Perl 入门和提高　　Lesson 3

周晓方

Sample

- @s里面是一个spice文本，请把电容的容量加倍，电阻的阻值减半；不考虑续行。
- 提示，电容卡的格式是"C名称 Node1 Node2 容量"

```
@s = ('RIN   1      2      10MEG',
      'EGAIN        3 0    1 2    100K',
      'RP1   3      4      1K',
      'CP1   4      0      1.5915UF',
      'ROUT  5      6      10');
s/^(C(\S+\s+){3})([0-9\.]+)/$1.($3*2)/e foreach @s;
/^R/ and s/([0-9\.]+)(\S*)$/($1\/2).$2/e foreach @s;
print join "\n", @s;
```

别忘了除法运算符/要转义

```
RIN       1         2         5MEG
EGAIN     3 0       1 2       100K
RP1       3         4         0.5K
CP1       4         0         3.183UF
ROUT      5         6         5
```

join, map, split, reverse, sort, grep

```
join("\n",  @array);           # return string
map("$_\n", @array);           # return array
   print join "\t", map $_*$_, 1..10;
   print join "\t", map int rand(20), 1..10;
split(/pattern/, "string");    # return array
   @list = split(//, "ABCDEFGHI"); # qw(A B C D E F G H I)
   @list = split(/:/, "12:34:56:78", 3); # qw(12 34 56:78)
reverse(@array);   看人下碟        # return array 数组反序
  scalar reverse(@array);      #合并成字符串，字符反序
  %new_hash = reverse %old_hash;#交换哈希表的key和val
  print scalar reverse "abc", "123"; ➔ 321cba
  print        reverse "abc", "123"; ➔ 123abc
sort(@array);                  # return array
   sort {$a cmp $b} @list;     # 用法很丰富，参考perlop
   sort {$a <=> $b} @list;
   sort {-($a <=> $b)} @list;
grep(/pattern/, @array);       # return filtered array
   @foo = grep(!/^#/, @bar);   # remove lines start with #
```

灵活运用Sort函数

- 已知hash表%std = (学号=>姓名，…)
  要求打印该表

- 无序的打印：

```
print "$_ : $std{$_}\n"
   foreach keys %std;
```

- 按学号顺序的打印：

```
print "$_ : $std{$_}\n"
   foreach sort {$a <=> $b} keys %std;
```

{…}括起来的是一段代码, 称为BLOCK

- 按姓名顺序的打印：

```
print "$_ : $std{$_}\n"
   foreach sort {$std{$a} cmp $std{$b}} keys %std;
```

注意, BLOCK和后续参数之间没有逗号！

- 想一想，带主键、次键的sort排序怎么写？

Hash表用于计数—单词统计

```
!/usr/bin/perl -w
use strict;
my @lines = <DATA>;     # Read in data
chomp @lines;           # Remove CRLF
my $line = lc join " ", @lines;
$line =~ s/[^a-z]+/ /g;        # remove ,:.()'
my @words = split / /, $line;
my @chars = split //, $line;
print "=====List of all words:\n";
print join "\t", @words;
my ($word, $char, $count, %words, %chars);
$words{$_}++ foreach @words;
$chars{$_}++ foreach @chars;
print "\n\n=====Word count:\n";
print "$word\t$count\n"
        while ($word, $count) = each %words;
print "\n\n=====Char count:\n";
print "$char\t$count\n"
        while ($char, $count) = each %chars;
1;
__END__
There is one minor difference: if variables
are declared with my in the initialization
section of the for, the lexical scope of those
……
```

```
=====List of all wo
=====Word count:
the       9
test      1
explicitly        1
you       2
normal    1
file      2
exactly 1
that      1
my        1
……
=====Char count:
w         1
a         27
r         26
x         4
d         11
y         9
u         4
h         16
……
```

回家作业

- 统计学号邮箱里邮件的发件方有效email地址，按邮件多少排序，邮件数量相同的，按email地址逆排序，学号-**03.pl**

- 户名、密码单独存在secret.txt中，不要发给我☺

- 用cpan命令安装YAML和Mail::POP3Client
  – YAML可以保存Perl的变量到文件，或从文件读出数据到变量
  – 支持标量、数组、散列和各种引用
  – YAML模块的保存用Dump、DumpFile，读取用Load、LoadFile

