火龙果•整理
uml.org.cn

PERL基础教程精华版

火龙果•整理
uml.org.cn

主要内容

- PERL简介
- PERL脚本的编写
- PERL变量
- Perl语法
- PERL与正则表达式
- PERL示例

火龙果•整理
uml.org.cn

PERL释义

Practical Extraction and Report Language

实用摘录和报告语言，但它其实不是缩写

Perl的历史

Larry Wall，1987.12.18

火龙果•整理
uml.org.cn

Perl的环境准备

- **Unix：大多数内置**
- **一些软件也内置perl，比如apache/oracle**
- **Windows：ActivePerl5.10.0**

火龙果•整理
uml.org.cn

Perl的工具

编辑工具：**notepad，vi**

IDE：Komodo，Perl Dev Kit

CGI：Top perl studio，Perl builder,perl edit，perlwiz，Mod_perl

火龙果•整理
uml.org.cn

Perl的功能

- 脚本语言，解释执行，无需编译
- 具有编译语言如c、Java的功能，又有shell脚本的方便
- 无数据类型区分，适于不太复杂的程序
- 适于不要求速度，不在乎内存CPU等系统资源的任务
- 强大的字符串处理功能
- 灵活或复杂的正则表达式
- 大多数平台支持，除了专用模块，可在不同平台运行

火龙果•整理
uml.org.cn

Perl的应用

- Web编程：CGI，XML处理
- 系统管理
- 网络编程（安全脚本）.
- 数据库管理
- 图像处理
- 其他众多的领域。。。

火龙果•整理
uml.org.cn

一个示例

```
#!/usr/bin/perl
Print “This is my first perl program\n”;
$a=<>;
Print $a;
```

- 第一行： #!/usr/bin/perl 由什么程序执行以下的内容
- 注释： #
- 输入：<>
- 输出： print
- $a：变量，无需指定数据类型

火龙果•整理
uml.org.cn

perl的四种变量

- Scalar：标量，以$开始，后面以字母或_开头，再后面可以是字母或数字
- array：数组，列表，以@开头
- Hash： 哈希，散列，以%开头
- 文件：大写字母

- 区分大小写，$Var, $VAR, $var
- 内置变量$/，$@等

火龙果•整理
uml.org.cn

字符串变量

- 由双引号或单引号标识的一组字符组成。
- 最少0个字符（“”为空串），最多可以占满内存，末尾不含null(‘\0’)
- “${str}ing” = $str + “ing” != $string
- 记住一些常用的转义字符
- print “the \$var is $var.”
- 注意单引号的用法：不替换、不转义

```
$var=“str”;
print “this is $var”; # ”this is str”
print ‘this is $var’;  # ‘this is $var’
```

火龙果•整理
uml.org.cn

变量初值

- 未创建时状态为undef，到达文件尾也为undef
- 说明变量为未定义：undef $a;
- 用在条件判断中：　if(undef $a)
- 代替不关心的变量：

  $s=“a:b:c:d"; ($a1,undef,undef,$d1)=split(/:/, $s);
- 如果有undef变量又不知在哪，可加-w参数进行提示

  #!/usr/bin/perl -w
- 创建后状态为defined　一般用在条件判断中　if(defined $a)
- 整数初值为0，字符串初值为空串””。一般未赋值就使用时

  $result = $undefined + 2;

火龙果•整理
uml.org.cn

相关函数

- length()：字符串长度
- uc, lc, ucfirst, lcfirst：改变大小写函数
- substr, index, pos：字符串函数
- sin等三角函数
- rand(), srand()：随机发生函数

```
$lastchar = chop($str) # 截去最后一个字符
$result = chomp($str) # 截去末尾的行分隔符（通常为”\n”），行分隔符由$/定义
```

火龙果•整理
uml.org.cn

控制结构

- if(condition1){} elsif(condition2){}else{}
- unless(){}
- until(){}
- do{} until()
- while(){}
- do {} while ()
- for(;;){}
- foreach循环语句

