linux学习园地03669

0Udhk5f

博客访问： 596242
博文数量： 729
博客积分： 6000
博客等级： 准将
技术积分： 5005
用 户 组： 普通用户
注册时间： 2008-09-18 11:52

Sobell 谈 bash 和 Linux 命令行命运

转载 分类： 2008-09-18 11:57:22

来自：yesky

【导读】《Linux 命令、编辑器和外壳编程实践指南》一书的作者 Mark G. Sobell 在接受 LinuxPlanet 栏目采访时谈了他本人对 Linux 命令行命运的看法。

LinuxPlanet(以下简称 LP):命令行死亡了吗?

答:不，根本就没有死亡。对于某些人和执行某些任务来说，使用图形界面更容易和更简洁。这实际上依赖于你要做什么和你是谁。图形用户界面和命令行之间的区别就像自动变档与变速杆一样。我使用变速杆是因为它能够让我更好地控制汽车，让我更多地感觉到汽车在做什么以及汽车是如何做到的。

当然，这个讨论假设你是以系统管理员的级别操作文件的。有些应用程序有用户图形界面，有些应用程序也许没有这种界面，或者只有非常原始的命令行界面。设法从命令行运行这些应用程序是没有意义的。

对命令行有好处的一件事情是它能够让你访问数百个工具软件。在命令行上面，你可以使用一个管道把工具软件结合在一起执行一项单个工具软件无法完成的工作。下面是从我的“Linux 命令、编辑器和外壳编程实践指南”一书中摘录的部分内容，谈了有关这些管道及其连接的过程:

“一个过程是 Linux 执行一个命令。过程之间的通信是 UNIX/Linux 的验证证明之一。一个通道(书写为垂直的直线“|”，在命令行中或者键盘上是一个垂直的实线)提供了这种通信最简单的方式。简单地说，一个通道接受一个工具软件的输出，然后把那个输出输入到其它工具软件。使用 UNIX/Linux 的词汇，这个通道接受了一个过程的标准输出，并把这个标准的输出作为另一个过程的标准输入。一个过程在屏幕上显示的大多数内容将发送给标准的输出。如果你没有重新定向这个输出，这个输出就在屏幕上显示出来。使用一个通道，你可以重新定向这个输出，这样它就变成了另一个工具软件的标准的输入。”

例如，你可以把列出目录中文件的命令“ls”与计算一个目录中的文件和字数的命令“wc -w”结合在一起使用:

```
$ ls | wc -w

45
```

在 Linux 系统管理领域，用户图形界面通常是建在命令行工具之外的，因此，你不能得到用户图形界面工具的好处。当然，除非你能使用鼠标。你在命令行下面能够完成的工作在图形用户界面系统管理工具中经常无法完成。

·Python中进行字符串拼接的常...

Bourne 和 Bourne 外壳程序

LP:你能讨论一下 bash(Bourne Again Shell)并且解释一下它与原来的 Bourne 外壳程序有什么区别吗?

答:这个外壳程序(shell)是命令行的解释程序，它分析你输入的命令行并且调用你申请的程序，并且把你在命令行中输入的参数传递给这个程序。这个外壳程序也是一种高级的编程语言。bash 是许多 Linux 系统默认的外壳程序。大多数 Linux 发布版软件还包含其它的外壳程序，甚至还有更多的外壳程序可供下载。

由 GNU 计划编写的 bash 包含了原始版本的 Bourne 外壳程序，那是 AT&T 公司发布的 UNIX 下面的第一个外壳程序。我曾经建议读者考虑使用 C 外壳程序作为他们的交互式外壳程序，因为它拥有原始版本的 Bourne 外壳程序所没有的一些重要功能。目前，bash 拥有所有这些功能，而且某些 bash 还包括命令完成、历史(这样你可以编辑和重复以前的命令)和工作控制(允许你在前端和后端之间转移工作)等功能。当然，你可以使用 bash 编写外壳程序脚本(批处理文件)。