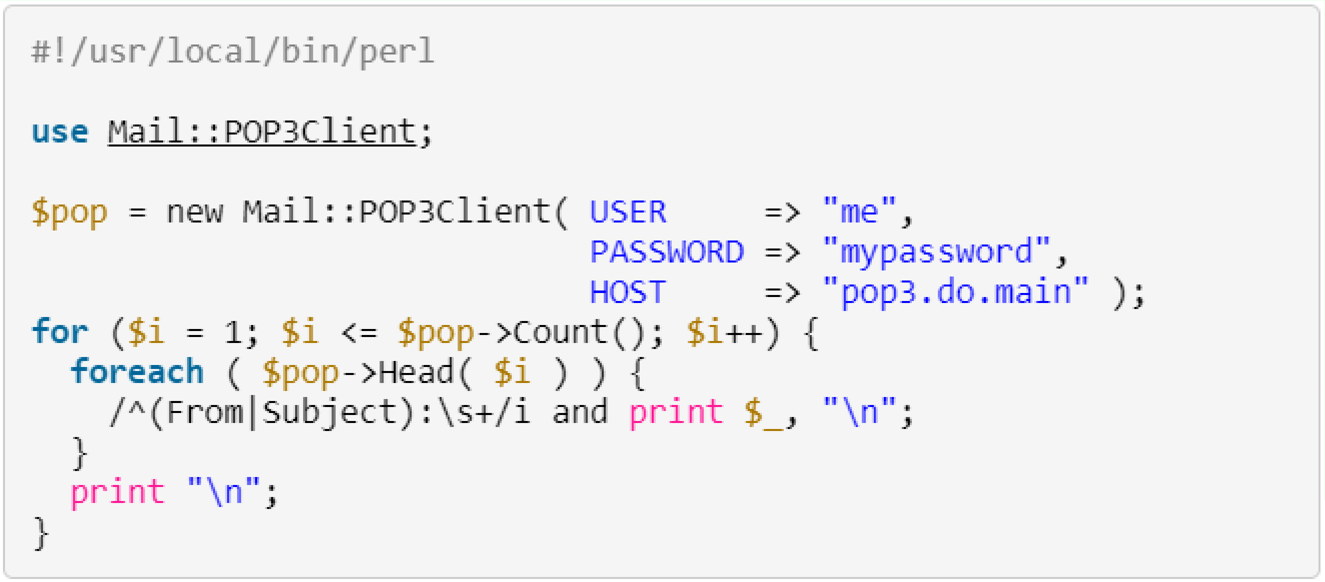
  – 类似的Perl模块还有JSON、XML、Data::Dumper、Storable等
  – 这里有很好的讨论：https://stackoverflow.com/questions/1876735/should-i-use-yaml-or-json-to-store-my-perl-data

  – 注：课件提到的模块，仅供扩充Perl知识面，考试不考特定的模块

Mail::POP3Client 和Email的头信息

- 查看CPAN的Mail::POP3Client网页实例 https://metacpan.org/pod/Mail::POP3Client
- 对照Email的头信息，大致是这个风格：

```
+OK 33556 octets
Received: by ajax-webmail-app2 (Coremail) ; Tue, 30 Sep 2012 00:56:20 +0800
 (GMT+08:00)
X-CM-HeaderCharset: UTF-8
X-Originating-IP: [180.160.159.212]
Date: Tue, 30 Sep 2012 00:56:20 +0800 (GMT+08:00)
From: =?UTF-8?B?5pu+5a6H?= <blabla@fudan.edu.cn>
To: "Dr. Yada Yada" <yadayada@fudan.edu.cn>
Subject: Re: Fw: Re: R
X-Priority: 3
X-Mailer: Coremail Web
 20131122(24254.5785.5
In-Reply-To: <54294A2A
References: <54294ABA.
X-SendMailWithSms: fal
X-CM-CTRLDATA: +q7cEzZ
Content-Type: multipar
        boundary="---
MIME-Version: 1.0
Message-ID: <3de62331.
```



```perl
#!/usr/local/bin/perl

use Mail::POP3Client;

$pop = new Mail::POP3Client( USER     => "me",
                             PASSWORD => "mypassword",
                             HOST     => "pop3.do.main" );

for ($i = 1; $i <= $pop->Count(); $i++) {
  foreach ( $pop->Head( $i ) ) {
    /^(From|Subject):\s+/i and print $_, "\n";
  }
  print "\n";
}
```

