Linux培训系列

————第二讲

将向您演示如何使用正则表达式在文件中搜索文本模式。接着，我们将向您介绍文件系统层次结构标准（Filesystem Hierarchy Standard，或者称为 FHS），并向您演示如何在您的系统上定位文件。然后，我们将通过在后台运行 Linux 进程、列出进程清单、从终端上拆离进程以及更多内容，向您演示如何完全控制 Linux 进程。最后，我们将向您简要介绍 shell 管道、重定向和文本处理命令。

红联Linux论坛是致力于Linux技术讨论的站点，目前网站收录的文章及教程基本能满足不同水平的朋友学习。

红联Linux门户： www.linux110.com
红联Linux论坛： www.linuxdiyf.com/bbs
下载:Linux电子书籍：
http://www.linux286.com/linux/linuxdzsj.htm

目录

正则表达式

正则表达式

发布时间 :2007-01-26 22:02:44

正则表达式（也称为“ regex ”或“ regexp ”）是一种用来描述文本模式的特殊语法。在 Linux 系统上，正则表达式通常被用来查找文本的模式，以及对文本流执行“ 搜索-替换 ”操作以及其它功能。

与 glob 的比较

发布时间 :2007-01-26 22:03:34

当我们看到正则表达式时，您可能发现正则表达式的语法看起来与我们上一篇教程，但是，不要让它欺骗您；它们的类似性只是表面的。虽然正则表达式和文件名匹配替换模式可能看上去相类似，但是它们是根本不同的两种类型。

简单子串

发布时间 :2007-01-26 22:04:08

记住那个警告，让我们看一下最基本的正则表达式，简单子串。为了这样做，我们要使用 grep，它是一个扫描文件内容来查找适合特定正则表达式的命令。grep 打印与正则表达式匹配的每一行，并忽略与之不匹配的每一行：

```
$ grep bash /etc/passwd
operator:x:11:0:operator:/root:/bin/bash
root:x:0:0::/root:/bin/bash
ftp:x:40:1::/home/ftp:/bin/bash
```

在上面的命令中，grep 的第一个参数是一个正则表达式；第二个参数是一个文件名。grep 读取 /etc/passwd 中的每一行并对它应用简单子串正则表达式 bash 来查找匹配项。如果找到一个匹配项，那么 grep 打印出整行；否则，忽略该行。

理解简单子串

发布时间 :2007-01-26 22:04:41

一般来说，如果您正在搜索一个子串，那么您可以不提供任何“ 特殊 ”字符，而只是逐字地指定文本。只有在子串包含 +、.、*、[、] 或 \（在这样的情况下，这些字符需要用引号括起来并在它们的前面使用反斜杠）才需要做特殊的事情。下面是简单子串正则表达式几个其它示例：

/tmp （扫描查找文字串 /tmp）
“ \[box\] ”（扫描查找文字串 [box]）
“ *funny* ”（扫描查找文字串 *funny*）
“ ld\.so ”（扫描查找文字串 ld.so）

元字符

发布时间 :2007-01-26 22:05:15

使用正则表达式，可以利用元字符来执行比我们至今已研究过的示例复杂得多的搜索。这些元字符中的一个是 .（点），它与任何单个字符匹配：

```
$ grep dev.hda /etc/fstab
/dev/hda3      /              reiserfs        noatime,ro 1 1
/dev/hda1      /boot          reiserfs        noauto,noatime,notail 1 2
/dev/hda2      swap           swap            sw 0 0
#/dev/hda4     /mnt/extra     reiserfs        noatime,rw 1 1
```

在本示例中，文字文本 dev.hda 没有出现在 /etc/fstab 中的任何一行中。但是，grep 扫描这些行时没有查找文字 dev.hda 字符串，而是查找 dev.hda 模式。请记住 . 将与任何单个字符相匹配。正如您看到的，. 元字符在功能上等价于 glob 扩展中 ? 元字符的工作原理。

使用 []

发布时间 :2007-01-26 22:05:52

如果我们希望与比 . 更具体一点地来匹配字符，那么我们可以使用 [和]（方括号）来指定要匹配的字符子集：

```
$ grep dev.hda[12] /etc/fstab
/dev/hda1       /boot           reiserfs        noauto,noatime,notail 1 2
/dev/hda2       swap            swap            sw 0 0
```

正如您看到的，这个特殊语法的作用与 “ glob ” 文件名扩展中的 [] 相同。同样，这是学习正则表达式的难点之一 — 这个语法与 “ glob ” 文件名扩展语法类似，但又不尽相同，它经常给学习正则表达式的人带来困惑。

使用 [^]

发布时间 :2007-01-26 22:06:21

通过使 [后面紧跟一个 ^，您可以使方括号中的意思相反。在本例中，方括号将与未列在方括号内的任意字符匹配。同样，请注意我们在正则表达式中使用 [^] ，而在 glob 中使用 [!] ：

```
$ grep dev.hda[^12] /etc/fstab
/dev/hda3       /               reiserfs        noatime,ro 1 1
#/dev/hda4      /mnt/extra      reiserfs        noatime,rw 1 1
```

