第09章 文本编辑器

讲师：武永亮

课程目标

- 了解CentOS的基本文本命令
- 掌握Vi及Vim的基本操作
- 掌握neno的基本操作

课程内容

- CentOS文本文件操作
- Vi及Vim文本编辑器
- 其他常用文本编辑器

常用的文本文件提取命令

命令	功能
cat、tac	滚屏显示文本文件内容
more、less	分屏显示文本文件内容
head、tail	显示文本文件的前或后若干行（横向截取文本文件内容）
cut	纵向切割出文本指定的部分（纵向截取文本文件内容）
grep	在文本文件中查找指定的字符串（按关键字提取文本文件中匹配的行）

文本显示命令举例

命令	举例
cat /etc/passwd	滚屏显示文件/etc/passwd的内容
cat -n /etc/passwd	滚屏显示文件/etc/passwd的内容，并显示行号
more /etc/passwd	分屏显示文件/etc/passwd的内容（注意空格键、回车键和q的使用）
more +10 /etc/passwd	从第10行分屏显示文件/etc/passwd的内容
less /etc/passwd	分屏显示文件/etc/passwd的内容（注意空格键、回车键、PgDn键、PgUp键和q的使用）
head –n 4 myalllist	显示文件myalllist前4行的内容
tail –n 4 myalllist	显示文件myalllist后4行的内容
tail –n +10 myalllist	显示文件myalllist从10行开始到文件尾的内容
tail -f /var/log/messages	跟踪显示不断增长的文件结尾内容（通常用于显示日志文件）

grep 简介

- grep（global search regular expression）是一个强大的文本搜索工具。grep 使用正则表达式搜索文本，并把匹配的行打印出来。
- UNIX 的 grep 家族包括 grep、egrep 和 fgrep：
  - ✓grep 使用 Basic regular expression (BRE) 书写匹配模式，等效于 grep -G
  - ✓egrep 使用 Extended regular expression (ERE) 书写匹配模式，等效于 grep -E
  - ✓fgrep 不使用任何正则表达式书写匹配模式(以固定字符串对待)，执行快速搜索，等效于 grep -F

grep 命令

- 格式
  - ✓grep [options] PATTERN [FILE...]
- 说明
  - ✓PATTERN 是查找条件
    - 可以是普通字符串
    - 可以是正则表达式，通常用单引号将RE括起来。
  - ✓FILE 是要查找的文件，可以是用空格间隔的多个文件，也可是使用Shell的通配符在多个文件中查找PATTERN，省略时表示在标准输入中查找。
  - ✓grep命令不会对输入文件进行任何修改或影响，可以使用输出重定向将结果存为文件。

grep命令选项

选项	说明
-c	只显示匹配行的次数
-i	搜索时不区分大小写
-n	输出匹配行的行号
-v	输出不匹配的行（反向选择）
-r	对目录（子目录）的所有文件递归地进行
-l	列出匹配PATTERN的文件名
--color=auto	对匹配内容高亮显示
-A NUM	同时输出匹配行的后 NUM 行
-B NUM	同时输出匹配行的前 NUM 行
-C NUM	同时输出匹配行的前、后各 NUM 行

grep正则表达式元数据

元数据	意义和范例
^word	搜寻以word开头的行。 例如：grep –n ‘^#’ regular.txt　搜寻以#开头的脚本注释行
word$	搜寻以word结束的行 例如：grep –n ‘.$’ regular.txt　搜寻以‘.’结束的行
.	匹配任意一个字符。 例如：grep –n ‘e.e’ regular.txt 匹配e和e之间有任意一个字符，匹配eee，eae，eve，但是不匹配ee
\	转义字符。 例如：grep –n ‘\’’ regular.txt　搜索单引号
*	前面的字符重复0到多次。 例如:grep –n ‘go*gle’ regular.txt　匹配gle，gogle，google，gooogle等等
[list]	匹配一系列字符中的一个。 例如：grep –n ‘g[lf]’ regular.txt　可匹配gl，gf。