附送本次作业部分代码

- 只要递交 学号-**03.pl**

secret.txt的内容就两行

```
--- '20300750999'
--- 'myp**sword_here'
```

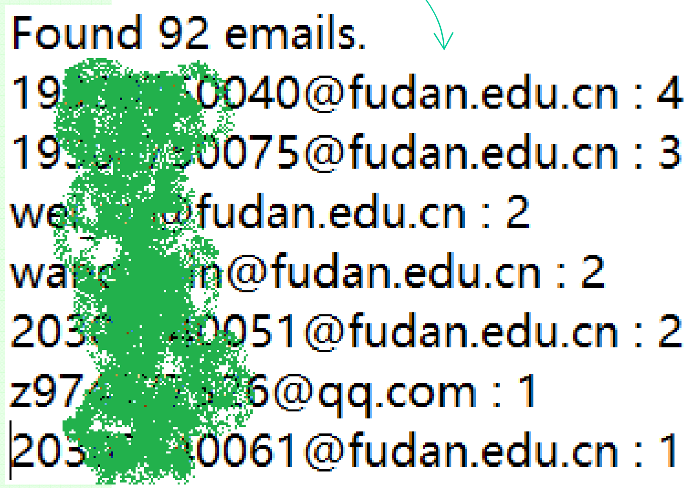
```
#!/usr/bin/perl -w
use strict;
use YAML qw(LoadFile);
use Mail::POP3Client;

my ($user, $pass) = LoadFile 'secret.txt';
my $pop = new Mail::POP3Client(
        HOST   => 'mail.fudan.edu.cn',
        USER   => $user,
        PASSWORD=> $pass,
        USESSL => 1,
        AUTH_MODE => 'PASS',
    );

my ($i, %header, $from, %senders);
my $cnt = $pop->Count();
print "Found $cnt emails.\n"; # 打-1是用户密码错
```

```
for ($i = 1; $i <= $cnt; $i++) {
    %header = ();
    (chomp, /^#这里匹配一行头信息
    and $header{lc $1} = $2)
        foreach $pop->Head($i);
    $from = $header{from};
    ...; # 这里提取有效的email地址
}
...; #这里排序打印%senders的信息

1;
```



```
Found 92 emails.
19[redacted]0040@fudan.edu.cn : 4
19[redacted]0075@fudan.edu.cn : 3
we[redacted]fudan.edu.cn : 2
wa[redacted]n@fudan.edu.cn : 2
203[redacted]0051@fudan.edu.cn : 2
z97[redacted]6@qq.com : 1
203[redacted]0061@fudan.edu.cn : 1
```

RE Rule 4: 反向引用

- Back tracking: $1, $2, $3,… $65536对应每对圆括号
  - $a=~/**(**[0-9]+**)**\.**(**[0-9]+**)**/; $int = **$1**; $frag=**$2**;
  - *List environ*: ($int, $frag) = ($a =~ /**(**[0-9]+**)**\.**(**[0-9]+**)**/);
    @int_frag_paires = ($a =~ /**(**[0-9]+**)**\.**(**[0-9]+**)**/g);
  - Note: either matches all or matches none. $1 $2 … are not cleared if the match fails. 这样写比较可靠:
    - `($a=~/([0-9]+)/) && ($num = $1);`
    - `$num = $1 if $a =~ /([0-9]+)/;`
    - `($int, $frag) = ($a =~ /([0-9]+)\.([0-9]+)/);`
  - Order of (…) : Outside ➔ inside, Left ➔ right.
  - `(?:`*RE*`)` 格式的圆括号不看作反向引用,只表示优先级
  - Last match, $+, `/Ver: (.*)|Rev: (.*)/ && ($rev = $+);`
  - Pre-match, entire match, post-match == $\` $& $'
  - *注意*：$0是$PROGRAM_NAME，不是反向引用

RE Rule 4: 反向引用-cont.

