

Info 文档内容全面并采用超链接。Info 页面可以输出为多种格式。相比之下，man page 针对打印输出进行了优化。Info 格式比 man page 更灵活，允许对复杂命令和概念进行详尽的说明。与 man page 相似，Info 节点可以从命令行通过 **pinfo** 命令来读取。

典型的 man page 具有少量的内容，侧重于一个特定的主题、命令、工具或文件。Info 文档则是一种综合性文档。Info 提供以下改进：

- 针对大型系统的单个文档，包含该系统的所有必要信息
- 超链接
- 完整的可浏览文档索引
- 全文搜索整个文档

一些命令和实用程序同时拥有 man page 和 info 文档；通常而言，Info 文档的内容将更加深入。使用 **man** 和 **pinfo** 比较 tar 文档中的区别：

```
[user@host ~]$ man tar
[user@host ~]$ pinfo tar
```

pinfo 阅读器比原始的 **info** 命令更加高级。要浏览特定主题，请使用 **pinfo topic** 命令。不带参数运行 **pinfo** 命令将打开顶级目录。安装了对应的软件包后，**pinfo** 中会提供新的文档。



注意

如果系统中没有针对您所请求的特定条目的 Info 主题，Info 就会改为查找匹配的 man page 并显示相应内容。

比较 GNU INFO 和 MAN PAGE 导航

pinfo 命令和 **man** 命令采用了略有不同的导航击键操作。下表比较了这两个命令的导航击键操作：

pinfo 和 man，键绑定比较

导航	PINFO	MAN
向前（向下）滚动一个屏幕	PageDown 或 空格键	PageDown 或 空格键
向后（向上）滚动一个屏幕	PageUp 或 b	PageUp 或 b
显示主题目录	D	-
向前（向下）滚动半个屏幕	-	D
显示主题的父节点	U	-
显示主题的顶部（上部）	HOME	G
向后（向上）滚动半个屏幕	-	U
向前（向下）滚动到下一超链接	向下箭头键	-
打开光标处的主题	Enter 键	-

导航	PINFO	MAN
向前（向下）滚动一行	-	向下箭头键或 Enter
向后（向上）滚动到上一超链接	向上箭头键	-
向后（向上）滚动一行	-	向上箭头键
搜索某种模式	/string	/string
显示主题中的下一节点（章节）	N	-
重复之前的向前（向下）搜索	/，再按 Enter	n
显示主题中的上一节点（章节）	P	-
重复之前的向后（向上）搜索	-	ShiftN
退出程序	Q	Q



参考文献

pinfo info (Info: 简介)

pinfo pinfo (pinfo 文档)

GNU 项目

<http://www.gnu.org/gnu/thegnuproject.html>

pinfo(1) 和 **info(1)** man page

► 指导练习

阅读 INFO 文档

在本练习中，您将通过使用命令行工具浏览 GNU Info 文档来查找存储在这些文档中的信息。

成果

您应能够使用命令行工具浏览 GNU Info 文档。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。

在 workstation 上，运行 **lab help-info start** 命令。

```
[student@workstation ~]$ lab help-info start
```

- 1. 在 workstation 上，不带参数启动 **pinfo**。

```
[student@workstation ~]$ pinfo
```

- 2. 导航至 **Common options** 主题。

按向上箭头键或向下箭头键，直到突出显示 **(coreutils) Common options**。

```
Basics
* Bash: (bash).                               The GNU Bourne-Again SHell.
* Common options: (coreutils)Common options.
```

图 4.2: Bash 文档

- 3. 按 **Enter** 键查看这一主题。

```
File: coreutils.info,  Node: Common options,  Next: Output of entire files,  Prev: Introduction,  Up: Top
2 Common options
*****
Certain options are available in all of these programs. Rather than writing identical descriptions for each of the programs, they are described here. (In fact, every GNU program accepts (or should accept) these options.)
```

Normally options and operands can appear in any order, and programs act as if all the options appear before any operands. For example, 'sort -r passwd -t :' acts like 'sort -r -t : passwd', since ':' is an option-argument of '-t'. However, if the 'POSIXLY_CORRECT' environment variable is set, options must appear before operands, unless otherwise specified for a particular command.

