

# Shell脚本编程

NSD SHELL

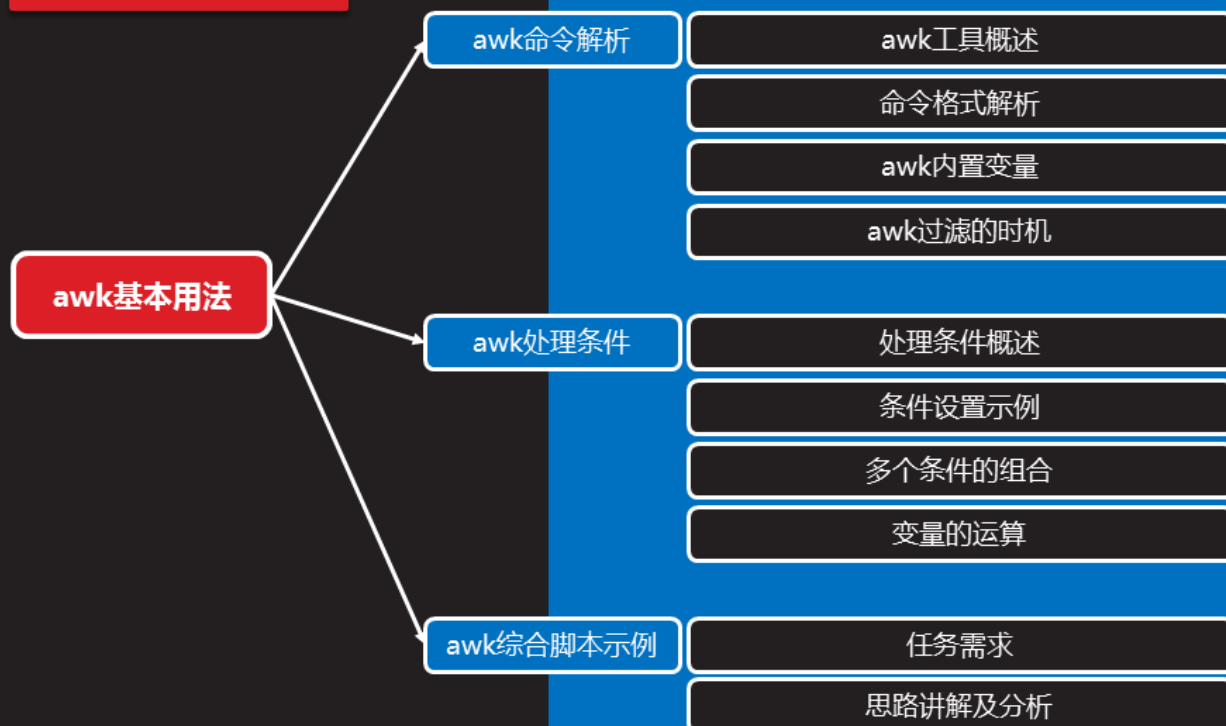
DAY06

# 内容

上午	09:00 ~ 09:30	作业讲解与回顾
	09:30 ~ 10:20	awk基本用法
	10:30 ~ 11:20	
	11:30 ~ 12:00	
下午	14:00 ~ 14:50	awk高级应用
	15:00 ~ 15:50	
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



## awk基本用法



# awk命令解析

## awk工具概述

- awk编程语言/数据处理引擎
  - 创造者：Aho、Weinberger、Kernighan
  - 基于模式匹配检查输入文本，逐行处理并输出
  - 通常用在Shell脚本中，获取指定的数据
  - 单独用时，可对文本数据做统计

# 命令格式解析

知识讲解

- 主要用法
  - 格式1：前置命令 | awk [选项] '[条件]{指令}'
  - 格式2：awk [选项] '[条件]{指令}' 文件.. ..

多条语句可以分号分隔

print 是最常用的指令

```
[root@svr5 ~]# cat test.txt
```

```
hello the world  
welcome to beijing
```

```
[root@svr5 ~]# awk '{print $1,$3}' test.txt
```

```
hello world  
welcome beijing
```



## 命令格式解析（续1）

知识讲解

- 常用命令选项
  - -F：指定分隔符，可省略（默认空格或Tab位）

```
[root@svr5 ~]# awk -F: '{print $1,$3}' /etc/passwd
```

```
root 0  
bin 1  
daemon 2  
adm 3  
....
```



## 命令格式解析（续2）

知识讲解

- 检查登录失败的IP地址有哪些

```
[root@svr5 ~]# awk '/Failed/{print $11}' /var/log/secure
192.168.2.254
192.168.2.100
... ..
```