区别语法

发布时间 :2007-01-26 22:06:54

注意下面一点很重要：方括号内部的语法根本不同于正则表达式其它部分中的语法。例如，如果在方括号内放置一个 . ，那么它允许方括号与文字 . 匹配，就象上面示例中的 1 和 2。比较起来，除非有 \ 作为前缀，否则方括号外面的文字 . 被解释为一个元字符。通过输入如下命令，我们可以利用这一事实来打印 /etc/fstab 中包含文字串 dev.hda 的所有行的列表：

$ grep dev[.]hda /etc/fstab

或者，我们也可以输入：

$ grep "dev\.hda" /etc/fstab

这两个正则表达式都不可能与您的 /etc/fstab 文件中的任何行相匹配。

“ * ”元字符

发布时间 :2007-01-26 22:07:25

某些元字符本身不匹配任何字符，但却修改前一个字符的含义。一个这样的元字符是 * （星号），它用来与前一个字符的零次或者多次重复出现相匹配。这里是一些示例：

ab*c（与 abbbbc 匹配但不与 abqc 匹配）
ab*c（与 abc 匹配但不与 abbqbbc 匹配）
ab*c（与 ac 匹配但不与 cba 匹配）
b[cq]*e（与 bqe 匹配但不与 eb 匹配）
b[cq]*e（与 bccqqe 匹配但不与 bccc 匹配）
b[cq]*e（与 bqqcce 匹配但不与 cqe 匹配）
b[cq]*e（与 bbbeee 匹配）
.*（与任何字符串匹配）
foo.*（与以 foo 开始的任何字符串相匹配）

ac 行与正则表达式 ab*c 相匹配，因为星号也允许前面的表达式（b）出现零次。请注意解释 * 正则表达式元字符所用的方法与解释 * glob 字符的方法根本不同。

行的开始和结束

发布时间 :2007-01-26 22:07:56

我们在这里要详细描述的最后几个元字符是 ^ 和 $ 元字符，它们用来分别与行的开始和结束相匹配。通过在正则表达式开始处使用一个 ^ ，您可以将您的模式“ 锚定 ”在行的开始。在下面的示例中，我们使用 ^# 正则表达式来与以 # 字符开始的任何行相匹配：

```
$ grep ^# /etc/fstab
# /etc/fstab: static file system information.
#
```

完整行正则表达式

发布时间 :2007-01-26 22:08:27

可以组合 ^ 和 $ 来与完整的行相匹配。例如，下面的正则表达式将与以 # 字符开始并以 . 字符结束的行相匹配，在其中间可以有任意多个其它字符：

```
$ grep '^#.*\.$' /etc/fstab
# /etc/fstab: static file system information.
```

在上面的示例中，我们用单引号将我们的正则表达式括起来以阻止 shell 解释 $ 。在不使用单引号的情况下，grep 甚至没有机会查看 $，$ 就从我们的正则表达式上消失了。

FHS 与查找文件

FHS 与查找文件－文件系统层次结构标准

发布时间 :2007-01-26 22:09:45

Filesystem Hierarchy Standard 是指定 Linux 系统上目录布局的文档。FHS 被设计来提供一个通用布局以简化与分布无关的软件开发。FHS 指定下列目录（直接来自 FHS 规范）：

/（根目录）
/boot（引导装入程序的静态文件）
/dev（设备文件）
/etc（主机特定的系统配置）
/lib（基本共享库和核心模块）
/mnt（临时挂装文件系统的挂装点）
/opt（附加的应用程序软件包）
/sbin（基本系统二进制文件）
/tmp（临时文件）
/usr（辅助层次结构）
/var（可变数据）

两个独立的 FHS 类别

发布时间 :2007-01-26 22:10:21

FHS 的布局规范基于存在两个独立的文件类别：可共享与不可共享以及可变与静态这一思想。可共享数据能在主机之间被共享；不可共享数据特定于给定主机（例如配置文件）。可变数据可以被修改；静态数据不可以被修改（除了在系统安装和维护阶段）。

下面的表格概述了四种可能的组合，并列出了与那些类别相符的目录示例。这个表还是直接取自 FHS 规范：

	可共享	不可共享
静态	/usr /opt	/etc /boot
可变	/var/mail /var/spool/news	/var/run /var/lock

/usr 中的辅助层次结构

发布时间 :2007-01-26 22:11:04

在 /usr 下，您会发现一个看上去与根文件系统非常相似的辅助层次结构。当机器打开并运行时，/usr 的存在并不重要，所以能在网络上共享它（“ 可共享 ”），或者从 CD-ROM 上挂装它（“ 静态 ”）。大多数 Linux 设置不利用 /usr 的共享，但是理解根目录中主层次结构和 /usr 中辅助层次结构之间的区别的用处是有价值的。

这就是我们要说的有关 Filesystem Hierarchy Standard 的所有内容。该文档本身非常具有可读性，所以您应该去看一下。我们承诺如果您读了它，那么您将对 Linux 文件系统理解得更多。

查找文件

发布时间 :2007-01-26 22:11:36

Linux 系统通常包含数十万个文件。可能您非常精明能干，从未丢失它们中的任何一个，但是更可能的是，您偶尔在查找一个文件时需要帮助。Linux 上有几个不同的工具用于查找文件。下面的演示将向您介绍它们，并帮助您选择适合您的工作的工具。