grep正则表达式元数据（续）

元数据	意义和范例
[n1-n2]	匹配一个字符范围中的一个字符。 例如：grep –n ‘[0-9]’ regular.txt　　匹配单个数字字符
[^list]	匹配字符集以外的字符 例如：grep –n ‘[^o]‘ regular.txt　　匹配非o字符
\{n1,n2\}	前面的字符重复n1或n2次 例如：grep –n ‘go\{2,3\}gle’ regular.txt　　匹配google，gooogle。
\<word	单词是的开头。 例如：grep –n ‘\<g’ regular.txt　　匹配以g开头的单词
word\>	匹配单词结尾 例如：grep –n ‘tion\>’ regular.txt　　匹配以tion结尾的单词

grep扩展正则表达式（补）

元数据	意义和范例
+	重复前面字符1到多次。 例如：grep –nE ’go+d’ regular.txt　匹配god，good，goood等等字符串。
?	匹配0或1次前面的字符 例如：grep –nE ‘go?d’ regular.txt　匹配gd，god
|	或（or）的方式匹配多个字串 例如：grep –nE ‘god|good’ regular.txt　匹配god或者good
()	匹配整个括号内的字符串，原来都是匹配单个字符 例如：grep –nE ‘g(oo|la)’ regular.txt　搜寻good或者glad

grep命令举例

- 在文件 myfile 中查找包含字符串 mystr的行
  - ✓$ grep mystr myfile
- 显示 myfile 中第一个字符为字母的所有行
  - ✓$ grep '^[a-zA-Z]' myfile
- 在文件 myfile 中查找首字符不是 # 的行（即过滤掉注释行）
  - ✓$ grep -v '^#' myfile

grep命令举例

- 搜索日志，查询有多少条503错误
  - ✓grep -c '503' /var/log/httpd/error_log-20141116
- 搜索含有 error 字样的行，并且输出行号
  - ✓grep -n 'error' /var/log/httpd/error_log-20141116
- 搜索没有 error 字样的行，并且输出行号
  - ✓grep -nv 'error' /var/log/httpd/error_log-20141116
- 过滤配置文件的注释符号#
  - ✓grep -v '#' /etc/httpd/conf/httpd.conf

常用的文本文件分析命令

命令	功能
wc	统计文本
sort	以行为单位对文本文件排序
uniq	删除文本文件中连续的重复的行
diff	比较两个文本文件的差异
diff3	比较三个文本文件的差异
patch	为文本文件打补丁
aspell	为文本文件做拼写检查（西文）

wc命令

- 功能：统计文本文件的行数、字数、字符数
- 格式：wc [选项] [<文件> …]
- 举例
  - ✓$ wc file
  - ✓$ wc -l file　　# 统计行数
  - ✓$ wc -w file　　# 统计字数
  - ✓$ wc -c file　　# 统计字符数
  - ✓$ wc -L file　　# 统计最长一行的长度

sort命令

- 功能：以行为单位对文件进行排序
- 格式：sort [选项] [<文件> …]
- 选项

-r	逆向排序
-f	忽略字母的大小写
-n	根据字符串的数值进行排序
-u	对相同的行只输出一行
-t c	选项使用c做为列的间隔符
-b	忽略前导的空格
-i	只考虑可打印字符
-k N	以第N列进行排序（默认以空格或制表符作为列的间隔符）

sort命令举例

- $ sort file
- $ sort -n file
- $ sort -fr file
- $ sort -u file
- $ sort file1 file2
- $ sort -br file1 file2
- $ sort -n -k 3 -t ':' /etc/passwd

常用的文本文件处理命令

命令	功能
tr	字符替换
sed	流编辑器，常用于字符串替换
paste	纵向合并多个文本
expand	将文件中的制表符转换为空格
unexpand	将文件中的空格转换为制表符
dos2unix	将DOS格式的文本转换成UNIX格式
unix2dos	将UNIX格式的文本转换成DOS格式
iconv	将文本从一种编码转换成另一种编码

sed 简介

- sed 是一个流编辑器（stream editor）。sed 是一个非交互式的行编辑器，它在命令行中输入编辑命令、指定被处理的输入文件，然后在屏幕上查看输出。输入文件可以是指定的文件名，也可以来自一个管道的 输出。
- 与 vi 不同的是 sed 能够过滤来自管道的输入。在 sed 编辑器运行的时候不必人工干涉，所以 sed 常常被称作批编辑器 。
- sed 默认不改变输入文件的内容，且总是将处理结果输出到标准输出，可以使用输出重定向将 sed 的输出保存到文件中。