- 实例：分割字符串为数组
  - 分解成单个字符
    ```
    @s = split // , $s;
    @s = ($s =~ /./g);
    ```
  - 分解成两个两个
    ```
    @s = ($s =~ /..|./g);
    @s = ($s =~ /.{1,2}/g);
    ```
  - 分解成三个、两个、三个、两个、…
    ```
    @s = ($s =~ /(.{1,3})(.{0,2})/g);
    ```
- Backtrack inside RE: \1\2(patten) and $1$2(replace)
  - $string =~ s"(far) (out)"$2 $1";
  - $s = "bballball"; $s =~ s"(b)\1(a..)\1\2"$1$2"; #$s➔'ball'

RE Rule 5:修饰

- Options for matching: g, i, m, o, s(单行), x
- Options for substitution: g, i, e, m, o, s, x
- i大小写无关的匹配
  - /yes/i 相当于/[yY][eE][sS]/
- g全局匹配
- o对RE编译一次,提高以后每次匹配的效率
- e替换部分作为代码执行，可以调用各种函数
- x在patten中可以放置空格和注解，可以分多行写
- m影响对^$的理解，m将字符串看作多行的(^匹配串开头和行开头, $匹配行尾和串尾)
- s影响对.的理解，看作一行(. 匹配任何字符, 包括\n).
- ms可以连用，分别对^$和.的处理产生影响

```
$m="a1a\nb2b\nc3c"; $m=~s/$/:/;    # a1a\nb2b\nc3c:
$m="a1a\nb2b\nc3c"; $m=~s/$/:/m;   # a1a:\nb2b\nc3c
$m="a1a\nb2b\nc3c"; $m=~s/$/:/mg;  # a1a:\nb2b:\nc3c:
$m="a1a\nb2b\nc3c"; $m=~s/$/:/sg;  # a1a\nb2b\nc3c:
$m="a1a\nb2b\nc3c"; $m=~s/$/:/msg; # a1a:\nb2b:\nc3c:
```

RE Rule 5:修饰 - cont.

```
$m="a1a\nb2b\nc3c"; $m=~s/^/:/;    # :a1a\nb2b\nc3c
$m="a1a\nb2b\nc3c"; $m=~s/^/:/m;   # :a1a\nb2b\nc3c
$m="a1a\nb2b\nc3c"; $m=~s/^/:/mg;  # :a1a\n:b2b\n:c3c
$m="a1a\nb2b\nc3c"; $m=~s/^/:/sg;  # :a1a\nb2b\nc3c

$m="a1a\nb2b\nc3c"; $m=~s/./:/;    # :1a\nb2b\nc3c
$m="a1a\nb2b\nc3c"; $m=~s/./:/g;   # :::\n:::\n:::
$m="a1a\nb2b\nc3c"; $m=~s/./:/mg;  # :::\n:::\n:::
$m="a1a\nb2b\nc3c"; $m=~s/./:/sg;  # :::::::::::
                                      $&就是匹配到的东西
$m="x=23;y=45";$m=~s/\d+/sprintf("%x",$&)/eg;#x=17;y=2d
```

- Options for translation: c(求补), d(删除), s(单个)

  tr是"翻译", 只认单个字母, 一一对应地翻译, 返回匹配的次数

```
$a = "111 222 33 4 5"; $a =~ tr/15/*/c;#111**********5
$a = "111 222 33 4 5"; $a =~ tr/12//s; #"1 2 33 4 5"
$a = "111 222 33 4 5"; $a =~ tr/12//d; #"  33 4 5"
$a = "111 222 33 4 5"; $a =~ tr/12//; # 111 222 33 4 5
$a = "111 222 33 4 5"; $a =~ tr/12/9/;# 999 999 33 4 5
$a = "111 222 33 4 5"; $a =~ tr/12/21/;#222 111 33 4 5
$a = "111 222 33 4 5"; $num = $a =~ tr/12//;#对1、2计数
```