PATH

发布时间 :2007-01-26 22:12:14

当您在命令行上运行程序时，bash 实际上搜索目录列表来查找您所请求的程序。例如，当您输入 ls，bash 实质上不知道 ls 程序位于 /usr/bin。但是，bash 引用一个名为 PATH 的环境变量，它是一个用冒号分隔的目录列表。我们可以检查 PATH 的值：

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/X11R6/bin
```

给定了 PATH 的值（您的可以不同），bash 将首先检查 /usr/local/bin，然后是 /use/bin 以搜索 ls 程序。ls 最有可能被保存在 /usr/bin 内，所以 bash 在那里停止。

修改 PATH

发布时间 :2007-01-26 22:12:46

您可以通过在命令行上为 PATH 指派元素来扩充它：

```
$ PATH=$PATH:~/bin
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/X11R6/bin:/home/agriffis/bin
```

您也可以除去 PATH 上的元素，尽管这不是那么容易，因为您不能引用现有的 $PATH。最好的办法是简单输入您想要的新 PATH：

```
$ PATH=/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:~/bin
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/agriffis/bin
```

关于“ which ”的一切

发布时间 :2007-01-26 22:13:17

通过使用 which，您能查看 PATH 中是否有给定程序。例如，我们通过下面的命令发现 Linux 系统没有（普通的）sense：

```
$ which sense
which: no sense in (/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/X11R6/bin)
```

在本示例中，我们成功定位 ls：

```
$ which ls
/usr/bin/ls
```

“ which -a ”

发布时间 :2007-01-26 22:13:47

最后，您应该知道 -a 标志，它使 which 向您显示您的 PATH 中给定程序的所有实例：

```
$ which -a ls
/usr/bin/ls
/bin/ls
```

whereis

发布时间 :2007-01-26 22:14:22

如果您不只对程序位置感兴趣，而且想要找到更多信息，那么可以尝试 whereis 程序：

$ whereis ls
ls: /bin/ls /usr/bin/ls /usr/share/man/man1/ls.1.gz

这里我们看到 ls 出现在两个常见二进制位置 /bin 和 /usr/bin 中。另外，我们被告知手册页定位在 /usr/share/man 。如果您要输入 man ls，那么这就是您将看到的手册页。

whereis 程序还具有搜索源代码、指定备用搜索路径和搜索不寻常项的能力。

find

发布时间 :2007-01-26 22:14:51

find 命令是您工具箱中的另一个工具。使用 find，您不会受限于程序；通过使用多种搜索标准，您能搜索您想要的任何文件。例如，要搜索 /usr/share/doc 目录下名为 README 的文件：

```
$ find /usr/share/doc -name README
/usr/share/doc/ion-20010523/README
/usr/share/doc/bind-9.1.3-r6/dhcp-dynamic-dns-examples/README
/usr/share/doc/sane-1.0.5/README
```

find 和通配符

发布时间 :2007-01-26 22:15:23

您可以在 -name 的参数中使用“ glob ”通配符，前提条件是您用双引号引用了它们或用反斜杠进行了转义（这样它们就能被完整地传递到 find 而不是被 bash 扩展）。例如，我们可能想要搜索带有扩展名的 README 文件：

```
$ find /usr/share/doc -name README\*
/usr/share/doc/iproute2-2.4.7/README.gz
/usr/share/doc/iproute2-2.4.7/README.iproute2+tc.gz
/usr/share/doc/iproute2-2.4.7/README.decnet.gz
/usr/share/doc/iproute2-2.4.7/examples/diffserv/README.gz
/usr/share/doc/pilot-link-0.9.6-r2/README.gz
/usr/share/doc/gnome-pilot-conduits-0.8/README.gz
/usr/share/doc/gimp-1.2.2/README.i18n.gz
/usr/share/doc/gimp-1.2.2/README.win32.gz
/usr/share/doc/gimp-1.2.2/README.gz
/usr/share/doc/gimp-1.2.2/README.perl.gz
[578 additional lines snipped]
```

在 find 中忽略大小写

发布时间 :2007-01-26 22:15:54

当然，您可能想要在搜索中忽略大小写：

```
$ find /usr/share/doc -name '[Rr][Ee][Aa][Dd][Mm][Ee]*'
```

或者，更简单：

```
$ find /usr/share/doc -iname readme\*
```

正如您看到的，您能使用 -iname 来进行不区分大小写的搜索。

find 和正则表达式

发布时间 :2007-01-26 22:16:24

如果您熟悉正则表达式，那么使用 -regex 选项将把输出限制成匹配某一模式的文件名。与 -iname 选项类似，它有一个相应的 -iregex 选项，该选项忽略模式中的大小写。例如：

