## 1. 磁盘与目录的容量

1. df : 列出文件系统的整体磁盘使用量

df [-ahikHTm] [目录或文件名]

-a : 列出所有文件系统，包括系统特有的/proc等文件系统

-h : 以人们较易阅读的GB、MB、KB等格式显示

-i : 不用硬盘容量，而以inode的数量来显示

-k\m : 以MB\KB的容量显示文件系统

1. du : 评估文件系统的磁盘使用量

du [-ahskm] 文件或目录名称

-a : 列出所有的文件与目录的容量，因为默认仅统计目录下面的文件量

-h : 以较易阅读的格式显示

-s : 列出总量，不列出每个个别的目录占用量

-S : 尚不理解

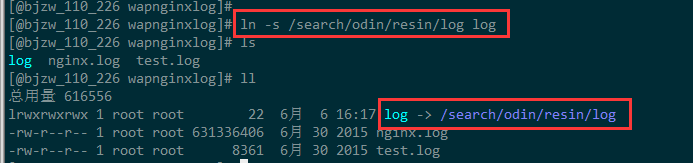
-k\m : 以MB\KB的容量显示文件系统

## 2. 连接文件：ln

ln [-sf] 源文件 目标文件(符号连接文件)

-s : 如果不加任何参数就进行硬链接，s是符号连接

-f : 如果目标文件存在，就主动将目标文件直接删除后再创建



## 3. 文件查询：which whereis locate find

1. which [-a] command : 寻找 “执行文件”
2. whereis [-bmsu] 文件或目录名

-b : 只找二进制文件

-m : 只找在manual路径下的文件

-s : 只找source源文件

-u : 查找不在上述三项内的其他特殊文件

1. locate [-ir] keyword (只要包含keyword的文件或目录名就会显示)

-i : 忽略大小写

-r : 后面可以接正则表达式的显示方式

1. find [path] [option] [action]

参数：

与时间有关 ： -atime、-ctime、-mtime

* -mtime n(n为数字) 在n天之前的“一天之内”被更改过的文件
* -mtime +n 列出在n天之前(不含n天)被更改过的文件名
* -mtime –n 列出在n天之内(含n天本身)被更改过的文件名
* -newer file 比file还要新的文件

与用户或用户组有关的参数：

* -uid/gid n: n为数字，是用户/用户组的ID
* -user/group name : name为用户名
* -nouser/nogroup ：寻找文件的所有者/用户组不在/etcpasswd 的人

与文件的权限及名称有关的参数：

* -name filename : 查找文件名为filename的文件，使用通配符时，需要用引号将filename括起来： find /etc –name ‘\*httpd\*’
* -size [+-] SIZE : 查找比SIZE大(+)或者小(-)的文件
* -type TYPE : 查找文件类型为TYPE的文件，一般正规文件(f)、设备文件(b,c)、目录(d)、连接文件(l)、socket(s)、FIFO(p)等
* -exec command: command 为其他命令， -exec后面可以再接其他的命令来处理查找到的结果。command不支持别名。find命令会将所有匹配到的文件一起传递给exec执行，但有些系统对能够传递给exec的命令有长度限制，会出现溢出错误。这时候可以使用xargs。

例子： find / -perm +7000 –exec ls –l {} \;

说明： {}：代表find找到的内容；-exec一直到\;（\为转义符号）是关键字，中间是find命令内的额外命令

* find命令与xargs结合可以实现强大的查找功能

**find . |** **xargs grep x: 查找当前目录下含有 x 的文件（输出是文件名）**

## 4. 压缩与解压

1. tar: (j：tar.bz2 z: tar.gz)

打包与压缩：tar [-j|-z] [cv] [-f 新建的文件名] filename

解压 : tar [-j|-z] [xv] [-f 新建的文件名] [-C 目录]

1. zip [-dDfFmr] [-b <dir>] [压缩文件][文件…] [-i|x <范本样式>]:

