shell day 05

=============================================

**一，正则表达式**

\w匹配数字、字母、下划线

egrep "roo\w" user //找roo后面是数字、字母、下划线的字符串

\s 匹配空格、tab键

egrep "roo\s" user //找roo后面是1个空格或者tab键打出来的空格的字符串，如果没有

就不输出

**二，sed其他指令**

a行下追加 i行上添加 c替换整行

sed 'a 666' user //所有行的下面追加666

sed '1a 666' user //第1行的下面追加666

sed '/^bin/a 666' user //在以bin开头的行的下面追加666

sed 'i 666' user //所有行的上面添加666

sed '5i 666' user //第5行的上面添加666

sed '$i 666' user //最后1行的上面添加666

sed 'c 666' user //所有行都替换成666

sed '1c 666' user //替换第1行为666

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

**三，awk**

**可以实现精确搜索并输出 ，逐行处理**

**1，前置指令 | awk 选项 条件 指令**

**2，awk 选项 条件 指令 被处理文档**

**选项 -F 定义分隔符**

**指令 print**

**条件 /字符串/**

**内置变量 $1第一列 $2第二列 $3第三列 ...**

**$0 所有列 NR行号 NF 列号**

[root@proxy opt]# cat abc.txt //准备素材

hello the world

welcome to beijing

awk '{print}' abc.txt //输出所有

awk '/to/{print}' abc.txt //输出有to的那行

awk '{print $2}' abc.txt //输出所有行的第2列

awk '/to/{print $1}' abc.txt //输出有to的那行的第1列

awk '{print $0}' abc.txt //输出所有行所有列

awk '{print $0,$1}' abc.txt //输出所有行所有列，第1列

awk '{print NR}' abc.txt //输出所有行的行号

awk '{print NR,$0}' abc.txt //输出所有行的行号，所有列

awk '{print NR,NF}' abc.txt //输出所有行的行号，列号(有几列)

awk '/^bin/{print NR}' user //找以bin开头的行，显示该行的行号

awk '/^bin/{print NR,$0}' user //找以bin开头的行，显示该行的

行号，所有列

awk '{print NF}' user //输出所有行的列号（每行有几列）

awk -F: '{print $1}' user //文档中如果没有空格，可以用F修改分

隔符

awk -F: '{print $1,$6}' user //使用冒号作为列的分隔符，显示

第1、6列

awk -F: '{print $1" 的家目录是 "$6}' user //还可以输出常量，加

双引号即可

awk -F: '{print $1" 的解释器是 "$7}' user

收集根分区剩余容量

df -h | awk '/\/$/{print $4}' //使用df -h 作为前置指令交给awk处理

找到以/结尾的行，并输出第4列

df -h | awk '/\/$/{print "根分区剩余容量是"$4}' //然后加常量输出

收集网卡流量信息

ifconfig eth0 | awk '/RX p/{print "eth0网卡接收的数量量是"$5}'

ifconfig eth0 | awk '/TX p/{print "eth0网卡发送的数量量是"$5}'

