

**注意**

如果安装了 vim-enhanced，普通用户可设置一个 shell 别名，以便在运行 **vi** 命令时自动获取 **vim** 命令。这不适用于 **root** 用户及其他 UID 小于 200 的用户（这些用户供系统服务使用）。

如果您正在以 **root** 用户身份编辑文件，并期望以增强模式运行 **vi**，可能会感到惊讶。同样，在安装了 vim-enhanced 的情况下，如果普通用户出于某种原因想使用简单的 **vi**，那么他们可能需要使用 **\vi** 来暂时覆盖别名。

高级用户可以使用 **\vi --version** 和 **vim --version** 来比较这两个命令的功能集。

Vim 管理模式

Vim 的一个不同寻常之处是它有几个运行模式，包括命令模式、扩展命令模式、编辑模式和可视模式。根据具体的模式，您可以发出命令、编辑文本或处理文本块。作为 Vim 新用户，您要始终了解当前的模式，因为击键在不同模式下具有不同的效果。

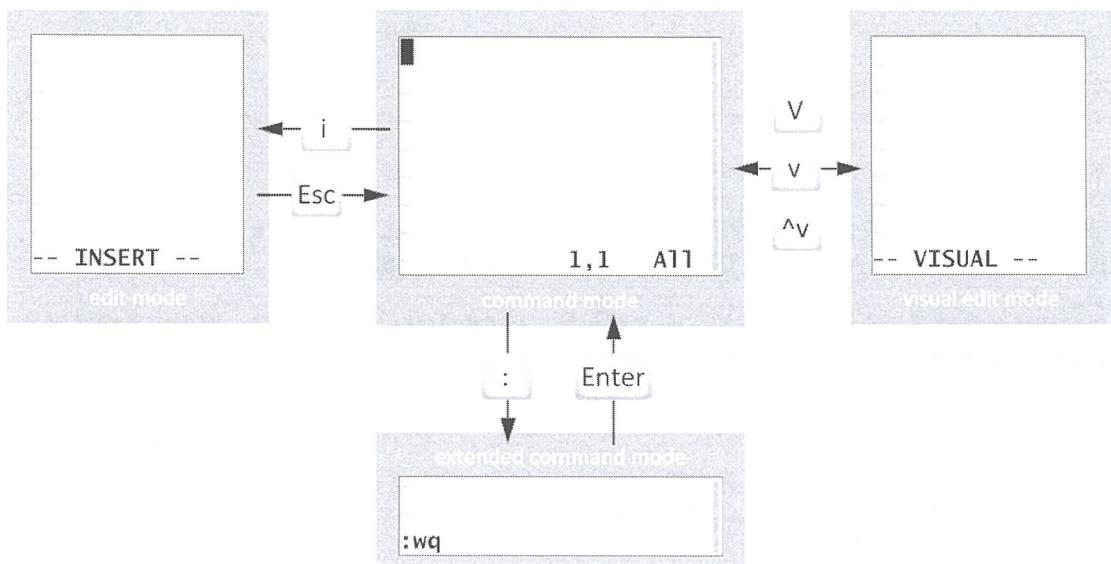


图 5.10: 在 Vim 模式间移动

第一次打开 Vim 时，它会以命令模式启动，可用于导航、剪切和粘贴，以及其他文本操作。通过单字符击键操作进入各个其他模式，访问特定的编辑功能：

- 按 **i** 键进入插入模式，其中键入的所有文本将变为文件内容。按 **Esc** 键返回到命令模式。
- 按 **v** 键进入可视模式，可在其中选择多个字符进行文本操作。使用 **Shift+V** 选择多行，使用 **Ctrl+V** 可选择文本块。用于进入可视模式的击键操作（**v**、**Shift+V** 或 **Ctrl+V**）也可用于退出。
- 按 **:** 键启动扩展命令模式，可以执行的任务包括写入文件（进行保存），以及退出 Vim 编辑器等。

**注意**

如果您不确定 Vim 所处的模式，可尝试按几次 **Esc** 以返回命令模式。在命令模式下按 **Esc** 并不会造成任何危害，因此多按几下按键也没有问题。

最低程度的基本 Vim 工作流

Vim 具有高效的组合击键操作，可进行高级编辑任务。虽然经练习后非常有用，Vim 的功能可能会难倒新用户。

i 键可使 Vim 进入插入模式。在此之后输入的所有文本都将被视为文件内容，直到您退出插入模式。**Esc** 键可退出插入模式，并让 Vim 返回到命令模式。**u** 键可撤销最近的编辑。按 **x** 键可删除单个字符。**:w** 命令可写入（保存）文件，并保留在命令模式中以进行更多编辑。**:wq** 命令可写入（保存）文件并退出 Vim。**:q!** 命令可退出 Vim，同时放弃上次写入以来进行的所有更改。Vim 用户必须学习这些命令才能完成任何编辑任务。

重排现有文本

在 Vim 中，复制和粘贴称为拖拉和放置，使用的命令字符是 **y** 和 **p**。首先将光标定位到要选择的第一个字符，然后进入可视模式。使用箭头键扩展可视选择。准备好时，按 **y** 将所选内容拖拉到内存中。将光标定位到新位置上，然后按 **p** 将所选内容放置到光标处。