```
$ find /etc -iregex '.*xt.*'
/etc/X11/xkb/types/extra
/etc/X11/xkb/semantics/xtest
/etc/X11/xkb/compat/xtest
/etc/X11/app-defaults/XTerm
/etc/X11/app-defaults/XTerm-color
```

请注意：不象许多程序，find 要求指定的正则表达式与整个路径匹配，而不只是该路径的一部分。为此，指定前导和尾随的 .* 是必要的；只使用 xt 是不够的。

find 和类型

发布时间 :2007-01-26 22:16:55

-type 选项允许您查找某一类型的文件系统对象。可能的 -type 参数是 b （块设备）、c（字符设备）、d（目录）、p（命名管道）、f（常规文件）、l（符号链接）和 s（套接字）。例如，要在 /uer/bin 中搜索包含字符串 vim 的符号链接：

```
$ find /usr/bin -name '*vim*' -type l
/usr/bin/rvim
/usr/bin/vimdiff
/usr/bin/gvimdiff
```

find 和 mtime

发布时间 :2007-01-26 22:17:33

-mtime 选项允许您根据最近一次的修改时间来选择文件。mtime 的参数以 24 小时为单位，当输入时带加号（表示“之后”）或者减号（表示“之前”）时，它最有用。例如，考虑如下情形：

```
$ ls -l ?
-rw-------   1 root     root            0 Jan  7 18:00 a
-rw-------   1 root     root            0 Jan  6 18:00 b
-rw-------   1 root     root            0 Jan  5 18:00 c
-rw-------   1 root     root            0 Jan  4 18:00 d
$ date
Mon Jan  7 18:14:52 EST 2002
```

您可以搜索在过去的 24 小时之内创建的文件：

```
$ find . -name \? -mtime -1
./a
```

或者您可以搜索在当前 24 小时周期之前创建的文件：

```
$ find . -name \? -mtime +0
./b
./c
./d
```

-daystart 选项

发布时间 :2007-01-26 22:18:03

如果您另外指定了 -daystart 选项，那么时间周期以今天的开始时为开始，而不是 24 小时之前。例如，这是昨天和前天创建的一组文件：

```
$ find . -name \? -daystart -mtime +0 -mtime -3
./b
./c
$ ls -l b c
-rw-------  1 root  root  0 Jan  6 18:00 b
-rw-------  1 root  root  0 Jan  5 18:00 c
```

-size 选项

发布时间 :2007-01-26 22:18:34

-size 选项允许您根据文件的大小来查找它们。缺省情况下，-size 的参数是 512 个字节的块，但是添加后缀可以使操作更简便。可用的后缀是 b（512 字节的块）、c（字节）、k（千字节）和 w（2 字节的字）。另外，您可以在前放置加号（“ 大于 ”）或者减号（“ 小于 ”）。

例如，要在 /usr/bin 中查找小于 50 个字节的常规文件：

```
$ find /usr/bin -type f -size -50c
/usr/bin/krdb
/usr/bin/run-nautilus
/usr/bin/sgmlwhich
/usr/bin/muttbug
```

处理找到的文件

发布时间 :2007-01-26 22:19:06

您可能在想如何处理所有这些找到的文件！不用担心，通过使用 -exec 选项，find 具有对它找到的文件进行操作的能力。这个选项接受命令行作为它的参数来执行，它以 ; 中断，并用文件名来替换任何出现的 {} 。下面这个示例可以帮助您完全理解它：

```
$ find /usr/bin -type f -size -50c -exec ls -l '{}' ';'
-rwxr-xr-x    1 root     root          27 Oct 28 07:13 /usr/bin/krdb
-rwxr-xr-x    1 root     root          35 Nov 28 18:26 /usr/bin/run-nautilus
-rwxr-xr-x    1 root     root          25 Oct 21 17:51 /usr/bin/sgmlwhich
-rwxr-xr-x    1 root     root          26 Sep 26 08:00 /usr/bin/muttbug
```

正如您看到的，find 是一个功能非常强大的命令。在 UNIX 和 Linux 开发的几年中，它获得了发展。find 中还有许多其它有用的选项。您可以在 find 手册页中学习它们。

定位

发布时间 :2007-01-26 22:19:35

我们已经学习了 which、whereis 和 find。您可能已经注意到执行 find 要花一些时间，因为它需要读取它正在搜索的每个目录。事实表明 locate 命令可以通过依靠外部数据库来加速操作。

locate 命令与路径名的任何部分相匹配，而不只是文件本身。例如：

```
$ locate bin/ls
/var/ftp/bin/ls
/bin/ls
/sbin/lsmod
/sbin/lspci
/usr/bin/lsattr
/usr/bin/lspgpot
/usr/sbin/lsof
```

使用 updatedb

发布时间 :2007-01-26 22:20:11

大多数 Linux 系统包含一个周期性的进程来更新这个数据库。如果您的系统在运行上述命令时返回如下错误，那么您需要运行 updatedb 来生成搜索数据库：

```
$ locate bin/ls
locate: /var/spool/locate/locatedb: No such file or directory
$ su
Password:
# updatedb
```

运行 updatedb 命令可能要花很长时间。如果您硬盘的噪音很大，那么将听到许多吵闹声，因为这正在为整个文件系统建立索引。

slocate

发布时间 :2007-01-26 22:20:41

在 Linux 的许多分发版（distribution）中，locate 命令已经被 slocate 所替代。通常有一个至 “ locate ” 的符号链接，这样您不需要记住拥有的是哪一个。slocate 代表 “ 安全定位（secure locate） ” 。它将许可权信息存储在数据库中，这样普通用户不能以别的方式窥探他们不能读取的目录。slocate 的用法信息在本质上与 locate 的信息相同，尽管输出可能不同（取决于正在运行命令的用户）。

进程控制

进程控制－启动 xeyes

发布时间 :2007-01-26 22:21:26

为了学习进程控制，我们首先需要启动一个进程：