- 检查内存的剩余容量

```
[root@svr5 ~]# free | awk '/Mem/{print $4}'
```

- 过滤网络流量

```
[root@svr5 ~]# ifconfig eth0 | awk '/RX p/{print $5}'
```



## awk内置变量

知识讲解

- 有特殊含义，可直接使用

变 量	用 途
FS	保存或设置字段分隔符，例如 FS= ":" ，与-F功能一样
\$n	指定分隔的第n个字段，如\$1、\$3分别表示第1、第3列
\$0	当前读入的整行文本内容
NF	记录当前处理行的字段个数（列数）
NR	记录当前已读入行的数量（行数）



## awk内置变量（续1）

知识讲解

```
[root@svr5 ~]# awk -F: '{print NR,NF}' /etc/passwd
```

```
[root@svr5 ~]# awk -F: '{print $NF}' /etc/passwd
```

输出每行最后一个字段

```
[root@svr5 ~]# awk -F: '{print "用户名:",$1,"解释器:",$7}' /etc/passwd
```

```
用户名: root 解释器: /bin/bash
```

```
用户名: bin 解释器: /sbin/nologin
```

```
... ..
```



## awk过滤的时机

知识讲解

- 在所有行前处理，**BEGIN{ }**
  - 读入第一行文本之前执行
  - 一般用来初始化操作
- 逐行处理，**{ }**
  - 逐行读入文本执行相应的处理
  - 是最常见的编辑指令块
- 在所有行后处理，**END{ }**
  - 处理完最后一行文本之后执行
  - 一般用来输出处理结果

可单独使用，  
也可以同时一起使用

## awk过滤的时机（续1）

知识讲解

```
[root@svr5 ~]# awk 'BEGIN {a=34;print a+12}'  
46                                //预处理不需要数据文件
```

```
[root@svr5 ~]# awk 'BEGIN{x=0}/\<bash$/ {x++}\nEND{print x}' /etc/passwd  
59                                //统计使用bash的用户个数
```

```
[root@svr5 ~]# awk 'BEGIN {print NR} END{print NR}' m.txt  
0                                //预处理时，行数为0  
2                                //全部处理完以后，行数为已读入文本的行数
```



## 案例1：使用awk提取文本

课堂练习

1. 练习awk工具的基本用法
2. 提取本机的网卡流量、根分区剩余容量，获取远程失败的IP地址
3. 格式化输出/etc/passwd文件：
  - 1) 只显示用户名、UID、宿主目录3列
  - 2) 给每列加标题，最后输出处理的总行数

User	UID	Home
root	0	/root
bin	1	/bin
daemon	2	/sbin
adm	3	/var/adm
.. ..		
Total 59 lines.		



# awk处理条件

---

## 处理条件概述

- 所有的行全部处理并输出吗？
- 怎么限制处理的条件？
- 根据多个条件来处理指定的行？



## 处理条件概述（续1）

知识讲解

- 格式回顾
  - awk [选项] '[条件]{编辑指令}' 文件.. ..
- 条件的表现形式？
  - 正则表达式
  - 数值/字符串比较
  - 逻辑比较
  - 运算符



## 条件设置示例

知识讲解

- 正则表达式
  - /正则表达式/
  - ~ 匹配、!~ 不匹配

```
[root@svr5 ~]# awk -F: '/^ro/{print}' /etc/passwd
root:x:0:0:root:/root:/bin/bash //列出以ro开头的用户记录
```

```
[root@svr5 ~]# awk -F: '$7!~/bash$/{print $1,$7}' /etc/passwd
bin /sbin/nologin //列出第7个字段不以bash结
daemon /sbin/nologin //尾的用户名、登录Shell
```

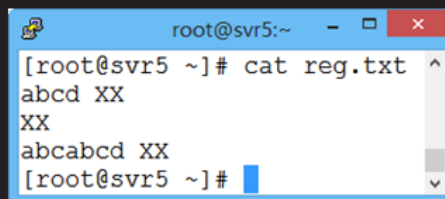


## 条件设置示例（续1）

知识讲解

- 数值比较

- == 等于、!= 不等于
- > 大于、>= 大于或等于
- < 小于、<= 小于或等于



```
root@svr5:~  
[root@svr5 ~]# cat reg.txt  
abcd XX  
XX  
abcbcd XX  
[root@svr5 ~]#
```

```
[root@svr5 ~]# awk 'NR==2{print}' reg.txt  
XX //输出第2行文本  
[root@svr5 ~]# awk '$2!="XX"{print}' reg.txt  
XX //输出第2列不是XX的行  
[root@svr5 ~]# awk 'NF>=2{print}' reg.txt  
abcd XX //输出包含2个及以上字段的行  
abcbcd XX
```