sed 的工作方式

- sed 以按顺序逐行的方式工作，过程为：
  - ✓从输入读取一行数据存入临时缓冲区，此缓冲区称为模式空间（pattern space）
  - ✓按指定的 sed 编辑命令处理缓冲区中的内容
  - ✓把模式空间的内容送往屏幕并将这行内容从模式空间中删除
  - ✓读取下面一行。重复上面的过程直到全部处理结束。

sed命令

- 格式
  - ✓ sed [选项] [-e] cmd1 [[-e cmd2] ... [-e cmdn]] [input-file]...
- 说明
  - ✓ 在命令行上执行sed编辑命令。可以指定多个编辑命令，每个编辑命令前都要使用 -e 参数，sed 将对这些编辑命令依次进行处理。若只有一个编辑命令时，-e 可以省略。
  - ✓ 每个sed的编辑命令cmdX均应使用单引号括起来。
  - ✓ input-file：sed 处理的文件列表，若省略，sed 将从标准输入中读取输入，也可以从输入重定向或管道获得输入。
- 选项
  - ✓ -r：使用扩展正则表达式进行模式匹配
  - ✓ -i：直接对输入文件进行sed的命令操作

sed命令举例

sed -i 's/Windows/Linux/g‘ myfile	将myfile中的所有Windows替换成Linux
sed 's/Windows/Linux/g' myfile	功能同上，但不对输入文件本身进行替换，仅在屏幕输出结果
sed 's/cc*/c/g' myfile	将myfile中所有连续出现的c都压缩成单个的c，仅在屏幕输出结果
sed 's/^[\t]*//' myfile	删除myfile中每一行前导的连续“空白字符”（空格，制表符），仅在屏幕输出
sed 's/ *$//' myfile	删除myfile中每行结尾的所有空格，仅在屏幕输出
sed 's/^$//' myfile	删除所有空白行，仅在屏幕输出
sed 's/^/> /' myfile	在每一行开头加上一个尖括号和空格（引用信息），仅在屏幕输出
sed 's/.*\///' myfile	删除路径前缀，仅在屏幕输出

iconv命令

- 功能：将文件从一种编码转换成另一种编码
- 格式：iconv [选项] <输入文件>
- 选项
  - ✓-f <encoding> : 指定原始文本编码。
  - ✓-t <encoding> : 指定要转换的编码。
  - ✓-o <output file> : 指定输出文件，而不是在标准输出上显示。
  - ✓-l : 列出所有已知编码字符集。

iconv命令举例

- 将编码为GB2312的inputfile文件转化为UTF-8编码的outputfile
  - ✓$ iconv -f GB2312 -t UTF-8 -o outputfile inputfile
- 又如
  - ✓$ iconv -l
  - ✓$ iconv -f ISO-8859-1 -t UTF-8 -o outputfile inputfile
  - ✓$ iconv -f GBK -t UTF-8 -o outputfile inputfile
  - ✓$ iconv -f BIG5 -t UTF-8 -o outputfile inputfile
  - ✓$ iconv -f UTF-8 -t GB2312 -o outputfile inputfile