图 4.3: 常见选项信息主题

- 4. 浏览这一个 Info 主题。了解长选项是否可以简写。

使用 **PageUp** 和 **PageDown** 键，浏览整个主题。是的，很多程序都允许简写长选项。

▶ 5. 确定 -- 符号用作命令参数时表示什么。

-- 符号表示命令选项的结尾和命令参数的开头，可以在 shell 的命令行解析器可能无法正确区分的复杂命令中。

▶ 6. 不退出 pinfo，上移到 **GNU Coreutils** 节点。

按 **u** 键，上移到主题的顶级节点。

▶ 7. 返回到顶级主题。

再次按 **u** 键。观察下述行为：位于主题节点的顶部时，上移可返回到节点目录。或者，在任何级别或主题中按 **d** 键可直接移到主题目录。

▶ 8. 搜索模式 **coreutils**，并选择该主题。

按 / 键，再输入搜索模式 “coreutils”。当主题突出显示时，按 **Enter** 键。

```
* Coreutils: (coreutils).          Core GNU (file, text, shell) utilities.
```

图 4.4: 搜索结果

▶ 9. 在顶部菜单中，通过按 **n** 键找到并选择 **Output of entire files**。浏览该主题。

使用 **Enter** 键来选择 **cat invocation**。使用方向键浏览该主题。

▶ 10. 向上移动两个级别，以返回到 **GNU Coreutils**。移到 **Summarizing files**。

按 **Enter** 键选择主题，再浏览该主题。

▶ 11. 按 **q** 退出 pinfo。

▶ 12. 再次从命令行使用 pinfo 命令，并将 **coreutils** 指定为目标主题。

```
[student@workstation ~]$ pinfo coreutils
```

▶ 13. 选择 **Disk usage** 主题。

按 **向下箭头** 键突出显示 **Disk usage**，然后按 **Enter** 键选择此主题。

▶ 14. 阅读 **df invocation** 和 **du invocation** 子主题。

使用方向键突出显示主题，使用 **PageUp** 和 **PageDown** 键浏览文本，然后按 **u** 键上移一个级别。完成时，按 **q** 键退出。

完成

在 workstation 上，运行 **lab help-info finish** 脚本来完成本练习。

```
[student@workstation ~]$ lab help-info finish
```

本引导式练习到此结束。

▶ 开放研究实验

在红帽企业 LINUX 中获取帮助

任务执行清单

在本实验中，您将查找信息以帮助自己完成 man page 和 GNU Info 文档中的任务。

成果

您应能够：

- 通过搜索 man page 和 Info 节点来查找相关命令。
- 了解常用文档命令的新选项。
- 使用适当的工具查看和打印文档及其他非文本格式文件。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。

在 workstation 上，运行 **lab help-review start** 命令。

```
[student@workstation ~]$ lab help-review start
```

1. 在 workstation 上，确定如何准备 man page 以进行打印。具体而言，查找用于打印的格式或渲染语言。
2. 为 **passwd** man page 创建格式化输出文件。将文件命名为 **passwd.ps**。确定文件内容格式。检查 **passwd.ps** 文件的内容。



注意

使用以下命令，创建 **passwd** man page 的格式化输出：

```
[student@workstation ~]$ man -t passwd > passwd.ps
```

> 符号将 man page 的内容重定向到 **passwd.ps** 文件。下一章中会详细讲解该命令。

3. 通过 **man**，了解用于查看或打印 PostScript 文件的命令。
4. 了解如何以预览模式使用 **evince(1)** 查看器。此外，确定如何在打开文档时显示特定的起始页面。
5. 使用您研究的各种 **evince** 选项查看您的 PostScript 文件。完成时关闭您的文档文件。
6. 利用 **man** 命令研究 **lp(1)**，以确定如何从特定的页面开始打印文档。不实际输入任何命令（因为没有打印机），了解通过一个命令行仅打印 PostScript 文件的第2和第3页的语法。
7. 使用 **pinfo**，查找有关 **evince** 查看器的 GNU Info 文档。

8. 使用 Firefox，打开系统的软件包文档目录，并浏览到 **man-db** 软件包子目录。查看提供的手册。
9. 使用 Firefox 浏览器，查找并浏览到 **initscripts** 软件包子目录。查看 **sysconfig.txt** 文件，它描述 **/etc/sysconfig** 目录中存储的重要系统配置选项。