火龙果•整理
uml.org.cn

foreach

- 语法：foreach $w(list|array){statement}
- ()内可以是数组@a，也可以是列表(1,2,3)
- 数组元素值可以修改，列表则是常量
- $w不影响本来已定义的变量$w，循环结束后恢复
- 可以用$a(@a)用相同的变量名称
- 示例：
  - foreach $a(@a){}用于数组
  - foreach $a(1,2,3,4){}用于列表
  - foreach $k(keys %h){}用于哈希/散列
  - foreach $a(@a[1,2,4]){}仅对数组部分元素
  - foreach (@a){}缺省循环变量为$_

火龙果•整理
uml.org.cn

循环控制

- last：退出循环
- next：进入下一循环
- redo：重新执行本次循环
- goto：跳转
- continue{statement}

火龙果•整理
uml.org.cn

单行条件语句与循环语句

- print $a if $a==0;
- print $a unless($a==0);
- print $a while ($a-->=0);
- print $a until ($a--==0)

- 用||，&&的条件语句：$a==0&&print $a;
  open(F,'file')||die "can't open";
- die函数：在控制台标准错误输出信息后退出程序。
- warn：　输出信息后不退出程序，只起警报作用。
- $!：内部变量，包含错误代码。
- $@：内部变量，包含错误信息。

火龙果•整理
uml.org.cn

列表——数组的形式

- 形式：(1,”a”,2.3, $a, $x+1)，其元素可以是数字、字符串、变量、表达式
- 空列表()，单元素列表(2)不同于标量2
- qw(1 $a str)
  - ()可以用其他符号表示，如<>
  - 元素可以是数值、变量、不带引号的字符串，中间用空格分开

火龙果•整理
uml.org.cn

范围表示的列表

..：范围运算符，每次增加1，如1..3

- (1..6)=(1,2,3,4,5,6)
- (1,2..5,6)=(1,2,3,4,5,6)
- (3..3)=(3)
- (2.4..5.3)=(2.4,3.4,4.4)
- (4.5..1.6)=()
- (“aa”..”ad”)=(“aa”,”ab”,”ac”,”ad”)
- $month=(’01’..’31’)
- ($a,$a+3)=(3,4,5,6) if $a=3

火龙果•整理
uml.org.cn

数组——列表的存储

- @a=(1,2,3)，不同于$a，初始值为()
- 元素形式：$a[0]表示第一个元素，索引从0开始，$a[-1]表示倒数第一个元素
- 数组的赋值：
  - @a=(1,2,3,4); @b=@a;
  - @b=(2,3); @a=(1,@a,4);
  - @a=<>; #从屏幕输入进行赋值，按下CTRL-d结束
  - 改变元素的值：$a[1]=3;
  - 超出数组大小的元素赋值：$a[5]=6; #自动增长，其他元素为NULL
  - 读取不存在的元素为空：$b = $a[6];

火龙果•整理
uml.org.cn

数组的读出

@a=(1,2,3);

- $a=$a[1];
- ($x, $y, $z)=@a; → $x=1, $y=2, $z=3;
- ($x, $y)=@a; → $x=1, $y=2;
- ($a,$b,$c,$d)=@a; → $a=1, $b=2, $c=3, $d="";
- $a=@a=$#a+1; # $a为数组长度，$#a为数组的最后一个元素的索引
- ($a)=@a; # 数组的第一个元素$a[0]
- 打印数组：print @a; # 元素直接相连
  print "@a"; # 元素之间用空格分开

火龙果•整理
uml.org.cn

数组片段

@a=(1..5)

- @sub=@a[0,1,3];
- @a[1,3]=(“a”,”b”);
- @b=(1,2,3); @sub=@a[@b];
- @a[1,2]=@a[2,1];
- @a[1,2,3]=@a[3,2,4];

火龙果•整理
uml.org.cn

数组操作函数

- sort：缺省按字母排序

$a, $b表示数组元素，@_代表数组本身

- reverse @a; # 取数组的逆序
- chop @a; # 每个元素截去最后一个字符
- shift(@a); # 删除数组第一个元素并返回该值，缺省对@ARGV数组
- unshift(@a); # 在数组头部添加元素，返回新数组长度
- push(@a,$a); # 在数组末尾添加元素
- pop(@a); # 删除数组末尾元素

