Perl 目录操作 Perl 特殊变量

Perl 错误处理

程序运行过程中，总会碰到各式各样的错误，比如打开一个不存在的文件。

程序运行过程中如果出现错误就会停止，我们就需要使用一些检测方法来避免错误，从而防止程序退出。

Perl 提供了多中处理错误发方法，接下来我们一一介绍。

if 语句

if 语句 可以判断语句的返回值，实例如下：

```
if(open(DATA, $file)){
   ...
}else{
   die "Error: 无法打开文件 - $!";
}
```

程序中变量 $! 返回了错误信息。 我们也可以将以上代码简化为如下代码：

```
open(DATA, $file) || die "Error: 无法打开文件 - $!";
```

unless 函数

unless 函数与 if 相反，只有在表达式返回 false 时才会执行，如下所示：

```
unless(chdir("/etc")){
   die "Error: 无法打开目录 - $!";
}
```

unless 语句在你要设置错误提醒时是非常有用的。我么也可以将以上代码简写为：

```
die "Error: 无法打开目录!: $!" unless(chdir("/etc"));
```

以上错误信息只有在目录切换错误的情况下才会输出。

三元运算符

以下是一个三元运算符的简单实例:

```
print(exists($hash{value}) ? '存在' : '不存在',"\n");
```

以上实例我们使用了三元运算符来判断哈希的值是否存在。

实例中包含了一个表达式两个值，格式为：**表达式？值一：值二。**

---

warn 函数

warn 函数用于触发一个警告信息，不会有其他操作，输出到 STDERR(标准输出文件)，通常用于给用户提示：

```
chdir('/etc') or warn "无法切换目录";
```

---

die 函数

die 函数类似于 warn, 但它会执行退出。一般用作错误信息的输出：

```
chdir('/etc') or die "无法切换目录";
```

Carp 模块

在 Perl 脚本中，报告错误的常用方法是使用 warn() 或 die() 函数来报告或产生错误。而对于 Carp 模块，它可以对产生的消息提供额外级别的控制，尤其是在模块内部。

标准 Carp 模块提供了 warn() 和 die() 函数的替代方法，它们在提供错误定位方面提供更多信息，而且更加友好。当在模块中使用时，错误消息中包含模块名称和行号。

carp 函数

carp函数可以输出程序的跟踪信息，类似于 warn 函数，通常会将该信息发送到 STDERR：

```
package T;

require Exporter;
@ISA = qw/Exporter/;
@EXPORT = qw/function/;
use Carp;

sub function {
   carp "Error in module!";
}
1;
```

在脚本调用以下程序:

```
use T;
function();
```

执行以上程序，输出结果为：

```
Error in module! at test.pl line 4
```

cluck 函数

cluck() 与 warn() 类似，提供了从产生错误处的栈回溯追踪。

```
package T;

require Exporter;
@ISA = qw/Exporter/;
@EXPORT = qw/function/;
use Carp qw(cluck);

sub function {
   cluck "Error in module!";
}
1;
```

在脚本调用以下程序:

```
use T;
function();
```

执行以上程序，输出结果为：

```
Error in module! at T.pm line 9
    T::function() called at test.pl line 4
```

croak 函数

croak() 与 die() 一样，可以结束脚本。

```
package T;

require Exporter;
@ISA = qw/Exporter/;
@EXPORT = qw/function/;
use Carp;

sub function {
   croak "Error in module!";
}
1;
```

在脚本调用以下程序:

```
use T;
function();
```

执行以上程序，输出结果为：

```
Error in module! at test.pl line 4
```

confess 函数

confess() 与 die() 类似，但提供了从产生错误处的栈回溯追踪。

```
package T;

require Exporter;
@ISA = qw/Exporter/;
@EXPORT = qw/function/;
use Carp;

sub function {
   confess "Error in module!";
}
1;
```

在脚本调用以下程序:

```
use T;
function();
```

执行以上程序，输出结果为：

```
Error in module! at T.pm line 9
    T::function() called at test.pl line 4
```

Perl 目录操作 Perl 特殊变量