评估

在 workstation 上，运行 **lab help-review grade** 以确认本练习是否成功。

```
[student@workstation ~]$ lab help-review grade
```

完成

在 workstation 上，运行 **lab help-review finish** 脚本来完成本练习。

```
[student@workstation ~]$ lab help-review finish
```

本实验到此结束。

► 解决方案

在红帽企业 LINUX 中获取帮助

任务执行清单

在本实验中，您将查找信息以帮助自己完成 man page 和 GNU Info 文档中的任务。

成果

您应能够：

- 通过搜索 man page 和 Info 节点来查找相关命令。
- 了解常用文档命令的新选项。
- 使用适当的工具查看和打印文档及其他非文本格式文件。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。

在 workstation 上，运行 **lab help-review start** 命令。

```
[student@workstation ~]$ lab help-review start
```

1. 在 workstation 上，确定如何准备 man page 以进行打印。具体而言，查找用于打印的格式或渲染语言。

- 1.1. 使用 **man man** 命令，确定如何准备 man page 以进行打印。

```
[student@worksation ~]$ man man  
...output omitted...
```

按 **q** 键退出 man page。



注意

man 通过 **-t** 使用 PostScript 准备 man page 以进行打印。

2. 为 **passwd** man page 创建格式化输出文件。将文件命名为 **passwd.ps**。确定文件内容格式。检查 **passwd.ps** 文件的内容。

注意

使用以下命令，创建 **passwd** man page 的格式化输出：

```
[student@workstation ~]$ man -t passwd > passwd.ps
```

> 符号将 man page 的内容重定向到 **passwd.ps** 文件。下一章中会详细讲解该命令。

2.1. 使用 **man -t** 命令，创建 **passwd** man page 的格式化文件。

```
[student@workstation ~]$ man -t passwd > passwd.ps  
[student@workstation ~]$ ls -al  
...output omitted...  
-rw-rw-r--. 1 student student 19947 Feb 26 11:14 passwd.ps  
...output omitted...
```

2.2. 使用 **file** 命令确定文件内容格式。

```
[student@workstation ~]$ file /home/student/passwd.ps  
passwd.ps: PostScript document text conforming DSC level 3.0
```

2.3. 使用 **less** 命令查看 **/home/student/passwd.ps** 文件。

```
[student@workstation ~]$ less /home/student/passwd.ps  
%!PS-Adobe-3.0  
%%Creator: groff version 1.22.3  
%%CreationDate: Tue Feb 26 11:14:40 2019  
%%DocumentNeededResources: font Times-Roman  
%%+ font Times-Bold  
%%+ font Times-Italic  
%%+ font Symbol  
%%DocumentSuppliedResources: procset grops 1.22 3  
...output omitted...
```

注意

file 的输出声称该文件采用 PostScript 格式，您已通过查看其内容进行了确认。注意 PostScript 信息中的标题行。使用 **q** 键退出 **less** 命令。

3. 通过 **man**，了解用于查看或打印 PostScript 文件的命令。

3.1. 通过 **man**，了解用于查看或打印 PostScript 文件的命令。

```
[student@workstation ~]# man -k postscript viewer  
evince (1) - GNOME document viewer  
evince-previewer (1) - show a printing preview of PostScript and PDF documents  
evince-thumbnailer (1) - create png thumbnails from PostScript and PDF documents  
gcm-viewer (1) - GNOME Color Manager Profile Viewer Tool  
gnome-logs (1) - log viewer for the systemd journal
```

```
grops (1)           - PostScript driver for groff  
pango-view (1)      - Pango text viewer  
pluginviewer (8)     - list loadable SASL plugins and their properties
```



注意

通过多个词语和 **-k** 选项查找与任一词语匹配的 man page；查找描述中包含“postscript”或“viewer”的 man page。注意输出中的 **evince(1)** 命令。

4. 了解如何以预览模式使用 **evince(1)** 查看器。此外，确定如何在打开文档时显示特定的起始页面。

- 4.1. 通过 **man evince** 命令，了解如何以预览模式使用查看器。

```
[student@workstation ~]$ man evince  
...output omitted...
```

