Perl 入门和提高　Lesson 5

周晓方
courses@xfzhou.homeftp.org

递归子程序和__SUB__

- 一个子程序直接或间接调用自己，称为递归
- 递归必须有退出条件。下面以阶乘为例：

```
1. sub naive_frac {
2.       my $n = shift;
3.       $n <= 1 ? 1 : $n * naive_frac($n - 1);
4. } # 斐波那契数列和阶乘的定义就是经典的递归
```

- perl 5.16之后，__SUB__是当前子程序的引用

```
1. use 5.16.0;
2. # or you can say:
3. # use feature 'current_sub';
4. my %frac = (0 => 1,);
5. sub frac {
6.       $frac{$_[0]} //
7.       ( $frac{$_[0]} = $_[0] * __SUB__->($_[0] - 1) );
8. }
```

空间换时间，避免重复计算

->表示想要调用这个函数啦

某函数的引用

( )内是参数列表

Program

- Perl程序的一般结构(务必用my定义变量)

```
#!/usr/local/bin/perl -w
use strict;
use lib "path";
use module_names;
require "sourcefile";

... main code ...

sub routine1 {...}
sub routine2 {...}
sub routine3 {...}
...
1;
__END__
... pod and/or data
```

回家作业(附件 ***学号-05.pl***)

1. 级数展开$\sin+\cos|_x$($|x|<\pi$)
2. 编写子程序sub tri，参数是x,*n*,返回(sin+cos)(x)的值，级数展开到多项式的*n*次项
3. 主程序调用tri(2,$n)，其中$n=(1..10)，运行结果如下：

```
0       1
1       3
2       1
3       -0.333333333333333
4       0.333333333333334
5       0.6
6       0.511111111111111
7       0.485714285714286
8       0.492063492063492
9       0.493474426807776
10      0.493192239858907
```

收敛慢∞→ 0.493150590278539

匿名子程序、代码之引用

要熟悉
$f->(args)
这种写法

- 匿名子程序(相当于子程序、函数的引用)

```
my $subref = sub {$_[0] + $_[1]};
my $add = $subref;      # subroutine-ref can be copied
print &{$add}(1,2),"\t", &$add(1,2), "\t", $add->(1,2);
```

- 找两个f:R→R，变成匿名子程序，例如

```
my ($sin, $cos) = (sub {sin($_[0])}, sub {cos($_[0]);});
```

- 尝试调用$sin和$cos

```
print join "\n", $sin->(1), $cos->(1),
    $sin->(1) ** 2 + $cos->(1) ** 2, "\n";
0.841470984807897
0.54030230586814
1
```

高阶玩法：Perl描述的线性空间

- 定义*va*两个R→R函数的和、*kv*一个R→R函数的数乘：

```
1.  sub va {
2.      my (@f) = @_;
3.      return sub {
4.              my $sum = 0;
5.              $sum += $_->($_[0]) foreach @f;
6.              $sum;
7.      }
8.  };
9.  sub kv {
10.     my ($k, $v) = @_;
11.     return sub { $k * $v->($_[0]) }
12. }
13. my $s_a_c = va($sin, $cos);
14. my $five_sac = kv(5, $s_a_c);
15. my $sa3c = va($sin, kv(3, $cos));
16. print join "\n", $s_a_c->(1),
17.             $five_sac->(1), $sa3c->(0.5);
18. 1; 按计算机世界的提法，Perl函数(的引用)既能作为参数传递，又能动态
    生成一个函数返回，所以Perl函数/子程序是first-class function
```

