Perl 包和模块 Perl POD 文档

Perl 进程管理

Perl 中你可以以不同的方法来创建进程。

本教程将讨论一些进程的管理方法。

- 你可以使用特殊变量 **$$** 或 **$PROCESS_ID** 来获取进程 ID。
- %ENV 哈希存放了父进程，也就是shell中的环境变量，在Perl中可以修改这些变量。
- **exit()** 通常用于退出子进程，主进程在子进程全部退出后再退出。
- 所有打开的句柄会在子程序中被 dup() 函数复制, 所有关闭进程所有句柄不会影响其他进程 。

反引号运算符

使用反引号运算符可以很容易的执行 Unix 命令。你可以在反引号中插入一些简单的命令。命令执行后将返回结果：

```
#!/usr/bin/perl

@files = `ls -l`;

foreach $file (@files){
   print $file;
}

1;
```

执行以上程序，输出结果如下：

```
drwxr-xr-x 3 root root 4096 Sep 14 06:46 9-14
drwxr-xr-x 4 root root 4096 Sep 13 07:54 android
-rw-r--r-- 1 root root  574 Sep 17 15:16 index.htm
drwxr-xr-x 3  544  401 4096 Jul  6 16:49 MIME-Lite-3.01
-rw-r--r-- 1 root root   71 Sep 17 15:16 test.pl
……
```

system() 函数

你也可以使用 **system()** 函数执行 Unix 命令, 执行该命令将直接输出结果。默认情况下会送到目前Perl的STDOUT指向的地方，一般是屏幕。你也可以使用重定向运算符 > 输出到指定文件：

执行以上程序，输出结果如下：

```
drwxr-xr-x 3 root root 4096 Sep 14 06:46 9-14
drwxr-xr-x 4 root root 4096 Sep 13 07:54 android
-rw-r--r-- 1 root root  574 Sep 17 15:16 index.htm
drwxr-xr-x 3  544  401 4096 Jul  6 16:49 MIME-Lite-3.01
-rw-r--r-- 1 root root   71 Sep 17 15:16 test.pl
.....
```

你需要注意命令包含环境变量如 $PATH 或 $HOME 的输出结果，如下所示：

实例

```
#!/usr/bin/perl
$PATH = "我是 Perl 的变量";
system('echo $PATH'); # $PATH 作为 shell 环境变量
system("echo $PATH"); # $PATH 作为 Perl 的变量
system("echo \$PATH"); # 转义 $
1;
```

执行以上程序，输出结果如下：

```
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin
我是 Perl 的变量
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin
```

fork() 函数

Perl fork() 函数用于创建一个新进程。

在父进程中返回子进程的PID，在子进程中返回0。如果发生错误（比如，内存不足）返回undef，并将$!设为对应的错误信息。

fork 可以和 exec 配合使用。exec 函数执行完引号中的命令后进程即结束。

实例

```
#!/usr/bin/perl
if(!defined($pid = fork())) {
# fork 发生错误返回 undef
die "无法创建子进程: $!";
}elsif ($pid == 0) {
print "通过子进程输出\n";
exec("date") || die "无法输出日期: $!";
} else {
# 在父进程中
print "通过父进程输出\n";
$ret = waitpid($pid, 0);
print "完成的进程ID: $ret\n";
}
1;
```

执行以上程序，输出结果如下：

```
通过父进程输出
通过子进程输出
2016年 6月19日 星期日 22时21分14秒 CST
完成的进程ID: 47117
```

如果进程退出时,会向父进程发送一个CHLD的信号后,就会变成僵死的进程,需要父进程使用wait和waitpid来终止。当然,也可以设置$SIG{CHLD}为IGNORG：

实例

```
#!/usr/bin/perl
local $SIG{CHLD} = "IGNORE";
if(!defined($pid = fork())) {
# fork 发生错误返回 undef
die "无法创建子进程: $!";
}elsif ($pid == 0) {
print "通过子进程输出\n";
exec("date") || die "无法输出日期: $!";
} else {
# 在父进程中
print "通过父进程输出\n";
$ret = waitpid($pid, 0);
print "完成的进程ID: $ret\n";
}
1;
```

执行以上程序，输出结果如下：

```
通过父进程输出
通过子进程输出
2016年 6月19日 星期日 22时30分56秒 CST
完成的进程ID: -1
```

Kill 函数

Perl **kill('signal', (Process List))**给一组进程发送信号。signal是发送的数字信号，9为杀掉进程。

首先看看linux中的常用信号,见如下列表:

```
信号名          值          标注          解释
————————————————————————————————————————————
HUP             1           A             检测到挂起
INT               2           A             来自键盘的中断
QUIT            3           A             来自键盘的停止
ILL               4           A             非法指令
ABRT          6           C             失败
FPE             8           C             浮点异常
KILL              9           AF            终端信号
USR1          10          A             用户定义的信号1
```

SEGV	11	C	非法内存访问
USR2	12	A	用户定义的信号2
PIPE	13	A	写往没有读取者的管道
ALRM	14	A	来自闹钟的定时器信号
TERM	15	A	终端信号
CHLD	17	B	子进程终止
CONT	18	E	如果被停止则继续
STOP	19	DF	停止进程
TSTP	20	D	tty键入的停止命令
TTIN	21	D	对后台进程的tty输入
TTOU	22	D	对后台进程的tty输出

以下实例向进程 104 和 102 发送 SIGINT 信号：

实例

```
#!/usr/bin/perl
kill('INT', 104, 102);
1;
```

← Perl 包和模块 Perl POD 文档 →