课程内容

- CentOS文本文件操作
- Vi及Vim文本编辑器
- 其他常用文本编辑器

Vi简介

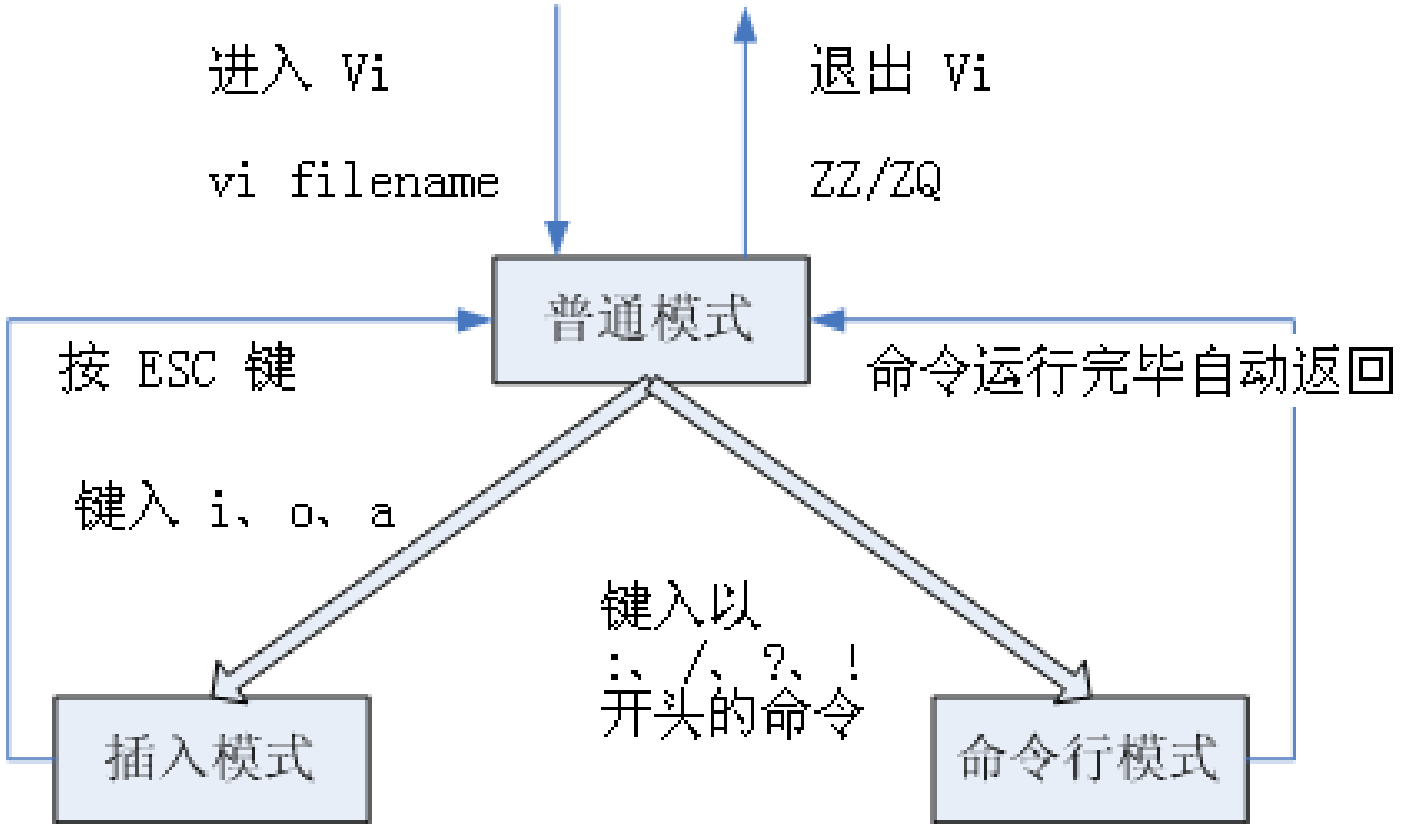
- Vi 是 “Visual interface” 的简称，它可以执行输出、删除、查找、替换、块操作等文本操作，而且用户可以根据用户的需要对其进行定制。
- Vi 不是一个排版程序，它不像 MS Word 或 WPS 那样可以对字体、格式、段落等其他属性进行编排，它只是一个文本编辑程序。
- Vi 是全屏幕文本编辑器，它没有菜单，只有命令。
- Vim 即 Vi IMproved，Vi 克隆版本之一。

进入Vi模式

命令	说明
vi	直接进入
vi filename	打开或新建文件filename，并将光标置于第一行首
vi +n filename	打开文件filename，并将光标置于第n行首
vi + filename	打开文件filename，并将光标置于最后一行首
vi +/pattern filename	打开文件filename，并将光标置于第一个与pattern匹配的串处
vi -r filename	打开上次用vi编辑时发生系统崩溃，恢复filename

Vi 的3种运行模式

- 普通(Normal)模式
- 插入(Insert)模式
- 命令行(Cmdline)模式



Vi 的 Normal 模式

- 在Shell中输入Vi启动编辑器时，即进入该模式。
  - ✓不管用户处于何种模式，只要按一下Esc 键，即可使 Vi 进入 Normal 模式
- 在该模式下，用户可以通过Vi命令，用于管理自己的文档。此时从键盘上输入的任何字符都被当做编辑命令来解释。
- 若输入的字符是合法的 Vi命令，则 Vi在接受用户命令之后完成相应的动作。但需注意的是，所输入的命令并不在屏幕上显示出来。若输入的字符不是 Vi的合法命令，Vi会响铃报警。