火龙果•整理
uml.org.cn

数组操作函数（二）

- join(连接符号, @a)把数组连接为一个字串

  @a=(‘a’,’b’); join(‘:’, @a)=“a:b”;

- split(/分隔符/, 分割串, 长度)

  分隔符：缺省为空格，可省略

  分割串：缺省为$_，可省略

  长度：可省略，缺省为全部分割

  $s=“a,b,c”; @a=split(/,/,$s); → @a=(‘a’,’b’,’c’);

  @a=split(/,/,$s, 2); → @a=(‘a’,’b’,’c’);

火龙果•整理
uml.org.cn

数组函数（三）

- splice函数： @ret = splice(@a, skip, length, @newlist);
  - 对数组@a进行操作，跳过skip个元素，然后用@newlist替换length个元素
  - @newlist长度可以不为length，但其替换长度总为length
  - 如果length=0表示为插入；如果@newlist=()则表示为删除
  - 当length和@newlist都省略时表示全部删除
- @found=grep(/pattern/, @search)对数组@search的每个元素进行搜索匹配pattern，匹配元素返回到@found
- map(expr, @list)对数组@list的每个元素进行expr运算，返回运算后的数组。元素用$_替代，如map($_+1, (1,2))→(2,3)

火龙果•整理
uml.org.cn

二维数组

@aoa=[[1,2,3],[‘a’,’b’,’c’]];

- 该数组的元素为两个数组
- 子数组访问： @{$aoa->[0]}→(1,2,3)
- 子数组元素列表：@{$aoa->[0]}[0,1,2]
- 子数组元素访问：$aoa->[0][0]

@a=(1,2,3);@b=(‘a’,‘b’,‘c’);

$aoa=[[@a],[@b]]; $aoa->[0][0];

火龙果•整理
uml.org.cn

关联数组：哈希/散列

- 关联数组的表示：%h=(1,’a’,2,’b’);
- 关联数组的下标为关键字key，由key得到的值为value
- 上式的意义是%h=(1=>‘a’,2=>’b’);
- 元素形式$h{1}=‘a’

火龙果•整理
uml.org.cn

关联数组的赋值

- %a=(“key1”,1,”key2”,2);
- %h=@a;
- @a=%h;
- %h1=%h2;
- ($a,%h)=@array;
- %h=(%first, %second);
- %h1=(%h2, ‘k’, ‘v’)
- 函数的返回：%h=split();
- @keys=(‘a’,’b’,’c’); @hash{@keys}=@hash{reverse @keys};

火龙果•整理
uml.org.cn

关联数组操作函数

- keys(%hash), values(%hash)分别返回键和值的列表，返回元素无顺序
- ($key, $value)=each(%hash)效率高于先用foreach $k (keys %h), 再用$hash{$k}
- exists $hash{‘key’}判断关键字是否存在
- undef(%h)相当于删除散列%h=();
- delete

火龙果•整理
uml.org.cn

关联数组的顺序

```
foreach $w(sort keys(%hash)) # 按照字符串排序
```

或者

```
foreach $w(sort {$a<=>$b} keys(%hash)) #数值排序
```

火龙果•整理
uml.org.cn

文件

- 存放于磁盘，用于读写访问，访问前必须先打开文件，结束时关闭文件

open(HANDLE, “>filename”) || die $!;

- 成功返回非零，失败返回零
- HANDLE：文件句柄，用来代表操作的文件。以字母开头，字母、数字、下划线组成，一般用大写字母
- 缺省打开的句柄STDIN, STDOUT, STDERR, 文件描述符为0，1，2。不必调用open就可以直接访问
  - STDIN：键盘输入，控制台。
  - STDOUT：屏幕，显示屏。
  - STDERR：错误输出，显示屏。

