mqxnongmin

随笔 - 1022 文章 - 0 评论 - 2

Unix shell输入输出重定向

敲代码的时候，适当地打印出一些进度或者日志信息经常能帮助我们跟踪程序的执行结果。可是，这些结果或者日志打印信息到屏幕上并不能作为以后检查问题的根据。这就是重定向的作用，敲代码的时候，我们能够方便的将相关的打印信息到屏幕或者是从键盘接收输入(这种优点就是避免直接操作文件)。利用重定向我们能够非常方便地将输入输出重定向到文件或者其他地方。

1、文件描写叙述符（以下部分来自维基百科）

文件描写叙述符(File descriptor)是计算机科学中的一个术语。是一个用于表述指向文件的引用的抽象化概念。

文件描写叙述符在形式上是一个非负整数。

实际上。它是一个索引值，指向内核为每个进程所维护的该进程打开文件的记录表。当程序打开一个现有文件或者创建一个新文件时，内核向进程返回一个文件描写叙述符。

在程序设计中，一些涉及底层的程序编写往往会环绕着文件描写叙述符展开。

可是文件描写叙述符这一概念往往仅仅适用于UNIX、Linux这种操作系统。

文件描写叙述符的优点主要有两个：基于文件描写叙述符的I/O操作兼容POSIX标准；在UNIX、Linux的系统调用中。大量的系统调用都是依赖于文件描写叙述符。文件描写叙述符的概念存在两大缺点：在非UNIX/Linux操作系统上(如Windows NT)，无法基于这一概念进行编程。由于文件描写叙述符在形式上只是是个整数，当代码量增大时，会使编程者难以分清哪些整数意味着数据，那些意味着文件描写叙述符，因此，完毕的代码可读性也就会变得非常差，这一点一般通过消除魔术数字来解决。

对于ANSI C规范中定义的标准库的文件I/O操作。ANSI C规范给出了一个解决方法，就是使用FILE结构体的指针。

其实。UNIX/Linux平台上的FILE结构体的实现中往往都是封装了文件描写叙述符变量在当中。

2、输入输出重定向

2.1 stdin、stdout和stderr

简单地说，文件描写叙述符是与已打开文件或设备相关联的整数，它们保持和已打开文件或设备的关联。

最为常见的文件描写叙述符是stdin、stdout和stderr，它们各自是0、1、2，是系统保留的文件描写叙述符，分别相应标准输入、标准输出和标准错误。Unix系统会默认打开这三个文件描写叙述符，并将stdin关联到键盘。将stdout和stderr关联到屏幕。这里将的输入输出重定向主要就是将这三个文件描写叙述符又一次定向到其他我们希望的文件或者设备。对于stdin。stdout和stderr重定向一般採用的操作符主要有<、>和>>，在没有指定的详细文件操作符的情况下，缺省是这种：command < file.txt相当于command 0< file.txt，也就是说默认是将文件重定向到文件描写叙述符0；command > file.txt相当于command 1> file.txt也就是说默认将文件描写叙述符1重定向到文件。>>和>同样。假设要重定向stderr，就要显示指定，比方command 2> file.txt，将command的错误信息输出到file.txt。

2.2 重定向输入

为了避免每次手动输入数据，我们能够将数据写入一个文件，然后使用重定向将输入重定向到该文件。这里为了演示用的是linux的cat命令，cat命令假设不加參数，会读取标准输入并将其输出到屏幕（实际上。cat用‘-’做參数也是这个效果，cat -），效果例如以下：

< 2020年11月 >

日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

最新评论

1. Re:ASP.NET中HttpApplication中ProcessRequest方法中运行的事件顺序；ASP.NET WebForm和MVC总体请求流程图
六六六
--望天hous
2. Re:c++ 常成员函数 和 常对象
this应该是指针常量，不是常量指针
--Uknowwhat

阅读排行榜

1. SQL Server无法连接到(local)问题的解决的方法(2616)
2. 全球（全国）土地利用数据集大全(2614)
3. unity3D iTween的使用(2111)
4. ORACLE物化视图具体解释(1963)
5. TLB的作用及工作过程(1835)

```
[lfqy@localhost ~]$ cat
Hello, world!
Hello, world!
Ni hao!
Ni hao!
I'm unhappy.
I'm unhappy.//这里按下ctrl+D相当于EOF
[lfqy@localhost ~]$
```