许多 Linux 系统外壳程序脚本是从“#!/bin/sh”开始的。这一行命令让脚本在外壳程序下运行。这个外壳程序不是 Bourne 外壳程序的一部分，而是一个指向 bash 的链接。

由于具有长期的和成功的历史，原始的 Bourne 外壳程序一直用来编写许多帮助管理 Unix 系统的外壳脚本。其中有些在 Linux 中出现的脚本称作 bash 脚本。虽然 bash 脚本包含了许多原始的 Bourne 外壳程序中所没有的扩展功能和特性，但是，bash 保持了对原始的 Bourne 外壳程序的兼容，因此你可以在 bash 下面运行 Bourne 外壳脚本。原始的 Bourne 外壳程序在 Unix 系统中称作 sh。在 Linux 系统中，sh 是指向 bash 的一个符号链接，以确保需要 Bourne 外壳程序的脚本能够运行。当被称作 sh 的时候，bash 尽最大的努力效仿原始的 Bourne 外壳程序。

LP:你会建议 Linux 的新手学习 bash 还是学习 TC 外壳程序?

答:如果你是一个顽固的 C 语言外壳程序员，你可以继续使用 TC 外壳程序(tcsh)。否则，我建议你使用 bash。几乎所有的控制 Linux 的管理外壳脚本程序都是由 bash 运行的。因此，如果你学习 bash，你将能够很容易地理解和修改这些脚本。

awk

LP:你为什么使用 awk?

答:这是一个很好的问题，特别是在很多人直接使用 Perl 语言的时候。这个工具软件简单而功能强大。在 Perl 出现之前，awk 一直是操作文件的工具之一。目前，awk 仍是有用的。GNU 版本的 awk 称作 gawk，有一些新的功能，使其成为一个非常有用的工具。下面是我的书中讨论的有关如何让 gawk 与协作进程之间相互通信的部分内容:

协作进程:双向 I/O

协作进程是与另一个进程并行运行的一个进程。从 3.1 版本开始，gawk 能够启动一个协作进程直接与后台进程交换信息。当你在客户机/服务器环境中工作，设置一个SQL前端和后端或者在一个网络上与一个远程系统交换数据的时候，协作进程是很有用的。gawk 句法通过在启动后台进程的程序名称前面添加一个运算符号“|&”来识别一个协助进程。

一个协助进程指令必须是一个过滤器(也就是说，它读取标准的输入并且写入标准的输出)，必须在完成一行输出之后就进行刷新，而不是积累很多行很以后再进行输出。当一个指令作为协作进程被启动之后，它将通过一个双向的通道与一个 gawk 程序连接，这样，你就可以对这个协作进程进行读写操作。

当与 tr 工具一起使用时，这个工具在完成每一行指令之后不刷新其输出。这个“to_upper”外壳脚本是不刷新其输出的tr指令的外壳。这个过滤器可以作为协作进程运行。对于读取的每一行指

令，“to_upper”写入这些行，并且把这些行翻译成大写字母和标准的输出。如果你要 “to_upper”显示调试的输出，可删除“set -x”前面的“#”。

```
$ cat to_upper

#!/bin/bash

#set -x

while read arg

do

echo "$arg" | tr '[a-z]' '[A-Z]'

done

$ echo abcdef | to_upper

ABCDEF
```

g6 程序启动“to_upper”作为一个协作进程。这个 gawk 程序读取标准的输入或者在命令行中指定的一个文件，把这个输入翻译成大写字母，并把翻译的数据写入一个标准的输出。

```
$ cat g6

{

print $0 |& "to_upper"

"to_upper" |& getline hold

print hold

}

$ gawk -f g6 < alpha

AAAAAAAAA

BBBBBBBBB

CCCCCCCCC

DDDDDDDDD
```

这个g6程序在括号之内有一个混合的指令，包含三个指令。由于没有执行方式，gawk 对于每一行输入内容都执行一次这个混合的指令。

第一个指令“print $0”把当前的记录发送到标准的输出。“|&”运算符把标准的输出从新指向名为“to_upper”的程序。“to_upper”作为一个协作进程在运行。这些程序的外边需要一个括号。第二个指令把来自“to_upper”的标准输出重新指向一个“getline”指令。这个指令将其标准的输出复制到这个名为“hold”的变量。第三个指令“print hold”把“hold”变量的这个内容发送到标准的输出。

这个工具的名称是“tr”

LP:你能不能更多地谈一下“tr”工具?

