训练稿——Linux常用重点命令

目录

[简介 1](#_Toc323592916)

[调用方式 2](#_Toc323592917)

[选项 2](#_Toc323592918)

[命令集合 2](#_Toc323592919)

[寻址 3](#_Toc323592920)

[基本用法 4](#_Toc323592921)

[文件读入写出 5](#_Toc323592922)

[附加-插入-修改文本 5](#_Toc323592923)

[删除文本 6](#_Toc323592924)

[替换文本 6](#_Toc323592925)

[转换文本 8](#_Toc323592926)

[补充 8](#_Toc323592927)

#### 简介

|  |
| --- |
| 简介 |
| 一个“非交互式的”字符流编辑器（stream editor）  Sed是一非交互性文本编辑器，它编辑文件或标准输入导出的文本拷贝，标准输入可能来自键盘、文件、重定向、字符串、变量或一管道文件  sed主要用来自动编辑一个或多个文件（替换、插入、删除、追加、更改……）；简化对文件的反复操作；编写转换程序等  【对进入的数据进行全局的处理，增加，删除，修改某些部分，得出结果】 |
| 处理过程 |
| 1.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）  2.接着用sed命令处理缓冲区中的内容，  3.处理完成后，把缓冲区的内容送往屏幕。  4.接着处理下一行或多行，这样不断重复，直到文件末尾  文件内容并没有改变，除非你使用重定向或写入命令存储输出 |
| 主要作用： |
| 1. 抽取域 2. 匹配正则表达式 3. 比较域 4. 增加、附加、替换 |

#### 调用方式

|  |
| --- |
| 调用方式： |
| 1. 命令行输入   sed [options] 'command' file(s)   1. 使用sed脚本   sed [options] -f scriptfile file(s)   1. 使sed脚本可执行   在脚本第一行具有sed命令解释器  Sed脚本文件 [选项] 输入文件  sed –f sedScriptFile targetFile |

#### 选项

|  |  |
| --- | --- |
| 选项: | |
| -n | --quiet, --silent 取消默认输出  不打印，不写编辑行到标准输出，缺省情况下打印所有行[编辑/未编  辑]p命令可以打印编辑行 |
| -f | 调用sed脚本 sed –f sedScriptFile targetFile |
| -c | 下一命令是编辑命令，使用多项编辑时加入此选项 |
| -e command | --expression=command    允许多条编辑命令 |
| -h | --help打印帮助，并显示bug列表的地址 |
| -V | --version 打印版本和版权信息 |

#### 命令集合

|  |  |
| --- | --- |
| 命令集合: | |
| a\ | 定位行号后附加新文本信息 append |
| b lable | 分支到脚本中带有标记的地方，如果分支不存在则分支到脚本的末尾 |
| c\ | 用新的文本改变本行的文本,用新文本替换定位文本 change |
| d | 从模板块（Pattern space）位置删除行 删除定位行 delete |
| D | 删除模板块的第一行 |
| i\ | 定位行号后插入 insert |
| h | 拷贝模板块的内容到内存中的缓冲区 |
| H | 追加模板块的内容到内存中的缓冲区 |
| g | 获得内存缓冲区的内容，并替代当前模板块中的文本 |
| G | 获得内存缓冲区的内容，并追加到当前模板块文本的后面 |
| l | 列表不能打印字符的清单 将非打印字符显示为两个数字的ASCII代码 |
| n | 读取下一个输入行，用下一个命令处理新的行而不是用第一个命令 |
| N | 追加下一个输入行到模板块后面并在二者间嵌入一个新行，改变当前行号码 |
| p | 打印匹配行 print |
| P（大写） | 打印模板块的第一行 |
| q | 退出Sed 第一个模式匹配完成后退出或立即退出 |
| r file | 从file中读行 从另一个文件中读文本 read |
| t label | **if**分支，从最后一行开始，条件一旦满足或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾 |
| T label | 错误分支，从最后一行开始，一旦发生错误或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾 |
| w file | 写并追加模板块到file末尾 写文本到一个文件 write |
| W file | 写并追加模板块的第一行到file末尾 |
| y | 传送字符 |
| ! | 表示后面的命令对所有没有被选定的行发生作用 |
| s/re/string | 用string替换正则表达式re 使用替换模式替换相应模式 |
| = | 打印匹配行行号 |
| # | 把注释扩展到下一个换行符以前 |
| {} | 定位执行命令组 |
| 替换标记 | １）g表示行内全面替换。  ２）p表示打印行。  ３）w表示把行写入一个文件。  ４）x表示互换模板块中的文本和缓冲区中的文本。  ５）y表示把一个字符翻译为另外的字符（但是不用于正则表达式） |