Normal模式下的基本操作

- a 在当前光标位置的右边添加文本
- i 在当前光标位置的左边添加文本
- A 在当前行的末尾位置添加文本
- I 在当前行的开始处添加文本(非空字符的行首)
- O 在当前行的上面新建一行
- o 在当前行的下面新建一行
- R 替换(覆盖)当前光标位置及后面的若干文本
- J 合并光标所在行及下一行为一行(依然在命令模式)

Normal模式下的基本操作

- G 用于直接跳转到文件尾
- x 删除光标所在的字符
- r 替换光标所在的字符
- ~ 切换光标所在字母的大小写
- dd、YY、p分别用于剪切、复制和粘贴一行文本
- u 取消上一次编辑操作（undo）
- . 重复上一次编辑操作（redo）
- ZZ 用于存盘退出Vi
- ZQ用于不存盘退出Vi

Vi 的 Insert 模式

- Normal 模式下输入插入命令 i、附加命令 a 、打开命令 o、修改命令 c、取代命令 r 或替换命令 s 等都可以进入 Insert 模式。
- 在该模式下，输入的字符都被Vi当做文件内容保存起来，并将其显示在屏幕上。在文本输入过程中，若想回到Normal模式下，按 Esc 键即可。

Vi 的 Command 模式

- Normal模式下，按冒号“:” ，进入 Command 模式，此时 Vi会在显示窗口的最后一行 (屏幕的最后一行) 显示一个 “:” 作为 Command 模式的提示符，等待输入命令。
- 常用命令如下：

Command 模式下的基本操作

- :w　保存当前编辑文件，但并不退出
- :w newfile 存为另外一个名为 “newfile” 的文件
- :wq 用于存盘退出Vi
- :q!用于不存盘退出Vi
- :q 用于直接退出Vi（未做修改）
- /和？用于查找字符串，查找过程中通过n查找下一处

Command 模式——设置 Vi 环境

- :set autoindent 缩进,常用于程序的编写
- :set noautoindent 取消缩进
- :set number 在编辑文件时显示行号
- :set nonumber 不显示行号
- :set tabstop=value 设置显示制表符的空格字符个数
- :set 显示设置的所有选项
- :set all 显示所有可以设置的选项

Vim

- Vim是一个基于Vi的功能强大、高度可定制的文本编辑器，在Vi的基础上改进和增加了很多特性。Vim是自由软件。
- Vim普遍被推崇为类Vi编辑器中最好的一个。1999 年Emacs被选为Linuxworld文本编辑分类的优胜者，Vim屈居第二。但在2000年2月Vim赢得了Slashdot Beanie的最佳开放源代码文本编辑器大奖，又将Emacs推至二线， 总的来看， Vim和Emacs在文本编辑方面都是非常优秀的。

Vim

- Vim新特征：
  - ✓多级撤消：Vi中按 u只能撤消上次命令，vim里可以无限撤消。
  - ✓易用性：Vi只能运行于Unix中, Vim可以运行于Unix,Windows ,Mac等多操作平台。
  - ✓语法加亮:Vim可以用不同的颜色来加亮你的代码。
  - ✓可视化操作:就是说Vim不仅可以在终端运行，也可以运行于x Window、 Mac os、 Windows。
  - ✓对vi的完全兼容：一般把Vim当成Vi来使用。

Vim

- Vim的安装
  - ✓yum -y install vim
- Vim快捷方式

移动光标相关

1、左移h、右移l、下移j、上移k
2、向下翻页ctrl + f，向上翻页ctrl + b
3、向下翻半页ctrl + d，向上翻半页ctrl + u
4、移动到行尾$，移动到行首0（数字），移动到行首第一个字符处^
5、移动光标到下一个句子 ），移动光标到上一个句子（
6、移动到段首{，移动到段尾}
7、移动到下一个词w，移动到上一个词b
8、移动到文档开始gg，移动到文档结束G
9、移动到匹配的{}.().[]处%
10、跳到第n行 ngg 或 nG 或 :n
11、移动光标到屏幕顶端H，移动到屏幕中间M，移动到底部L
12、读取当前字符，并移动到本屏幕内下一次出现的地方 *
13、读取当前字符，并移动到本屏幕内上一次出现的地方 #