答:哦，tr，好的。首先想到的事情是这是一个微不足道的问题的答案。命名一个 Linux 工具。这个工具仅接收来自标准输入的输入，从来不接收作为来自命令行变量的文件的输入。这个怪物只是有时候有用，但是，当它有用的时候，它是非常有用的。下面是摘录一些有关“tr”的内容:

tr 工具读取标准输入中每一个输入的字符，把字符镜像为一个替代的字符并删除原来的字符或者把不再管那个字符。这个工具读取标准的输入并且写入标准的输出。

tr 工具一般与两个参数一起使用，string1 (字符串1)和 string2 (字符串2)。每个字符在这两个字符串中的位置是非常重要的:每一次tr发现在其输入的string1中的一个字符的时候，它都要用 string2 中相对应的字符取代那个字符。

使用一个参数，string1 和“--delete”(删除指令)的选项，tr 删除在 string1 中指定的字符。这个“squeeze- repeats ”(缩减连续重复的字符)选项使用一个出现的字符取代在 string1 中连续出现的字符，例如 abbc 将变成 abc。

你可以使用一个连字符代表在 instring1 或者 string2 中的一系列字符。这两个命令行在下面的例子中产生同样的结果:

$ echo abcdef | tr 'abcdef' 'xyzabc'

xyzabc

$ echo abcdef | tr 'a-f' 'x-za-c'

xyzabc

下面这个例子演示了隐藏文本的流行的做法。这个方法通常称作“ROT13”(rotate 13)，因为它用第十三个字母代替第一个字母，用第十四个字母代替第二个字母，以此类推。

$ echo The punchline of the joke is ... |

> tr 'A-M N-Z a-m n-z' 'N-Z A-M n-z a-m'

Gur chapuyvar bs gur wbxr vf ...

为了使这个文本再次智能化，把给 tr 的参数顺序倒过来:

$ echo Gur chapuyvar bs gur wbxr vf ... |

> tr 'N-Z A-M n-z a-m' 'A-M N-Z a-m n-z'

The punchline of the joke is ...

这个“--delete”选项使 tr 删除选择的字符:

$ echo If you can read this, you can spot the missing vowels! |

> tr --delete 'aeiou'

If y cn rd ths, y cn spt th mssng vwls!

在下面的例子中，tr 替换几个字符并且产生与单个字符相同的一对儿字符:$ echo tennessee | tr --squeeze-repeats 'tnse' 'srne'

serene

下一个例子用一个“新文件”字符替换在 draft1 文件中的没有按照字母顺序排列的每一个字符。输出是一个词汇列表，每一行一个单词:

$ tr --complement --squeeze-repeats '[:alpha:]' 'n' < draft1

最后一个例子是使用字符类提升那里的字符串“hi there”:

$ echo hi there | tr '[:lower:]' '[:upper:]'

HI THERE

总结

LP:最后还有什么想法吗?

答:我想说命令行并不适用于每一个人。命令行适用于那些要求亲手操作和对他们驯服的野兽有更强的控制力的那些用户。学习外壳程序能够做什么已经成功了一半。成功的另一半是学习一些与 Linux 发布版一起推出的许多工具的知识。你不需要了解每一个指令的每一个参数。了解每一个指令名称的含义和每个指令基本上能做什么就足够了。你可以阅读我的书中有关人和信息的内容或者有关指令参考的部分。查看 tac 工具并且在一开始的时候会对这个工具名称的起源感到可笑。

----------next----------

上一篇：孙玉芳：红旗旗手倒在途中
下一篇：背景知识之Linux背后的人