔，打印第1列用户名**

**例2：指定多个分隔符**

**[root@xuegod63 ~]# echo "12AxAbADXaAD52" | awk -F"[aA]" '{print $6}'**

**D52**

**例3：使用FS指定分隔符**

**[root@xuegod63 ~]# echo "12AxAbADXaAD52" | awk 'BEGIN {FS="aA"} {print $2}'**

**D52**

**例4：过滤出本系统的IP地址**

**[root@xuegod63 ~]# ifconfig ens33 | grep netmask**

**inet 192.168.1.63 netmask 255.255.255.0 broadcast 192.168.1.255**

**[root@xuegod63 ~]# ifconfig ens33 | grep netmask | awk '{print $2}'**

**192.168.1.63**

**（2）关系运算符的使用**

**例1：**

**[root@xuegod63 ~]# echo "3 2 3 4 5" > a.txt**

**[root@xuegod63 ~]# awk '{print $1+10}' a.txt**

**13**

**例2：**

**[root@xuegod63 ~]# echo "one two three four" | awk '{print $4}'**

**four**

**[root@xuegod63 ~]# echo "one two three four" | awk '{print $NF}'**

**four**

**[root@xuegod63 ~]# echo "one two three four" | awk '{print $(NF-2)}' #打印倒数第3列**

**two**

**[root@xuegod63 ~]# echo "one two three four" | awk '{print $(NF/2-1)}'**

**one**

**例2：打印出passwd文件中用户UID小于10的用户名和它登录使用的shell**

**参数： $NF 最后一列**

**[root@xuegod63 ~]# awk -F: '$3<10{print $1 $NF}' /etc/passwd #直接输出格式太乱**

**root/bin/bash**

**bin/sbin/nologin**

**daemon/sbin/nologin**

**adm/sbin/nologin**

**lp/sbin/nologin**

**sync/bin/sync**

**shutdown/sbin/shutdown**

**halt/sbin/halt**

**mail/sbin/nologin**

**[root@xuegod63 ~]# awk -F: '$3<10{print $1 "<======>" $NF}' /etc/passwd #awk格式化输出**

**root<======>/bin/bash**

**bin<======>/sbin/nologin**

**daemon<======>/sbin/nologin**

**在$1和$NF之间加一下\t tab**

**[root@xuegod63 ~]# awk -F: '$3<10{print $1"\t"$NF}' /etc/passwd**

**注：awk 最外面使用了单引号'' ,里面都使用双引号“”**

**.输出多个列时，可以加，分隔一下.**

**[root@xuegod63 ~]# awk -F: '$3<10{print $1,$NF}' /etc/passwd**

**例2：打印出系统中UID大于1000且登录shell是/bin/bash的用户**

**[root@xuegod63 ~]# awk -F: '$3>=1000 && $NF=="/bin/bash"{print $1"\t"$NF}' /etc/passwd**

**mk /bin/bash**

### （3）在脚本中的一些应用

**例：统计当前内存的使用率**

**[root@xuegod63 ~]#**

**[root@xuegod63 ~]# cat user\_cache.sh**

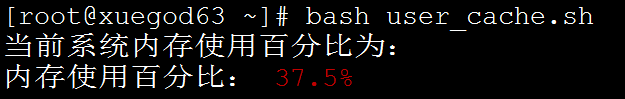
**#!/bin/bash**

**echo "当前系统内存使用百分比为："**

**USEFREE=`free -m | grep -i mem | awk '{print $3/$2\*100"%"}'`**

**echo -e "内存使用百分比： \e[31m${USEFREE}\e[0m"**

**[root@xuegod63 ~]# bash user\_cache.sh**



# 24.3 awk高级应用

**命令格式：**

**awk [-F | -f | -v ] ‘BEGIN {} / / {command1;command2} END {}’file**

**-F 指定分隔符**

**-f 调用脚本**

**-v 定义变量**

**‘{}’ 引用代码块**

**{…} 命令代码块，包含一条或多条命令**

**BEGIN 初始化代码块**

**/ str / 匹配代码块，可以是字符串或正则表达式**

**{print A；print B} 多条命令使用分号分隔**

**END 结尾代码块**

**在awk中，pattern有以下几种：**

1. **empty空模式，这个也是我们常用的**
2. **/regular expression/ 仅处理能够被这个模式匹配到的行**

**例:打印以root开头的行**

**[root@xuegod63 ~]# awk -F: '/^root/{print $0}' /etc/passwd**

**root:x:0:0:root:/root:/bin/bash**

**3) 行范围匹配 startline,endline**

**例1：输出行号大于等于3且行号小于等于6的行**

**[root@xuegod63 ~]# awk -F: '(NR>=3&&NR<=6){print NR,$0}' /etc/passwd**

**3 daemon:x:2:2:daemon:/sbin:/sbin/nologin**

**4 adm:x:3:4:adm:/var/adm:/sbin/nologin**

**5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin**

**6 sync:x:5:0:sync:/sbin:/bin/sync**

**内置变量的特殊用法：**

* **$0       表示整个当前行**
* **NF     字段数量 NF(Number 数量 ; field 字段)**
* **NR      每行的记录号，多文件记录递增 Record [ˈrekɔ:d]**
* **\t        制表符**
* **\n      换行符**
* **~        匹配**
* **!~      不匹配**
* **-F'[:#/]+'  定义三个分隔符**

**例1：使用NR行号来定位，然后提取IP地址**

**注：这个思路很好，之前都是通过 过滤关键字来定位，这次是通过行号，多了一种思路**

**[root@xuegod63 ~]# ifconfig ens33 | awk -F " " 'NR==2{print $2} '**

**192.168.1.63**

**注：NR==2表示行号**

**例2：NR与FNR的区别**

**[root@xuegod63 ~]# awk '{print NR"\t" $0}' /etc/hosts /etc/hostname**

**1 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**2 ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**3 192.168.1.63 xuegod63.cn**

**4 192.168.1.64 xuegod64.cn**

**5 192.168.1.64 xuegod62.cn**

**6**

**7 xuegod63.cn**

**[root@xuegod63 ~]# awk '{print FNR"\t" $0}' /etc/hosts /etc/hostname**

**1 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**2 ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**3 192.168.1.63 xuegod63.cn**