-b <dir>: 指定暂时存放文件的目录

-D : 压缩文件中不建立目录名称

-d : 从压缩文件中删除指定的文件

-f : 更新既有文件，并将不在压缩文件中的文件压缩到文件中

-F : 尝试修复已损坏的文件

-i<范式样本> : 只压缩符合条件的文件

-x<范本样式> 压缩时排除符合条件的文件

-m : 将文件压缩并加入压缩文件后，删除原始文件，即把文件移到压缩文件中

-r : 递归处理，将指定目录下的所有文件和子目录一并处理

1. windows文件转换成Linux下可执行文件：dos2unix

## 6. 数据流重定向

1. 1> : 以覆盖的方式将正确的数据输出到指定的文件或设备上
2. 1>> : 以累加的方式将正确的数据输出到指定的文件或设备上
3. 2> : 以覆盖的方式将错误的数据输出到指定的文件或设备上
4. 2>> : 以累加的方式将错误的数据输出到指定的文件或设备上
5. command > list 2>&1 ： 将正确信息和错误信息都输入到list文件中
6. command 2> /dev/null : 不保存错误信息
7. 命令执行判断依据： ; && ||
8. com1;com2 不考虑命令的相关性，连续执行命令
9. com1&&com2 前一个命令执行正确($?=0)，才执行第二个命令
10. com1||com2 前一个命令执行不正确($?!=0)，才执行第二个命令

## 7. 管道命令(pipe) : |

每个 | 后面接的第一个数据必须是能够接受standard input数据的命令，而且管道命令只能处理standard output，对于error output会忽略

管道命令以行为处理单位

1. cut –d ‘分隔符’ –f fields 将同一行里面的数据按照某些字符进行分解，

-d : 后边接分隔符， 与-f一起使用

-f : 依据-d的分隔符将一段信息切割成为数段，有-f取出第几段的意思

last | cut –d ‘ ‘ –f 1,2

1. grep [-cinv] [--color=auto] ‘查找字符串’ filename ： 主要用正则

-c : 计算找到字符串的次数

-i : 忽略大小写

-n : 给出行号

grep -–color=auto ‘manpath’ /etc/man.config

1. 排序命令：

sort [-fbMnrtuk] [file or stdin]

-f: 忽略大小写

-b: 忽略最前面的空格

-M: 以月份的名字排序

-n: 使用“纯数字”进行排序（默认是文字类型）

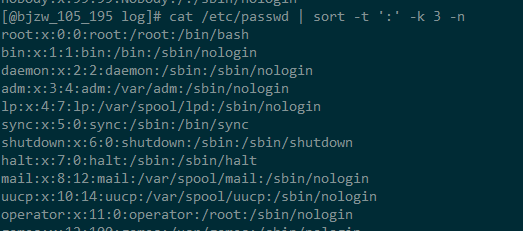
-r: 反向排序

-u: 同umiq

-t: 分隔符，默认是[tab]分割

-k: 以哪个区间来进行排序

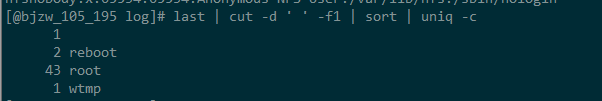
下列显示了以第三个字段的数字排序



uniq [-ic]

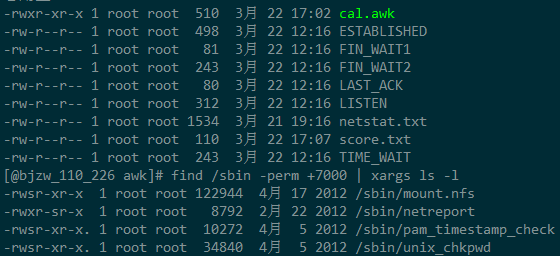
-i: 忽略大小写字符的不同

-c: 进行计数



4. 参数代换： xargs

xargs 可以读入stdin的数据，并且以空格符或者换行符进行粪便，将stdin的数据分割成arguments。对于不支持管道的命令，可以通过xargs来提供该命令使用stdin

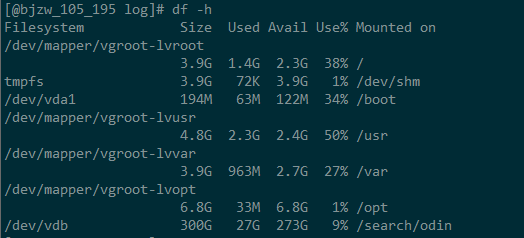


## 8. 磁盘检查

1. df [-ahikHTm] [目录或文件名]: 列出文件系统的整体磁盘使用量

-h: 以较易阅读的GB、MB、KB等格式自动显示

-i: 不使用硬盘容量，而以inode的数量显示



2.du [-ahskm] 文件或目录名:

