linux日常用到的命令

# grep

grep是Linux中最常用的"文本处理工具"之一，grep与sed、awk合称为Linux中的三大剑客；

作用：文本搜索工具，根据用户指定的"模式(过滤条件)"对目标文本逐行进行匹配检查;打印匹配到的行;

模式：由正则表达式的元字符及文本字符所编写出的过滤条件;

## 常用选项

--color=auto：显示颜色;

-i, --ignore-case：忽略字符大小写;

-o, --only-matching：只显示匹配到的部分;

-n, --line-number：显示行号;

-v, --invert-match：反向显示,显示未匹配到的行;

-E, --extended-regexp：支持使用扩展的正则表达式;

-q, --quiet, --silent：静默模式,即不输出任何信息;

-w, --word-regexp：整行匹配整个单词;

-c, --count：统计匹配到的行数; print a count of matching lines;

-B, --before-context=NUM：print NUM lines of leading context 后#行

-A, --after-context=NUM：print NUM lines of trailing context 前#行

-C, --context=NUM：print NUM lines of output context 前后各#行

## 常用操作

# 搜索出含有"basic"字符串(-i:不区分大小写)

grep -i 'basic' testfile.txt

# 搜索出所有含有"basic"字符串并全部替换成"BasicInplace"字符串

grep -l 'basic' testfile.txt | xargs sed -i 's/basic/BasicInplace/g'

# 搜索出含有"basic"字符串(-i:不区分大小写),并打印行号

grep -i -n 'basic' testfile.txt

# 搜索出含有"basic"字符串(-i:不区分大小写),并打印行号,关键字"basic"颜色标记

grep -i -n --color 'basic' testfile.txt

# 打印"basic"字符串(-i:不区分大小写)显示的次数

grep -i -c 'basic' testfile.txt

# 打印'basic'字符串(-i:不区分大小写),但不打印整行

grep -i -o 'basic' testfile.txt

# 打印'basic'字符串和它上一行的信息

grep -B1 'basic' testfile.txt

# 打印'basic'字符串和它下一行的信息

grep -A1 'basic' testfile.txt

# 打印'basic'字符串和它下一行及下一行的信息

grep -C1 'basic' testfile.txt

# 精确匹配出'basic'字符串

grep -w 'basic' testfile.txt

# 匹配出不包含'basic'字符串

grep -v 'basic' testfile.txt

# 同时匹配'test'和'basic'字符串

grep -e 'test' -e 'basic' testfile.txt

# 显示/etc/passwd文件中不以/bin/bash结尾的行

grep -v "/bin/bash$" /etc/passwd

# 找出/etc/passwd文件中的两位数或三位数

grep "\<[0-9]\{2,3\}\>" /etc/passwd

egrep "\<[0-9]{2,3}\>" /etc/passwd

# 或操作:满足任意条件(word1、word2和word3之一)将匹配

grep -E "word1|word2|word3" testfile.txt

egrep 'word1|word2|word3' testfile.txt //用egrep同样可以实现

awk '/word1|word2|word3/' testfile.txt //awk 的实现方式

# 与操作:必须同时满足三个条件(word1、word2和word3)才匹配

grep word1 testfile.txt | grep word2 |grep word3

awk '{if(($0~"word1") && ($0~"word2")) printf("%d: %s\n", NR, $0)}' testfile.txt

# 其他格式操作

grep -i pattern files //不区分大小写地搜索。默认情况区分大小写，

grep -l pattern files //只列出匹配的文件名，

grep -L pattern files //列出不匹配的文件名，

grep -w pattern files //只匹配整个单词，而不是字符串的一部分（如匹配‘magic’，而不是‘magical’），

grep -C number pattern files //匹配的上下文分别显示[number]行

# 基本正则表达式的元字符

### 字符匹配

.：匹配任意单个字符;

[]：匹配指定范围内的任意单个字符;

[^]：匹配指定范围外的任意单个字符;