RE Rule 6: The (?…) stuffs

- (?=*<RE>*), look ahead匹配但不吃掉
- (?!*<RE>*),下一组文本不match时,才匹配
- (?<=*<RE>*), lood behind, also (?<!*<RE>*)

```
$a = "cat housecat catch crazycats";
$a =~ s/(?<=\s)cat(?=\s)/CAT/g;     # 不变
$a =~ s/(?<!\s)cat(?=\s)/CAT/g; #CAT houseCAT catch
   carzycats
```

- (?:*<RE>*),使用(…)但不计入反向引用中
- (?xims-xims: *<RE>*)

```
/Answer: ((?i)yes)/; # 'Answer: yes', '…YES', '…Yes', etc
/Answer: ((?-i)yes)/i; # 'answer: yes' only, not "…YES"
```

- (?#)comment, replaced by m/…/**x** now.

RE Rule 7 返回值

- 替换运算符
  - ($s =~ s/a/b/)返回1(匹配)或' '(不匹配)
  - ($s=~s/a/b/g)返回匹配替换的次数或' '(不匹配)
- 匹配运算符

  这对括号可有可无

  - @matches = ($line =~ m[(\d*\.\d+)]g)
  - @mathces = ($line =~ m[(.{1,3})(.{0,2})]g) 这里括号不能省
  - ($var, $eq, $val) = ($line =~ /(\w+)\s*(=)\s*(\w+)/);
  - $b = ($line =~ /cat/) 返回1(匹配)或' '(不匹配)
  - $b = ($line =~ /cat/g) 同上,返回1(匹配)或' '(不匹配)
- 标量环境中的匹配——特殊的迭代操作, pos( )函数

  -
    ```
    $line="BEGIN<d1>BEGIN<d2>BEGIN<d3>";
    while ($line =~ m"BEGIN(.*?)(?=BEGIN|$)"sg)
      {push(@blocks, $1); print pos ($line), ", ";}
    ```
  - *print out "9; 18; 27; ", @blocks is ('<d1>', '<d2>', '<d3>');*

Quote-like operators

- ?RE?　　Match only once between ***reset*** call.
- m/ /, s///, tr///, y///(same as tr///)
- q/string/相当于'string'　　qq/str/相当于"str"
- qr/PATTEN/imosx 生成一个正则表达式
  - `$re=qr/$pattern/; $string =~ $re; $string =~ /$re/; $string =~ /foo${re}bar/; #甚至可以嵌套在其他RE里面`
- qx/command/　　执行命令，相当于`command`
- qw/word1 word2 word3 …/　　返回字符串数组
- HERE document (like those in UNIX shell scripts)

```
$date = q!March 2nd, 2006!;
print << "HERE";    # same as print << HERE
Many lines here with interplation
Today is $date
HERE
print << 'THERE';
Many lines here but no interplation
Today is March 2nd, 2006
THERE
```

两者必须成对出现，拼写必须完全相同

后面必须马上跟回车符，不能有其他字符

RE，一些注意事项

- 特定字符什么时候理解为元字符
  - '|'理解为元字符：/cat|dog|piggy/
  - '|'理解为普通字符：/[cat|dog|piggy]/相当于/[acdgiopty|]/
  - /a|z|-/、/[az-]/、/[-az]/、/[a\-z]/是等价的
- '|'从左到右原则，并非越长越好原则
  - /foo|foot/匹配'barefoot'，匹配到的是'foo'部分
- 避免反向引用出现歧义
  - 实例：遇到数字串，添加000: s/(\d+)/${1}000/g
- 进一步的阅读
  - use re 'debug';参考《Programming Perl》3rd ed, 5.9.3章节
  - use String::Approx;参考《Perl Cookbook》2002, 6.13章节
  - RE和状态机, H. R. Lewis等 "Elements of the Theory of Computation, 2ed" ， Prentice Hall, 1998