按 **q** 键退出 man page。



注意

-w（或 **--preview**）选项可在预览模式中打开 **evince**。**-i** 选项用于指定起始页面。

5. 使用您研究的各种 **evince** 选项查看您的 PostScript 文件。完成时关闭您的文档文件。

- 5.1. 使用 **evince** 命令打开 **/home/student/passwd.ps**

```
[student@workstation ~]$ evince /home/student/passwd.ps
```

- 5.2. 使用 **evince -w /home/student/passwd.ps** 命令，以预览模式打开文件。

```
[student@workstation ~]$ evince -w /home/student/passwd.ps
```

- 5.3. 使用 **evince -i 3 /home/student/passwd.ps** 命令，打开文件并显示第 3 页。

```
[student@workstation ~]$ evince -i 3 /home/student/passwd.ps
```



注意

evince 普通模式可以全屏显示演示风格内容，**evince** 预览模式则可用于快速浏览和打印。请注意顶部的打印图标。

6. 利用 **man** 命令研究 **lp(1)**，以确定如何从特定的页面开始打印文档。不实际输入任何命令（因为没有打印机），了解通过一个命令行仅打印 PostScript 文件的第 2 和第 4 页的方法。

- 6.1. 使用 **man lp** 命令确定如何打印文档的特定页面。

```
[student@workstation ~]$ man lp  
...output omitted...
```

按 **q** 键退出 man page。



注意

从 **lp(1)**，您了解到 **-P** 选项用于指定页面。**lp** 命令假脱机至默认打印机，仅发送第 2 到第 3 页这一范围。因此，一个有效答案为 **lp passwd.ps -P 2-3**。

7. 使用 **pinfo**，查找有关 **evince** 查看器的 GNU Info 文档。

7.1. 使用 **pinfo command** 来查找有关 **evince** 查看器的 GNU Info 文档。

```
[student@workstation ~]$ pinfo evince
```



注意

请注意，显示的应该是 **evince(1)** man page。当请求的主题没有相应的 GNU 文档节点时，**pinfo** 文档查看器将查找相关的 man page。按 **q** 退出。

8. 使用 Firefox，打开系统的软件包文档目录，并浏览到 **man-db** 软件包子目录。查看提供的手册。

8.1. 使用 **firefox /usr/share/doc** 来查看系统文档。浏览到 **man-db** 子目录。单击手册来查看它们。

```
[student@workstation ~]$ firefox /usr/share/doc
```



注意

可以为任何常用目录创建书签。浏览到 **man-db** 目录后，单击以打开并查看文本版本的手册，然后将它关闭。单击以打开 PostScript 版本。正如之前观察的，**evince** 是系统用于 PostScript 和 PDF 文档的默认查看器。您可能希望以后返回到这些文档，学习更多与 **man** 相关的知识。完成时，关闭 **evince** 查看器。

Index of file:///usr/share/doc/man-db/			
Up to higher level directory			
Name	Size	Last Modified	
File: ChangeLog	51 KB	12/12/16	1:44:30 PM GMT+1
File: NEWS	60 KB	11/7/18	4:46:16 PM GMT+1
File: README	12 KB	12/11/16	12:44:45 AM GMT+1
File: man-db-manual.ps	129 KB	11/7/18	4:47:06 PM GMT+1
File: man-db-manual.txt	70 KB	11/7/18	4:47:01 PM GMT+1

9. 使用 Firefox 浏览器，查找并浏览到 **initscripts** 软件包子目录。查看 **sysconfig.txt** 文件，它描述 **/etc/sysconfig** 目录中存储的重要系统配置选项。

9.1. 在 Firefox 浏览器中，查找 **initscripts** 软件包子目录。

注意浏览器对查找和查看本地系统文档的用处。完成时，关闭文档和 Firefox。

评估

在 workstation 上，运行 **lab help-review grade** 以确认本练习是否成功。

```
[student@workstation ~]$ lab help-review grade
```

完成

在 workstation 上，运行 **lab help-review finish** 脚本来完成本练习。

```
[student@workstation ~]$ lab help-review finish
```