### 位置锚定

^：行首锚定;用于模式的最左侧;匹配开头的字符;

# grep '^root' /etc/passwd 匹配以root开头的行

$：行尾锚定;用于模式的最右侧;匹配结尾的字符；

# grep 'r.\*h$' /etc/passwd 匹配以r开头以h结尾的行

^$:空行

^[[:space:]]\*$:空行或包含空白字符的行;

单词：非特殊字符组成的连续字符(字符串)都称为单词；

\<或\b：词首锚定，用于单词模式的左侧;界定单词的左边界;

\<hello 用于匹配以hello开头的单词

\>或\b：词尾锚定，用于单词模式的右侧;

hello\> 用于匹配以hello结尾的单词

\<PATTERN\>：匹配完整单词;

\<hello\> 可精确匹配单词hello，而不是helloworld

### 分组及引用

\(\)：将一个或多个字符捆绑在一起，当作一个整体进行处理;括号不能交叉，但可以嵌套;

\(xy\)\*ab

注意：分组括号中的模式匹配到的内容会被正则表达式引擎自动记录于内部变量中，这些变量为：

\1：模式从左侧起，第一个左括号以及与之匹配的右括号之间的模式所匹配到的字符;

\2：模式从左侧起，第二个左括号以及与之匹配的右括号之间的模式所匹配到的字符;

\3:

...

# vim lovers.txt

he loves his lover.

he likes his lover.

she likes her liker.

she loves her liker.

# grep "\(l..e\).\*\1" lovers.txt

# grep "^\(r..t\).\*\1" /etc/passwd

### 后向引用：\1表示后向引用，引用前面第一个括号所匹配到的内容

\d：匹配一个数字;等价于[0-9];

\w：匹配字母、数字和下划线;

\W：匹配非字母、数字和下划线;

\n：换行符;

\r：回车;

\t：制表符;tab

\f：换页符;

\s：空白字符;

\S：非空白字符;

# Sed

sed是一种流编辑器，它是文本处理中非常中的工具，能够完美的配合正则表达式使用，功能不同凡响。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”（pattern space），接着用sed命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有 改变，除非你使用重定向存储输出。Sed主要用来自动编辑一个或多个文件；简化对文件的反复操作；编写转换程序等。

## 常用操作

1.定位行

sed命令用来处理文本，在处理前首先要找得到待处理的行，这是逻辑上必须的。所以需要首先定位，然后对定位到的各行进行各种处理，包括插入，删除，替换等。

sed -n '10p' testfile // sed命令默认会打印出经过处理后所有的文本，-n选项则不打印这种默认文本。10为要定位的行，找到之后执行p命令，打印这一行。

sed -n '1,10p' testfile // 打印1-10行。

sed -n '2,$p' testfile // 打印第二行到最后一行。

sed -n '3~2p' testfile // 打印第3，5，7，9，…….行，你懂得。

sed -n '/Barry/,10p' testfile // 首先找到第一个匹配/Barry/（可以是其他基本正则表达式）的行，打印从这一行到第10行。

sed -n '1,3!p' testfile // 打印1-3行以外的行("!"的作用)。

sed -n '1{n;p}' testfile // 首先定位到第一行，然后执行命令n，定位到第一行的下一行，然后执行p打印，可见定位之后可以用 {}执行命令组合。

sed -n 'p' testfile // 什么都不写，则定位全部的行。

采用a,b这种模式定位时，首先定位a（如果超出范围或者未找到，就不会去查找b），然后定位b，如果b这一行在a的上面会出现什么状况呢？例如：

sed -n '/Barry/,/Hello/p' testfile

该首先会定位到/Barry/,如果/Hello/这一行在/Barry/这一行上面，则只打印/Barry/这一行。为什么会这样？原来，sed定位是不回退的，即找到了/Barry/这一行之后，就从这一行下面找/Hello/，当然找不到了，所以只打印了/Barry/这一行。