#### 寻址

|  |  |
| --- | --- |
| 寻址: | |
| sed 命令可以指定零个、一个或两个地址。每个地址都是一个描述模式、行号、或者行寻址  符号的正则表达式。  如果没有指定地址，那么命令将应用于每一行。  如果只有一个地址，那么命令将应用于与这个地址匹配的任意行。  如果指定了由逗号分隔的两个地址，那么命令应用于匹配第一个地址的第一行和他后面的  行，直到匹配第二个地址的行（包括此行）。  如果地址后面跟有感叹号（!），那么命令就应用于不匹配该地址的所有的行。 | |
| 定位命令 | |
| x | 行x |
| X,y | 行x到行y |
| /pattern/ | 模式 |
| /pattern/pattern/ | 两个模式 |
| /pattern/,x | 模式+行【在给定行号上查询模式】 |
| X,y /pattern/ | 通过行号和模式查询匹配行 |
| X,y! | 不包含指定行号 |

#### 基本用法

|  |
| --- |
| 示例：[dream为数据文件] |
| 基本用法——匹配&打印 |
| 1. 显示第二行   $sed –n ‘2p’ dream   1. 显示一到三行   $sed –n ‘1,3p’ dream   1. 匹配模式   $sed –n ‘/dreamb/’p dream   1. 使用模式和行号进行匹配   $sed –n ‘4,/The/’p dream  【只能加一个行号，不能变为区间】   1. 打印匹配行行号   $sed –n ‘/dream/=’ dream   1. 匹配元字符   $sed –n ‘/\$/’p dream   1. 显示整个文件   $sed –n ‘1,$p’ dream   1. 匹配字符串   $sed –n ‘/.\*ing/’p dream   1. 首行   $sed –n ‘1p’ dream   1. 行号打印   $sed –e ‘/music/=’ quote.txt 整个文件并打印匹配行号  $sed –n ‘/music/=’ quote.txt 只打印匹配行号  sed –n “/^title/p” auction.xml#精确匹配以title开头，打印此行  sed -n "/pict/,5p" auction.xml #指定具体行匹配  sed –n “/\<doc\>/p” auction.xml #匹配正则表达式  sed –n “/.\*is/p” auction.xml#匹配任意字母  sed -n -e "/^title/p" -e "/^title/=" auction.xml #打印行号及匹配行 |
| 文件读入写出 |
| 文件读入写出 |
| 读取文件:r命令  $ sed '/test/r file' example  file里的内容被读进来，显示在与test匹配的行后面，如果匹配多行，则file的内容将显示在所有匹配行的下面。  写入文件：w命令    可以使用重定向  $ sed -n '/test/w file' example  在example中所有包含test的行都被写入file里  注： 在命令和文件名之间必须有一个空格。每个脚本最多只能打开10个文件 |
| 附加-插入-修改文本 |
| 附加/插入/修改文本 |
| 使用符号a\，可以将指定文本一行或多行附加到指定行，若不指定放置位置，默认放入第一行。  注：附加操作，结果到标准输出，不能被编辑，必须存到另一个文件，再运行一个sed编辑  插入文本 i\ 修改文本 c\   1. 匹配行后插入一行   $sed ‘/dreamb/a\”appended line”’ dream  【会自动换行】   1. 插入文本   $sed ‘2 i\ “inserted line”’ dream  【在第二行之前加一行】  sed "/user/ i\test" auction.xml  在匹配行前插入test   1. 修改文本   $sed ‘3 c\ “changed line”’ dream  【整行替换掉】  【注：可以用行,/正则/ 搞定】 |
| 删除文本 |
| 删除文本 |
| 命令格式: [address[,address]] d   1. 删除行   $sed ‘1d’ dream  $sed ‘1,3d’ dream  $ sed '2,$d' example #第二行到末尾所有行  $ sed '$d' example-----删除example文件的最后一行   1. 删除匹配行   $sed ‘/dreamb/d’ dream  $sed –n ‘/BEGIN/,/END/d’ myfile |
| 替换文本 |
| 替换文本 |
| 格式：[address[,address]] s/pattern-find/replacement-pattern/[g,p,w,n]  n 1到512之间的一个数字，表示对本模式中指定模式第n次出现的情况进行替换。  g 对模式空间所有出现的情况进行全局更改【缺省只替换首次出现的模式 】  p 打印模式空间的内容  w file |