本实验到此结束。

总结

在本章中，您学到了：

- man page 可使用 **man** 命令进行查看并提供有关 Linux 系统组件（如文件、命令和功能）的信息。
- 按照惯例，当引用 man page 时，页面名称的后面跟着放在括号内的章节号。
- Info 文档可使用 **pinfo** 命令进行查看，它们由一个超文本节点集合组成，可提供有关整个软件包的信息。
- **man** 和 **pinfo** 采用的导航击键操作略有不同。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

章 5

创建、查看和编辑文本文件

目标

通过命令行输出或在编辑器中创建、查看和编辑文本文件。

培训目标

- 通过 shell 重定向将命令输出或错误保存到文件中，并利用管道通过多个程序处理命令输出。
- 使用 **vim** 编辑器创建和编辑文本文件。
- 使用 shell 变量来帮助运行命令，并编辑 Bash 启动脚本以设置 shell 和环境变量，从而修改 shell 以及从 shell 运行的程序的行为。

章节

- 将输出重定向到文件或程序（及测验）
- 从 Shell 提示符编辑文本文件（及引导式练习）
- 更改 Shell 环境（及引导式练习）

实验

创建、查看和编辑文本文件

将输出重定向到文件或程序

培训目标

学完本节后，您应能够通过 shell 重定向将输出或错误保存到文件中，并利用管道通过多个程序处理命令输出。

标准输入、标准输出和标准错误

一个运行的程序（或称为进程）需要从某位置读取输入并将输出写入某位置。从 shell 提示符运行的命令通常会从键盘读取其输入，并将输出发送到其终端窗口。

进程使用称为文件描述符的编号通道来获取输入并发送输出。所有进程在开始时至少要有三个文件描述符。标准输入（通道 0）从键盘读取输入。标准输出（通道 1）将正常输出发送到终端。标准错误（通道 2）将错误消息发送到终端。如果程序打开连接至其他文件的单独连接，则可能要使用更大编号的文件描述符。

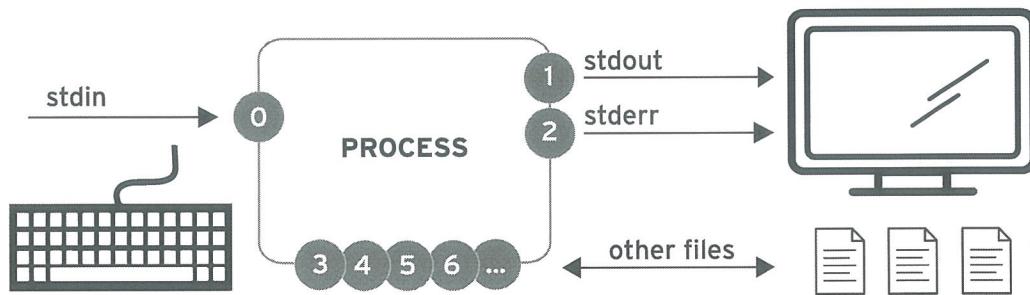


图 5.1: 进程 I/O 通道（文件描述符）

通道（文件描述符）

编号	通道名称	描述	默认连接	用法
0	stdin	标准输入	键盘	仅读取
1	stdout	标准输出	终端	仅写入
2	stderr	标准错误	终端	仅写入
3+	filename	其他文件	无	读取和/或写入

将输出重定向到文件

I/O 重定向可更改进程获取其输入或输出的方式。该进程并不是从键盘获取输入，也不是将输出和错误发送到终端，而是执行文件读取或写入。重定向允许您将消息保存到通常发送给终端窗口的文件中。或者，您也可以使用重定向来丢弃输出或错误，这样它们就不会显示在终端上或保存下来。

重定向 **stdout** 可以阻止进程输出显示在终端上。如下表所示，仅重定向 **stdout** 不会阻止 **stderr** 错误消息显示在终端上。如果文件不存在，则会创建文件。如果文件确实存在，但是所需的重定向没有附加到文件，则该文件的内容将被覆盖。

如果想丢弃消息，特殊文件 `/dev/null` 以静默方式丢弃重定向到其自身的通道输出，并且始终是空文件。

输出重定向操作符

用法	说明	视觉辅助
<code>> file</code>	重定向 <code>stdout</code> 以覆盖文件	
<code>>> file</code>	重定向 <code>stdout</code> 以附加到文件	
<code>2> file</code>	重定向 <code>stderr</code> 以覆盖文件	
<code>2> /dev/null</code>	将 <code>stderr</code> 错误消息重定向到 <code>/dev/null</code> ，从而将它丢弃	
<code>> file 2>&1</code>	重定向 <code>stdout</code> 和 <code>stderr</code> 以覆盖同一个文件	
<code>&> file</code>		
<code>>> file 2>&1</code>	重定向 <code>stdout</code> 和 <code>stderr</code> 以附加到同一个文件	
<code>&>> file</code>		