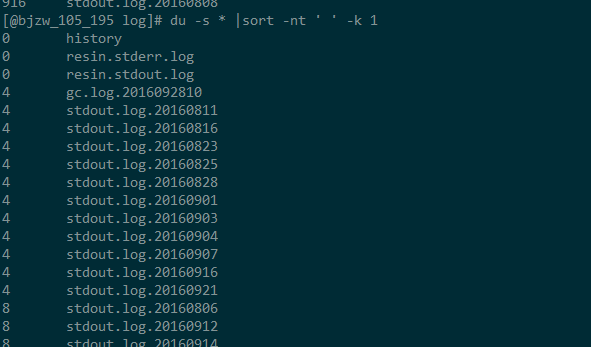
-a: 列出所有的文件和目录容量

-h: 以较易阅读的GB、MB、KB等格式自动显示

-s: 列出总量，不列出每个各别目录的占用容量

-S: 不包括子目录下的总计

下面是按照文件的大小排序



## 9. awk

* 1. 使用awk输出第1列和第4列： **awk ‘{print $1, $4}’ filename**
* 单引号中的大括号内的就是awk语句，其只能被单引号包含
* $1, $2, ..., $n代表第几列， $0代表整行
  1. awk格式化输出，同C语言中的printf相同

**awk ‘{printf “%-8s %-8s\n”, $1, $2}’**

* 1. 过滤记录
* 过滤条件为：第三列的值为0 && 第六列的值为LISTEN(支持的运算符有: !=, >, <, >=, <=) :

**awk ‘$3==0 && $6==”LISTEN” ‘ filename**

* **awk ‘$3>0 {print $0}’ filename**
* 如果需要表头： **awk ‘$3==0 || NR==1’ filename**
  1. 内建变量

|  |  |
| --- | --- |
| 变量名 | 描述 |
| $0 | 当前记录，整行的内容 |
| $1-$n | 当前记录的第n个字段，字符间有FS分割 |
| FS | 输入字段的分隔符，默认是空格或TAB |
| NF | 当前记录中的字段的个数，即列数 |
| NR | 已经读出的记录数，即行号；从1开始，若有多个文件，该值也不断累加 |
| FNR | 各个文件自己的行号 |
| RS | 输入的记录分隔符， 默认为换行符 |
| OFS | 输出的字段分隔符，默认是空格 |
| ORS | 输出记录的分隔符，默认是换行符 |
| FILENAME | 当前输入文件的名字 |

如果要输出行号

awk ‘$3==0 ||NR==1{printf “%02s %s %-20s\n, **NR**, **FNR**, $4}’

* 1. 指定分隔符
* awk ‘[BEGIN{FS=”:”} {print $1,$3,$6}’ filename
* awk –F: ‘{print $1, $3, $6}’ filename
* 使用多个分隔符分割： awk –F ‘[:;]’
  1. 字符串匹配

awk中可以使用正则表达式对字符串进行匹配：

* 示例中，匹配的是第6个字段含有FIN的行， ~表示模式开始， /.../ 中的内容是正则表达式

awk ‘$6 ~ /FIN/ {print NR, $1}’ OFS=”\t” filename

awk ‘$6 !~ /FIN/ {print NR, $1}’ filename (取反)

* 也可以像grep一样去匹配第一行

awk ‘/FIN/’ filename 取反：awk ‘!/FIN/’ filename

* 1. 拆分文件: awk拆分文件，使用重定向就可以
     + 按第6个字段拆分成以该字段为名的文件：

awk ‘NR!=1{print $4, $5 > $6}’ filename

* + - 使用if...else... awk是一个脚本解释器

awk ‘NR!=1{

if($6 ~ /TIME|ESTABLISHED/) print > “1.txt”;

else if ($6 ~/LISITEM/) print > “2.txt”;

else print > “3.txt”

} filename

* 1. 统计
     + 计算所有sh文件和log文件的文件大小综合

ls -l \*.out \*.sh | awk '{sum+=$5} END {print sum}'