火龙果•整理
uml.org.cn

文件访问模式

- 只读：open(F, “<filename”);或者open(F, “filename”); 文件不存在则打开失败
- 只写：open(F, “>filename”); 文件不存在则创建新文件，存在则清空重写
- 追加：open(F, “>>filename”);在存在的文件后面追加内容
- 读写：open(F, “+<filename”);可读可写，文件不存在则失败，否则覆盖原文件
- 读写：+>，文件不存在则创建，存在则清空再写
- 读写：+>>，文件不存在则创建，存在则追加
- 管道：|，

open(F,“| cat>hello”);把文件F的输出(print F $a)作为｜后的输入。

open(F,“comm|”);把comm的输出作为F的输入。以下的内容只要读出。comm为命令。

火龙果•整理
uml.org.cn

文件缓冲

	缓冲	无缓冲
打开：	open,sysopen	sysopen
关闭	close	
读	<>,readline	sysread
写	print	syswrite
定位	tell,seek	sysseek

火龙果•整理
uml.org.cn

读文件

- $line=<file>读一行到line，指针后移一行。缺省读到$_中。$/='\n'，为行分隔符，遇到它则为一行结束，行包含$/。可用chomp($s)去除此标志，行尾不含$/则不去除字符。可设置$/为其他字符串，遇到$/为行结束，chomp去除此字符串。
- @array=<file>文件内容全部读出，每行为一个元素。含回车。
- 当从STDIN中读时，可省略为<>。
- read(F,$in,len[,$offset])读入$in
- sysread(F,$in,len[,$offset]) getc(F)读一个字符

火龙果•整理
uml.org.cn

命令行参数

@ARGV：全局，$ARGV[0]是第一个参数，不是程序名。

<>是对$ARGV的引用。@ARGV一旦赋值，原值丢失。

1。第一次看到＜＞时，打开以$ARGV[0]中的文件。无参数则打开STDIN读。所以可以省略。

2。shift(@ARGV),元素数量减少一个。

3。＜＞读打开的文件中的所有行。

4。再读第二个参数表示的文件。

文件尾检测：eof和eof()。文件结束返回真。

@ARGV＝（”file1”,”file2”）;

while($line=<>){if(eof){print 'eof';}}

读取file1到末尾时，下一循环打开下一文件。每次读完一个文件输出eof.

if(eof()){print 'eof');}所有文件都读完才输出eof。

火龙果•整理
uml.org.cn

写文件

- print F (“str”);

  F文件句柄，后面为空格，省略F为STDOUT。

  str输出内容。可用单引号’，不进行变量替换，不加引号，计算出变量的值再输出。

  ()可省略。这是函数的特点。

- printf(“format str”,$a,$b…);同c中的printf，格式化串包含%m.nf的格式指示，后面依次是相应的值列表。

- write用于格式化输出。不是read的相应操作。
- syswrite(F,$data,length,$offset);同sysread

火龙果•整理
uml.org.cn

文件测试

–op expr

if(-e “file1”){print STDERR (“file1\n”);}文件是否存在。

-b是否为块设备　　-c是否为字符设备
-d是否为目录　　-e是否存在
-f是否为普通文件　　-g是否设置了setgid位
-k是否设置了sticky位　-l是否为符号链接
-o是否拥有该文件　　-p是否为管道
-r是否可读　　-s是否非空
-t是否表示终端　　-u是否设置了setuid位
-w是否可写　　-x是否可执行　　-z是否为空文件
-A距上次访问多长时间-B是否为二进制文件
-C距上次访问文件的inode多长时间　　-M距上次修改多长时间
-O是否只为“真正的用户”所拥有　　-R是否只有“真正的用户”可读
-S是否为socket　　-T是否为文本文件
-W是否只有“真正的用户”可写　　-X是否只有“真正的用户”可执行
-s 返回文件长度，-A-C-M返回天数。