**重要**

重定向操作的顺序非常重要。以下序列将标准输出重定向到 **file**，然后将标准错误作为标准输出重定向到相同位置 (**file**)。

```
> file 2>&1
```

但是，下一个序列以相反的顺序执行重定向。这会将标准错误重定向到标准输出的默认位置（终端窗口，因此没有任何更改），然后仅将标准输出重定向到 **file**。

```
2>&1 > file
```

因此，有些人更倾向于使用合并重定向运算符：

&>file	而不是	>file 2>&1
----------------------	-----	-----------------------------

&>>file	而不是	>>file 2>&1 (在 Bash 4 / RHEL 6 和更高版本中)
--------------------------	-----	--

但是，某些系统管理员和程序员也会使用与 **bash** 相关的其他 shell（称为“兼容 Bourne 的 shell”）对命令进行脚本编制，这些人认为应避免使用较新的合并重定向运算符，因为这些运算符并没有标准化，也没有在所有这些 shell 中实施，而且还存在其他限制因素。

本课程的作者对该主题采取中立立场，因此您可能会在课程中遇到这两种语法。

输出重定向示例

通过重定向，可以简化许多日常管理任务。在思考下列示例时，请参考前面的表格：

- 保存时间戳以供日后参考。

```
[user@host ~]$ date > /tmp/saved-timestamp
```

- 将一个日志文件的最后 100 行复制到另一文件。

```
[user@host ~]$ tail -n 100 /var/log/dmesg > /tmp/last-100-boot-messages
```

- 将四个文件连接为一个。

```
[user@host ~]$ cat file1 file2 file3 file4 > /tmp/all-four-in-one
```

- 将主目录的隐藏文件名和常规文件名列到文件中。

```
[user@host ~]$ ls -a > /tmp/my-file-names
```

- 将输出附加到现有文件。

```
[user@host ~]$ echo "new line of information" >> /tmp/many-lines-of-information
[user@host ~]$ diff previous-file current-file >> /tmp/tracking-changes-made
```

- 接下来的几个命令会生成错误消息，因为普通用户无法访问某些系统目录。观察错误消息如何被重定向。在终端上查看普通命令输出时，将错误重定向到文件。

```
[user@host ~]$ find /etc -name passwd 2> /tmp/errors
```

- 将进程输出和错误消息保存到单独的文件中。

```
[user@host ~]$ find /etc -name passwd > /tmp/output 2> /tmp/errors
```

- 忽略并丢弃错误消息。

```
[user@host ~]$ find /etc -name passwd > /tmp/output 2> /dev/null
```

- 将输出和生成的错误消息存储在一起。

```
[user@host ~]$ find /etc -name passwd &> /tmp/save-both
```

- 将输出和生成的错误附加到现有文件。

```
[user@host ~]$ find /etc -name passwd >> /tmp/save-both 2>&1
```

构建管道

管道是一个或多个命令的序列，用竖线字符 (|) 分隔。管道将第一个命令的标准输出连接到下一个命令的标准输入。

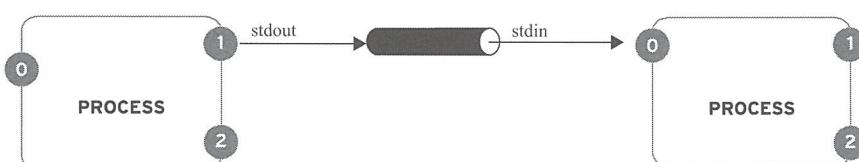


图 5.8: 进程 I/O 传送

在某个进程输出到终端之前，管道允许另一个进程操作和格式化该进程的输出。为便于理解，可以这样想象一下：数据正在通过管道从一个进程“流”向另一个进程，并且在其流过的管道中每个命令都会略微对其做些改动。