* + - 统计各个connection的状态

awk ‘NR!=1{a[$6]++;}END {for (i in a) print i ”,” a[i];}’ file

* + - 统计每个用户的进程占了多少内存

ps aux | awk 'NR!=1{a[$1]+=$6;} END { for(i in a) print i ", " a[i]"KB";}'

## 10. awk 脚本

awk脚本中有一个END关键字，END的意思是“处理完所有的行的标识”， BEGIN是执行前的意思。

* + - BEGIN {这里放的是执行前的语句}
    - END {这里放的是处理完所有的行后要执行的语句}
    - {这里放的是处理每一行时要执行的语句}

### 1. 执行脚本

可以通过 awk –f cal.awk score.txt 执行下边脚本（也可以通过./cal.awk score.txt 需要cal.awk有x权限）

|  |
| --- |
| 1 #!/bin/awk -f  2 # 运行前  3 BEGIN {  4 math = 0  5 english = 0  6 computer = 0  7 printf "NAME NO. MATH ENGLISH COMPUTER TOTLA\n"  8 printf "---------------------------------------------\n"  9 }  10 #运行中  11 {  12 math += $3  13 english += $4  14 computer += $5  15 printf "%-6s%-6s %4d%8d%8d %8d\n", $1,$2,$3,$4,$5,$3+$4+$5  16 }  17 # 运行后  18 END {  19 printf "---------------------------------------------\n"  20 printf "TOTAL:%10d %8d %8d \n", math, english, computer  21 printf "AVERAGE: %10.2f%8.2f%8.2f\n",math/NR, english/NR, computer/NR  22 } |

### 2. 环境变量

在脚本中使用环境变量，需要使用-v参数和ENVIRON，使用ENVIRON的环境变量需要export。

awk –v val=$x ‘{print $1, $2, $3, $4+val, $5+ENVIRON[“y”]}’ s.txt

例子：

#从文件中找出长度大于80的行

awk ‘length>80’ file

# 按连接数查看客户端IP

netstat –ntu | awk ‘{print $5}’ |cut –d: f1 | sort |uniq –c | sort –nr

#打印99乘法表

seq 9 | sed ‘H;g’ |awk –v RS=’’ ‘{for(i=1;i<=NF;i++) printf(“%Dx%D=%d%s”, i, NR, i\*NR, i==NR?”\n”:\t”)}’

## 11. sed

sed全名叫做stream editor（流编辑器），用程序的方式来编辑文本，基本上是正则模式的使用。

### 1. 用s命令做替换

如果使用单引号，是没有办法通过 \’ 来转义的，所以一般使用双引号，在双引号中可以使用 \” 来转义。

|  |
| --- |
| sed “s/my/Wang’s/g” pets.txt [不会对原始文件做修改]  sed –i “s/my/Wang’s/g” pets.txt [直接修改原始文件]  sed –i “s/^/#/g” pets.txt [在每行最前面添加一个 #]  sed –i “s/$/ --- /g” pets.txt [在每行最后添加 --- ]  sed –i “3s/my/your/g” pets.txt [只替换第3行的内容, 行从0开始]  sed –i “3,6s/my/your/g” pets.txt [替换3到6行的内容]  sed ‘s/s/S/1’ my.txt [只替换每行的第一个s]  sed ‘s/s/S/2’ [替换每行的第二个s]  sed ‘s/s/S/3g’ [替换每行的第3个及3个以后的s] |

### 2. 多个匹配

当对不同行进行不同的处理时，可以用多个匹配来替换

|  |
| --- |
| # 将第一行到第三行的没有替换成your，将3行以后的This替换成That  sed ‘1,3s/my/your/g; 3,$s/This/That/g’ my.txt  sed –e ‘1,3s/my/your/g’ –e ‘3,$s/This/That/g’ my.txt[-e:令行参数]  # 使用’&’来当做被匹配的变量，然后可以在其左右进行操作  sed ‘s/my/[&]/g’ my.txt [将’my’替换为’[my]’] |

### 3. 捕获匹配

捕获匹配是使用圆括号将匹配内容包裹起来，在替换时可以使用 \1，\2 来表示匹配的内容。

|  |
| --- |
| # 括号需要使用 ‘\’ 来转义  sed ‘s/This is my \([^.]\*\),.\*is\(.\*\)/\1:\2/g’ my.txt |