```
$ xeyes -center red
```

您将注意到弹出一个 xeyes 窗口，红色眼球跟随您的鼠标在屏幕上移动。您还可能注意到在终端上没有新的提示符。

停止进程

发布时间 :2007-01-26 22:22:01

为了恢复提示符，您可以输入 Control-C（通常写为 Ctrl-C 或 ^C）：

```
^C
$
```

您获得了一个新的 bash 提示符，但 xeyes 窗口消失了。事实上，整个进程已被杀死。如果不通过 Control-C 来杀死它，我们还可以使用 Control-Z 来使它只是停止：

```
$ xeyes -center red
^Z
[1]+  Stopped                 xeyes -center red
$
```

这次您获得了一个新的 bash 提示符，并且 xeyes 窗口依然保留着。但是，如果您试图稍微移动这个窗口，那么将注意到眼球在某一位置被冻结了。如果 xeyes 窗口被另一个窗口遮盖，然后又出现，那么您将看到它根本不会重绘眼睛。进程没有做任何操作。事实上，它是“ 被停止了 ”。

fg 和 bg

发布时间 :2007-01-26 22:22:39

为了使进程“ 重新活动 ”并再次运行，我们能用 bash 内置的 fg 使它在前台运行：

```
$ fg

xeyes和xeyes


^Z
[1]+  Stopped                 xeyes -center red
$
```

现在用 bash 内置的 bg 来继续在后台运行它：

```
$ bg
[1]+ xeyes -center red &
$
```

好极了！现在 xeyes 进程在后台运行，并且出现了新的正在工作的 bash 提示符。

使用“ & ”

发布时间 :2007-01-26 22:23:11

如果我们一开始想要在后台启动 xeyes（而不是使用 Control-Z 和 bg），那么我们只须在 xeyes 命令行的最后添加一个“ & ”：

```
$ xeyes -center blue &
[2] 16224
```

多个后台进程

发布时间 :2007-01-26 22:23:47

现在红色和蓝色 xeyes 都在后台运行。我们可以用 bash 内置的 jobs 列出这些作业：

```
$ jobs -l
[1]- 16217 Running                 xeyes -center red &
[2]+ 16224 Running                 xeyes -center blue &
```

左列中的号码是当作业被启动时，bash 指定给它们的作业号码。作业 2 有一个 +（加号），这表示它是 “ 当前作业 ”，这意味着输入 fg 将把它带到前台。您也可以通过指定作业号码将指定的作业带到前台，换句话说，fg 1 使红色 xeyes 成为前台任务。下一列是包含在列表中的进程标识或者是 pid，由于 jobs 的 -l 选项可得到该列表。最后，这两个作业当前都是 “ Running ”，并且它们的命令行都列在右边。

介绍信号

发布时间 :2007-01-26 22:24:22

为了杀死、停止或者继续运行进程，Linux 使用了一种称为 “ 信号 ” 的特殊通讯方式。通过将某一信号发送给进程，您可以使它中断、停止或执行其它操作。这就是当您输入 Control-C、Control-Z 或使用 bg 或 fg 内置命令时真正执行的操作 — 您正使用 bash 将一个特殊信号发送给进程。还可以通过使用 kill 命令并在命令行上指定 pid（进程标识）来发送这些信号：

```
$ kill -s SIGSTOP 16224
$ jobs -l
[1]- 16217 Running                 xeyes -center red &
[2]+ 16224 Stopped (signal)        xeyes -center blue
```

正如您看到的，kill 不一定 “ 杀死 ” 进程，尽管它能这样做。通过使用 “ -s ” 选项，kill 能将任何信号发送给进程。当分别将 SIGINT、SIGSTOP 或 SIGCONT 信号发送给进程时，Linux 就杀死、停止或继续运行这些进程。您还可以将其它信号发送给进程；这些信号中的一些可能是用与应用程序相关的方法来解释的。

SIGTERM 和 SIGINT

发布时间 :2007-01-26 22:24:52

如果您想杀死进程，那么有几种选择。缺省情况下，kill 发送 SIGTERM，它与 Control-C 著名的 SIGINT 不同，但是通常具有相同的结果：

```
$ kill 16217
$ jobs -l
[1]- 16217 Terminated              xeyes -center red
[2]+ 16224 Stopped (signal)        xeyes -center blue
```

功能强大的 kill

发布时间 :2007-01-26 22:25:28

进程可以自己选择或者由于被停止或由于某种原因“ 被阻塞 ”而忽略 SIGTERM 和 SIGINT。在这些情况下可能需要使用功能强大的 SIGKILL 信号。进程不能忽略 SIGKILL：