注意

管道和 I/O 重定向都可以操作标准输出和标准输入。重定向会向文件发送标准输出或从文件获取标准输入。管道会将一个进程的标准输出发送到另一个进程的标准输入。

管道示例

本示例取 **ls** 命令的输出并使用 **less** 在终端上以一次一屏的方式显示输出。

```
[user@host ~]$ ls -l /usr/bin | less
```

ls 命令的输出传送到 **wc -l**，用于统计从 **ls** 收到的行数并将该行数显示在终端。

```
[user@host ~]$ ls | wc -l
```

在此管道中，**head** 将输出 **ls -t** 输出内容的前 10 行，并且最终结果会重定向到一个文件。

```
[user@host ~]$ ls -t | head -n 10 > /tmp/ten-last-changed-files
```

管道、重定向和 tee 命令

当重定向与管道组合时，shell 会首先设置整个管道，然后重定向输入/输出。如果在管道的中间使用了输出重定向，则输出将转至文件，而不是前往管道中的下一个命令。

在本示例中，**ls** 命令的输出将转至文件，并且 **less** 不会在终端上显示任何内容。

```
[user@host ~]$ ls > /tmp/saved-output | less
```

tee 命令克服了这个限制。在管道中，**tee** 将其标准输入复制到其标准输出中，并且还将标准输出重定向到指定为命令参数的文件。如果您将数据想象成流经管道的水，那么可将 **tee** 视觉化为管道中的 T 形接头，它负责输出在两个方向上的流向。

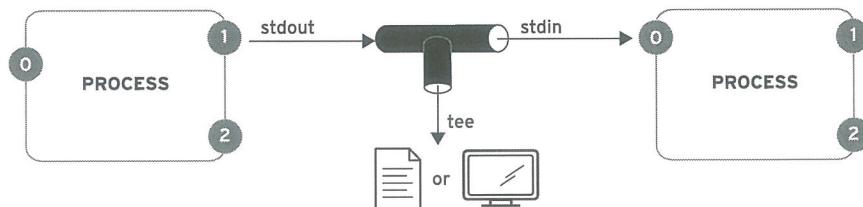


图 5.9: 利用 tee 处理 I/O 传送

使用 tee 命令的管道示例

本示例将 **ls** 命令的输出重定向到文件，并且将输出传递到 **less** 以便在终端上以一次一屏的方式显示。

```
[user@host ~]$ ls -l | tee /tmp/saved-output | less
```

如果在管道末尾使用了 **tee**，则可以保存命令的最终输出并且同时输出到终端。

```
[user@host ~]$ ls -t | head -n 10 | tee /tmp/ten-last-changed-files
```



重要

可通过管道来重定向标准错误，但是不能使用合并重定向运算符（&> 和 &>>）执行此操作。

以下是通过管道重定向标准输出和标准错误的正确方法：

```
[user@host ~]$ find -name / passwd 2>&1 | less
```



参考文献

info bash (GNU Bash 参考手册)

- 第 3.2.2 节：管道
- 第 3.6 节：重定向

info coreutils 'tee invocation' (GNU coreutils 手册)

- 第 17.1 节：重定向输出到多个文件或进程

bash(1)、**cat(1)**、**head(1)**、**less(1)**、**mail(1)**、**tee(1)**、**tty(1)**、**wc(1)** 手册

► 小测验

将输出重定向到文件或程序

选择以下问题的正确答案：

► 1. 哪一个答案向终端显示输出并忽略所有错误？

- a. &>file
- b. 2> &>file
- c. 2>/dev/null
- d. 1>/dev/null

► 2. 哪一个答案将输出发送到文件并将错误发送到另一文件？

- a. >file 2>file2
- b. >file 1>file2
- c. >file &2>file2
- d. | tee file

► 3. 哪一个答案将输出和错误都发送到文件，创建文件还是覆盖其内容？

- a. | tee file
- b. 2 &>file
- c. 1 &>file
- d. &>file

► 4. 哪一个答案将输出和错误发送到同一文件，以确保保留现有文件内容？

- a. >file 2>file2
- b. &>file
- c. >>file 2>&1
- d. >>file 1>&1

► 5. 哪一个答案会丢弃通常发送到终端的所有消息？

- a. >file 2>file2
- b. &>/dev/null
- c. &>/dev/null 2>file
- d. &>file

► 6. 哪一个答案将输出同时发送到屏幕和文件？

- a. &>/dev/null
- b. >file 2>file2
- c. | tee file
- d. | < file

► 7. 哪一个答案将输出保存到文件并丢弃错误消息？

- a. &>file
- b. | tee file 2> /dev/null
- c. > file 1> /dev/null
- d. > file 2> /dev/null

