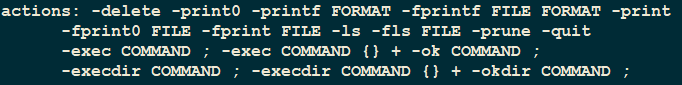
# Linux下的文本处理

## 使用find查找与操作文件

命令选项与参数  
一般格式  
  
pathname指定了查找路径，后面中括号的内容指定了找到文件以后进行的操作，默认在当前文件夹下查找，并将找到的结果打印至标准输出

动作  
-print 打印至标准输出  
-exec 找到文件以后执行命令 find /home -type f -exec ls -l {} \;  
-ok同-exec，但是执行每条命令时都会让用户选择是否执行  
-execdir   


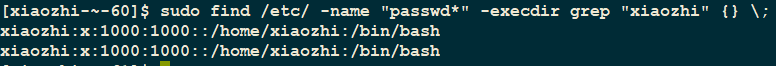
选项  
-name 按照文件名称查找文件  
-perm 按照文件权限查找  
-prune 不在当前指定的目录中查找  
-user 按照文件属主查找  
-group 按照文件属组查找  
-mtime -n or +n 文件更改时间  
-newer file1 ！ file2 比文件file1新但是比文件file2旧  
-type b|d|c|p|l|f 按文件类型  
-follow 跟踪符号链接

应用实例

**查找并列出当前目录下普通文件并打印该文件详细信息**  

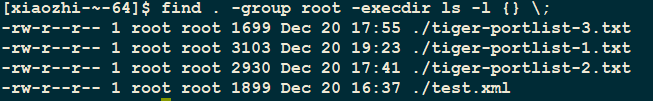

注意{}后面有空格

**查找log目录下更改时间在5日以前的文件**  
  
以时间查找  
-atime +n/-n 存取时间+n代表n天以前，-n代表n天以内  
-ctime  
-mtime  
-cmin  
-amin  
-mmin

**查找etc目录下文件名包含passwd的文件并挑选出指定用户**  


按权限查找当前目录下对任何人可执行的文件并打印详细信息  

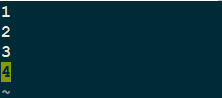

查找大小为0的文件  


查找属于root用户组的文件  


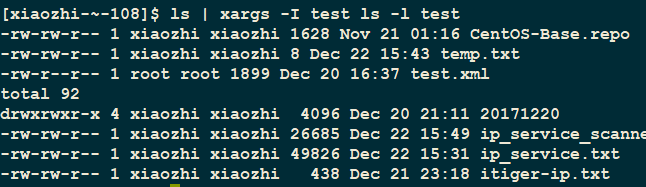
Xargs处理参数  
xargs可以处理输入数据，经常用于命令行参数的格式化，可以将多行数据变为单行，单行数据变为多行，处理数据以供命令执行

命令选项与参数  
-n 3将单行数据变为多行数据，且每行三个  
-I “string” 使用string替换每个输入数据，string可供后来命令使用  
-d “X” 重定义定界符 “X”

### 实例

多行变单行  
  


重定义分隔符  


使用I参数  


## Sort排序

命令参数  
-u 去除重复行  
-r 反向排序  
-o 将排序结果输出至文件（重定向操作时，如果重定向输出至源文件，则源文件会被清空，sort -o解决了这个问题，可以将排序结果输出至源文件）  
-n 以数值来排序而不是ascII  
-k -t k参数指定排序依据的第几列，t参数指定每个字段的分隔符  
-f 忽略大小写  
-c -C 检查文件是否已排序，-C返回第一个乱序数据后返回1，-c直接返回1

## Uniq消除重复行

### 参数与选项

默认过滤重复行，保证每行数据仅输出一次

-u 仅显示出现一次的行

-d 仅显示重复出现的行

-c 打印重复次数

## Tr处理命令参数

**参数与选项**  
tr 选项 第一字符集 第二字符集  
  
第一字符集为指定要删除或替换的原字符集，第二字符集用与替换的字符  
-d 删除属于第一字符集的字符  
-s 只显示重复字符中的一个

## Cut切割数据，选取出一行中某些字段

**参数与选项**  
-d 指定分隔符默认tab  
-f 指定选出哪个域  
-c 以字符为单位分割  
-b 以字节为单位分割

## Paste

将两个文本拼接到一起  


Wc统计文件  
-l 行数  
-w 单词数  
-c 字符数

Awk

awk用于在命令行下对文本数据进行扫描和处理。逐行扫描文件，从第一行到最后一行，寻找匹配特定模式的行，在这些行上进行用户操作  
pattern {action}  
begin,end模式分别代表处理第一行之前和处理最后一行之后进行的操作