火龙果•整理
uml.org.cn

正则表达式(模式匹配)

regular expression，规则表达式

模式匹配，在字符串中寻找特定序列的字符。

指定模式：由斜线包含，/def/即模式def。

匹配操作符 =~、!~

检验匹配是否成功

=~字符串是否匹配模式，匹配则为真，没有匹配则为假。!~不匹配为真，匹配为假。

```
$question=“expleaseding”
$question =~ /please/
$question!~/edit/
```

火龙果•整理
uml.org.cn

正则表达式的使用

用于条件判断：

```
if ($question =~ /please/) {  print ("Thank you for being polite!\n");  }
   else { print ("That was not very polite!\n");  }
```

grep：正则表达式只对简单变量匹配，如果是数组@a=~/abc/，则’2’=~/abc/。用grep(/abc/,@a);对数组中的每个元素匹配。

split(/abc/,$line)根据模式匹配分割字符串。

模式匹配的3种类型：
m//模式匹配，s///匹配并替换，tr///逐一替换，翻译

火龙果•整理
uml.org.cn

模式匹配之一：元字符

+ 一个或多个相同的前导字符(模式)。如：/de+f/指def、deef、deeeeef等。是对前一个匹配模式的重复，不是匹配后的字符的重复。如/d[eE]+/，匹配de,dee,dE,dEE,deE,dEe。不是匹配了e后再重复eee，就没有eE了。相当于/d[eE][eE][eE]…/。

* 匹配0个、1个或多个相同字符

？匹配0个或1个该前一个字符

. 匹配除换行外的所有单个字符，通常与*合用 .*所有任意数量字符。与前一字符结合，可不出现字符。相当于….

匹配指定数目的字符

{}指定所匹配字符的出现次数。如：/de{1,3}f/匹配def,deef和deeef；/de{3}f/匹配deeef；/de{3,}f/匹配不少于3个e在d和f之间；/de{0,3}f/匹配不多于3个e在d和f之间。

火龙果•整理
uml.org.cn

模式匹配之二：选择

- [] 匹配一组字符中的任一个。/a[0123456789]c/将匹配a加一个数字加c的字符串。与+联合使用例：/d[eE]+f/匹配def、dEf、deef、dEdf

- [^]。^表示除其之外的所有字符，如：/d[^deE]f/匹配d加非d,e,E字符加f的字符串

- [0-9] [a-z] [A-Z]　/a[0-9]c/　匹配任意字母或数字[0-9a-zA-Z]

- 字符"|"指定两个或多个选择来匹配模式。每个选择都是一个匹配或一组。不是单个字符。如：/def|ghi/匹配def或ghi。/x|y+/匹配x或y+。

例：检验数字表示合法性

```
if ($number =~ /^-?\d+$|^-?0[xX][\da-fa-F]+$/) {print ("$number is a legal integer.\n");}
else {print ("$number is not a legal integer.\n");  }
```

其中 ^-?\d+$ 匹配十进制数字，^-?0[xX][\da-fa-F]+$ 匹配十六进制数字。

火龙果•整理
uml.org.cn

转义符和定界符

模式中包含通常被看作特殊意义的字符，须在其前加斜线"\"。如：/*+/中*即表示字符*，而不是上面提到的一个或多个字符的含义。斜线的表示为/\\/。在PERL5中可用字符对\Q和\E来转义。从\Q开始到\E间的字符为原始字符，无特殊含义。

\d 任意数字 [0-9]　　\D 除数字外的任意字符 [^0-9]

\w 任意单词字符 [_0-9a-zA-Z] \W 任意非单词字符 [^_0-9a-zA-Z]

\s 空白 [\r\t\n\f]　　\S 非空白 [^ \r\t\n\f]

例：/[\da-z]/匹配任意数字或小写字母。

定界：^ 或 \A 仅匹配字符串首　　$ 或 \Z 仅匹配字符串尾

\b 匹配单词边界　　\B 单词内部匹配

/^def/只匹配行以def打头的字符串，/def$/只匹配以def结尾的字符串，结合起来的/^def$/只匹配字符串def

^$和\A,\Z在多行匹配时用法不同。

火龙果•整理
uml.org.cn

示例

例1：检验变量名的类型：

```
if ($varname =~ /^\$[A-Za-z][_0-9a-zA-Z]*$/) {
  print ("$varname is a legal scalar variable\n");
} elsif ($varname =~ /^@[A-Za-z][_0-9a-zA-Z]*$/) {
  print ("$varname is a legal array variable\n");
} elsif ($varname =~ /^[A-Za-z][_0-9a-zA-Z]*$/) {
  print ("$varname is a legal file variable\n");
} else {
  print ("I don't understand what $varname is.\n");
}
```