► 解决方案

将输出重定向到文件或程序

选择以下问题的正确答案：

► 1. 哪一个答案向终端显示输出并忽略所有错误？

- a. &>file
- b. 2>>file
- c. 2>/dev/null
- d. 1>/dev/null

► 2. 哪一个答案将输出发送到文件并将错误发送到另一文件？

- a. >file 2>file2
- b. >file 1>file2
- c. >file &2>file2
- d. | tee file

► 3. 哪一个答案将输出和错误都发送到文件，创建文件还是覆盖其内容？

- a. | tee file
- b. 2 &>file
- c. 1 &>file
- d. &>file

► 4. 哪一个答案将输出和错误发送到同一文件，以确保保留现有文件内容？

- a. >file 2>file2
- b. &>file
- c. >>file 2>&1
- d. >>file 1>&1

► 5. 哪一个答案会丢弃通常发送到终端的所有消息？

- a. >file 2>file2
- b. &>/dev/null
- c. &>/dev/null 2>file
- d. &>file

► 6. 哪一个答案将输出同时发送到屏幕和文件？

- a. &>/dev/null
- b. >file 2>file2
- c. | tee file
- d. | <file

► 7. 哪一个答案将输出保存到文件并丢弃错误消息？

- a. &>file
- b. | tee file 2>/dev/null
- c. > file 1>/dev/null
- d. > file 2>/dev/null

从 SHELL 提示符编辑文本文件

培训目标

学完本节后，您应能够使用 **vim** 编辑器从命令行创建和编辑文本文件。

使用 VIM 编辑文件

Linux 的一个重要设计原则是信息和配置设置通常都存储在基于文本的文件中。这些文件可以采用各种结构方式，如设置列表、类似于 INI 的格式、结构化 XML 或 YAML 等等。但是，文本文件的优点在于能用任何简单的文本编辑器进行查看和编辑。

Vim 是随 Linux 和 UNIX 系统分发的 **vi** 编辑器的改进版本。Vim 对于有经验的用户而言具有很高的可配置性和效率，其包含分屏编辑、颜色格式和突出显示编辑文本等功能。

为何要掌握 Vim?

您应该了解如何使用至少一种能从纯文本 shell 提示符使用的文本编辑器。在这种情况下，您就可以从终端窗口，或是通过从 **ssh** 或 Web 控制台远程登录的方式来编辑基于文本的配置文件。然后，您无需访问图形桌面即可编辑服务器上的文件，实际上，该服务器可能根本不需要运行图形桌面环境。

但是，为何要掌握 Vim 而不是其他可能的选择呢？主要原因在于，只要存在文本编辑器，几乎总会在服务器上安装 Vim。这是因为 **vi** 是由 POSIX 标准指定的，而 Linux 及其他很多类似 UNIX 的操作系统大体上都遵循此标准。

另外，Vim 经常被用作 **vi** 在其他常见操作系统或发行版上的实现。例如，macOS 目前默认包含 Vim 的轻量级安装。因此，掌握 Linux Vim 技能也可以帮助您完成其他方面的工作。

启动 Vim

在红帽企业 Linux 中，Vim 可通过两种不同的方式进行安装。安装方式的不同会影响到可供使用的功能和 Vim 命令。

您的服务器上可能仅安装了 vim-minimal 软件包。这属于一个轻量级安装，它仅包含核心功能集和基本 **vi** 命令。在这种情况下，您可以用 **vi filename** 打开文件进行编辑，本节中讨论的所有核心功能均可供您使用。

或者，您的服务器上可能安装有 vim-enhanced 软件包。此时会提供一套更完整的功能、一个在线帮助系统和一个教程程序。要想在该增强模式下启动 Vim，请使用 **vim** 命令。

```
[user@host ~]$ vim filename
```

无论采用哪种方式，在本节中讨论的核心功能都将适用于这两个命令。