**4 192.168.1.64 xuegod64.cn**

**5 192.168.1.64 xuegod62.cn**

**6**

**1 xuegod63.cn**

**注：对于NR来说，在读取不同的文件时，NR是一直加的 ；**

**对于FNR来说，在读取不同的文件时，它读取下一个文件时，FNR会从1开始重新计算的**

**例3：使用3种方法去除首行**

**方法1：[root@xuegod63 ~]# route -n | grep -v ^Kernel**

**Destination Gateway Genmask Flags Metric Ref Use Iface**

**0.0.0.0 192.168.1.1 0.0.0.0 UG 100 0 0 ens33**

**192.168.1.0 0.0.0.0 255.255.255.0 U 100 0 0 ens33**

**192.168.122.0 0.0.0.0 255.255.255.0 U 0 0 0 virbr0**

**方法2：sed 1d #删除第1行的内容**

**[root@xuegod63 ~]# route -n | sed 1d**

**方法3： awk**

**[root@xuegod63 ~]# route -n | awk 'NR!=1 {print $0}'**

**例4：匹配，使用awk查出以包括root字符的行 ， 有以下3种方法**

**[root@xuegod63 ~]# awk -F: "/root/{print}" /etc/passwd**

**[root@xuegod63 ~]# awk -F: "/root/" /etc/passwd**

**[root@xuegod63 ~]# awk -F: '/root/{print $0}' /etc/passwd**

**做一个不匹配root行：**

**[root@xuegod63 ~]# awk -F: '!/root/{print $0}' /etc/passwd**

**以root开头的行：**

**[root@xuegod63 ~]# awk -F: '/^root/{print $0}' /etc/passwd**

**以bash结尾的行：**

**[root@xuegod63 ~]# awk -F: '/bash$/{print $0}' /etc/passwd**

**root:x:0:0:root:/root:/bin/bash**

**mk:x:1000:1000:mk:/home/mk:/bin/bash**

**=====以下知识，大家了解一下======**

**例5：条件表达式**

**表达式?if-true:if-false 问号前面是条件，如果条件为真执行if-true,为假执行if-false**

**例1：如果passwd中UID小于10，则给变量USER赋值成aaa，否则赋值成bbb**

**[root@xuegod63 ~]# awk -F: '{$3<10? USER="aaa":USER="bbb";print $1,USER}' /etc/passwd**

**root aaa**

**bin aaa**

**daemon aaa**

**adm aaa**

**lp aaa**

**sync aaa**

**shutdown aaa**

**halt aaa**

**mail aaa**

**operator bbb**

**。。。**

**用if(条件){命令1；命令2}elif(条件){命令；}else{命令}中，在比较条件中用( )扩起来，在AWK中，如果条件为1为真，0为假**

**例：如果UID大于10 ，则输出user=>用户名，否则输出pass=>用户名**

**[root@xuegod63 ~]# awk -F: '{if($3<10){print "user=>"$1}else{print "pass=>"$1}}' /etc/passwd**

**user=>root**

**user=>bin**

**user=>daemon**

**user=>adm**

**user=>lp**

**user=>sync**

**user=>shutdown**

**user=>halt**

**user=>mail**

**pass=>operator**

**~        匹配**

**!~      不匹配**

**例：查出行号小于等于5且包括bin/bash的行**

**[root@xuegod63 ~]# awk -F: '{if($3<=5 && $NF ~ "bin/bash"){print $1,$NF}}' /etc/passwd**

**root /bin/bash**

**例6：变量**

1. **用-v指定 var=value 变量名区分大小写的**
2. **在程序中直接定义**
3. **在awk里，使用变量不用加$符号。**

**[root@xuegod63 ~]# var="test"**

**[root@xuegod63 ~]# awk 'BEGIN{print "'$var'"}' #引用变量时，使用单引号+双引号括起来**

**test**

**例7：格式化输出**

**printf命令：格式化输出 printf “FORMAT”,item1,item2.......**

**format使用注意事项：**

**1、其与print命令的最大不同是，printf需要指定format样式**

**2、format用于指定后面的每个item的输出格式**

**3、printf语句不会自动打印换行符；\n**

**4、format格式的指示符都以%开头，后跟一个字符；如下：**

**%c: 显示字符的ASCII码**

**%d, %i：十进制整数**

**%e, %E：科学计数法显示数值**

**%f: 显示浮点数**

**%g, %G: 以科学计数法的格式或浮点数的格式显示数值；**

**%s: 显示字符串**

**%u: 无符号整数**

**%%: 显示%自身**

**例1：输入passwd文件中的第1列内容，输出时不会换行**

**[root@xuegod63 ~]# awk -F: '{printf "%s",$1}' /etc/passwd ##不会自动换行**

**rootrootbindaemonadmlpsyncshutdownhaltmailoperatorgamesftpnobodysystemd-networkdbuspolkitdabrtlibstoragemgmtrpccolordsaslauthrtkitchronyqemutssusbmuxdgeocluerpcusernfsnobodyradvdsetroubleshootpulsegdmgnome-initial-setupsshdavahipostfixntptcpdumpmk[root@xuegod63 ~]#**