字典变量成为匿名函数的私有静态变量

每次调用*va*都返回一个新的匿名函数。形成**Closure**闭包

高阶函数 := 吃进函数 且/或 吐出函数

相当于数学上的泛函、算符、算子

Same is for *kv*

```
1.38177329067604
6.90886645338018
3.11217322427532
```

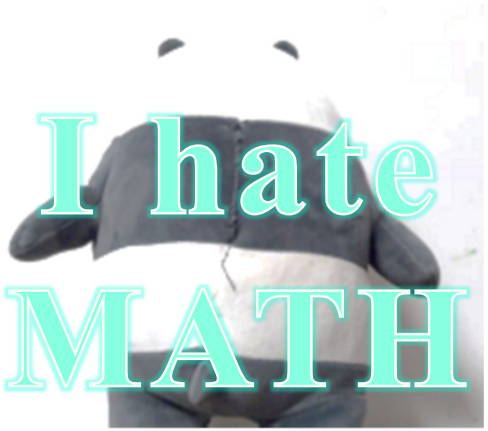
继续：在区间[$-\pi,\pi$]上定义R→R函数的范数$\|f\|$和内积$\langle f1, f2\rangle$

范数简单地定义为[$-\pi,\pi$]上等间隔采样201个点的函数值的平方和

```
1. use constant LOW => -3.14159;  # Perl定义常数的方法之一
2. use constant HIGH => 3.14159;  # 常数不带$、@等前缀
3. use constant SAMPLES => 201;   # 名称按惯例用全大写
4. sub norm {       # Euclidean欧氏结构。注：数学上“范数”应该开平方根，此处略
5.     my ($f) = @_;
6.     my $step = (HIGH - LOW) / (SAMPLES - 1);
7.     my $v = ($f->(HIGH)) ** 2;
8.     my ($x, $res) = (LOW, $v);
9.     foreach (0 .. SAMPLES - 2) {
10.            $v = ($f->($x)) ** 2;
11.            $res += $v;
12.            $x += $step;
13.    }
14.    sqrt($res / SAMPLES);
15.}
16.
```

norm(·)
“距离”可以有不同的定义

```
4. sub norm_DIY {# DIY, mix of Manhattan/maximum
5.       my ($f) = @_;
6.       my $step = (HIGH - LOW) / (SAMPLES - 1);
7.       my $v = abs($f->(HIGH));
8.       my ($x, $max, $res) = (LOW, $v, $v);
9.       foreach (0 .. SAMPLES - 2) {
10.              $v = abs($f->($x));
11.              $max = $v if $v > $max;
12.              $res += $v;
13.              $x += $step;
14.      }
15.      0.5 * ($max + $res) / SAMPLES;
16.}
17.  #注：这个norm_DIY缺少“双线性”属性，不是“好”norm
```



继续(2)：内积可以用范数来定义，

$\langle f_1, f_2 \rangle := (\|f_1+f_2\|^2 + \|f_1-f_2\|^2)/4$

满足$\langle f, f \rangle == \|f\|^2 >= 0$

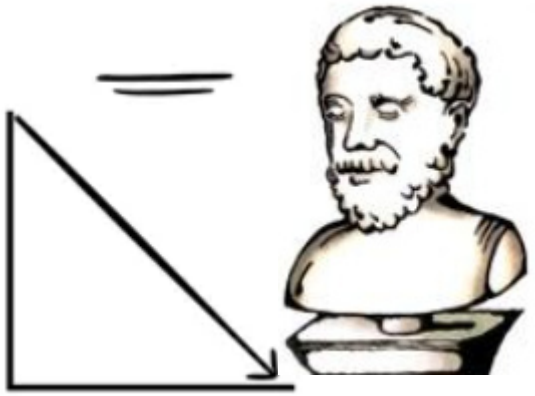
ref: www.zhihu.com/question/366117005

```
19.sub inner {
20.    my ($f1, $f2) = @_;

21.    my $normadd = norm(sub {$f1->($_[0]) + $f2->($_[0])});
22.    my $normsub = norm(sub {$f1->($_[0]) - $f2->($_[0])});

23.    0.25 * ($normadd * $normadd - $normsub * $normsub);
24.}

25.print  map "$_\n", map inner(@$_), [$sin, $sin],
26.           [$cos, $cos], [$sin, $cos];
```

```
0.497512857903854
0.502487142096146
-5.55111512312578e-17
```