```
$ kill 16224
$ jobs -l
[2]+ 16224 Stopped (signal)        xeyes -center blue
$ kill -s SIGKILL
$ jobs -l
[2]+ 16224 Interrupt               xeyes -center blue
```

nohup

发布时间 :2007-01-26 22:26:00

您启动作业的终端被称为这个作业的控制终端。当您注销时，一些 shell（缺省情况下不是 bash）将向这些后台作业传送 SIGHUP 信号，从而导致这些进程退出。为了保护进程以免产生这种行为，当您启动进程时，请使用 nohup：

```
$ nohup make &
$ exit
```

使用 ps 来列出进程

发布时间 :2007-01-26 22:26:36

我们较早使用的 jobs 命令只列出了从您 bash 会话上启动的进程。为了查看您系统上所有的进程，请使用同时带有 a 和 x 选项的 ps：

```
$ ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:04 init [3]
    2 ?        SW     0:11 [keventd]
    3 ?        SWN    0:13 [ksoftirqd_CPU0]
    4 ?        SW     2:33 [kswapd]
    5 ?        SW     0:00 [bdflush]
```

我们只列出了开始的几个进程，因为通常它是一个非常长的列表。这提供给您整台机器正在执行的进程的一个快照，但您可能要筛选掉一些信息。如果您要省略 ax，那么将只看到您拥有的并具有控制终端的进程。命令 ps x 将显示您所有的进程，甚至那些没有控制终端的进程。如果您要使用 ps a，那么可以获取附加在终端上的每人的进程列表。

查看森林（层次结构）和树

发布时间 :2007-01-26 22:27:06

您也可以列出有关每个进程的不同信息的列表。使用 --forest 选项可以很容易地查看进程的层次结构，它将向您显示系统上的各种进程是如何相互关联的。当一个进程启动一个新进程时，那个新进程被称为“ 子 ”进程。在 --forest 列表中，父进程出现在左侧，而子进程作为分支出现在右侧：

```
$ ps x --forest
  PID TTY      STAT   TIME COMMAND
  927 pts/1    S      0:00 bash
 6690 pts/1    S      0:00  \_ bash
26909 pts/1    R      0:00      \_ ps x --forest
19930 pts/4    S      0:01 bash
25740 pts/4    S      0:04  \_ vi processes.txt
```

“ u ”和“ l ”ps 选项

发布时间 :2007-01-26 22:27:41

“ u ”或“ l ”选项也可以被添加到“ a ”和“ x ”的任何组合中以包含关于每个进程的更多信息：

```
$ ps au
USER       PID %CPU %MEM    VSZ  RSS TTY      STAT START    TIME COMMAND
agriffis   403  0.0  0.0  2484    72 tty1     S     2001    0:00 -bash
chouser    404  0.0  0.0  2508    92 tty2     S     2001    0:00 -bash
root       408  0.0  0.0  1308   248 tty6     S     2001    0:00 /sbin/agetty 3
agriffis   434  0.0  0.0  1008     4 tty1     S     2001    0:00 /bin/sh /usr/X
chouser    927  0.0  0.0  2540    96 pts/1    S     2001    0:00 bash
```

```
$ ps al
  F   UID   PID  PPID PRI  NI    VSZ  RSS WCHAN  STAT TTY        TIME COMMAND
100  1001   403     1   9   0   2484   72 wait4  S    tty1       0:00 -bash
100  1000   404     1   9   0   2508   92 wait4  S    tty2       0:00 -bash
000     0   408     1   9   0   1308  248 read_c S    tty6       0:00 /sbin/ag
000  1001   434   403   9   0   1008    4 wait4  S    tty1       0:00 /bin/sh
000  1000   927   652   9   0   2540   96 wait4  S    pts/1      0:00 bash
```

使用“ top ”

发布时间 :2007-01-26 22:28:12

如果您正在连续多次运行 ps，并尝试观察其中的变化，那么您可能想要用 top。top 显示了持续更新的进程列表，以及一些有用的摘要信息：

```
$ top
 10:02pm  up 19 days,  6:24,  8 users,  load average: 0.04, 0.05, 0.00
75 processes: 74 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  1.3% user,  2.5% system,  0.0% nice, 96.0% idle
Mem:   256020K av,  226580K used,   29440K free,       0K shrd,    3804K buff
Swap:  136544K av,   80256K used,   56288K free                  101760K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
  628 root      16   0  213M  31M  2304 S       0  1.9 12.5  91:43 X
26934 chouser   17   0  1272 1272  1076 R       0  1.1  0.4   0:00 top
  652 chouser   11   0 12016 8840  1604 S       0  0.5  3.4   3:52 gnome-termin
  641 chouser    9   0  2936 2808  1416 S       0  0.1  1.0   2:13 sawfish
```

nice

发布时间 :2007-01-26 22:28:52

每个进程都有一个优先级设置，Linux 用它来确定：该进程相对于与系统上其它进程的运行速度。通过使用 nice 命令来启动进程，您能设置进程的优先级：

$ nice -n 10 oggenc /tmp/song.wav

因为优先级设置称为 nice，所以很容易记住一个更大的值对于其它进程是非常友好的，从而允许它们获取对 CPU 的优先访问权。缺省情况下，用 0 设置来启动进程，所以上面的 10 设置意味着 oggenc 将欣然放弃对 CPU 的访问权，而把它交给其它进程。一般来说，这意味着 oggenc 将允许其它进程以它们正常的速度运行，而不管 oggenc 突然多么迫切地需要访问 CPU。您可以在上面的 ps 和 top 列表的 NI 列下看到这些 nice 值（niceness）的级别。

renice

发布时间 :2007-01-26 22:29:24

只有在您启动进程时，nice 命令才改变它的优先级。如果您想要更改正在运行的进程 nice 值设置，那么使用 renice：