**参数**  
-F fs 指定域分隔符fs  
-v var=val 处理之前设置一个变量var  
-f file 从文本中读取awk指令

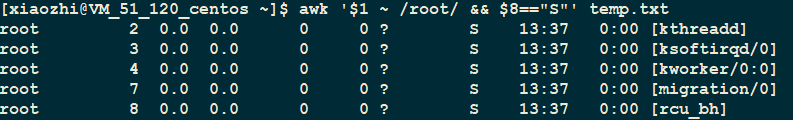
**操作指令**  
读入的每一行称为一个记录，记录被字段分隔符拆分成域

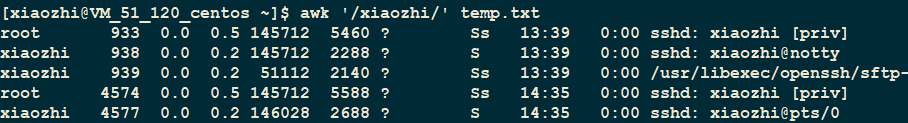
**内置变量**  
FILENAME 当前处理的文件名称  
FNR 当前输入文档的当前记录编号  
NR 输入流的当前记录编号  
FS 字段分隔符  
OFS 输出字段分隔符  
ORS 输出记录分隔符  
RS 输入记录分隔符c

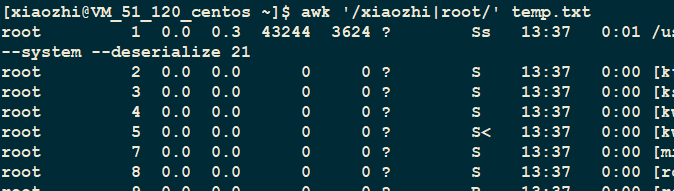
**操作符**  
<,>,<=,>=,!~  
!~表示反向匹配

**内置字符串函数**  
gsub（r，s） 在整个$0中用s替换r  
index（s，t） 返回s中第一个t的位置  
length（s） 返回s的长度  
match（s，r） 测试s是否包含匹配r的字符串  
split（s，a，fs） 在fs上将s分解为数组a  
substr（s，p） 返回s从p开始的后缀部分  
printf 格式化输出

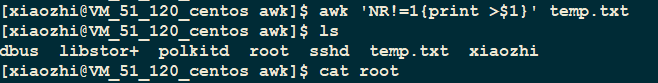
**实例**

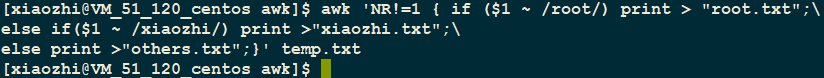
使用匹配  


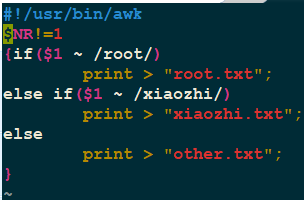
匹配一行  


匹配或  


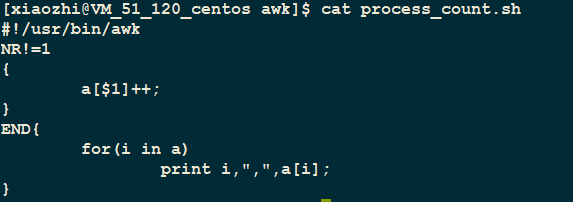
取反  


拆分文件  


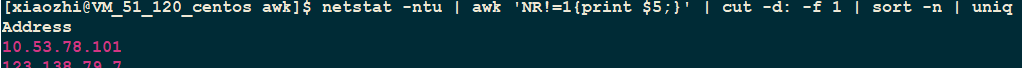
使用if-else if-else  


使用awk脚本  
  


统计信息  
(当前目录下文件大小总和）

按用户名统计进程个数  
  


查找长度大于90的行  


按连接数查看客户端IP  


## Grep工具

**简介**  
grep是一种文本搜索工具，以行为处理单位，搜索匹配的文本

**参数与选项**  
-c 计算次数  
-I 忽略大小写  
-n 输出行号  
-v 反向选择  
-r 递归查找

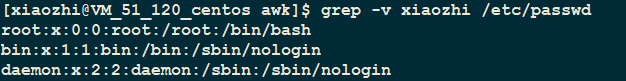
**匹配正则表达式**  
[]中匹配一个字符，^字符加在[]里表示反向选择  
^匹配行首，$匹配行尾  
.代表任意字符  
\*代表重复前一个字符0次或任意多次  
+代表重复前一个字符一次或多次  
?代表重复一次或0次

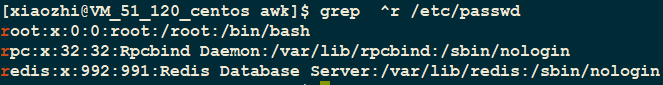
**Grep -E扩展正则表达式，**  
grep -E “A|B”