**例2：换行输出**

**[root@xuegod63 ~]# awk -F: '{printf "%s\n",$1}' /etc/passwd**

**例3：在输出的字母前面添加自定义字符串USERNAME:**

**[root@xuegod63 ~]# awk -F: '{printf "USERNAME: %s\n",$1}' /etc/passwd**

**USERNAME: root**

**USERNAME: bin**

**USERNAME: daemon**

**USERNAME: adm**

**例4：对$1和$NF都做格式化输出**

**[root@xuegod63 ~]# awk -F: '{printf "USERNAME: %s %s\n",$1,$NF}' /etc/passwd**

**USERNAME: root /bin/bash**

**USERNAME: bin /sbin/nologin**

**USERNAME: daemon /sbin/nologin**

**例5：对$1和$NF都做格式化输出，在$1和$NF两者之间添加一串====字符进行输入**

**[root@xuegod63 ~]# awk -F: '{printf "USERNAME: %s=========%s\n",$1,$NF}' /etc/passwdUSERNAME: root=========/bin/bash**

**USERNAME: bin=========/sbin/nologin**

**awk修饰符：**

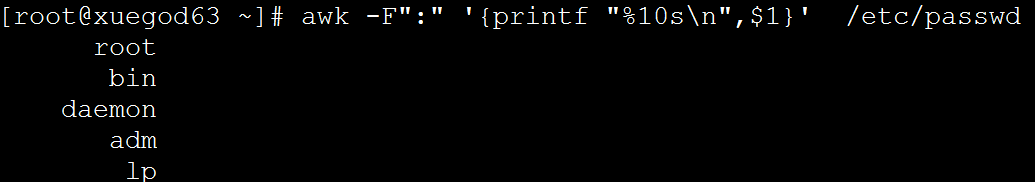
**N: 显示宽度；**

**-: 左对齐；**

**一个字母占一个宽度。默认是右对齐**

**例1：显示时用10个字符串右对齐显示。如果要显示的字符串不够10个宽度，以字符串的左边自动添加。一个字母占一个宽度。默认是右对齐**

**[root@xuegod63 ~]# awk -F":" '{printf "%10s\n",$1}' /etc/passwd**



**例2：使用10个宽度，左对齐显示**

**[root@xuegod63 ~]# awk -F":" '{printf "%-10s\n",$1}' /etc/passwd**

**root**

**bin**

**daemon**

**adm**

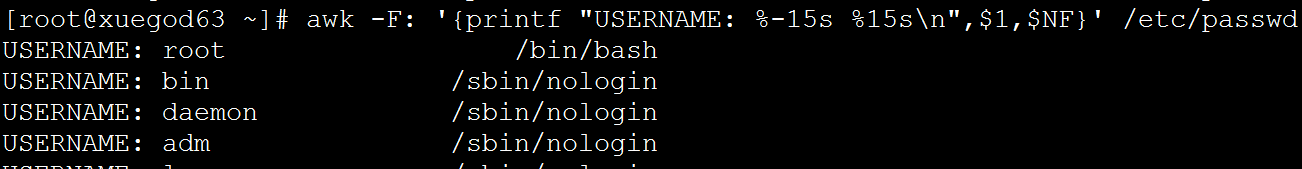
**lp**

**例3：第1列使用15个字符宽度左对齐输出，最后一列使用15个字符宽度右对齐输出**

**[root@xuegod63 ~]# awk -F: '{printf "USERNAME: %-15s %15s\n",$1,$NF}' /etc/passwd**

**USERNAME: root /bin/bash**

**USERNAME: bin /sbin/nologin**



**例4：使用开始和结束模块来格式化输出**

**[root@xuegod63 ~]# cat test.awk**

**BEGIN{**

**print "UserId\t\t\tShell"**

**print "-------------------------------"**

**FS=":"**

**}**

**$3>=500 && $NF=="/sbin/nologin"{**

**printf "%-20s %-20s\n", $1,$NF**

**}**

**END{**

**print "--------------------------------"**

**}**