```
$ ps l 641
  F   UID   PID  PPID PRI  NI    VSZ  RSS WCHAN  STAT TTY        TIME COMMAND
000  1000   641     1   9   0   5876 2808 do_sel S    ?          2:14 sawfish
$ renice 10 641
641: old priority 0, new priority 10
$ ps l 641
  F   UID   PID  PPID PRI  NI    VSZ  RSS WCHAN  STAT TTY        TIME COMMAND
000  1000   641     1   9  10   5876 2808 do_sel S    ?          2:14 sawfish
```

文本处理

文本处理－再述重定向

发布时间 :2007-01-26 22:30:38

可以使用“ > ”操作符将命令的输出重定向到一个文件，如下所示：

```
$ echo "firstfile" > copyme
```

除了将输出重定向到一个文件之外，我们也可以利用 shell 强大的名为管道的特性。通过使用管道，我们能将一个命令的输出传递给另一个命令的输入。考虑下面示例：

```
$ echo "hi there" | wc
      1       2       9
```

| 字符用来将左侧命令的输出连接到右侧命令的输入。在上面的示例中，echo 命令打印出后面跟有换行符的字符串 hi there。那个输出通常出现在终端上，但是管道将它重定向到 wc 命令，该命令显示它输入中的行、字和字符的数量。

管道示例

发布时间 :2007-01-26 22:31:12

这里是另一个简单示例：

$ ls -s | sort -n

在本例中，ls -s 通常打印终端上当前目录的列表，并在每个文件前面打印它的大小。但是我们已通过管道将输出传递给 sort -n，它根据文件大小对输出排序。这是在您的主目录中查找大型文件的一个实际有用的方法！

下列的示例更复杂，但是它们演示了通过可以使用管道实现的强大功能。我们将抛弃我们还未讨论的一些命令，但不要让它使您放慢脚步。请集中精力理解管道的工作原理，这样您才能将它们应用到日常 Linux 任务中。

解压缩管道

发布时间 :2007-01-26 22:31:47

通常为了解压缩并解包文件，您可以执行以下操作：

```
$ bzip2 -d linux-2.4.16.tar.bz2
$ tar xvf linux-2.4.16.tar
```

这个方法的缺点是需要在您的磁盘上创建一个中间的未压缩文件。由于 tar 具备从其输入上直接读的能力（而不是指定一个文件），所以我们可以使用管道来产生相同的最终结果：