2.命令

在定位之后当然是执行指定的命令了。看实例：

sed '1,3d' testfile // 删除1-3行，打印剩余的文本，注意，此时文件本身并没有改变。d命令是删除命令。

sed -n '5,10a helloworld' testfile // 在5-10行每一行后面都增加一行 helloworld。a命令在行后插入一行。

sed -n '5,10i helloworld' testfile // 在5-10行每一行前面都增加一行 helloworld。i命令在行前插入一行。

sed -n ‘5~3c helloworld’ testfile //把5，8，11，14，17，20，23，26……依次替换为helloworld。c

命令替换行。

sed -n '5,10c helloworld' testfile // 把5-10行整体替换成一行helloworld。当替换目标是连续的行时，把整体替换成一行。

sed -n -r '5,10s/yes|Yes/no/gp' testfile // 5-10行的yes或者Yes被替换成no。 -r 选项使sed支持扩展正则表达式。

s命令是替换命令，通常格式是line1,line2s/字符串1/字符串2/g，即把line1~line2的字符串1全部用字符串2代替，字符串1可以是正则表达式。这里不再对正则表达式详细介绍。如果末尾不加g字符，则表示只替换每行的第一个匹配的字符串。

sed '1,10s/$/helloworld/' testfile // 在1~10行中，替换每行的行尾设置为helloworld。$是正则表达式的行尾标志。

sed '5,10y/abc/ABC/' testfile // 5-10行的a被换成A，b被换成B，c被换成C，类似tr命令的功能。y命令替换以字符为单位，而s命令以字符串为单位。

sed '/My/r textfile' testfile // r命令是读命令，sed使用该命令将一个文本文件中的内容加到当前行的下面。

sed -n '1,10w newfile' testfile // 将1-10行写入newfile。w命令可以将指定的行写入指定的文件。

sed -n '/root/{=;p}' testfile // 定位到/root/行之后，打印行号，然后打印这一行。"="是打印行号的命令。

3.特殊选项

sed -i '1,10d' testfile // 直接在文件中删除1~10行。-i选项表示对真实文件进行操作，所做的更改将保存到文件中。

sed -s '1d' testfile1 testfile2 testfile3 // 此处有三个文件，如果不加-s，则将三个文件读入，当作一个整体处理，-s选项使分别处理。

sed -e '/root/p' -e '/Barry/d' testfile // 此处用-e选项可以进行多次处理，第一次打印/root/行，第二次删除/Barry/行。

sed -e '/root/{n;p}' -e '/Barry/{n;d}' testfile // 每次处理时，可以对每次匹配到的行执行多个命令。命令之间用";"分隔。

# 配置开机字启用脚本

## 方法一：修改/etc/rc.local

$ vim /root/scripts/auto.sh

$ cat /root/scripts/auto.sh

#!/bin/bash

/bin/echo $(/bin/date +%F\_%T) >> /tmp/auto.log

# /etc/rc.local，该文件为链接文件

# 修改/etc/rc.local文件,在最后一行加入:

/bin/bash /root/scripts/auto.sh 2>&1

$ tail -n 1 /etc/rc.local

# 修改完成后重启查/tmp/auto.log

## 方法二：chkconfig管理

$ vim /etc/init.d/test

#!/bin/bash

/bin/bash /root/scripts/auto.sh

$ chmod +x /etc/init.d/test

# 添加到chkconfig，开机自启动

$ chkconfig --add test

$ chkconfig --list test

test 0:off 1:off 2:off 3:on 4:off 5:off 6:off

# 从chkconfig管理中删除test

$ chkconfig --del test

# Vim的分屏功能

## 在shell里打开几个文件并且分屏

# 大O表示垂直分割(vertical),小o表示水平分割(默认horizontal),后面的n表示分几个屏,实际上我觉得不用写,默认按后面要分割的文件数来决定分几个屏

vim -On file1 file2 ...