| 1. 替换   $sed ‘s/dreamb/DREAMB’ dream  【问题：这里是如何执行的?报错】   1. 全局替换   $sed ‘s/The/Wow!/g’ dream   1. 替换后重定向到文本   $sed ‘s/dreamb/Dream/w sed.out’ dream  【只输出替换行】   1. 读取部分写入另一个文件   $sed ‘1,2 w filedt’ dream  $sed ‘/dream/ w filedt’ dream  【注： 在命令和文件名之间必须有一个空格。每个脚本最多只能打开10个文件。】   1. 首次匹配后退出   $sed ‘/dream/q’ dream  【遇到匹配之前的内容还是要输出的】   1. 显示文件中的控制字符   格式: [ address[,address]]l  $sed –n ‘1,$l’ dream 【小写字母l】   1. 去除行首数字   $sed ‘s/^[0-9]//g’ dream   1. Shell向sed传值   $NAME=”go there”  $REP=”GO”  $echo $NAME | sed “s/go/$REP/g” 这里必须双引号【此时sed单引号不被解析】   1. 使用替换修改字符串(title前面加test)（注意和附件文本不要混淆）   sed -n "s/^title/test &/p" auction.xml   1. $ sed 's#10#100#g' example   不论什么字符，紧跟着s命令的都被认为是新的分隔符，所以，“#”在这里是分隔符，代替了默认的“/”分隔符。表示把所有10替换成100   1. $ sed 's/^192.168.0.1/&localhost/' example   &符号表示替换换字符串中被找到的部份。所有以192.168.0.1开头的行都会被替换成它自已加 localhost，变成192.168.0.1localhost |
| 转换文本 |
| 转换文本 |
| 语法: [address]y/abc/xyz/  功能：字母表式的转换  eg1  $ cat test.txt  1 cow  2 cow  3 pig  4 cow  $ sed 'y/cp/wd/' test.txt  1 wow  2 wow  3 dig  4 wow  （c转换成w，p转换成d）  eg2  $ sed '1,10y/abcde/ABCDE/' example  把1--10行内所有abcde转变为大写，注意，正则表达式元字符不能使用这个命令 |
| 补充 |
| sed补充: |
| 分组 |
| sed 使用大括号“{}”将一个地址嵌套在另一个地址中，或者在相同的地址上应用多个命令。  左大括号必须在行末，而且右大括号本身必须单独占一行。要确保在大括号之后没有空格。  可以使用大括号将编辑命令括起来以对某个范围的行应用多个命令。  $cat test.txt  1 cow  2 cow  3 pig  4 cow  $ cat test.sed  /^1/,/^3/{  s/cow/pig/  /^2/d  }  $ sed -f test.sed test.txt  1 pig  3 pig  4 cow  输出到不同文件:  /^1/,/^3/{  /^1/w 1out.txt  /^2/w 2out.txt  /^1/,/^3/w allout.txt  } |
| 下一个 |
| sed '/test/{ n; s/aa/bb/; }' example  如果test被匹配，则移动到匹配行的下一行，替换这一行的aa，变为bb，并打印该行，然后继续 |
| 退出 |
| $ sed '10q' example-----打印完第10行后，退出sed |
| 保持和获取 |
| $ sed -e '/test/h' -e '$G example  在sed处理文件的时候，每一行都被保存在一个叫模式空间的临时缓冲区中，除非行被删除或者输出被取消，否则所有被处理的行都将 打印在屏幕上。接着模式空间被清空，并存入新的一行等待处理。  在这个例子里，匹配test的行被找到后，将存入模式空间  h命令将其复制并存入一个称为保 持缓存区的特殊缓冲区内。  当到达最后一行后，G命令取出保持缓冲区的行，然后把它放回模式空间中，且追加到现在已经存在于模式空间中 的行的末尾  在这个例子中就是追加到最后一行。简单来说，任何包含test的行都被复制并追加到该文件的末尾。 |
| 保持和互换 |
| $ sed -e '/test/h' -e '/check/x' example  互换模式空间和保持缓冲区的内容。也就是把包含test与check的行互换 |
| 同时执行多个命令 |
| $ sed -e '1,5d' -e 's/test/check/' example  (-e)选项允许在同一行里执行多条命令。如例子所示，第一条命令删除1至5行，第二条命令用check替换test。命令的执 行顺序对结果有影响。如果两个命令都是替换命令，那么第一个替换命令将影响第二个替换命令的结果。  $ sed --expression='s/test/check/' --expression='/love/d' example |

|  |
| --- |
| **简单脚本示例**： |
| vi test.sed  s/pig/cow/g  [:~$] sed -f test.sed test.txt  cow  [:~$] cat test.txt  pig  简单脚本示例：  [:~$] cat test.sh  sed '/^1/,/^3/{  s/cow/pig/  }' test.txt  [:~$] ./test.sh  1 pig  2 pig  3 pig  4 cow |