```
$ bzip2 -dc linux-2.4.16.tar.bz2 | tar xvf -
```

哇！我们压缩的 tarball 已经被解压缩了，而且不需要中间文件。

更长的管道

发布时间 :2007-01-26 22:32:17

这是另一个管道示例：

$ cat myfile.txt | sort | uniq | wc -l

我们使用 cat 将 myfile.txt 的内容传递给 sort 命令。当 sort 命令接收到这个输入时，它对所有的输入行排序，以便它们按字母次序排列，然后它将输出发送到 uniq。uniq 除去任何重复行，将筛选后的输出发送到 wc -l。我们在前面就已经看到 wc 命令了，但它没有带命令行选项。当给定 -l 选项时，它只打印它输入中的行数，而不包括字和字符。用您喜爱的文本编辑器尝试创建两个测试文件，然后使用这个管道来查看您获得了什么样的结果。

马上开始文本处理

发布时间 :2007-01-26 22:32:51

现在我们着手快速查看标准 Linux 文本处理命令。因为我们正在本教程中讨论许多内容，所以没有足够的篇幅来提供每个命令的示例。因此，鼓励您阅读每个命令的手册页（例如，通过输入 man echo），并花一些时间使用每个命令来学习它们及其选项的工作原理。通常，这些命令将任何文本处理的结果打印到终端，而不是修改任何指定文件。

在快速查看了标准 Linux 文本处理命令之后，我们将详细讨论输出和输入重定向。那么，是的，隧道的尽头就是光明。:)

echo

echo 将它的参数打印到终端。如果您想要嵌入反斜杠转义序列，那么使用 -e 选项；例如 echo -e "foo\nfoo" 将打印 foo，然后打印一个换行，接着再打印 foo。使用 -n 选项告知 echo 省略缺省情况下附加到输出的最后一个换行。

cat、sort 和 uniq

发布时间 :2007-01-26 22:33:23

cat

cat 将指定为参数的文件内容打印到终端。作为管道的第一个命令，这是很方便的，例如，cat foo.txt | blah。

sort

sort 按字母次序打印在命令行上指定的文件内容。当然，sort 也接受用管道传送的输入。输入 man sort 来熟悉控制排序行为的各种选项。

uniq

uniq 获取已排序的文件或数据流（通过管道）并除去重复行。

wc、head 和 tail

发布时间 :2007-01-26 22:33:52

wc

wc 打印出指定文件或输入流（来自管道）中的行、字和字节的数量。输入 man wc 来学习如何精调显示的内容。

head

head 打印出文件或流的前十行。使用 -n 选项来指定应显示的行数。

tail

打印出文件或流的最后十行。使用 -n 选项来指定应显示的行数。

tac、expand 和 unexpand

发布时间 :2007-01-26 22:34:23

tac

tac 与 cat 类似，但它以逆向顺序打印所有行，换句话说，先打印最后一行。

expand

expand 将输入制表符转换为空格。使用 -t 选项来指定制表符停止位。

unexpand

unexpand 将输入空格转换为制表符。使用 -t 选项来指定制表符停止位。

cut、nl 和 pr

发布时间 :2007-01-26 22:34:54

cut

cut 从输入文件或流的每个行上抽取出由字符限定的字段。

nl

nl 将行号添加到输入的每个行上。这对于打印输出很有用。

pr

pr 将文件分解为多个页面的输出；通常用于打印。

tr、sed 和 awk

发布时间 :2007-01-26 22:35:33

tr

tr 是字符转换工具；它用来将输入流中的某些字符映射成输出流中的某些其它字符。

sed

sed 是一个功能强大的面向流的文本编辑器。

awk

awk 是一种方便的面向行的文本处理语言。

od、split 和 fmt

发布时间 :2007-01-26 22:36:09

od

od 将输入流转换为八进制或十六进制的“ 转储 ”格式。

split

split 将较大的文件拆分成许多较小、更易处理的块。

fmt

fmt 对段落重新格式化以便在其边缘处进行换行。这个能力被构建到大多数文本编辑器中，但是应知道它仍是一个好工具。

paste、join 和 tee

发布时间 :2007-01-26 22:36:41

paste

paste 获取两个或更多文件作为输入，连接输入文件上的每个后续行，并输出结果行。它对于创建文本的表或列是很有用的。

join

join 与 paste 类似，但它在每个输入行中使用一个字段（缺省情况下是第一个字段）来匹配一在单行上合并的字段。

tee

tee 将它的输入打印到文件和屏幕。当您想创建某些日志记录，但还想在屏幕上看时，这很有用。

快要结束了！重定向

发布时间 :2007-01-26 22:37:19

与在 bash 命令行上使用 > 类似，您也可以使用 < 来将文件重定向到一个命令。对于许多命令，您能简单地在命令行上指定文件名，但是，一些命令只对标准输入起作用。

Bash 和其它 shell 支持 “ 本地文件（herefile） ” 的概念。这允许您在命令调用后面几行中为命令指定输入，同时用一个标记值来使它终止。下面是一个示例：

```
$ sort <<END
apple
cranberry
banana
END
apple
banana
cranberry
```

在我们的示例中，我们输入字 apple、cranberry 和 banana，后跟 “ END ” 来表示输入的结束。然后 sort 程序以字母次序返回我们的字。

使用 >>

发布时间 :2007-01-26 22:37:53

您也许认为 >> 在某些方面与 << 相类似，但事实上不是。它只意味着将输出附加到文件上，而不是象 > 那样进行覆盖。例如：

```
$ echo Hi > myfile
$ echo there. > myfile
$ cat myfile
there.
```

哇！我们丢失了“ Hi ”部分！我们的本意是这样：

```
$ echo Hi > myfile
$ echo there. >> myfile
$ cat myfile
Hi
there.
```

这样要好得多！

Linux文章汇集

海量Linux技术文章

发布时间 :2006-11-24 16:50:29

下面是linux技术文章快速入口。需要联网：
Linux 技术交流

http://www.linuxdiyf.com/bbs/forum-3-1.html

Linux 应用

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=1

Linux 安装及学习指导

http://www.linuxdiyf.com/bbs/forum-45-1.html

Linux 系统安装

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=45&filter=type&typeid=11

Linux 学习指导

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=45&filter=type&typeid=12

Linux 软件安装

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=45&filter=type&typeid=13

shell

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=3

Linux 壁纸

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=4

红旗

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=5

Redhat

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=6

SuSE

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=7

Linux 认证

http://www.linuxdiyf.com/bbs/forumdisplay.php?fid=3&filter=type&typeid=9

Linux下载分享{酷件、书籍、视频分享 }

http://www.linuxdiyf.com/bbs/forum-6-1.html

服务器应用

http://www.linuxdiyf.com/bbs/forum-7-1.html

数据库应用

http://www.linuxdiyf.com/bbs/forum-8-1.html

Linux 编程与内核

http://www.linuxdiyf.com/bbs/forum-9-1.html

UniX 技术文章

http://www.linuxdiyf.com/bbs/forum-32-1.html

Linux 业界声音、新闻

http://www.linuxdiyf.com/bbs/forum-11-1.html

Linux 人才招聘信息

http://www.linuxdiyf.com/bbs/forum-46-1.html

网络转载，感谢原创作者！
制作：红联Linux论坛
祝您阅读愉快！