vim -on file1 file2 …

## 在vim里打开一个分屏

# 创建空白分屏

:new

# 打开任意文件：

:vsplit(:vsp) filename

:sp(split) filename

# 打开当前文件：

ctrl+w 和 s(split)

ctrl+w 和 v(vsplit)

## 关闭一个分屏

# 只保留当前

:only

# 只剩最后一个分屏以后推出：

ctrl+w 和 q(quit)

## 在文件之间切换

# 文件切换

Ctrl+6 //两文件间的切换

:bn //下一个文件

:bp //上一个文件

:ls //列出打开的文件，带编号

:b1~n //切换至第n个文件

# 移动光标，也就是切换分屏；也可以移动分屏，比如将左分屏移动到右边。

ctrl+w 和 w（各种切换，只有两个分屏的时候还是比较方便的）

ctrl+w 和 h(H) 左

ctrl+w 和 j(J) 下

ctrl+w 和 k(K) 上

ctrl+w 和 l(L) 右

## 改变分屏尺寸的操作

ctrl+w 和 < 左

ctrl+w 和 > 右

ctrl+w 和 + 上

ctrl+w 和 - 下

ctrl+w 和 = 恢复均等

## 直接打开目录

# vim不仅可以打开文件，而且可以打开目录

:e ./harttle " 编辑该目录

:Explore . " 浏览该目录

:Sexplore . " 在水平分割窗口中浏览该目录

:Vexplore . " 在垂直分割窗口中浏览该目录

## 万能方法

# 如果你找不到合适的Vim命令，随时都可以在Vim中直接执行Bash命令。只需要在Ex模式中添加前导的!字符，例如：

:!ls //列出文件

:!rm foo.txt //删除文件

# 查看系统信息

uname -a # 查看内核/操作系统/CPU信息的linux系统信息

head -n l /etc/issue # 查看操作系统版本

cat /proc/cpuinfo # 查看CPU信息

hostname # 查看计算机名的linux系统信息命令

lspci -tv # 列出所有PCI设备

lsusb -tv # 列出所有USB设备的linux系统信息命令

lsmod # 列出加载的内核模块

env # 查看环境变量资源

free -m # 查看内存使用量和交换区使用量

df -h # 查看各分区使用情况

du -sh # 查看指定目录的大小

grep MemTotal /proc/meminfo # 查看内存总量

grep MemFree /proc/meminfo # 查看空闲内存量

uptime # 查看系统运行时间、用户数、负载

cat /proc/loadavg # 查看系统负载磁盘和分区

mount | column -t # 查看挂接的分区状态

fdisk -l # 查看所有分区

swapon -s # 查看所有交换分区

hdparm -i /dev/hda # 查看磁盘参数(仅适用于IDE设备)

dmesg | grep IDE # 查看启动时IDE设备检测状况网络

ifconfig # 查看所有网络接口的属性

iptables -L # 查看防火墙设置

route -n # 查看路由表

netstat -lntp # 查看所有监听端口

netstat -antp # 查看所有已经建立的连接

netstat -s # 查看网络统计信息进程

ps -ef # 查看所有进程

top # 实时显示进程状态用户

w # 查看活动用户

id # 查看指定用户信息

last # 查看用户登录日志

cut -d: -f1 /etc/passwd # 查看系统所有用户

cut -d: -f1 /etc/group # 查看系统所有组

crontab -l # 查看当前用户的计划任务服务

chkconfig –list # 列出所有系统服务

chkconfig –list | grep on # 列出所有启动的系统服务程序

rpm -qa # 查看所有安装的软件包

cat /proc/cpuinfo ：查看CPU相关参数的linux系统命令

cat /proc/partitions ：查看linux硬盘和分区信息的系统信息命令

cat /proc/meminfo ：查看linux系统内存信息的linux系统命令

cat /proc/version ：查看版本，类似uname -r

cat /proc/ioports ：查看设备io端口

cat /proc/interrupts ：查看中断

cat /proc/pci ：查看pci设备的信息

cat /proc/swaps ：查看所有swap分区的信息