Vim

- Vim的安装
  - ✓yum -y install vim
- Vim快捷方式

查找替换相关

1、光标向后查找关键字 #或者g#
2、光标向前查找关键字 *或者g*
3、当前行查找字符 fx, Fx, tx, Tx
4、基本替换 :s/s1/s2 （将下一个s1替换为s2）
5、全部替换 :%s/s1/s2
6、只替换当前行 :s/s1/s2/g
7、替换某些行 :n1,n2 s/s1/s2/g
8、搜索模式为 /string，搜索下一处为n，搜索上一处为N
9、制定书签 mx, 但是看不到书签标记，而且只能用小写字母
10、移动到某标签处 `x，1旁边的键
11、移动到上次编辑文件的位置 `.
PS：.代表一个任意字符 *代表一个或多个字符的重复
正则表达式的内容将会在后续文章中整理

Vim

- Vim的安装
  - ✓yum -y install vim
- Vim快捷方式

编辑操作相关

1、光标后插入a, 行尾插入A
2、后插一行插入o，前插一行插入O
3、删除字符插入s， 删除正行插入S
4、光标前插入i，行首插入I
5、删除一行dd，删除后进入插入模式cc或者S
6、删除一个单词dw，删除一个单词进入插入模式cw
7、删除一个字符x或者dl，删除一个字符进入插入模式s或者cl
8、粘贴p，交换两个字符xp，交换两行ddp
9、复制y，复制一行yy
10、撤销u，重做ctrl + r，重复.
11、智能提示 ctrl + n 或者 ctrl + p
…

Vim

- Vim的安装
  - ✓yum -y install vim
- Vim快捷方式

窗口操作相关

1、分隔一个窗口:split或者:vsplit
2、创建一个窗口:new或者:vnew
3、在新窗口打开文件:sf {filename}
4、关闭当前窗口:close
5、仅保留当前窗口:only
6、到左边窗口 ctrl + w, h
7、到右边窗口 ctrl + w, l
8、到上边窗口 ctrl + w, k
9、到下边窗口 ctrl + w, j
10、到顶部窗口 ctrl + w, t
11、到底部窗口 ctrl + w, b…

课程内容

- CentOS文本文件操作
- Vi及Vim文本编辑器
- 其他常用文本编辑器

常用的文本编辑器

- Vim：Vi编辑器的扩展，大部分Linux的都支持
- nano：更简单的文本编辑器
- Emacs：Emacs，著名的集成开发环境和文本编辑器
- gedit:gedit是一个GNOME桌面环境下兼容UTF-8的文本编辑器。

nano

- nano是一个字符终端的文本编辑器，比vi/vim要简单得多，比较适合Linux初学者使用。某些Linux发行版的默认编辑器就是nano。 nano命令可以打开指定文件进行编辑，默认情况下它会自动断行。

nano

- 在线安装：
  - ✓执行yum -y install nano
- 语法
  - ✓nano [选项] [[+行,列] 文件名]...
- 选项
  - ✓-A –smarthome 开启智慧型 HOME 按键功能
  - ✓-B –backup 储存既有文件的备份
  - ✓-C <目录> –backupdir=<目录> 用以储存独一备份文件的目录
  - ✓-D –boldtext 用粗体替代颜色反转

nano的用法

- 光标控制
  - ✓移动光标：使用用方向键移动。
  - ✓选择文字：按住鼠标左键拖到。
- 快捷键
  - ✓复制一整行：Alt+6
  - ✓剪贴一整行：Ctrl+K
  - ✓粘贴：Ctrl+U
  - ✓搜索：按Ctrl+W
  - ✓翻页：Ctrl+Y到上一页，Ctrl+V到下一页
  - ✓保存：使用Ctrl+O来保存所做的修改
  - ✓退出：按Ctrl+X

课程总结

- CentOS文本文件操作
- Vi及Vim文本编辑器
- 其他常用文本编辑器

本章思考题

- Vi的3种运行模式是什么？如何切换？
- Vim的基本用法及操作快捷键？
- neno文本编辑器的基本用法及操作快捷键

THANK YOU!