例2：\b在单词边界匹配：/\bdef/匹配def和defghi等以def打头的单词，但不匹配abcdef。/def\b/匹配def和abcdef等以def结尾的单词，但不匹配defghi，/\bdef\b/只匹配字符串def。注意：/\bdef/可匹配$defghi，因为单词包括字母数字下划线， $并不被看作是单词的部分。

例3：\B在单词内部匹配：/\Bdef/匹配abcdef等，但不匹配def；/def\B/匹配defghi等；/\Bdef\B/匹配cdefg、abcdefghi等，但不匹配def,defghi,abcdef。

火龙果•整理
uml.org.cn

模式的重用

当模式中匹配相同的部分出现多次时，可用括号括起来，用\n来多次引用，以简化表达式。
把匹配的值存起来以后再用，和+模式的重复不同。
只在本次匹配可用。还可以在匹配外引用。
例：/\d{2}([\W])\d{2}\1\d{2}/ 匹配12-05-92，26.11.87，07 04 92等但不匹配12-05.92
注意：/\d{2}([\W])\d{2}\1\d{2}/ 不同于/(\d{2})([\W])\1\2\1/，后者只匹配形如17-17-17的字符串，而不匹配17-05-91等。

火龙果•整理
uml.org.cn

模式变量

在模式匹配后调用重用部分的结果可用变量$n，全部的结果，匹配模式用变量$&，包含不在括号中的。匹配处之前的部分用变量$`，匹配处之前的部分用变量$'。也可用列表一次取得。

```
$string = "This string contains the number 25.11.";
$string =~ /-?(\d+)\.?(\d+)/; # 匹配结果为25.11
$integerpart = $1; # now $integerpart = 25
$decimalpart = $2; # now $decimalpart = 11
$totalpart = $&; # now totalpart = 25.11
$_ = "This string contains the number 25.11.";
@result =~ /-?(\d+)\.?(\d+)/; 匹配得到的变量形成列表，可赋值给数组。
```

当匹配失败，$1的内容不确定，可能是从前匹配的内容。为避免匹配失败要进行是否匹配成功的判断，或直接赋值。

($m1,$m2)=($name=~/(ab).*(c)) 把()内的匹配值直接赋与$m1,$m2，不改变$1的值。

嵌套使用：/((aaa)*)/，最外层的括号为$1，内层为$2，$3。

火龙果•整理
uml.org.cn

匹配选项

g 匹配所有可能的模式，根据懒惰规则不加g只匹配一处。返回到数组中。

```
@matches = "balata" =~ /.a/g; # @matches = ("ba", "la", "ta")
```

匹配的循环：每次匹配记住上次的位置

```
while ("balata" =~ /.a/g) {
  $match = $&;
  print ("$match\n");
}
```

结果为：

```
ba
la
ta
```

当要匹配的字符串改变时重新开始搜索。

当使用了选项g时，可用函数pos来控制下次匹配的偏移：

```
$offset = pos($string);下一个匹配开始的位置
pos($string) = $newoffset;从此位置开始搜索匹配
```

火龙果•整理
uml.org.cn

匹配选项

i 忽略模式中的大小写：/de/i 匹配de,dE,De和DE。

m 将待匹配串视为多行，^符号匹配字符串的起始或新的一行的起始；$符号匹配任意行的末尾。以下例只匹配第一行为a，否则无匹配;

```
$line='a
b
c';
$line=~/^(.*)$/m;
```

s 将待匹配串视为单行 。.可以匹配\n。

/a.*bc/s匹配字符串axxxxx \nxxxxbc，但/a.*bc/则不匹配该字符串。

o 仅只执行一次变量替换

```
$var = 1;  $line = <STDIN>;
while ($var < 10) { $result = $line =~ /$var/o; $line = <STDIN>;$var++;}
```