以下将上面的内容写入文件test.txt。然后将cat的输入重定向到该文件，这里用到的是"<"，这个操作符：

```
[lfqy@localhost ~]$ cat <test.txt
Hello, world!
Ni hao!
I'm unhappy.
[lfqy@localhost ~]$
```

实际上，像上面说的cat 0<test.txt效果也一样。

当然，这里仅仅是用cat做一个样例。想用cat查看一个文件的内容何须这么麻烦。

2.2 重定向输出

2.2.1 重定向标准输入和标准错误

为了测试标准错误，首先新建了一个test文件夹，然后在里面创建了三个文件f1.txt、f2.txt和f.txt。

在f.txt中写入上面的几句话，然后将f1.txt和f2.txt的权限设置为000（chmod 000 f1.txt）。以下開始分别将标准输出和标准错误重定向。

```
1、没有重定向之前
[lfqy@localhost test]$ cat f*
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
Hello, world!
Ni hao!
I'm unhappy.
2、重定向标准输出
[lfqy@localhost test]$ cat f* > stdout.txt
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
[lfqy@localhost test]$ cat stdout.txt
Hello, world!
Ni hao!
I'm unhappy.
3、重定向标准输出和标准错误
[lfqy@localhost test]$ cat f* 2> stderr.txt
Hello, world!
Ni hao!
I'm unhappy.
[lfqy@localhost test]$ cat stderr.txt
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
4、同一时候分别将标准输出重定向到stdout.txt，将标准错误重定向到stderr.txt
[lfqy@localhost test]$ rm stderr.txt stdout.txt
[lfqy@localhost test]$ cat f* 2> stderr.txt >stdout.txt
[lfqy@localhost test]$ cat stderr.txt stdout.txt
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
Hello, world!
Ni hao!
I'm unhappy.
```

假设输出重定向的文件不存在，该文件会默认被创建。假设文件存在并有内容，当中原来的内容都会被清空。

当然，假设想要新的重定向的内容追加在原来内容的后面。可将上面的'>'换成'>>'，能够达到效果。

2.2.2 将标准输出和标准错误同一时候重定向到一个文件

这个实现起来有好几种办法。方法例如以下：

```
1、cat f* &> stdall.txt
[lfqy@localhost test]$ cat f* &> stdall.txt
[lfqy@localhost test]$ cat stdall.txt
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
Hello, world!
Ni hao!
I'm unhappy.
2、cat f* 1> stdall0.txt 2>&1
[lfqy@localhost test]$ cat f* 1> stdall0.txt 2>&1
[lfqy@localhost test]$ cat stdall0.txt
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
Hello, world!
Ni hao!
I'm unhappy.
反例：
[lfqy@localhost test]$ cat f* 2>&1 1> stdall0.txt
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
```

注意上面的反例。不能是cat f* 2>&1 1> stdall0.txt（由于假设先将标准错误重定向到标准输出，而这时标准输出没有被重定向。仍然是输出到屏幕，所以会导致将标准错误重定向到屏幕。），而必须是cat f* 1> stdall0.txt 2>&1（先将标准输出重定向到文件。这时候。全部到标准输出的内容都会被重定向到文件，因此后面重定向到标准输出的标准错误也会输出到文件。）。

上面1用的是一个略微特殊的操作符&>能够同一时候重定向标准输出和标准错误；2是将标准错误重定向到标准输出，然后重定向标准输出来达到目的。

二者的实现思路稍有不同。这里之所以要用cat f* 1> stdall0.txt 2>&1，之所以要在1前面加一个&，是由于1作为一个系统默认创建的文件描写叙述符。要用&来引用它。以下应该也会讲到。

2.2.3 新技能get

(1) 丢弃不想要的输出

有一个相似于垃圾桶的设备文件：/dev/null。将不想要的输出重定向到该文件就能够了。

```
1、丢弃标准输出
[lfqy@localhost test]$ cat f* > /dev/null
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
2、丢弃标准错误
[lfqy@localhost test]$ cat f* 2> /dev/null
Hello, world!
Ni hao!
I'm unhappy.
3、都丢弃（以下三种效果同样）
[lfqy@localhost test]$ cat f* 2> /dev/null 1>/dev/null
[lfqy@localhost test]$ cat f* &> /dev/null
[lfqy@localhost test]$ cat f* 1>/dev/null 2>&1
```