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

**四，awk的条件**

**1，/字符串/ 还可以使用正则 ~ 包含 !~不包含**

awk -F: '$6~/root/{print}' user //输出第6列包含root的行

awk -F: '$6~/bin/{print}' user //输出第6列包含bin的行

awk -F: '$6!~/bin/{print}' user //输出第6列不包含bin的行

**2，使用数字或者字符串**

== != > >= < <=

awk -F: '$3<3{print}' user //输出第3列小于3的行

awk -F: '$3<=3{print}' user //输出第3列小于等于3的行

awk -F: 'NR==2{print}' user //输出第2行

awk -F: 'NR>2{print}' user //输出行号大于2的行

**3，逻辑组合 &&并且 ||或者**

awk -F: 'NR==2||NR==4{print}' user //找行号是2或者4的行

awk -F: 'NR==2||NR==40{print}' user //如果只有一个条件满足就显示一个

awk -F: '$7~/bash/&&$3<=500{print}' user //找第7列包含bash并且第3列

小于等于500的行

awk 'NR==2&&NR==4{print}' user //找行号既是2又是4的行，不存在，无输出

awk -F: '$7~/bash/&&NR<=3{print}' user //找第7列包含bash并且行号是1~3的

awk -F: '$7~/bash/||NR<=3{print}' user //找第7列包含bash或者行号是1~3的

awk -F: '$1~/root/' user //找第1列包含root的行

如果有用户叫root6，也会搜到，比较宽松的搜索方式，如果

任务就是{print}的话可以省略不写

awk -F: '$1=="root"' user //找第1列完全等于root的行

多一个字符少一个字符都不行，比较严格的搜索方式

**4，运算**

awk 'NR%2==0{print NR,$0}' user //在条件中使用运算

，找到将行号除以2余数等于0的行，然后输出该行的

行号和所有列，相当于输出偶数行

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

**五，awk处理时机，可以执行额外任务**

BEGIN任务 执行1次，读取文档之前执行

逐行任务 执行n次，读取文档时执行

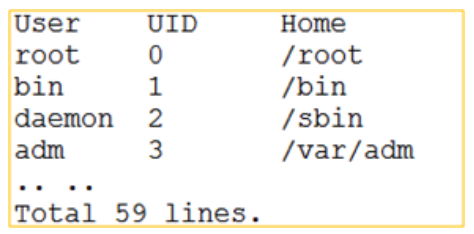
END任务 执行1次，读取文档之后执行

BEGIN{ 任务1 }{ 任务2 }END{ 任务3 }

awk -F: 'BEGIN{print "ok"}{print $1}END{print "ok"}' user

awk 'BEGIN{print NR}{print NR}END{print NR}' user

**利用awk处理时机，输出下列内容**



awk 'BEGIN{print "User\tUID\tHome"}' //第1步输出表头信息

awk -F: '{print $1"\t"$3"\t"$6}' user //第2步输出内容

awk 'END{print "总计"NR"行" }' user //第3步输出结尾

awk -F: 'BEGIN{print "User\tUID\tHome"}{print $1"\t"$3"\t"$6}END{print "总计"NR"行"}' user //合在一起写

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

使用awk统计网站访问量

setenforce 0 //关闭selinux

systemctl stop firewalld //关闭防火墙

systemctl restart httpd //开启网站服务

使用浏览器多访问几次网站，包括本机用curl

curl 192.168.4.7:82 //如果端口没改过就不用敲

awk '{print $1}' /var/log/httpd/access\_log //初步统计，不完美

**六，使用awk数组+for循环实现高级搜索**

**数组 相当于可以存储多个值的特殊变量**

**数组名[下标]=下标对应的值**

awk 'BEGIN{a[1]=10;a[2]=20;print a[2],a[1]}' //使用awk测试数组，首先

创建数组a，下标1对应值是10，下标2对应值是20，然后输出下标是2与

下标是1的值

awk 'BEGIN{a["abc"]="abcabc";a["xyz"]="xyzxyz";print a["xyz"]}' //数组的下标

和值都可以不是数字，测试时加双引号即可

以上信息是手工输入，实际情况中通常是从文档收集

准备一个文档，里面有6行，每行分别是abc、xyz、abc、opq、xyz、abc 然后

按照awk逐行处理的工作特点使用awk '{a[$1]++}' shu.txt 走完每一行得到下列结果

但不会输出到屏幕

a[$1]++ a[abc]++ a[abc]=1

a[$1]++ a[xyz]++ a[xyz]=1

a[$1]++ a[abc]++ a[abc]=2

a[$1]++ a[opq]++ a[opq]=1

a[$1]++ a[xyz]++ a[xyz]=2

a[$1]++ a[abc]++ a[abc]=3

如果要输出到屏幕可以使用命令awk '{a[$1]++}END{print a[abc]}' shu.txt

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

**根据上述操作得知使用数组可以收集信息，但收集完了之后查看确不方便，可以用for循环实现。方法如下：**

**for(变量名 in 数组名){print 变量名} //这个格式可以查看数组的所有下标**

awk '{a[$1]++}END{for(i in a){print i,a[i]}}' shu.txt //使用逐行任务与数组

收集文档shu.txt中的信息，然后在END任务中使用for循环显示所有

数组a的下标与值

awk '{ip[$1]++}END{for(i in ip){print i,ip[i]}}' /var/log/httpd/ac

cess\_log //将上述的文件替换成网站的日志，就可以

最终用来查看日志得到可以得到哪个ip来访以及来访的次数

awk '{ip[$1]++}END{for(i in ip){print i,ip[i]}}' /var/log/httpd/ac

cess\_log | sort -nr -k 2 //使用sort命令增加排序

功能，-n是以数字形式排序，-r是降序, -k是指定为

第几列排序

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

/var/log/secure是安全日志，如果有人登陆时输入错误密码的话

信息会记录下来，这种信息可以用awk抓取出来，方法如下：

awk '/Failed password for root/{ip[$11]++}END{for(i in ip){print i,ip[i]}}'

/var/log/secure //统计安全日志中访问root账户密码输入错误

的ip地址与次数

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

**编写脚本，可以查看系统的各种参数信息**

#!/bin/bash

while :

do

clear

free -h | awk '/^Mem:/{print "剩余内存容量是"$4}'

df -h | awk '/\/$/{print "根分区剩余容量是"$4}'

awk 'END{print "用户总数是"NR"个"}' /etc/passwd

who | awk 'END{print "登录用户数量是"NR"个"}'

uptime | awk '{print "cpu的15分钟平均负载是"$NF}'

rpm -qa | awk 'END{print "安装的软件包数量是"NR"个"}'

sleep 3

done