shell day 06

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

**awk中的条件**

1. **使用正则作为条件**

**/ ^ / / $/ ~包含 !~不包含**

awk -F: '/bin/{print}' user //输出有bin的行,无论bin在第几列

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

awk -F: '/bin/' user //以上指令还可以这样输入，使用awk时

如果指令就是{print}，且前面写了条件的情况下可以省略{print}不写

awk -F: '$6~/bin/' user

awk -F: '$6!~/bin/' user

**2，使用数字和字串作为条件**

==相等 !=不相等 >大于 >=大于等于 <小于 <=小于等于

awk -F: 'NR==3{print}' /etc/passwd //输出第3行

awk -F: 'NR<3{print}' /etc/passwd //输出第1~2行

awk -F: '$3<10{print}' /etc/passwd //输出id号是0~9的行

awk -F: '$3>=1000{print}' /etc/passwd //输出普通用户的行

awk -F: '$7=="/bin/bash"{print}' /etc/passwd //输出解释器是/bin/bash的

行,注意如果对比内容不是变量,要用双引号(数字不用)

awk -F: '$7!="/bin/bash"{print}' /etc/passwd //上述情况的取反搜索

**3，逻辑组合 &&并且 ||或者**

awk -F: '$3>=10&&$3<=20' /etc/passwd //找id号范围是10~20的行

awk -F: 'NR>=2&&NR<=10' /etc/passwd //找2~10行

awk -F: '$3<5||$3>1000' /etc/passwd //找id号是0~4或者1001以上的行

awk 'NR<5||NR>10{print NR}' /etc/passwd //找1~4行或者11以上的行输出

行号

awk -F: '$1=="root"||$1=="bin"' /etc/passwd //找第1列是root或者bin的

awk -F: 'NR>5||NR<100' /etc/passwd //找所有行

awk -F: 'NR<5&&NR>100' /etc/passwd //逻辑错误，没有输出

**4，使用运算作为条件**

awk 'BEGIN{print 1+1}' //常规运算

awk 'BEGIN{print 2-1}'

awk 'BEGIN{print 2\*2}'

awk 'BEGIN{print 2/2}'

awk 'BEGIN{print 2%2}'

seq 200 | awk '$1%7==0' //找到能被7整除的行

seq 200 | awk '$1~/7/||$1%7==0' //找被7整除或包含7的行

seq 200 | awk '$1~/7/&&$1%7==0' //找被7整除并且包含7的行

**在awk中使用if**

**单分支，如果满足条件，就执行指令，不满足就不执行任何指令**

awk '{if(条件){指令}}' //基本格式

awk '{if( ){ }}' //使用时可以先写框架

awk -F: '{if($7~/bash/){print}}' user //逐行检查user文档，如果

第7列包含bash，就输出该行

**双分支，如果满足条件就执行指令，不满足就执行else后面的指令**

awk '{if(条件){指令}else{指令}}' //基本格式

awk '{if( ){ }else{ }}END{ }' //先写框架，多增加END任务

awk -F: '{if($7~/bash/){x++}else{y++}}END{print x,y}' /etc/passwd //统

计系统中使用bash作为解释器的用户，和没有使用bash的用户数量, 判断

如果每找到一行的$7包含bash，就把x+1，否则y+1，最后使用end输出

x与y的值

**多分支，满足第一个条件就执行第一个条件后面的指令,不满足**

**的话看第二个条件，都不满足就执行最后else后面的指令**

awk '{if(条件){指令}else if(条件){指令}else{指令}}' //基本格式

awk -F: '{if( ){ }else if( ){ }else{ }}END{ }' //使用时可

以先写出框架，多增加END任务输出最终结果

awk -F: '{if($7~/bash/){x++}else if($7~/nologin/){y++}else{z+

+}}END{print x,y,z}' /etc/passwd //找使用bash或者nologin

作为解释器的账户数量,分别用变量x与y显示,其他解释器

用z显示

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

**使用awk统计网站访问量：**

1，准备工作：

systemctl stop httpd

yum -y remove httpd

yum -y install httpd

systemctl restart httpd //开启服务

netstat -ntulp | grep :80 //查询当前80端口是否被httpd占用

systemctl stop firewalld

curl 192.168.4.7 //使用包括自身在内的linux系统访问该网站

tail -1 /var/log/httpd/access\_log //查看最后一条httpd访问日志

awk '{print $1}' /var/log/httpd/access\_log

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

数组，相当于可以存储多个值的变量

数组名[下标]=某下标对应的值

awk 'BEGIN{a[1]=10;print a[1]}' //在awk中利用BEGIN任务创建了

一个数组名字叫a，下标是1，对应的值是10，然后输出该数组对

应下标是1的值

awk 'BEGIN{a[1]=10;a[2]=20;print a[1]}' //定义多个值，输出其中1个

awk 'BEGIN{a[1]=10;a[2]=20;print a[2]}'

**awk中的for循环，可以遍历数组(查看数组所有内容)**

**for(变量名 in 需要遍历的数组名){执行任务} //变量在每次**

**循环时,代表数组的不同下标**

awk 'BEGIN{a[1]=10;a[2]=20;a[3]=30;print a[1],a[2],a[3]}' //创建

数组a,并添加多个下标与对应的值,然后逐个输出

awk 'BEGIN{a[1]=10;a[2]=20;a[3]=30;for(i in a){print a[i]}}' //创建

数组a,并使用for循环显示数组的所有下标

awk 'BEGIN{a[1]=10;a[2]=20;a[3]=30;for(i in a){print i,a[i]}}' //创建

数组a,并使用for循环显示数组的所有下标与对应的值

[root@svr7 opt]# cat abc.txt

abc

xyz

opq

abc

abc

xyz

**高级用法示例：**

awk '{a[$1]++}END{for(i in a){print a[i],i}}' abc.txt //将abc.txt

文档中信息的第1列收集到数组a中，然后累计出现的次数，最后交

给for循环遍历(显示)所有内容

**使用awk的数组进行逐行任务时收集数据的思路过程：**

用逐行任务{a[$1]++}走完所有行的第1列得到

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

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

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

a[abc] a[abc]++ a[abc]=2

a[abc] a[abc]++ a[abc]=3

a[xyz] a[xyz]++ a[xyz]=2

下课休息 16:20回