当前拓扑下sin和cos是正交的

Πυθαγόρας



继续(3)：找一个容易下手的线性子空间：

- 实数域上单变量x的多项式环R[x]，这里只用到R[x]的加法和数乘，及范数和内积。
- 用array-ref表示实系数多项式环R[x]；向量和多项式一一对应，如$[1,2,-3,0,4]^T$表示实系数多项式$1+2x-3x^2+4x^4$;
- 先做一个计算多项式的子程序，参数是x和向量v;
  再做一个(又一个高阶函数 **higher-order function**)能够"烧制"特定多项式函数的子程序poly(·)：参数是向量v，返回一个v对应的匿名多项式子程序：

```
27.sub rx {
28.   my ($x, $v) = @_;
29.   my ($p, $s)=(1, 0);
30.   foreach (@$v) {
32.     $s += $_ * $p;
33.     $p *= $x;
34.   }
35.   $s;
36.}
```

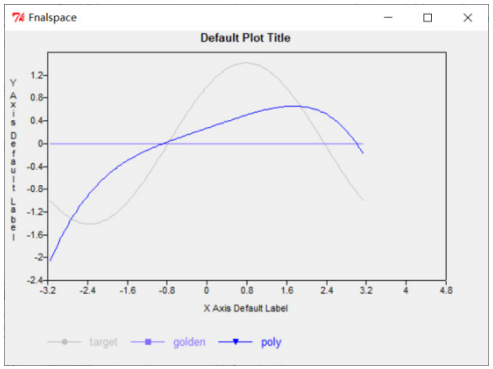
```
38.sub poly {
39.   my @v = @{$_[0]};
40.   return sub {
41.     rx($_[0], [@v]);
42.   };
43.}
```

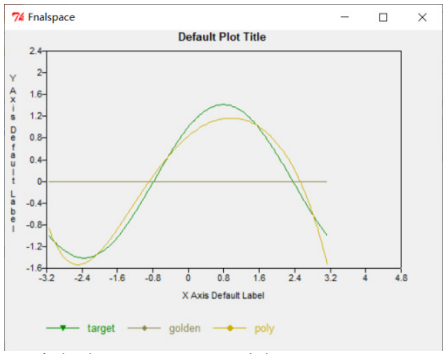
继续(4)：随机走动，找相似，在多项式子空间内拟合超越函数

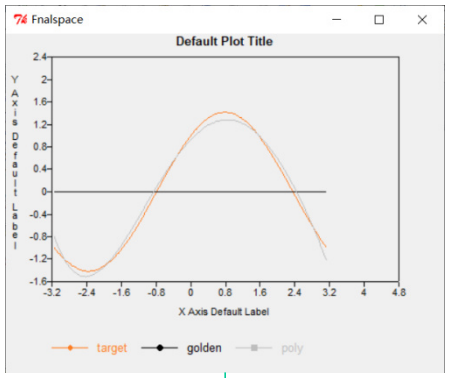
```
27.sub optimizePoly {
28.    my($target, $v, $step, $ntry, $eps) = @_;
29.    my $n = scalar @$v;     # 优化过程中，多项式的维度不变
30.    my($ferr,$error,$dim,$try,$tryv,$trye,$bestv,$beste);
31.    my ($golden) = sub {0};
32.    $error = norm(va($target, kv(-1, poly($v))));  # ||t - p(v)||
33.    while (1) {
34.     ($beste, $bestv) = ($error);
35.     for $dim (0 .. $n - 1) {    # 分别尝试随机优化各个维度的分量
36.       $try = $ntry;
37.       while ($try-- > 0) {# 和$tryv = $v不同，这是deep-copy，
38.          $tryv = [ @$v ]; # 引用新的匿名数组，并复制了@$v的每个分量
39.          $tryv->[$dim] += ($step - 2*rand $step) / frac($dim);
40.          $trye = norm(va($target, kv(-1, poly($tryv))));
41.          ($beste, $bestv) = ($trye, $tryv) if $trye < $beste;
42.       }
43.     }
44.     last unless $bestv;   # 走投无路则退出，很可能落在局部最优点里了
45.     ($error, $v) = ($beste, $bestv);
46.     $step *= 0.98;  # 谨慎地缩小搜索范围
47.    }
48.    $error;
49.}                  # 本例算法收敛速率底，仅供Perl教学之用
```

