正则表达式就是处理字符串的方法； 他的工具有：vim grep awk sed 等工具支持；

表达式方式有基础正则表达式与扩展正规表达式；  （正则表达式与通配符完全不同！！！）

   特殊符号：

[:alnum:]           //代表0-9 a-z A-Z

[:alpha:]           //代表 A-Z a-z

[:digit:]            //代表0-9

[:lower:]           //代表a-z

[:upper:]         //代表A-Z

grep   [-A] [-B] [--color=auto]   '搜寻字符串'  filename     (grep 高级用法）

grep   -A     //后面可以加数字，为after的意思

grep  -B      //后面可以加数字，为before 的意思

grep  --color=auto      //可以正确的那个选取数据列出颜色

     例题： 用dmesg列出内核信息，用grep 找出eth的哪行，并颜色标示，列出行号， 前3行后两行列出

            dmesg | grep -n --color=auto 'eth' -A2 -B3          // -n 显示行号， --color=auto 颜色标示

  基础正则表达式字符：

^WORD        //待查找的字符串（word）在行首

      例题： 查找行首为# 开始的那一行，并列出行号

         grep -n   ' ^#' test.txt

WORD$      //待查找的字符串 （word) 在行尾

     例题：  将行尾为！ 的那一行打印出来，并列出行号

       grep  -n  ‘！$' test.txt

^$               //当首尾同时出现时，表示的是空行的意思

        grep   -n   '^$'   test.txt              //这个的意思就是找到空格的那一行

.               //代表一定有一个任意字符的字符

   例题：查找字符串可以是 eve ,eae ,eee ,

        grep -n 'e.e' test.txt

\              //代表转义字符，将特殊符号的特殊意义去除

   例题： 查找含有   '     的那一行

    grep -n \'   test.txt

\*           //代表重复零个到无穷多个的前一个字符  （\*可以是0个）

   例题：   查找含有es  ess esss 等字符串，

       grep  -n  'ess\*'   test.txt

[list]       //代表字符集合的RE字符里面找出想要选取的字符

      例题：  查找含有 gl  gd  的那一行

           grep -n 'g[ld]'  test.txt

[n1-n2]          //代表字符集合的RE字符里面找出想要选取的字符范围

            例题： 查找含有任意数字的那一行

                grep -n '[0-9]'  text.txt

[^list]            //代表从字符集合的RE字符里面找出不要的字符串或范围

            例题：查找的字符串可以是oog ood  但是不能是oot

                 grep -n  'oo[^t]'   test.txt

\{n,m\}       //代表连续n到m个的前一个RE字符

            例题：   在g与g 之间有2个到3个的o存在的字符串

                  grep  -n  'go\{2,3\}' test.txt

sed    可以将数据进行替换，删除，新增，选取特定行等功能

sed    [-nefr]   [动作]

sed  -n      安静模式，sed特殊处理的那一行才会被列出来

sed  -e      直接在命令行模式上进行sed 的动作编辑

sed  -r      sed的动作支持的是扩展性正则表达式的语法

动作说明：   a  新增  c  替换  d  删除 i 插入  p 打印  s 替换   （如：1，20s/old/new/g 就是）

a    新增 ，  a后面可以接字符串 （目前下一行）

i     新增，   i 后面可以接字符串  （目前上一行）

d    删除 ， d 后面通常不接任何参数

p   打印，  p  会与参数sed -n 一起运行

c    替换，  c 可以接字符串，这些字符串可以替换n1,n2之间的行

s   替换 ， s 可以直接进行替换的工作  s可以搭配正则表达式  （如：1，20s/old/new/g 就是）

      例题：  将/etc/passwd 的内容列出并打印行号，同时，将2-5行删除

              nl  /etc/passwd | sed '2,5d'           （sed '/abc/d'       //删除abc在的行）

      例题：  在第二行后加上  drink tea

             nl /etc/passwd | sed '2a drink tea‘

     例题： 将第2-5行的内容替换成为NO 2-5 number

            nl /etc/passwd | sed '2,5c NO 2-5 number'

     例题：  仅列出5-7行

           nl /etc/passwd | sed -n '5,7p'         (切记一定要加-n )

部分数据的查找并替换：（sed  's/要被替换的字符串/新的字符串/g’）

\*\*\*           sed   's/^.\*addr:/ /g'       //旧为首字母为任意字符 零个到无穷大 addr:   替换为 ： 空格 就是无的意思

扩展正规表达式   egrep

 +                   //代表重复一个或一个以上的前一个RE字符

         例题：  查找god  good   goood 等字符串

                   egrep -n 'go+d'  test.txt

?                 //代表零个或一个的前一个RE字符

         例题：   查找gd god 两个字符串

                  egrep -n 'go?d' test.txt

|                //代表用或的方式找出数个字符串

        例题：  查找gd  good dog 三个字符串

                egrep   -n 'gd|good|dog‘  test.txt

(  )            //代表找出组字符串

        例题：查找出glad  good  这两个字符串

                egrep  -n  'g(la|oo)d' test.txt

awk       常用与段的数据处理工具

复杂暂时不学

diff       文件比较工具

diff   [-bBi]    新文件  修改过的文件

diff  -b   //忽略一行当中有多个空白的区别

diff  -i    //忽略大小写

           例题： 比较passwd.old 与passwd.new区别

           diff   passwd.old passwd.new

           例题： 比较不同 并制作补丁文件

           diff   -Naur  passwd.old passwd.new > passwd.pathch

          patch -p0 < passwd.patch      //更新旧版本数据   （-p  后面的N表示几层目录，先在本层所用0) 通常用于跟新旧文件，让新文件与旧文件做对比，并吧对比出来的数据添加到旧的文件内，这样话就新旧文件一样了！！！

          patch -R -p0 < passwd.patch    //还原恢复旧文件的内容