## 多个条件的组合

知识讲解

- 逻辑比较测试

- && 逻辑与：期望多个条件都成立
- || 逻辑或：只要有一个条件成立即满足要求

```
[root@svr5 ~]# awk -F: '$3>=0&&$3<2{print $1,$3}' /etc/passwd  
//列出UID小于2的用户信息  
root 0  
bin 1  
[root@svr5 ~]# awk -F: '$3==1||$3==7{print $1,$3}' /etc/passwd  
//列出UID为1或7的用户信息  
bin 1  
halt 7
```



## 变量的运算

知识讲解

- 运算符

— +、-、\*、/、%

— ++、--、+=、-=、\*=、/=

```
[root@svr5 ~]# awk 'NR%2==1{print}' reg.txt
```

```
abcd XX
```

//输出奇数行文本

```
abcabcd XX
```

```
[root@svr5 ~]# awk 'BEGIN{i=0} {i+=NF} END{print i}' reg.txt
```

```
5
```

//统计文本的总字段个数

```
[root@svr5 ~]# seq 200 | awk 'BEGIN{i=0} ($0%3==0)&& \
```

```
($0%13==0){i++} END{print i}'
```

```
5
```

//计算能同时被3和13整除的整数个数



## 案例2：awk处理条件

课堂练习

1. 列出UID间于1~1000的用户详细信息
2. 输出/etc/hosts文件内以127或192开头的记录
3. 列出100以内整数中7的倍数或是含7的数



# awk综合脚本示例

## 任务需求

- 根据/etc/passwd提取密码串
  - 找到使用bash作登录Shell的本地用户
  - 列出这些用户的shadow密码记录
  - 按每行“用户名 --> 密码记录” 保存结果

知识讲解

```
root --> $1$vEpH83MN$n1aJDFq5Sia1dzyJQyWs3/  
nick --> $1$ZrLQZB3g$phYBuzHeU9YiqgFQWZZaf0
```



## 思路讲解及分析

### 参考思路、分析

#### 知识讲解

```
root@svr5:~  
[root@svr5 ~]# cat getupwd-awk.sh  
#!/bin/bash  
## 创建空文件  
> /tmp/getupwd.log  
## 提取用户名列表  
awk -F: '/:\\bin\\bash${print $1}' /etc/passwd > /tmp/users.tmp  
## 通过for循环遍历用户名、查询密码记录，保存结果  
for NAME in $(cat /tmp/users.tmp)  
do  
    grep "^$NAME:" /etc/shadow | awk -F: '{print $1" --> "$2 | \  
    "cat >> /tmp/getupwd.log"}'  
done  
echo "用户分析完毕，请查阅文件 /tmp/getupwd.log" ## 完成后提示  
[root@svr5 ~]#  
[root@svr5 ~]# head -2 /tmp/getupwd.log  
root --> $1$vEpH83MN$n1aJDFq5SialdzyJQyWs3/  
nick --> $1$ZrLQZB3g$phYBuzHeU9YiqgFQWZZaf0  
[root@svr5 ~]#
```