试一试

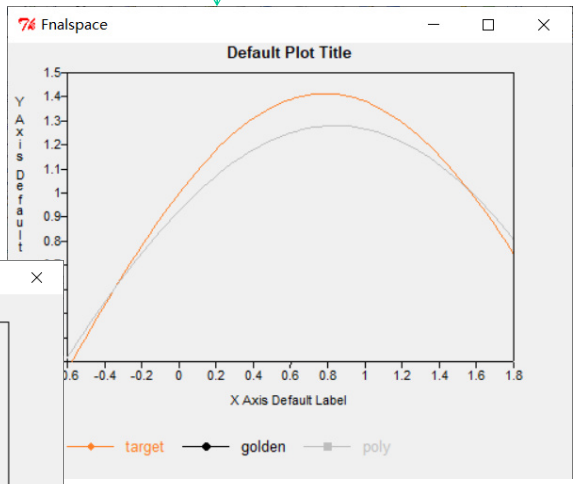
```
my $v = [(0) x 6]; #采样欧式范数
print optimizePoly($s_a_c, $v, 1.5, 300, 1.0e-5);
```



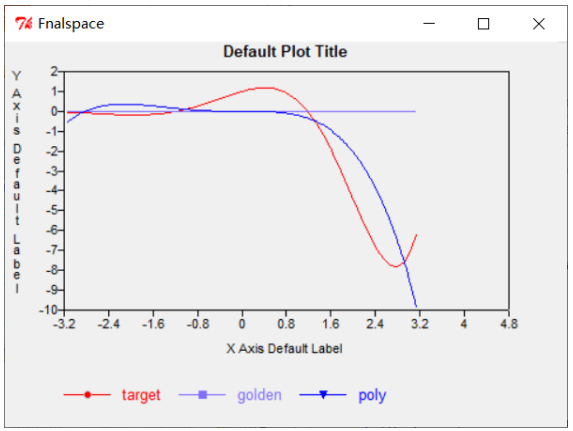


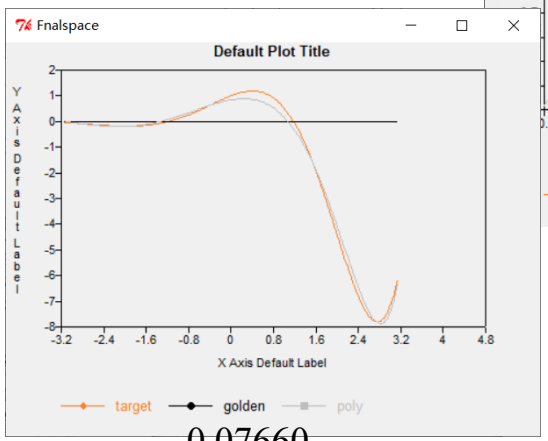


细节




```
my $v = [(0) x 10]; #采样DIY范数
print optimizePoly(sub {
        my $x = shift;
        cos($x*4/3) * exp ($x*0.8)
    }, $v, 2.5, 500, 1.0e-5);
```