Vim 中的可视模式

可视模式是一种突出显示和操作文本的绝佳方式。它有三种击键操作：

- 字符模式：**v**
- 行模式：**Shift+v**
- 块模式：**Ctrl+v**

字符模式可突出显示文本块中的句子。屏幕的底部会出现 **VISUAL** 一词。按 **v** 可进入可视字符模式。按 **Shift+v** 可进入行模式。屏幕的底部会出现 **VISUAL LINE** 一词。

可视块模式非常适合于操作数据文件。从光标位置，按 **Ctrl+v** 可进入视觉块。屏幕的底部会出现 **VISUAL BLOCK** 一词。使用箭头键可突出显示要更改的部分。



注意

Vim 具有很多功能，但首先应掌握其基本工作流。您并不需要很快地了解整个编辑器及其功能。通过练习熟悉这些基础知识，然后通过学习其他 Vim 命令（击键）来扩展您的 Vim 词汇表。

本节的练习将向您介绍 **vimtutor** 命令。本教程随 vim-enhanced 一起提供，是学习 Vim 核心功能的绝佳方式。



参考文献

vim(1) man page

vim 中的 **:help** 命令（如果安装了 vim-enhanced 软件包）。

Vim 编辑器

<http://www.vim.org/>

Vim 可视模式入门

<https://opensource.com/article/19/2/getting-started-vim-visual-mode>

► 指导练习

从 SHELL 提示符编辑文本文件

在本练习中，您将使用 **vimtutor** 练习 vim 编辑器中的基本编辑技巧。

成果

您应能够：

- 使用 Vim 编辑文件。
- 通过 **vimtutor** 来掌握 Vim。

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 edit-vim start** 命令。此脚本将验证目标服务器是否正在运行。

```
[student@workstation ~]$ lab edit-vim start
```

► 1. 使用 ssh 命令来登录 servera。

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

► 2. 打开 **vimtutor**。阅读“欢迎”屏幕，再执行 Lesson 1.1。

```
[student@servera ~]$ vimtutor
```

在该演示中，使用键盘上的箭头键进行导航。最初开发 **vi** 时，用户无法依赖箭头键或箭头键的键盘映射来移动光标。因此，**vi** 原来设计了使用标准字符键的命令来移动光标，如可方便分组的 **H**、**J**、**K** 和 **L**。

以下是记忆它们的一种方式：

hang back、**jump down**、**kick up** 和 **leap forward**。

► 3. 在 **vimtutor** 窗口中，执行 Lesson 1.2。

这节课介绍如何退出且放弃不需要的更改。所有更改都将丢失。有时，最好将关键文件保留为错误状态。

► 4. 在 **vimtutor** 窗口中，执行 Lesson 1.3。

Vim 拥有快速高效的击键操作，用于删除准确数量的词语、行、句子和段落。不过，任何编辑工作都可以通过使用 **x** 单字符删除操作来完成。

► 5. 在 **vimtutor** 窗口中，执行 Lesson 1.4。

对于大多数编辑任务，要按的第一个键是 **i** 键。

► 6. 在 **vimtutor** 窗口中，执行 Lesson 1.5。

在该课程中，仅教授了 **i**（插入）命令，作为进入编辑模式的击键操作。此 **vimtutor** 课时演示可用于更改进入插入模式时的光标位置的其他击键操作。在插入模式中，键入的所有文本都是文件内容。

► 7. 在 **vimtutor** 窗口中，执行 Lesson 1.6。

键入 **:wq** 以保存文件并退出编辑器。

► 8. 在 **vimtutor** 窗口中，阅读 Lesson 1 Summary。

vimtutor 命令包括另外六节多步课程。这几节课不分配为本课程的一部分，但您随时可以自己进行探索来了解更多信息。

► 9. 从 **servera** 退出。

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

完成

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

```
[student@workstation ~]$ lab edit-vim finish
```

本引导式练习到此结束。

更改 SHELL 环境

培训目标

学完本节后，您应能够设置 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。

使用 SHELL 变量

Bash shell 允许您设置 shell 变量，您可以使用这些变量来帮助运行命令或修改 shell 的行为。您还可以将 shell 变量导出为环境变量，它们会在程序启动时自动复制到从该 shell 运行的程序中。您可以使用变量来帮助更轻松地运行带有长参数的命令，或者将常用设置应用于从该 shell 运行的命令。

Shell 变量对于特定 shell 会话是唯一的。如果您打开了两个终端窗口，或者通过两个独立的登录会话登录同一远程服务器，那么您在运行两个 shell。每个 shell 都有自己的一组 shell 变量值。

为变量分配值

使用以下语法将值分配给 shell 变量：

```
VARIABLENAME=value
```

变量名称可以包含大写或小写字母、数字和下划线字符 (_)。例如，以下命令可设置 shell 变量：

```
[user@host ~]$ COUNT=40
[user@host ~]$ first_name=John
[user@host ~]$ file1=/tmp/abc
[user@host ~]$ _ID=RH123
```

请记住，此更改仅影响在其中运行命令的 shell，而不影响您可能在该服务器上运行的任何其他 shell。

您可以使用 **set** 命令列出当前设置的所有 shell 变量。（它还会列出所有 shell 函数，您可以忽略它们。）此列表足够长，您可能希望将输出通过管道传输到 **less** 命令，以便您可以一次查看一页。

```
[user@host ~]$ set | less
BASH=/usr/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:
force_fignore:histappend:interactive_comments:progcomp:promptvars:sourcepath
BASHRC_SOURCE=Y
...output omitted...
```

使用变量扩展检索值

您可以使用变量扩展来指代您已设置的变量值。为此，请在变量名称前加上美元符号 (\$)。在以下示例中，**echo** 命令显示输入的其余命令行，但是在执行变量扩展之后。

