

1. perl语法速查

1.1 技巧文章

- [cpan](#)
 - [bit-vector](#)
 - [text-table](#)
- [clipboard复制中文字符到剪切板乱码](#)
- [perl_crlf](#)
- [perl向上取整](#)
- [perl打包linux](#)
- [perl打包windows](#)
- [perl非贪婪匹配](#)
- [windows平台perl推荐](#)
- [使用cpanm安装包](#)
- [剪贴板分析工时](#)
- [用perl产生verilog文件](#)

1.2 perl单行命令

```
# linux里面可以用"
perl -e "print 0;"
perl -E "print 0;"

perl -e "$i = int(rand(100000)); print $i" # 单行命令产生随机数
```

1.3 正则表达式

1.3.1 匹配多次

/a{5,15}/将匹配5个到15个a中的任意一个（包括5，和15）

/(fred){3,}/将在一行中有3个或者更多个 fred

/\w{8}/将严格的匹配8个word

星号(*)等同于{0,}，表示0个或多个。

加号(+)等同于{1,}，表示1个或多个。

而问号(?)则等同于{0,1}。

1.3.2 匹配非

[^def]将匹配上这三个字符中之外的任意单个字符

1.4 number

浮点数

```
1.25
255.000
255.0
7.25e45 # 7.25 times 10 to the 45th power (a big number)
-6.5e24 # negative 6.5 times 10 to the 24th
        # (a big negative number)
-12e-24 # negative 12 times 10 to the -24th
        # (a very small negative number)
-1.2E-23 # another way to say that the E may be uppercase
```

整数

```
0
2001
-40
255
61298040283768
61_298_040_283_768 可自动识别下划线，与上面数字相同，方便人眼观察。
```

```
0xff # FF hex, also 255 decimal 十六进制数
```

1.5 list

```
@ar = ();
push @ar, $xx;
push @ar, $yy; # 将变量压到数组最后
$aa = shift @ar; # 取出数组第一个元素给变量，数组长度减1
$bb = pop $ar; # 取出数组最后一个元素给变量，数组长度减1
```

```
@a = (1, 2, 3);
$a = (21, 22, 33);
print "a = $a\n"; # 结果为33,不清楚为什么为这样，不要将数组直接给普通变量
print "a = @a\n";
```

输出：

```
a = 33
a = 1 2 3
```

```
@a = ((1, 2), (3, 4));
for $i (@a) {
    print "$i\n";
}
```

输出：

```
1
2
```

3
4

1.6 hash

```
%hash = ();

$hash{"a"} = 1;
$hash{"b"} = 2;
$hash{"c"} = 3;

for $key (keys %hash) {
    $value = $hash{$key};
    print "key = $key, value = $value\n";
}

delete $hash{"a"};

for $key (sort keys %hash) {
    $value = $hash{$key};
    print "key = $key, value = $value\n";
}

my @descending = reverse sort { $a <=> $b } @some_numbers;
my @descending = sort { $b <=> $a } @some_numbers;

for $key (sort { $a <=> $b } keys %hash) {
    $value = $hash{$key};
    print "key = $key, value = $value\n";
}

for $key (sort { $hash{$a} <=> $hash{$b} } keys %hash) {
    #按值的升序排列
}
for $key (sort { $hash{$a}->{idx} <=> $hash{$b}->{idx} } keys %hash) {
    #按ref值的升序排列
}
for $key (sort { $hash{$b} <=> $hash{$a} } keys %hash) {
    #按值的降序排列
}
for $key (sort { $hash{$b}->{idx} <=> $hash{$a}->{idx} } keys %hash) {
    #按ref值的降序排列
}
```

1.7 reference

1.7.1 ref for array

```
@a = (1, 2, 3);
print "@a\n";      # result: 1 2 3

%hash = ();
$hash{"a"} = \@a; # reference of @a

$ref = $hash{"a"};
print "$ref\n";    # result: ARRAY(0x25d7338)
print "@$ref\n";  # result: 1 2 3
```

```
@a = (1, 2, 3);
@b = (4, 5, 6);
@c = (7, 8, 9);
@o = (\@a, \@b); # 数组放进数组
print "@o\n";
push @o, \@c;
for $i (@o) {
    print "@$i\n";
}
输出：
ARRAY(0x654b20) ARRAY(0x65f568)
1 2 3
4 5 6
7 8 9
```

1.7.2 ref for hash

t.pl

```
sub new_block {
    return {
        name => "",
        id => "",
        blk => {},
    }
}

sub new_hash {
    return {
    }
}

$blk = &new_block;
$blk->{name} = "hello";
$blk->{id} = 1;
```

```

$blk2 = &new_hash;
$blk2->{a} = "a0";
$blk2->{b} = "b0";

$blk->{blk} = $blk2;

print "-----\n";
foreach $key (sort keys %{$blk}) {
    print "$key  $blk->{$key}\n";
}
print "\n";

print "-----\n";
foreach $key (sort keys %{$blk->{blk}}) {
    print "$key  $blk->{blk}->{$key}\n";
}
print "\n";

```

运行结果：

```

D:\>perl t.pl
-----
blk  HASH(0x100a950)
id  1
name  hello

-----
a  a0
b  b0

```

1.7.3 hash包array

```

@a = (1, 2, 3);
@b = (11, 22, 33);
$hash{a} = \@a;
$hash{b} = \@b;
foreach $key (sort keys %hash) {
    $ar_ref = $hash{$key};
    for $i (@{$ar_ref}) {
        print "$i\n";
    }
}
$c = @{$hash{b}}[2]; #输出数组b的第2个值
print "c = $c\n";