0.07660

介绍一组实用的Tk作图模块

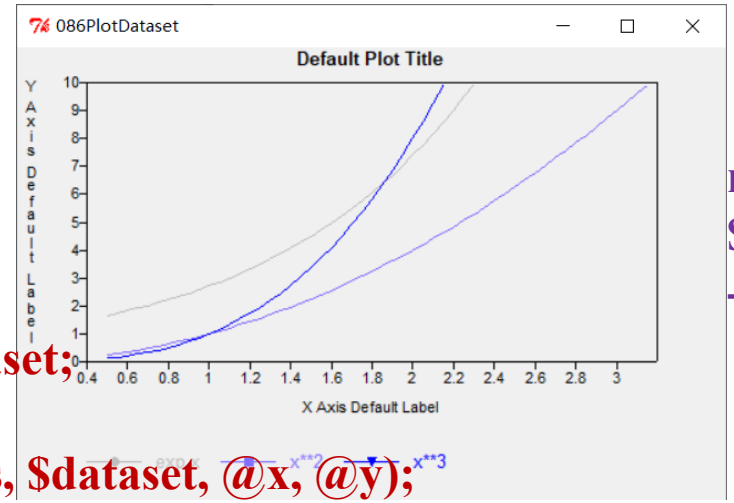
- LineGraphDataset对象用于在Tk里面作曲线图
  - cpan里面要这样安装：install LineGraphDataset
  - 每个对象（数据集）有一个名字-name，可以只带一组-yData，也可以-xData / -yData都加入
  - 该模块new会报一堆warning，请直接忽略。或者找到 ??/cpan/build/Tk-LineGraphDataset-0.01-?/子目录，修改LineGraphDataset.pm文件new方法中类似的7处，再gmake install

```
$self->{-color} = "none" if($self->{-color} eq undef);
$self->{-y1} = 0         if($self->{-y1}    eq undef);
$self->{-yAxis} = "Y"    if($self->{-yAxis} eq undef);
```

```
$self->{-color} = "none" unless defined $self->{-color};
$self->{-y1} = 0         unless defined $self->{-y1};
$self->{-yAxis} = "Y"    unless defined $self->{-yAxis};
```

- Tk::PlotDataset将多个LineGraphDataset对象画在一张图中，可以用鼠标放大、缩小，可以点击高亮单条曲线。
  - 方法有PlotDataset、plot，用pack( )放到Tk窗口之中

Sample code Tk::PlotDataset



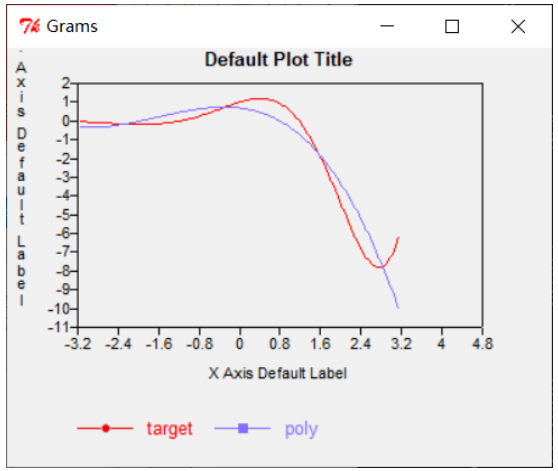

```
#!/usr/bin/perl -w
use strict;

use Tk;
use Tk::PlotDataset;
use Tk::LineGraphDataset;

my ($fn, $f, $x, $y, @dss, $dataset, @x, @y);
foreach (
        ['exp x', sub {exp $_[0]}],
        ['x**2', sub {$_[0]**2}],
        ['x**3', sub {$_[0]**3}],
        ) {
        ($fn, $f) = @$_;
        @x = @y = ();
        foreach $x (map $_ / 20, 10..100) {
                $y = $f->($x);
                last if $y > 10;
                push @x, $x;
                push @y, $y;
        }
        next unless @x;
        $dataset = LineGraphDataset
        -> new(
                -name => $fn,
                -xData => [@x],
                -yData => [@y],
        );
        push @dss, $dataset;
}

my $m = MainWindow->new;
my $graph = $m->PlotDataset(
        -width => 800,
        -height => 500,
        )->pack;
$graph->addDatasets(@dss);
$graph->plot;
MainLoop;
1;
```