例如，以下命令可将变量 COUNT 设为 40。

```
[user@host ~]$ COUNT=40
```

如果输入命令 **echo COUNT**，它会显示出字符串 **COUNT**。

```
[user@host ~]$ echo COUNT  
COUNT
```

但如果输入命令 **echo \$COUNT**，则会显示出变量 **COUNT** 的值。

```
[user@host ~]$ echo $COUNT  
40
```

更实际的示例可能是使用变量来指代多个命令的长文件名。

```
[user@host ~]$ file1=/tmp/tmp.z9pXW0HqcC  
[user@host ~]$ ls -l $file1  
-rw----- 1 student student 1452 Jan 22 14:39 /tmp/tmp.z9pXW0HqcC  
[user@host ~]$ rm $file1  
[user@host ~]$ ls -l $file1  
total 0
```



重要

如果变量名称旁边有任何尾随字符，您可能需要使用花括号来保护变量名称。您始终都能在变量扩展中使用花括号，但您也会看到很多不需要它们并且将它们省略的示例。

在以下示例中，第一个 **echo** 命令尝试扩展不存在的变量 **COUNTx**，这不会导致错误，而是不返回任何内容。

```
[user@host ~]$ echo Repeat $COUNTx  
Repeat  
[user@host ~]$ echo Repeat ${COUNT}x  
Repeat 40x
```

使用 Shell 变量配置 Bash

一些 shell 变量在 Bash 启动时设置，但可以进行修改来调整 shell 的行为。

例如，两个影响 shell 历史记录和 **history** 命令的 shell 变量是 **HISTFILE** 和 **HISTFILESIZE**。如果设置了 **HISTFILE**，它将指定文件的位置，以便在退出时保存 shell 历史记录。默认情况下，这是用户的 **~/.bash_history** 文件。**HISTFILESIZE** 变量指定应将历史记录中的多少个命令保存到该文件中。

另一个例子是 **PS1**，这是一个控制 shell 提示符外观的 shell 变量。如果更改此值，它将改变 shell 提示符的外观。**bash(1)** man page 的“PROMPTING”部分中列出了提示符支持的多个特殊字符扩展。

```
[user@host ~]$ PS1="bash\$ "
bash$ PS1="[\u@\\h \w]\$ "
[user@host ~]$
```

有关上述示例应注意两点：首先，由于 PS1 设置的值是提示符，因此以尾随空格结束提示符几乎是可取的。其次，每当变量的值包含某种形式的空格（包括空格、制表符或回车）时，该值必须用引号括起来，单引号或双引号均可；这不是可选的。如果省略引号，则会出现意外结果。检查上面的 PS1 示例，并注意它遵循了建议（尾随空格）和规则（引号）。

使用环境变量配置程序

shell 提供了一个环境，供您从该 shell 中运行程序。例如，此环境包括有关文件系统上当前工作目录的信息、传递给程序的命令行选项，以及环境变量的值。程序可以使用这些环境变量来更改其行为或其默认设置。

不是环境变量的 Shell 变量只能由 shell 使用。环境变量可以由 shell 以及从该 shell 运行的程序使用。



注意

上一节中了解到的 **HISTFILE**、**HISTFILESIZE** 和 **PS1** 不需要导出为环境变量，因为它们仅供 shell 本身使用，而不由您从该 shell 运行的程序使用。

您可以使 shell 中定义的任何变量变为环境变量，方法是将它标记为使用 **export** 命令导出。

```
[user@host ~]$ EDITOR=vim
[user@host ~]$ export EDITOR
```

您可以在一个步骤中设置和导出变量：

```
[user@host ~]$ export EDITOR=vim
```

应用程序和会话使用这些变量来确定其行为。例如，shell 启动时自动将 **HOME** 变量设置为用户主目录的文件名。这可用于帮助程序确定保存文件的位置。

另一个例子是 **LANG**，它设定的是区域设置。这会调整程序输出的首选语言，字符集，日期、数字和货币的格式，以及程序的排序顺序。如果设置为 **en_US.UTF-8**，区域设置将使用美国英语及 UTF-8 Unicode 字符编码。如果设为其他设置，如 **fr_FR.UTF-8**，它将使用法语 UTF-8 Unicode 编码。

```
[user@host ~]$ date
Tue Jan 22 16:37:45 CST 2019
[user@host ~]$ export LANG=fr_FR.UTF-8
[user@host ~]$ date
mar. janv. 22 16:38:14 CST 2019
```

另一个重要环境变量是 **PATH**。**PATH** 变量包含一个含有程序的目录的冒号分隔列表：

```
[user@host ~]$ echo $PATH  
/home/user/.local/bin:/home/user/bin:/usr/share/Modules/bin:/usr/local/bin:/usr/  
bin:/usr/local/sbin:/usr/sbin
```

当您运行 **ls** 等命令时，shell 会按照顺序逐一在这些目录中查找可执行文件 **ls**，并且运行它找到的第一个匹配文件。（在典型的系统中，这是 **/usr/bin/ls**。）

您可以轻松地将其他目录添加到 PATH 的末尾。例如，您在 **/home/user/sbin** 中可能具有您希望像常规命令一样运行的可执行程序或脚本。针对当前会话，您可以在 PATH 的末尾添加 **/home/user/sbin**，如下所示：

```
[user@host ~]$ export PATH=${PATH}:/home/user/sbin
```