（2）既保存输出到文件，又让输出显示在屏幕

有时候。我们要将程序的输出存档，而又想看到程序的输出(上面的重定向没办法看到输出，仅仅能后来查看文件)。

这里。用到一个命令tee，它能够读取标准输入的内容，然后将其既输出到标准输出又写入到文件（tee - read from standard input and write to standard output and files）。这里要注意的是。这里须要

将命令的输出送到管道，然后由tee来接收管道的输入。而仅仅有标准输出的内容能够经过管道，所以先要将标准错误重定向到标准输出。

```
[lfqy@localhost test]$ cat f* 2>&1 | tee tee.txt
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
Hello, world!
Ni hao!
I'm unhappy.
[lfqy@localhost test]$ cat tee.txt
cat: f1.txt: Permission denied
cat: f2.txt: Permission denied
Hello, world!
Ni hao!
I'm unhappy.
```

这里会自己主动创建不存在的文件。并清空已存在文件原来的内容，假设想用追加模式，请使用cat f* 2>&1 | tee -a tee.txt。

3、创建自己定义的文件描写叙述符

0、1和2是系统保留的文件描写叙述符。我们也能够自己创建自己定义的文件描写叙述符。自己定义的文件描写叙述符主要有三种模式：仅仅读模式。截断模式和追加模式。

假定已经创建了文件描写叙述符num，能够使用&num来引用它，上文中用过&1就是这个道理。

3.1 仅仅读模式

使用"exec num<filename"创建仅仅读模式的文件描写叙述符；仅仅读方式的文件描写叙述符仅仅能读取一次。如需二次读取。须要对文件描写叙述符又一次创建。

```
[lfqy@localhost test]$ exec 5<f.txt
[lfqy@localhost test]$ cat <&5
Hello, world!
Ni hao!
I'm unhappy.
[lfqy@localhost test]$ cat <&5//二次读取的时候没有不论什么输出
```

3.2 截断模式

截断模式就是指在创建该文件描写叙述符的时候。相应文件里原来的内容将会被清空。使用"exec num>filename"的模式创建截断模式截断模式的文件描写叙述符创建之后，全部输出到该文件描写叙述符的内容都会依次追加在相应文件的后面，仅仅是该文件描写叙述符创建之前的内容被清空了。

```
[lfqy@localhost test]$ cat f.txt
Hello, world!
Ni hao!
I'm unhappy.
[lfqy@localhost test]$ exec 7>f.txt
[lfqy@localhost test]$ cat f.txt
[lfqy@localhost test]$ echo "Love" >&7
[lfqy@localhost test]$ echo "Happy" >&7
[lfqy@localhost test]$ cat f.txt
Love
Happy
```

3.3 追加模式

使用"exec num>>filename"创建追加模式的文件描写叙述符，和截断模式不同，在创建该文件描写叙述符的时候。相应文件里原来的内容不会被清空，全部输出到该文件描写叙述符的内容都会依次追加在相应文件的后面。

```
[lfqy@localhost test]$ cat f.txt
Love
Happy
[lfqy@localhost test]$ exec 8>>f.txt
[lfqy@localhost test]$ cat f.txt
```

```
Love
Happy
[lfqy@localhost test]$ echo "Friendship" >&7
[lfqy@localhost test]$ echo "Flair" >&7
[lfqy@localhost test]$ cat f.txt
Love
Happy
Friendship
Flair
```

好吧，关于文件描写叙述符今天就到这里了。后面学到新的内容会再补上。

mqxnongmin
关注 - 0
粉丝 - 3
+加关注

« 上一篇： 关于浏览器被http://www.51jetso.com/劫持
» 下一篇： 2014年百度之星资格赛第一题Energy Conversion

posted @ 2019-03-24 21:23 mqxnongmin 阅读(240) 评论(0) 编辑 收藏

相关博文：
· Posthttp://unix/api/shutdown:dialunix.gosuv.sock:connectionrefused
· Unix哲学
· UNIX编程GetAddrInfo笔记
· Unix套接字接口
· unix网络编程套接字
» 更多推荐...