```

输出：

```
1
2
3
11
22
33
c = 33
```

1.7.4 array包hash

```
$hash_a{a} = 1;
$hash_b{a} = 2;
@a = (\%hash_a, \%hash_b);
for $i (@a) {
    print "$i\n";
}
for $i (@a) {
    print "$i->{a}\n";
}
```

输出：

```
HASH(0x6efcc0)
```

```
HASH(0x6efd20)
```

```
1
2
```

```
$hash_a{a} = 1;
$hash_b{a} = 2;
@a = (%hash_a, %hash_b);
for $i (@a) {
    print "$i\n";
}
```

输出：

```
a
1
a
2
```

```
@ar = ();
sub new_hash {
    return {}
}
for ($i=0; $i<3; $i++) {
    $hash = &new_hash;
    push @ar, $hash;
    $hash->{value} = $i;
    $hash->{value2} = "value_$i";
}
for $i (@ar) {
```

```

    print "$i\n";
    print "$i->{value}\n";
    print "$i->{value2}\n";
}

```

输出：

```

HASH(0x1e9520)
0
value_0
HASH(0x24d42b8)
1
value_1
HASH(0x24d4390)
2
value_2

```

1.8 字符串操作

```

"hello" . "world"           # same as "helloworld"
"hello" . ' ' . "world"    # same as 'hello world'
'hello world' . "\n"      # same as "hello world\n"

"fred" x 3                  # is "fredfredfred"
"barney" x (4+1)           # is "barney" x 5, or "barneybarneybarneybarneybarney"
5 x 4.8                    # is really "5" x 4, which is "5555"

```

1.9 文件和目录操作

1.9.1 读写文件

```

#open读文件
open(my $fh, "<", "input.txt")
  or die "Can't open < input.txt: $!";
and for writing:

#open写文件
open(my $fh, ">", "output.txt")
  or die "Can't open > output.txt: $!";

#close文件
close($handle)
  || warn "close failed: $!";

```

```

#写文件
print $fh "hello\n";

```

```

#读文件

```

```
while ($line =<$fh>) {  
    print $line;  
}
```

1.9.2 readdir

```
# 读目录  
my $dir_to_process = '/etc';  
opendir my $dh, $dir_to_process or die "Cannot open $dir_to_process: $!";  
foreach $file (readdir $dh) {  
    print "one file in $dir_to_process is $file\n";  
}  
closedir $dh;
```

1.9.3 删除文件 unlink

```
unlink 'slate', 'bedrock', 'lava';  
unlink qw(slate bedrock lava);  
unlink glob '*.o';
```

1.9.4 rename

```
rename 'old', 'new';
```

1.9.5 mkdir rmdir

```
mkdir 'fred', 0755  
rmdir $temp_dir;
```

1.9.6 chmod

```
chmod 0755, 'fred', 'barney';
```

1.9.7 chown

```
chown $user, $group, glob '*.o';
```

1.9.8 获取当前目录PATH

有两种方法：

```
use Cwd;
```

```
my $dir = getcwd;  
#$dir中即为当前目录的完整路径信息。
```

```
my $dir = $ENV{'PWD'};  
#ENV是一个散列，用于存放环境变量，PWD是Linux的环境变量，表示当前所在目录。
```

1.10 perl处理json

```
1. use JSON;  
2. use Data::Dumper;  
3. local $Data::Dumper::Sortkeys = 1;  
4.  
5.  
6.  
7.  
8. ##### use JSON; # imports encode_json, decode_json, to_json  
   and from_json.  
9. #####  
10. ##### # simple and fast interfaces (expect/generate UTF-8)  
11. #####  
12. ##### $utf8_encoded_json_text = encode_json  
    $perl_hash_or_arrayref;  
13. ##### $perl_hash_or_arrayref = decode_json  
    $utf8_encoded_json_text;  
14. #####  
15. ##### # OO-interface  
16. #####  
17. ##### $json = JSON->new->allow_nonref;  
18. #####  
19. ##### $json_text = $json->encode( $perl_scalar );  
20. ##### $perl_scalar = $json->decode( $json_text );  
21. #####  
22. ##### $pretty_printed = $json->pretty->encode( $perl_scalar );  
    # pretty-printing  
23.  
24.  
25.  
26. my %rec_hash = ( 'a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5 );  
27. my $json = encode_json \%rec_hash;  
28. print "$json\n";  
29.  
30. $ha_ref = decode_json $json;  
31.  
32. print Dumper($json);  
33. print Dumper($ha_ref);  
34.  
35.  
36. # using custom sort function
```

```
37. # local $Data::Dumper::Sortkeys = sub { [ sort keys %{\+shift} ] };
38.
39. print Dumper {
40.     car => {
41.         '10' => 'y',
42.         '11' => 'y',
43.         '12' => 'y',
44.         '13' => 'y'
45.     },
46.     bus => {
47.         '10' => 'y',
48.         '11' => 'y',
49.         '12' => 'y',
50.         '13' => 'y'
51.     },
52.     tr => {
53.         '10' => 'y',
54.         '11' => 'y',
55.         '12' => 'y',
56.         '13' => 'y'
57.     }
58. }
```

UTF8与GBK的转换

```
use Encode;
gbk转uft-8
$line = encode("utf-8", decode("gbk", $line));
或
$line = encode_utf8(decode("gbk", $line));
utf-8转gbk
$line = encode("gbk", decode("utf8", $line));
uft-8转gb2312
$line = encode("gb2312", decode("utf8", $line));
```