双引号调用外部Shell命令



## 案例3：awk综合脚本应用

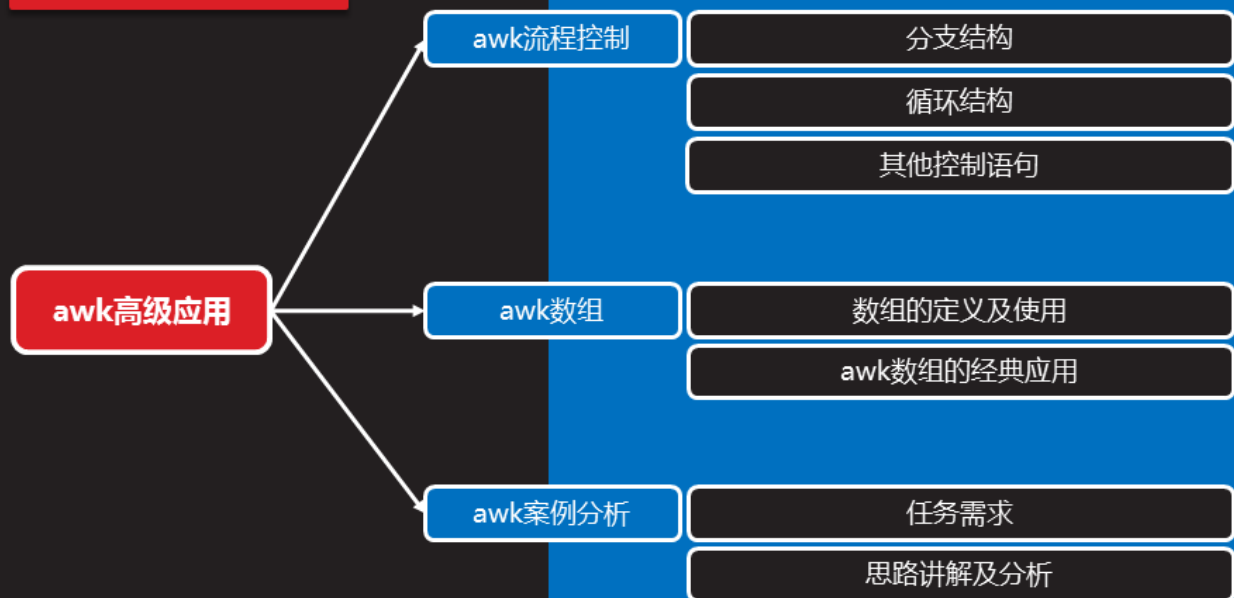
编写脚本getupwd-awk.sh，实现以下需求：

- 1) 找到使用bash作登录Shell的本地用户
- 2) 列出这些用户的shadow密码记录
- 3) 按每行“用户名 --> 密码记录”保存到getupwd.log

#### 课堂练习



## awk高级应用



## awk流程控制

# 分支结构

知识讲解

- 单分支
  - `if(条件){编辑指令}`
- 双分支
  - `if(条件){编辑指令1}else{编辑指令2}`
- 多分支
  - `if(条件){编辑指令1}else if(条件){编辑指令2}.. .. else{编辑指令N}`



## 分支结构（续1）

知识讲解

- 应用示例
  - 统计UID小于或等于500的用户个数
  - 统计UID大于500的用户个数

```
[root@svr5 ~]# awk -F: 'BEGIN{i=0;j=0}{if($3<=500){i++} \
else{j++}}END{print i,j}' /etc/passwd
37 22
```



# awk数组

## 数组的定义及使用

- 定义数组
  - 格式：数组名[下标]=元素值
- 调用数组
  - 格式：数组名[下标]
- 遍历数组
  - 用法：for(变量 in 数组名){print 数组名[变量]}



## 数组的定义及使用（续1）

- 用法示例：
  - 为数组name赋值两个元素，值分别为jim、tom

```
[root@svr5 ~]# awk 'BEGIN{name[0]="jim";name[1]="tom"; print  
name[0],name[1]}'  
jim tom
```

知识讲解



## 案例4：awk流程控制

- if分支结构（双分支、多分支）
- 练习awk数组的使用

课堂练习



# awk案例分析

## 任务需求

- 针对Web访问日志计算访问量排名
  - 获得结果：客户机的地址、访问次数
  - 按照访问次数排名

知识讲解

```
[root@svr5 ~]# less /var/log/httpd/access_log
192.168.4.5 -- [08/May/2015:10:35:27 +0800] "GET /pxe/centos6
HTTP/1.1" 404 287 "-" "ELinks/0.12pre5 (textmode; Linux; 79x21-2)"
192.168.4.110 -- [08/May/2015:10:35:58 +0800] "GET / HTTP/1.1"
403 3985 "-" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0;
rv:11.0) like Gecko LBBROWSER"
```

此Web日志中的第1个字段，即对应客户机的IP地址



## 思路讲解及分析

知识讲解

- 利用awk提取客户机IP地址、计算访问次数
  - 以\$1做下标，定义数组ip
  - 最后利用for循环输出数组下标、对应数组元素的值

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



## 思路讲解及分析（续1）

知识讲解

- 利用sort对提取结果排序
  - -n：按数字升序排列
  - -k：针对指定的列进行排序
  - -r：反向排序

```
awk ' {ip[$1]++} END{for(i in ip) {print ip[i],i}}' ... | sort -nr
```



## 案例5：awk扩展应用

课堂练习

1. 分析Web日志的访问量排名，要求如下：

- 获得结果：客户机的地址、访问次数
- 按照访问次数排名



### 总结和答疑



# awk分隔符

---

## 问题现象

- 故障错误信息

```
[root@svr5 ~]# awk '{print $1,$3}' /etc/passwd | head -2  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin
```

## 原因分析

知识讲解

- 分析故障
  - 报错信息：输出的信息不是希望的数据
- 分析故障原因
  - 默认awk分隔符为空格和tab键
  - 通过-F指定分隔符



## awk引号

---

## 问题现象

- 故障错误信息

```
[root@svr5 ~]# awk -F: "{print $1,$3}" /etc/passwd | head -2  
awk: cmd. line:1: {print ,}  
awk: cmd. line:1:      ^ syntax error
```

知识讲解



## 原因分析

- 分析故障
  - 报错信息 : awk: cmd. line:1: ^ syntax error
- 分析故障原因
  - awk的条件和指令需要使用单引号

知识讲解