**[root@xuegod63 ~]# awk -f test.awk /etc/passwd**

**UserId Shell**

**-------------------------------**

**polkitd /sbin/nologin**

**libstoragemgmt /sbin/nologin**

**colord /sbin/nologin**

**saslauth /sbin/nologin**

**chrony /sbin/nologin**

**geoclue /sbin/nologin**

**nfsnobody /sbin/nologin**

**setroubleshoot /sbin/nologin**

**gnome-initial-setup /sbin/nologin**

**--------------------------------**

# 24.4 实战-shell脚本实战

## 24.4.1 检查服务器是否受到DDOS攻击脚本

**思路：通过：netstat 查看网络连接数。如果一个IP地址对服务器建立很多连接数（比如一分钟产生了100个连接），就认为发生了DDOS**

**搭建环境：**

**[root@xuegod63 ~]# yum install httpd -y #安装apache**

**[root@xuegod63 ~]# systemctl start httpd #启动服务**

**[root@xuegod63 ~]# echo "xuegod" > /var/www/html/index.html #创建apache默认网站首页**

**[root@xuegod63 ~]# vim ddos-test.sh #写入以下内容**

**#!/bin/bash**

**netstat -ntu | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort -n**

**注释：**

**netstat -ntu | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort -n**

**截取外网IP和端口 截取外网的IP以：为分隔符 ｜排序 ｜ 排除相同的记录 ｜ 排序并统计**

**测试，模拟DDOS**

**ab命令：做压力测试的工具和性能的监控工具**

**语法： ab -n 要产生的链接数总和 -c 同时打开的客户端数量 http：//链接**

**安装ab命令：**

**[root@xuegod63 ~]# rpm -qf `which ab ` #这个安装apache时，会自动安装上**

**httpd-tools-2.2.15-15.el6.x86\_64**

**模拟DDOS： 启动10个客户端对网站首页发起1000次访问**

**[root@xuegod64 ~]# ab -n 1000 -c 10 http://192.168.1.63/index.html**

**互动： 如果你要对方发生DDOS攻击，你会攻击什么样的页面？**

**#访问一个页面比较大，页面越大，消耗服务器带宽就越大，攻击效果越明显**

**[root@xuegod63 ~]# ./ddos-test.sh #检查DDOS**

**1 Address**

**1 servers)**

**2 192.168.1.6**

**1003 192.168.1.63**

**总结：**

**24.1 shell中的色彩处理**

**24.2 awk基本应用**

**24.3 awk高级应用**

**24.4 实战-shell脚本实战**