要列出特定 shell 的所有环境变量，请运行 **env** 命令：

```
[user@host ~]$ env  
...output omitted...  
LANG=en_US.UTF-8  
HISTCONTROL=ignoredups  
HOSTNAME=host.example.com  
XDG_SESSION_ID=4  
...output omitted...
```

设置默认文本编辑器

EDITOR 环境变量指定要用作命令行程序的默认文本编辑器的程序。如果不指定，很多程序都使用 **vi** 或 **vim**，但您可以根据需要覆盖此首选项：

```
[user@host ~]$ export EDITOR=nano
```



重要

按照惯例，shell 自动设置的环境变量和 shell 变量具有使用全部大写字符的名称。如果要设置自己的变量，您可能需要使用由小写字符组成的名称来帮助避免命名冲突。

自动设置变量

如果希望在 shell 启动时自动设置 shell 或环境变量，您可以编辑 Bash 启动脚本。当 Bash 启动时会运行几个包含 shell 命令的文本文件，以初始化 shell 环境。

运行的确切脚本取决于 shell 的启动方式，是交互式登录 shell、交互式非登录 shell 还是 shell 脚本。

假设是默认的 **/etc/profile**、**/etc/bashrc** 和 **~/.bash_profile** 文件，如果您要更改启动时影响所有交互式 shell 提示符的用户帐户，请编辑 **~/.bashrc** 文件。例如，您可以通过编辑要读取的文件，将该帐户的默认编辑器设置为 **nano**：

```
# .bashrc  
  
# Source global definitions
```

```
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific environment
PATH="$HOME/.local/bin:$HOME/bin:$PATH"
export PATH

# User specific aliases and functions
export EDITOR=nano
```



注意

调整影响所有用户帐户的设置的最佳方式是添加名称以 **.sh** 结尾的文件，并在该文件中包含对 **/etc/profile.d** 目录的更改。要这样做，您需要以 **root** 用户身份登录。

取消设置和取消导出变量

要完全取消设置和取消导出变量，请使用 **unset** 命令：

```
[user@host ~]$ echo $file1
/tmp/tmp.z9pXW0HqcC
[user@host ~]$ unset file1
[user@host ~]$ echo $file1

[user@host ~]$
```

要取消导出变量但不取消设置它，请使用 **export -n** 命令：

```
[user@host ~]$ export -n PS1
```



参考文献

bash(1)、**env(1)** 和 **builtins(1)** man page

► 指导练习

更改 SHELL 环境

在本练习中，您将使用 shell 变量和变量扩展来运行命令，并且设置环境变量来调整新 shell 的默认编辑器。

成果：

您应能够：

- 修改用户配置文件。
- 创建 shell 变量。
- 创建环境变量。

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 edit-shell start** 命令。此脚本将验证目标服务器是否正在运行。

```
[student@workstation ~]$ lab edit-shell start
```

► 1. 将 student 用户的 PS1 shell 变量更改为 **[\u@\h \t \w]\$**（记住将 PS1 的值放在引号内，并在美元符号后放入一个尾随空格）。这样会在提示符上添加时间。

1.1. 在 workstation 上，使用 ssh 命令登录 servera。

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

1.2. 使用 Vim 编辑 **~/.bashrc** 配置文件。

```
[student@servera ~]$ vim ~/.bashrc
```

1.3. 将 PS1 shell 变量及其值添加到 **~/.bashrc** 文件。记住在您设置的值的末尾包含一个尾随空格，并将包括尾随空格在内的整个值放在引号中。

```
...output omitted...  
# User specific environment and startup programs  
PATH=$PATH:$HOME/.local/bin:$HOME/bin  
PS1='[\u@\h \t \w]$ '  
export PATH
```

1.4. 从 servera 退出，然后使用 ssh 命令重新登录，以更新命令提示符。

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera 14:45:05 ~]$
```

- 2. 为本地 shell 变量分配值。变量名称可以包含大写或小写字母、数字和下划线字符。检索变量值。

- 2.1. 创建一个名为 `file` 且值为 `tmp.zdkei083` 的新变量。`tmp.zdkei083` 文件存在于 `student` 主目录中。

```
[student@servera 14:47:05 ~]$ file=tmp.zdkei083
```

- 2.2. 检索 `file` 变量的值。

```
[student@servera 14:48:35 ~]$ echo $file  
tmp.zdkei083
```

- 2.3. 使用变量名称 `file` 和 `ls -l` 命令列出 `tmp.zdkei083` 文件。使用 `rm` 命令和 `file` 变量名称删除 `tmp.zdkei083` 文件。确认它已被删除。

```
[student@servera 14:59:07 ~]$ ls -l $file  
-rw-rw-r--. 1 student student 0 Jan 23 14:59 tmp.zdkei083  
[student@servera 14:59:10 ~]$ rm $file  
[student@servera 14:59:15 ~]$ ls -l $file  
ls: cannot access 'tmp.zdkei083': No such file or directory
```

- 3. 为 `editor` 变量分配一个值。使用一个命令使变量成为环境变量。

```
[student@servera 14:46:40 ~]$ export EDITOR=vim  
[student@servera 14:46:55 ~]$ echo $EDITOR  
vim
```

- 4. 从 `servera` 退出。

```
[student@servera 14:47:11 ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

完成

在 `workstation` 上，运行 `lab edit-shell finish` 脚本来完成本练习。

```
[student@workstation ~]$ lab edit-shell finish
```