**实例**

挑选出当前文件夹下包含指定字符串的文件  

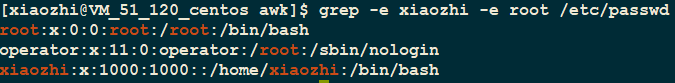

显示匹配行的行号  


反向查找  


匹配开头  


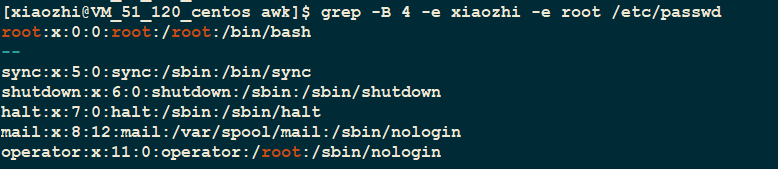
匹配结尾  

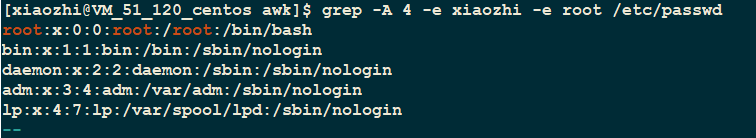

去除空行  

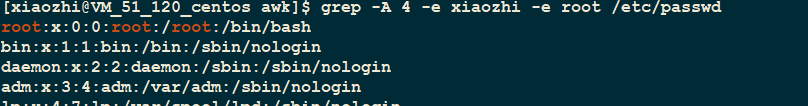

多模式匹配  


计数  

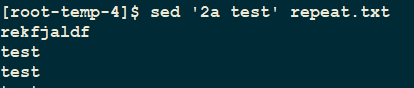
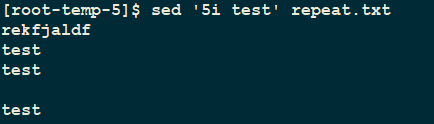
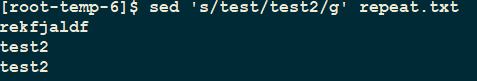
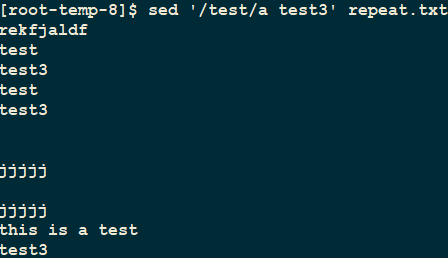
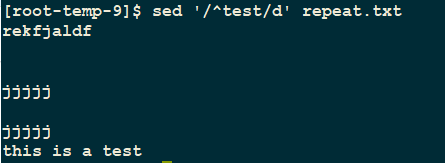
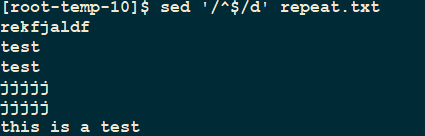

指定匹配的文件

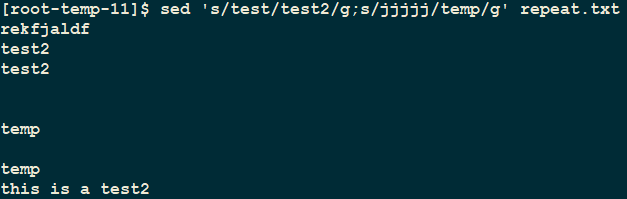
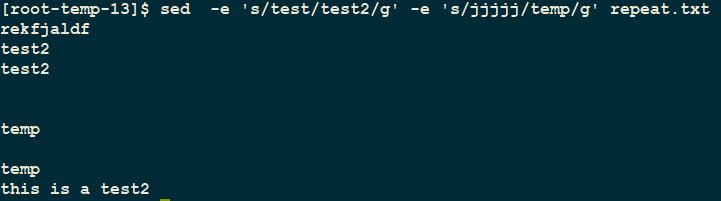
显示匹配的前四行  


显示匹配行的后四行  


显示前后各四行  


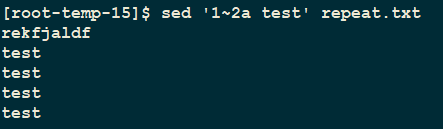
Sed工具  
address{  
command1  
command2  
}  
a append 追加  
d delete 删除  
s substitution 替换

**实例**  
在第二行后添加文本test  
  
在第五行前添加文本test  
  
将文本中所有的test替换为test2  
  
匹配包含test的行，并在该行后添加指定文本  
  
删除以test开头的行  
  
删除空白行  


**执行多个命令**  
可以将命令写入文件中，使用-f选项指定文件，也可以使用；分割，也可以使用-e参数  
  


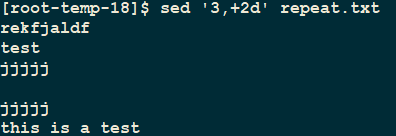
**匹配范例**

Number 指定文件唯一行号  


First~step 表示以first开始，操作步长为step  


/regexp/匹配正则  


Addr1，addr2匹配地址1到地址2的所有行  


Addr1，+N以及后面的N行内容  


正则表达式