### 4. sed的命令

N命令： 将下一行的内容纳入当前缓冲区做匹配。即将奇数行与其下一行的偶数行拼成一行做匹配

i命令： 是insert命令，用来在之前添加一行

a命令：是append命令，用来在之后添加一行

c命令：用来将匹配行替换为指定内容

d命令：用来删除匹配行

p命令：打印命令行 （可以做为grep命令）

|  |
| --- |
| sed ‘N;s/my/your/’ pets.txt  # 其中的1i表明，其要在第1行前插入一行（insert）  sed "1 i This is my monkey, my monkey's name is wukong" my.txt  # 其中的1a表明，其要在最后一行后追加一行（append）  sed "$ a This is my monkey, my monkey's name is wukong" my.txt  # 注意其中的/fish/a，这意思是匹配到/fish/后就追加一行  sed "/fish/a This is my monkey, my monkey's name is wukong" my.txt  sed "/my/a ----" my.txt  # 将第二行替换为c后边的内容  sed "2 c This is my monkey, my monkey's name is wukong" my.txt  sed "/fish/c This is my monkey, my monkey's name is wukong" my.txt  # 删除匹配行  sed '/fish/d' my.txt  # 使用-n，可以不让sed把处理时的信息输出  sed –n '/fish/p' my.txt  # 从一个模式到另一个模式  sed -n '/dog/,/fish/p' my.txt  #从第一行打印到匹配fish成功的那一行  $ sed -n '1,/fish/p' my.txt |

### 5. 命令打包

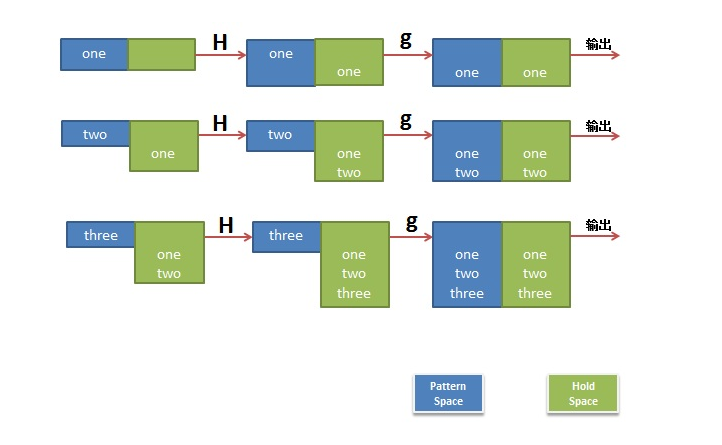
sed中的cmd可以是多个，他们可以用分号分开，用大括号括起来做为嵌套命令。

|  |
| --- |
| # 对3行到第6行，执行命令/This/d  sed '3,6 {/This/d}' pets.txt  # 对3行到第6行，匹配/This/成功后，再匹配/fish/，成功后执行d命令  sed '3,6 {/This/{/fish/d}}' pets.txt  # 从第一行到最后一行，如果匹配到This，则删除之；如果前面有空格，则去除空格  sed '1,${/This/d;s/^ \*//g}' pets.txt |

### 6. Hold Space

Hold Space 有四个命令：

* g: 将hold space中的内容拷贝到pattern space中，原来pattern space内的数据被清空
* G：将hold space汇总的内容append到pattern space\n后
* h：将pattern space中的内容拷贝到hold space中，hold space原有内容被清空
* H：将pattern space中的内容append到hold space\n 之后
* x: 交换pattern space和hold space的内容



7. 查看程序和监听的端口

1. sudo lsof -i -P -n | grep LISTEN
2. sudo netstat -tulpn | grep LISTEN
3. sudo nmap -sTU -O IP地址

# 碰到过的问题

1. /var 目录显示使用为100%，但是使用【du -h --max-depth=1 /var】命令查看var下的文件，没有发现占用空间大的文件，使用【lsof -n |grep delete】命令查看哪些进行使用了已经删除的文件，这些文件删除，但是没有释放空间，解决方案是删除进程或者reboot