Perl 引用 Perl 文件操作

Perl 格式化输出

Perl 是一个非常强大的文本数据处理语言。

Perl 中可以使用 format 来定义一个模板，然后使用 write 按指定模板输出数据。

Perl 格式化定义语法格式如下：

```
format FormatName =
fieldline
value_one, value_two, value_three
fieldline
value_one, value_two
.
```

参数解析：

- **FormatName** ：格式化名称。
- **fieldline** ：一个格式行，用来定义一个输出行的格式,类似 @,^,<,>,| 这样的字符。
- **value_one,value_two.....** ：数据行，用来向前面的格式行中插入值,都是perl的变量。
- **.** ：结束符号。

以下是一个简单是格式化实例：

实例

```
#!/usr/bin/perl
$text = "google runoob taobao";
format STDOUT =
first: ^<<<<< # 左边对齐，字符长度为6
$text
second: ^<<<<< # 左边对齐，字符长度为6
$text
third: ^<<<< # 左边对齐，字符长度为5，taobao 最后一个 o 被截断
$text
.
write
```

执行以上实例输出结果为：

```
first: google
second: runoob
third: taoba
```

格式行(图形行)语法

- 格式行以 @ 或者 ^ 开头，这些行不作任何形式的变量代换。
- @ 字段(不要同数组符号 @ 相混淆)是普通的字段。
- @,^ 后的 <, >,| 长度决定了字段的长度，如果变量超出定义的长度,那么它将被截断。
- <, >,| 还分别表示,左对齐,右对齐,居中对齐。
- ^ 字段用于多行文本块填充。

值域格式

值域的格式，如下表所示：

格式	值域含义
@<<<	左对齐输出
@>>>	右对齐输出
@|||	中对齐输出
@##.##	固定精度数字
@*	多行文本

每个值域的第一个字符是行填充符，当使用@字符时，不做文本格式化。

在上表中，除了多行值域@*，域宽都等于其指定的包含字符@在内的字符个数，例如：

```
@###.##
```

表示七个字符宽，小数点前四个，小数点后两个。

实例如下：

实例

```
#!/usr/bin/perl
format EMPLOYEE =
===================================
@<<<<<<<<<<<<<<<<<<<<<<< @<<
$name, $age
@#####.##
$salary
===================================
.
select(STDOUT);
$~ = EMPLOYEE;
@n = ("Ali", "Runoob", "Jaffer");
@a = (20,30, 40);
@s = (2000.00, 2500.00, 4000.000);
$i = 0;
foreach (@n){
$name = $_;
```

```
$age = $a[$i];
$salary = $s[$i++];
write;
}
```

以上实例输出结果为：

```
 ===================================
Ali                     20
  2000.00
 ===================================
 ===================================
Runoob                  30
  2500.00
 ===================================
 ===================================
Jaffer                  40
  4000.00
 ===================================
```

格式变量

- $~ ($FORMAT_NAME) ：格式名字 $^ ($FORMAT_TOP_NAME) ：当前的表头格式名字存储在
- $% ($FORMAT_PAGE_NUMBER) ：当前输出的页号
- $= ($FORMAT_LINES_PER_PAGE) ：每页中的行数
- $| ($FORMAT_AUTOFLUSH) ：是否自动刷新输出缓冲区存储
- $^L ($FORMAT_FORMFEED) ：在每一页(除了第一页)表头之前需要输出的字符串存储在

以下是一个简单是使用 $~ 格式化的实例：

实例

```
#!/usr/bin/perl
$~ = "MYFORMAT"; # 指定缺省文件变量下所使用的格式
write; # 输出 $~ 所指定的格式
format MYFORMAT = # 定义格式 MYFORMAT
================================
Text # 菜鸟教程
================================
.
write;
```

执行以上实例输出结果为：

```
 ================================
       Text # 菜鸟教程
 ================================
```

```
================================
      Text # 菜鸟教程
================================
```

如果不指定$~的情况下，会输出名为STDOUT的格式：

实例

```
#!/usr/bin/perl
write; # 不指定$~的情况下会寻找名为STDOUT的格式
format STDOUT =
~用~号指定的文字不会被输出
---------------
STDOUT格式
---------------
.
```