本引导式练习到此结束。

▶ 开放研究实验

创建、查看和编辑文本文件

任务执行清单

在本实验中，您将使用 **vim** 编辑器编辑文本文件。

成果

您应能够：

- 使用 Vim 执行文件编辑。
- 使用可视模式简化文件编辑。

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 edit-review start** 命令。

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

1. 将 **student** 的主目录中所有内容的长列表（包括隐藏目录和文件）重定向到名为 **editing_final_lab.txt** 的文件中。
2. 使用 Vim 编辑该文件。
3. 删除前三行。使用大写 **V** 进入基于行的可视模式。
4. 删除第一行上的列。使用小写 **v** 进入可视模式。小写 **v** 仅在一行中选择字符。应该将位于 **-rw-** 后面的列删除。
5. 删除剩余行中的列及随后的点 (**"**)。使用可视块模式。使用 **Ctrl+V** 控制序列进入可视块模式。使用此键序列选择多行上的字符块。应该将位于 **-rw-** 后面的列删除。
6. 使用可视块模式删除第四列。
7. 使用可视块模式删除时间列，但保留所有行中的月和日。
8. 删除 **Desktop** 行和 **Public** 行。使用大写 **V** 进入可视行模式。
9. 使用 **:wq** 命令保存并退出文件。制作一个备份，并使用日期（秒数）创建唯一文件名。
10. 在文件中附加一条虚线。虚线中应至少包含 12 个短划线。
11. 附加 **Documents** 目录的目录列表。通过一个命令行，在终端上列出目录列表并将它发送到 **editing_final_lab.txt** 文件。
12. 确认目录列表位于实验文件的底部。

评估

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

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

完成

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

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

本实验到此结束。

► 解决方案

创建、查看和编辑文本文件

任务执行清单

在本实验中，您将使用 **vim** 编辑器编辑文本文件。

成果

您应能够：

- 使用 Vim 执行文件编辑。
- 使用可视模式简化文件编辑。

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 edit-review start** 命令。

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

1. 将 **student** 的主目录中所有内容的长列表（包括隐藏目录和文件）重定向到名为 **editing_final_lab.txt** 的文件中。



注意

输出可能与显示的示例不完全匹配。

在 **workstation** 上，从 **student** 主目录，使用 **ls -al** 命令将所有内容的长列表重定向到名为 **editing_final_lab.txt** 的文件。

```
[student@workstation ~]$ ls -al > editing_final_lab.txt
```

2. 使用 Vim 编辑该文件。

```
[student@workstation ~]$ vim editing_final_lab.txt
```

3. 删除前三行。使用大写 **V** 进入基于行的可视模式。

使用箭头键将光标定位到第一行中的第一个字符。使用 **Shift+V** 进入基于行的可视模式。使用向下箭头键两次向下移动，以选择前三行。利用 **x** 删选行。

```
student@workstation:~$ total 32
drwxr-xr-x. 15 student student 4096 Jan 22 10:31 .
drwxr-xr-x. 3 root root 21 Jan 9 18:57 ..
-rw-r--r--. 1 student student 310 Jan 21 15:44 .bash_history
-rw-r--r--. 1 student student 18 Oct 12 23:56 .bash_logout
-rw-r--r--. 1 student student 141 Oct 12 23:56 .bash_profile
-rw-r--r--. 1 student student 312 Oct 12 23:56 .bashrc
drwxr-xr-x. 8 student student 201 Jan 14 14:16 .cache
drwxr-xr-x. 10 student student 203 Jan 14 14:16 .config
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Desktop
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Documents
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Downloads
-rw-rw-r--. 1 student student 0 Jan 22 10:45 editing_final_lab.txt
-rw-r--r--. 1 student student 16 Jan 14 14:15 .esd_auth
-rw-r--r--. 1 student student 310 Jan 14 14:16 .ICEauthority
drwxr-xr-x. 3 student student 19 Jan 14 14:16 .local
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Music
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Pictures
drwxrw----. 3 student student 19 Jan 14 14:16 .pki
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Public
drwxr-xr-x. 2 student student 73 Jan 14 15:35 .ssh
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Templates
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Videos
-rw-r--r--. 1 student student 1095 Jan 14 15:34 .viminfo
.

-- VISUAL LINE --

```

4. 删除第一行上的列。使用小写 **v** 进入可视模式。小写 **v** 仅在一列中选择字符。应该将位于 **-rw-** 后面的列删除。

使用箭头键将光标定位到第一个字符。使用小写 **v** 进入可视模式。使用箭头键将光标定位到最后一个字符。利用 **x** 删除所选内容。

```
student@workstation:~$ total 32
drwxr-xr-x. 15 student student 4096 Jan 22 10:31 .
drwxr-xr-x. 3 root root 21 Jan 9 18:57 ..
-rw-r--r--. 1 student student 310 Jan 21 15:44 .bash_history
-rw-r--r--. 1 student student 18 Oct 12 23:56 .bash_logout
-rw-r--r--. 1 student student 141 Oct 12 23:56 .bash_profile
-rw-r--r--. 1 student student 312 Oct 12 23:56 .bashrc
drwxr-xr-x. 8 student student 201 Jan 14 14:16 .cache
drwxr-xr-x. 10 student student 203 Jan 14 14:16 .config
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Desktop
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Documents
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Downloads
-rw-rw-r--. 1 student student 0 Jan 22 10:45 editing_final_lab.txt
-rw-r--r--. 1 student student 16 Jan 14 14:15 .esd_auth
-rw-r--r--. 1 student student 310 Jan 14 14:16 .ICEauthority
drwxr-xr-x. 3 student student 19 Jan 14 14:16 .local
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Music
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Pictures
drwxrw----. 3 student student 19 Jan 14 14:16 .pki
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Public
drwxr-xr-x. 2 student student 73 Jan 14 15:35 .ssh
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Templates
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Videos
-rw-r--r--. 1 student student 1095 Jan 14 15:34 .viminfo
.

-- VISUAL --