第一次匹配1，第二次值为2，但仍匹配1。

x 忽略模式中的空白 。格式清晰

/\d{2} ([\W]) \d{2} \1 \d{2}/x等价于/\d{2}([\W])\d{2}\1\d{2}/。

火龙果•整理
uml.org.cn

匹配符号的优先级

象操作符一样，转义和特定字符也有执行次序

() 模式内存

+ * ? {} 出现次数

^ $ \b \B 锚

| 选项

火龙果•整理
uml.org.cn

扩展匹配模式

- (?<c>pattern)，其中c是一个字符，pattern是起作用的模式或子模式。

1、(?:pattern)不存贮括号内的匹配内容

括号内的子模式将存贮在内存中，此功能即取消存贮该括号内的匹配内容，如/(?:a|b|c)(d|e)f\1/中的\1表示已匹配的d或e，而不是a或b或c。

2、 /(?option)pattern/内嵌模式选项

通常模式选项置于其后，有四个选项：i、m、s、x可以内嵌使用，等价于/pattern/option。/(?i)[a-z]+/=/[a-z]+/i

3、(?#注释)模式注释

PERL5中可以在模式中用?#来加注释，如：

if ($string =~ /(?i)[a-z]{2,3}(?# match two or three alphabetic characters)/ { ... }

4、(?)取消贪婪

"a12b38b" /a.*b/ 全部匹配，当/a(.*?)b/时匹配a12b。

同样的有*?,+?,??,{x}?,{x,}?,{x,y}?,

火龙果•整理
uml.org.cn

扩展模式匹配

5、/pattern(?=string)/肯定的和否定的预见匹配.?= ?!

匹配后面为string的模式，相反的，(?!string)匹配后面非string的模式，如：

```
$string = "25abc8";
$string =~ /abc(?=[0-9])/;
$matched = $&; # $&为已匹配的模式，为abc，不是abc8
```

例1。$line=“block1 first block2 second block3 third”

```
$line=~/block\d(.*?)(?=block\d|$)/g;print $1;
```

例2。使用while

```
$line="begin <data1> begin <data2> begin <data3>";
while($line=~/begin(.*?)(?=begin|$)/sg)
{ push(@blocks,$1);}
```

火龙果•整理
uml.org.cn

替换操作

s/pattern/replace/，将字符串中与pattern匹配的部分换成replace。替代字符串不是模式。如$string = "abc123def";

```
$string =~ s/123/456/; # now $string = "abc456def";
```

- 在替换部分可使用模式变量$n，如s/(\d+)/[$1]/，但在替换部分不支持模式的特殊字符，如{},*,+等，如s/abc/[def]/将把abc替换为[def]。

替换操作符的选项： g,i,m,o,s,x,e

e 替换字符串作为表达式。e选项把替换部分的字符串看作表达式，在替换之前先计算其值，如：

```
$string = "0abc1";
$string =~ s/[a-zA-Z]+/$& x 2/e; # now $string = "0abcabc1"
```

火龙果•整理
uml.org.cn

翻译操作

- tr/string1/string2/。string1中的第一个字符替换为string2中的第一个字符，把string1中的第二个字符替换为string2中的第二个字符，依此类推。如：$string = "abcdefghicba";
- $string =~ tr/abc/def/; # now string = "defdefghifed"
- 当string1比string2长时，其多余字符替换为string2的最后一个字符；当string1中同一个字符出现多次时，将使用第一个替换字符。

翻译操作符的选项：c 翻译所有未指定字符，d 删除所有指定字符，s 把多个相同的输出字符缩成一个

$string =~ tr/\d/ /c;把所有非数字字符替换为空格。
$string =~ tr/\t //d；删除tab和空格；
$string =~ tr/0-9/ /cs；把数字间的其它字符替换为一个空格。