执行以上实例输出结果为：

```
---------------
  STDOUT格式
---------------
```

以下实例我们通过添加报表头部信息来演示 $^ 或 $FORMAT_TOP_NAME 变量的使用：

实例

```
#!/usr/bin/perl
format EMPLOYEE =
===================================
@<<<<<<<<<<<<<<<<<<<<<<< @<<
$name, $age
@#####.##
$salary
===================================
.
format EMPLOYEE_TOP =
===================================
Name Age
===================================
.
select(STDOUT);
$~ = EMPLOYEE;
$^ = EMPLOYEE_TOP;
@n = ("Ali", "Runoob", "Jaffer");
@a = (20,30, 40);
@s = (2000.00, 2500.00, 4000.000);
$i = 0;
foreach (@n){
$name = $_;
$age = $a[$i];
$salary = $s[$i++];
```

```
write;
}
```

以上实例输出结果为：

```
 ===================================
Name                    Age
 ===================================
 ===================================
Ali                     20
  2000.00
 ===================================
 ===================================
Runoob                  30
  2500.00
 ===================================
 ===================================
Jaffer                  40
  4000.00
 ===================================
```

我们也可以使用 $% 或 $FORMAT_PAGE_NUMBER 为报表设置分页：

实例

```
#!/usr/bin/perl
format EMPLOYEE =
===================================
@<<<<<<<<<<<<<<<<<<<<<<< @<<
$name, $age
@#####.##
$salary
===================================
.
# 添加分页 $%
format EMPLOYEE_TOP =
===================================
Name Age Page @<
$%
===================================
.
select(STDOUT);
$~ = EMPLOYEE;
$^ = EMPLOYEE_TOP;
@n = ("Ali", "Runoob", "Jaffer");
@a = (20,30, 40);
@s = (2000.00, 2500.00, 4000.000);
$i = 0;
foreach (@n){
$name = $_;
$age = $a[$i];
```

```
$salary = $s[$i++];
write;
}
```

以上实例输出结果为：

```
==================================
Name                    Age Page 1
==================================
==================================
Ali                     20
  2000.00
==================================
==================================
Runoob                  30
  2500.00
==================================
==================================
Jaffer                  40
  4000.00
==================================
```

输出到其它文件

默认情况下函数write将结果输出到标准输出文件STDOUT，我们也可以使它将结果输出到任意其它的文件中。最简单的方法就是把文件变量作为参数传递给write，如：

```
write(MYFILE);
```

以上代码write就用缺省的名为MYFILE的打印格式输出到文件MYFILE中。

但是这样就不能用$~变量来改变所使用的打印格式。系统变量$~只对默认文件变量起作用，我们可以改变默认文件变量，改变$~，再调用write。

实例

```
#!/usr/bin/perl
if (open(MYFILE, ">tmp")) {
$~ = "MYFORMAT";
write MYFILE; # 含文件变量的输出，此时会打印与变量同名的格式，即MYFILE。$~里指定的值被忽略。
format MYFILE = # 与文件变量同名
==================================
输入到文件中
==================================
.
close MYFILE;
}
```

执行成功后，我们可以查看 tmp 文件的内容，如下所示：

```
$ cat tmp
================================
      输入到文件中
================================
```

我们可以使用select改变默认文件变量时，它返回当前默认文件变量的内部表示，这样我们就可以创建子程序，按自己的想法输出，又不影响程序的其它部分。

实例

```
#!/usr/bin/perl
if (open(MYFILE, ">>tmp")) {
select (MYFILE); # 使得默认文件变量的打印输出到MYFILE中
$~ = "OTHER";
write; # 默认文件变量，打印到select指定的文件中，必使用$~指定的格式 OTHER
format OTHER =
================================
使用定义的格式输入到文件中
================================
.
close MYFILE;
}
```

执行成功后，我们可以查看 tmp 文件的内容，如下所示：

```
$ cat tmp
================================
      输入到文件中
================================
================================
  使用定义的格式输入到文件中
================================
```

← Perl 引用 Perl 文件操作 →