```

5. 删除剩余行中的列及随后的点 ("."). 使用可视块模式。使用 **Ctrl+V** 控制序列进入可视块模式。使用此键序列选择多行上的字符块。应该将位于 -rw- 后面的列删除。

使用箭头键将光标定位到第一个字符。使用 **Ctrl+V** 控制序列进入可视块模式。使用箭头键将光标定位到列中最后一行的最后一个字符。利用 **X** 删除所选内容。

```
student@workstation:~$ 
File Edit View Search Terminal Help
-rw- 1 student student 310 Jan 21 15:44 .bash_history
-rw-r--r--. 1 student student 18 Oct 12 23:56 .bash_logout
-rw-r--r--. 1 student student 141 Oct 12 23:56 .bash_profile
-rw-r--r--. 1 student student 312 Oct 12 23:56 .bashrc
drwx-----. 8 student student 201 Jan 14 14:16 .cache
drwxr-xr-x. 10 student student 203 Jan 14 14:16 .config
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Desktop
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Documents
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Downloads
-rw-rw-r--. 1 student student 0 Jan 22 10:45 editing_final_lab.txt
-rw-----. 1 student student 16 Jan 14 14:15 .esd_auth
-rw-----. 1 student student 310 Jan 14 14:16 .ICEAuthority
drwx-----. 3 student student 19 Jan 14 14:16 .local
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Music
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Pictures
drwxrw----. 3 student student 19 Jan 14 14:16 .pki
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Public
drwx-----. 2 student student 73 Jan 14 15:35 .ssh
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Templates
drwxr-xr-x. 2 student student 6 Jan 14 14:16 Videos
-rw-----. 1 student student 1095 Jan 14 15:34 .viminfo
~
~
~
~
~
-- VISUAL BLOCK --
20x7 21,11 All
```

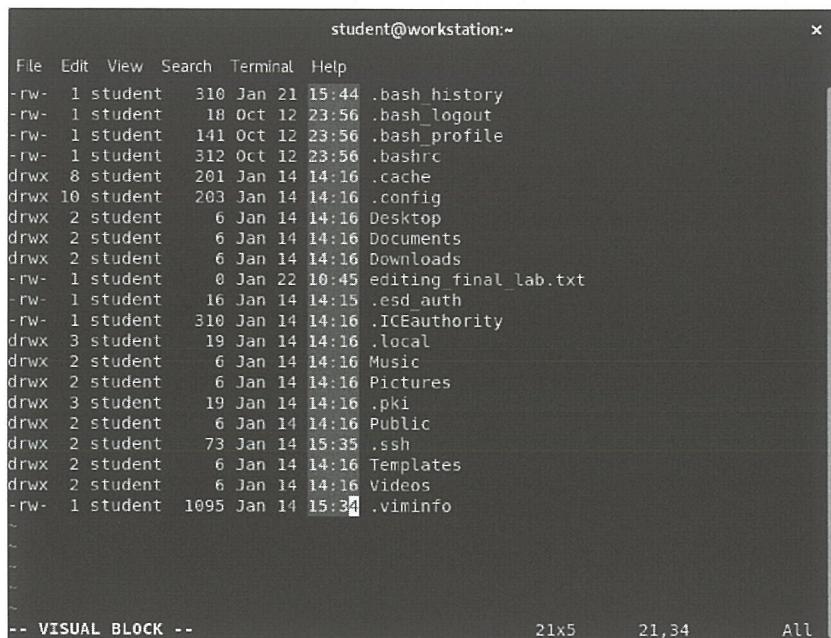
6. 使用可视块模式删除第四列。

使用箭头键将光标定位到第四列的第一个字符。使用 **Ctrl+V** 进入可视块模式。使用箭头键将光标定位到第四列的最后一个字符和相应的行。利用 **X** 删除所选内容。

```
student@workstation:~$ 
File Edit View Search Terminal Help
-rw- 1 student student 310 Jan 21 15:44 .bash_history
-rw- 1 student student 18 Oct 12 23:56 .bash_logout
-rw- 1 student student 141 Oct 12 23:56 .bash_profile
-rw- 1 student student 312 Oct 12 23:56 .bashrc
drwx 8 student student 201 Jan 14 14:16 .cache
drwx 10 student student 203 Jan 14 14:16 .config
drwx 2 student student 6 Jan 14 14:16 Desktop
drwx 2 student student 6 Jan 14 14:16 Documents
drwx 2 student student 6 Jan 14 14:16 Downloads
-rw- 1 student student 0 Jan 22 10:45 editing_final_lab.txt
-rw- 1 student student 16 Jan 14 14:15 .esd.auth
-rw- 1 student student 310 Jan 14 14:16 .ICEAuthority
drwx 3 student student 19 Jan 14 14:16 .local
drwx 2 student student 6 Jan 14 14:16 Music
drwx 2 student student 6 Jan 14 14:16 Pictures
drwx 3 student student 19 Jan 14 14:16 .pki
drwx 2 student student 6 Jan 14 14:16 Public
drwx 2 student student 73 Jan 14 15:35 .ssh
drwx 2 student student 6 Jan 14 14:16 Templates
drwx 2 student student 6 Jan 14 14:16 Videos
-rw- 1 student student 1095 Jan 14 15:34 .viminfo
~
~
~
~
~
-- VISUAL BLOCK --
21x7 21,23 All
```