回家作业 学号-06.pl

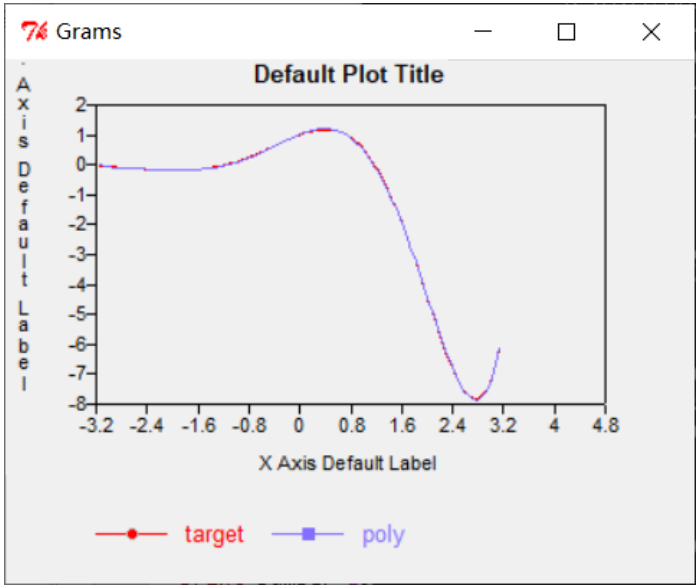
- 如何更科学的拟合函数，思路说明：
  - 区间依然取 [-π, π]，欧式范数的内积公式
  - 指定维度n ($n > 0$)，从标准基$\{\vec{b}_i\}$:={1, x^1, x^2,…, x^{n-1}}出发(这个不是正交基)，Gram-Schmidt Process找到多项式的一组标准正交基$\{\vec{e}_i\}$
  - 提示：以分量而言，$\{\vec{e}_i\}$是方阵、 $\{\vec{b}_i\}$是I_n方阵
  - 用内积得到目标函数在$\{\vec{e}_i\}$上的各个分量m_i
  - 将m_i基从$\{\vec{e}_i\}$变换回$\{\vec{b}_i\}$，得到$\{\vec{b}_i\}$上的多项式w_i 提示：$w_i = \{\vec{e}_i\}\vec{m}$
  - 显示误差、多项式系数，目标函数曲线和拟合多项式曲线

举个例子

- 取$*n*=4; $target = sub {
  my $x = shift; cos($x*4/3) * exp ($x*0.8)};
- $e = **gramschmidt**(4); 得[[1,0,0,0]^T, [0,0.5486,0,0]^T, [-1.118, 0, 0.3365, 0]^T, [0, -1.2571, 0, 0.2102]^T]，此处特意标了转置^T，提示数学上该存贮要看作Fortran顺序
- $m = **project**($target, $e); 得 *本页的数值都已四舍五入*
  [-1.2765, -1.9484, -1.7611, -0.5830]
- $w = **transcoord**($m, $e); 得
  [0.6925, -0.3360, 0.5926, -0.1225]
- 作图



不太理想？取n=7试试



0.80761

0.02609

作业要求 学号-06.pl

- 用前面例子的inner.pm
- 维数和目标函数单独放入task.inc
- 学号-06.pl 参考如下架构：

task.inc是1；结尾的脚本

```perl
$n = 4;
$target = sub {
    my $x = shift;
    cos($x*4/3) * exp ($x*0.8);
};
1;
```

```perl
#!/usr/bin/perl -w
use strict;
use lib '.';
use inner;
use Data::Dumper;

our ($n, $target);
eval {require "task.inc"}
    or die "Error on task.inc\n$@";
```

require有点像C的#include，Perl事实上会运行该脚本，并期待返回真值，否则会出错。注意主脚本用了**our**变量！

…定义子程序…

```perl
my $e = gramschmidt($n);
my $m = project($target, $e);
my $w = transcoord($m, $e);
my $err = norm(va($target, kv(-1, poly($w))));
# 这里把多项式、误差打印出来
plot(target => $target, poly => poly($w));
1;
```

Subroutine III——Prototype

- sub name (prototype_format) {BLOCK}

```
sub func1($) {…}#只接受一个标量参数的函数
sub func2($$;$){#头两个参数是必须的,后一个是可选的
}
sub mygrep(&@) # &代码 mygrep {/1+/} $a, $b
sub myopen(*$) # *句柄 myopen HANDLE, $name
sub mypush (\@@) {  # \@ is array-ref
  my $list = shift; # \% is hash-ref, etc
  push @$list, @_;  # @ is array, last one
}          # \@ \% became + for later perl
sub pi () { 3.1416 } # constanct function
print pi + 2; # print 5.1416, not 3.1416
# take it as print(pi(+2)) otherwisw
```