7. 使用可视块模式删除时间列，但保留所有行中的月和日。

使用箭头键将光标定位到第一个字符。使用 **Ctrl+V** 进入可视块模式。使用箭头键将光标定位到时间列末行的最后一个字符。利用 **X** 删除所选内容。

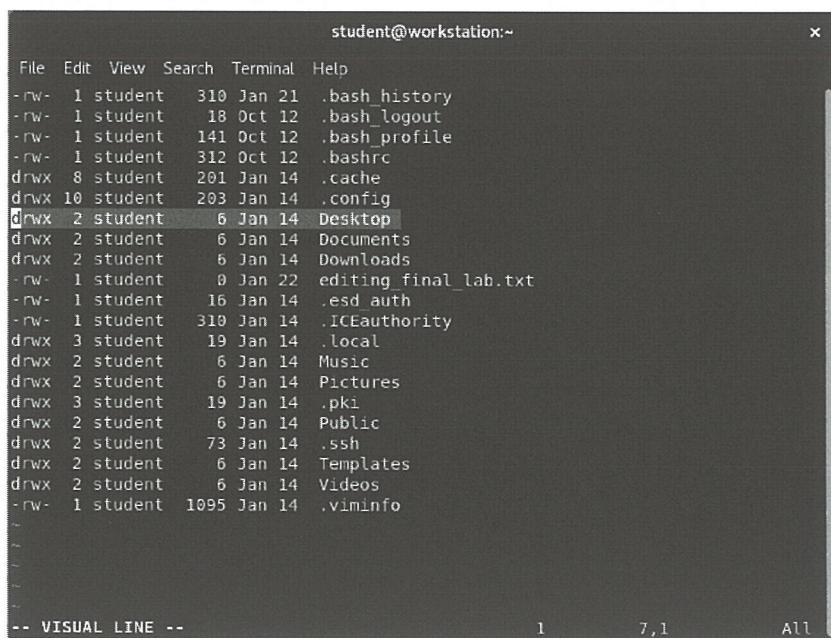


```
student@workstation:~$ ls
-rw- 1 student 310 Jan 21 15:44 .bash_history
-rw- 1 student 18 Oct 12 23:56 .bash_logout
-rw- 1 student 141 Oct 12 23:56 .bash_profile
-rw- 1 student 312 Oct 12 23:56 .bashrc
drwx 8 student 201 Jan 14 14:16 .cache
drwx 10 student 203 Jan 14 14:16 .config
drwx 2 student 6 Jan 14 14:16 Desktop
drwx 2 student 6 Jan 14 14:16 Documents
drwx 2 student 6 Jan 14 14:16 Downloads
-rw- 1 student 0 Jan 22 10:45 editing_final_lab.txt
-rw- 1 student 16 Jan 14 14:15 .esd_auth
-rw- 1 student 310 Jan 14 14:16 .ICEauthority
drwx 3 student 19 Jan 14 14:16 .local
drwx 2 student 6 Jan 14 14:16 Music
drwx 2 student 6 Jan 14 14:16 Pictures
drwx 3 student 19 Jan 14 14:16 .pki
drwx 2 student 6 Jan 14 14:16 Public
drwx 2 student 73 Jan 14 15:35 .ssh
drwx 2 student 6 Jan 14 14:16 Templates
drwx 2 student 6 Jan 14 14:16 Videos
-rw- 1 student 1095 Jan 14 15:34 .viminfo
~

-- VISUAL BLOCK --
21x5      21,34      All
```