不带\的@和%必须放在最后,通吃后面的所有参数

- 若用了函数原形，要在调用前就定义函数头
- 常数可以用use constant pi=>3.1415更方便
  - 这样定义的常数不用带前缀$,直接用pi

使用函数原形的一个例子程序

```
#!/usr/bin/perl -w
use strict;
my(@list);

sub pi_proto () {3.1416}
sub pi_normal {3.1416}

sub mypush (\@@) {
   my $list = shift;
   push @$list, @_;
}

print "pi_proto + 2 is ", pi_proto + 2, "\n";
print "pi_normal + 2 is ", pi_normal + 2, "\n";
mypush @list, 1, 2, 3, 4;
print "\n";
print "list is (", (join ", ", @list), ")\n";
1;
```

- 运行结果

```
pi_proto + 2 is 5.1416
pi_normal + 2 is 3.1416
list is (1, 2, 3, 4)
```

较新版Perl增加了+，表示\@或\%

```
sub mypush (+@) {
      my $list = shift;
      push @$list, @_;
}
```

Subroutine IV匿名和左值函数

- Define and call an anonymous subroutine

```
my $subref = sub {$_[0] + $_[1]};
my $add = $subref;  # make a copy of subroutine-ref
print &{$add}(1,2),"\t", &$add(1,2), "\t", $add->(1,2);
```

- Lvalue subroutine (experimental)

```
sub Lsub : lvalue {
   my($index)=shift;
   $_[$index]; # don't say return $_[$index] here
}
my @a = qw(I my teaching perl.);
$" = " ";
print "@a\n";
Lsub(2, @a) = 'learning'; #函数返回作值，修改为'learning'
print "@a\n";  # @a的值已经被修改
```

Kindly Reminder

- 应该知道，但没有充分理由不建议使用这些内容：
  - 循环的continue块
  - local和our变量
  - 函数原型，匿名函数，左值函数

File Handling --- Standard Handles

- Standard file handles (Already opened):
  - **DATA** data following **__END__**
  - **STDIN** input, can be redirected
  - **STDOUT** output, can be redirected
  - **STDERR** output, can't be redirected, for error messages
- Write to file, "**print**"

```
print STDERR $err_msg;
print @lines;        # default is STDOUT
print FD $a, $b;     # not print FD, $a…
```

- Read file, the **<FILEHANDLE>**, diamond operator "**<>**"

```
while($line = <STDIN>){}; # read one line
@lines=<>; # read file listed in @ARGV
           # one by one. When @ARGV is
           # empty, read from STDIN
```

File Handling --- User files

- Open a user file
  - For read: `open(FD, "<filename");`
  - For write: `open(FD, ">filename");`
  - For append: `open(FD, ">>filename");`
  - For pipe read: `open(FD, "可执行文件名|");`
  - For pipe write: `open(FD, "|可执行文件名");`
  - `+>` 截断文件并打开读写 `+<` 不截断文件并打开读写
    等等有很多有趣的用法，详见`perl`文档`perlfunc`的`open`函数说明
- Close file: `close(FD);`

```
open(FD,"ls -la |"); print <FD>;   # DOS dir
```

- $oldFD = select(FD); # choose a default output

Sample

- Find out the ed2k link in html files

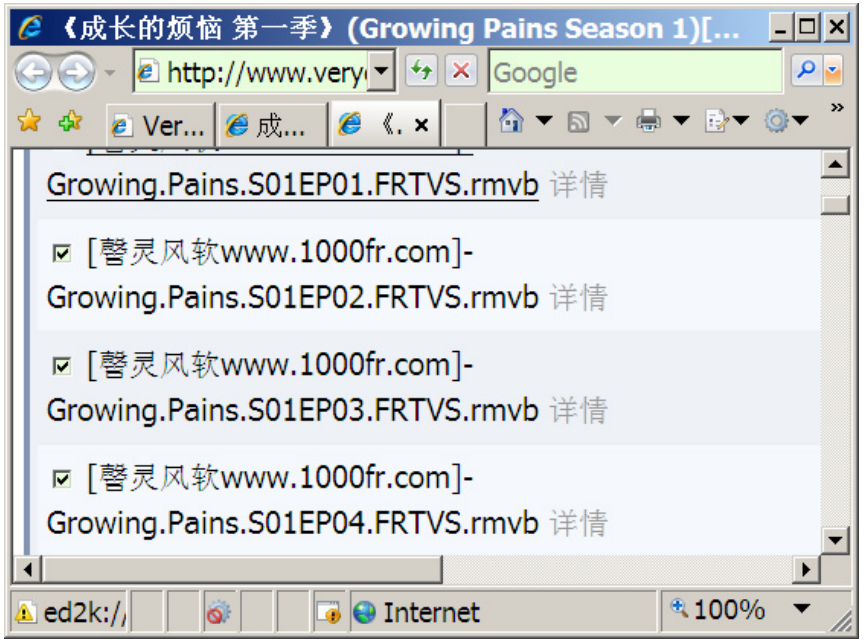
特征

```
<a
  href="ed2k://|file|%5B...S01EP01.FRTVS.rmvb|110711118|dc6cc75a
  e43d224f2ed77e4ae00228d0|h=VPJ7LDHO5R6ECOVFG2SYSS74O2ZMMKLT|/"
  ed2k="ed2k://|file|%5B%E...
```

最新可改用**wget --no-cookie --no-check-certificate https://..**

```
httpget http://www.verycd.com/topics/146912/ | perl xed2k.pl
```

```
#!/usr/bin/perl;
use strict;
my @lines = grep /href=\"ed2k\:/,
    <>;
chomp @lines;
foreach (@lines) {
   $_ =~ s/.*ed2k\:/ed2k:/;
   $_ =~ s/\".*//;
}
print "$_\n" foreach @lines;
1;
```



想想如何将这个脚本和电驴下载工具互动起来？

File Hanle --- Binary mode files

- **binmode(FD)**
- **read(FD, $var, $len, $offset)**

```
# read 20 bytes from file bin.dat
open(FILE, "bin.dat");
binmode(FILE);
read(FILE, $buffer, 20);
close(FILE);
```

- **seek(FD, $pos, $start)**
- **tell(FD) # position of FD**
- **sysopen(), sysread(), syswrite()**
- **$integer = fileno(FD);** .....

Offset in $var, not to the file

0: Begin of file
1: Current position
2: End of file

File Handle --- File Tests, Glob

- Check if a file/dir/symbolic_link exists

```
die "File missing!" if not -e "data.txt";
```

- Check if *name* is a directory: `-d "name"`
- Check if *name* is a regular file: `-f "name"`
- Lots more -*x* operations in perlfunc

- Glob, return a list of matched file/dir names

```
@txt_files = <*.txt>;
@root_files= <c:/*.*>; # PC/MS-OS only
@gif_files = glob("*.gif");
```

这里的*?是文件名通配符，不能按照perlRE规则来理解.

The UNIX *fork*( ) and *exec*( )

- *fork*() The only way to start a new process.
  - Return child pid to parent, return 0 to child.
  - Copy all data sections. Code are usually shared.
  - File descriptors are shared.
- *exec*(str) loads a new image (code), never return.

```
if (0 == ($child_pid = fork)) {
        #in child
        exec("my_command.exe", "my_arg1", "my_arg2");
        # no more code here, since exec never returns!
} else {
        # in parent
        #... parent can do other things.
}
```

- *system*(str) run a child process and wait for return.
- *wait*(), *waitpid*(pid, FLAG)

File System Pipelines in Perl

- pipe( ) then fork( ), close one side use the other

```
pipe READHANDLE, WRITEHANDLE or die "Can't open pipe.";

my $message;
my $child_pid = fork();

die "Fail to fork!\n" if not defined $child_pid;

if ($child_pid == 0) {
        # in child
        print "I'm child. My pid is $$.\n";
        close READHANDLE;
        $message = "Hello father. " . int rand 100 . ".";
        print "Child send \"$message\" to father.\n";
        print WRITEHANDLE $message;
        close WRITEHANDLE;
} else {
        # in parent
        print "I'm father, pid is $$. Child pid is $child_pid.\n"
        close WRITEHANDLE;
        $message = join "", <READHANDLE>;
        print "Father received : $message\n";
#       waitpid $child_pid, 1;
}
```