8. 删 除 **Desktop** 行和 **Public** 行。使用大写 **V** 进入可视行模式。

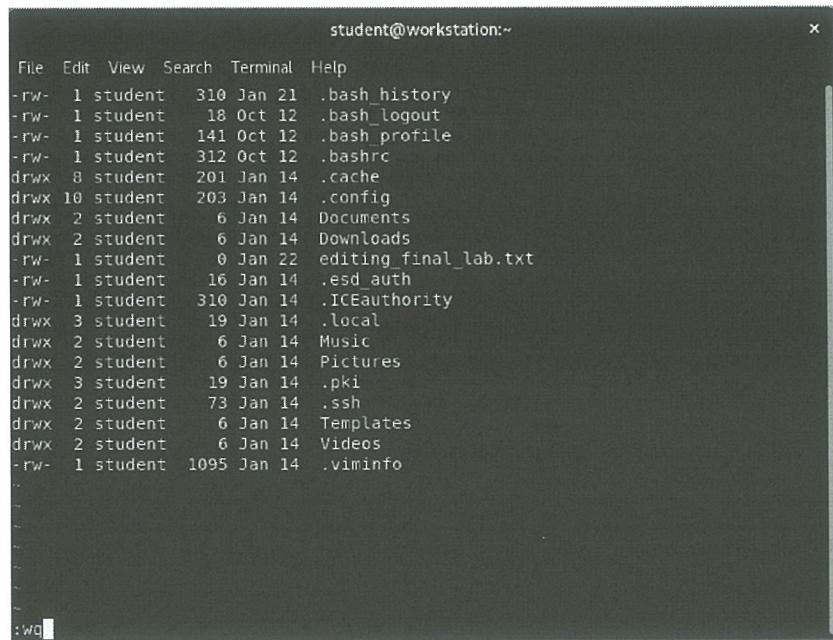
使用箭头键将光标定位到 **Desktop** 行的任何字符上。使用大写 **V** 进入可视模式。选中整行。利用 **X** 删除所选内容。对 **Public** 行重复此操作。



```
student@workstation:~$ ls
-rw- 1 student 310 Jan 21 15:44 .bash_history
-rw- 1 student 18 Oct 12 23:56 .bash_logout
-rw- 1 student 141 Oct 12 23:56 .bash_profile
-rw- 1 student 312 Oct 12 23:56 .bashrc
drwx 8 student 201 Jan 14 14:16 .cache
drwx 10 student 203 Jan 14 14:16 .config
drwx 2 student 6 Jan 14 14:16 Desktop
drwx 2 student 6 Jan 14 14:16 Documents
drwx 2 student 6 Jan 14 14:16 Downloads
-rw- 1 student 0 Jan 22 10:45 editing_final_lab.txt
-rw- 1 student 16 Jan 14 14:15 .esd_auth
-rw- 1 student 310 Jan 14 14:16 .ICEauthority
drwx 3 student 19 Jan 14 14:16 .local
drwx 2 student 6 Jan 14 14:16 Music
drwx 2 student 6 Jan 14 14:16 Pictures
drwx 3 student 19 Jan 14 14:16 .pki
drwx 2 student 6 Jan 14 14:16 Public
drwx 2 student 73 Jan 14 15:35 .ssh
drwx 2 student 6 Jan 14 14:16 Templates
drwx 2 student 6 Jan 14 14:16 Videos
-rw- 1 student 1095 Jan 14 15:34 .viminfo
~

-- VISUAL LINE --
1      7,1      All
```

9. 使用 :wq 命令保存并退出文件。制作一个备份，并使用日期（秒数）创建唯一文件名。



```
student@workstation:~$ ls -l
total 128
drwxr-xr-x 2 student staff 68 Jan 14 .
drwxr-xr-x 2 student staff 68 Jan 14 ..
-rw-r--r-- 1 student staff 1095 Jan 14 editing_final_lab.txt
student@workstation:~$ cp editing_final_lab.txt \
editing_final_lab_$(date +%s).txt
```

```
[student@workstation ~]$ cp editing_final_lab.txt \
editing_final_lab_$(date +%s).txt
```

10. 在文件中附加一条虚线。虚线中应至少包含 12 个短划线。

```
[student@workstation ~]$ echo "-----" \
>> editing_final_lab.txt
```

11. 附加 **Documents** 目录的目录列表。通过一个命令行，在终端上列出目录列表并将它发送到 **editing_final_lab.txt** 文件。

```
[student@workstation ~]$ ls Documents/ | tee -a editing_final_lab.txt
lab_review.txt
```

12. 确认目录列表位于实验文件的底部。

```
[student@workstation ~]$ cat editing_final_lab.txt
total 128
drwxr-xr-x 2 student staff 68 Jan 14 .
drwxr-xr-x 2 student staff 68 Jan 14 ..
-rw-r--r-- 1 student staff 1095 Jan 14 editing_final_lab.txt
```

```
drwx 2 student 6 Jan 14 Pictures  
drwx 3 student 19 Jan 14 .pki  
drwx 2 student 73 Jan 14 .ssh  
drwx 2 student 6 Jan 14 Templates  
drwx 2 student 6 Jan 14 Videos  
-rw- 1 student 1095 Jan 14 .viminfo  
-----  
lab_review.txt
```

评估

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

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

完成

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

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

本实验到此结束。

总结

在本章中，您学到了：

- 运行程序或进程涉及三个标准通信通道：标准输入、标准输出和标准错误。
- 您可以使用 I/O 重定向从文件中读取标准输入，或者将输出或错误从进程写入文件。
- 管道可用于将标准输出从一个进程连接到另一个进程的标准输入，并可用于格式化输出或构建复杂命令。
- 您应该了解如何使用至少一种命令行文本编辑器，一般都会安装 Vim。
- Shell 变量可以帮助您运行命令，它们对于特定 shell 会话而言是唯一的。
- 环境变量可以帮助您配置 shell 的行为或 shell 启动的进程。

章 6

管理本地用户和组

目标

创建、管理和删除本地用户和组，以及管理本地密码策略。

培训目标

- 描述 Linux 系统上用户和组的用途。
- 切换到超级用户帐户来管理 Linux 系统，并使用 **sudo** 命令授予其他用户超级用户访问权限。
- 创建、修改和删除本地定义的用户帐户。
- 创建、修改和删除本地定义的组帐户。
- 为用户设置密码管理策略，并且手动锁定和解锁用户帐户。

章节

- 描述用户和组概念（及测验）
- 获取超级用户访问权限（及引导式练习）
- 管理本地用户帐户（及引导式练习）
- 管理本地组帐户（及引导式练习）
- 管理用户密码（及引导式练习）

实验

管理本地 Linux 用户和组

描述用户和组概念

培训目标

学完本节后，您应能够描述 Linux 系统上用户和组的用途。

什么是用户？

用户帐户用于在可以运行命令的不同人员和程序之间提供安全界限。

用户使用用户名向人类用户标识自己并增加操作的便利性。在内部，系统通过分配的唯一标识号（用户 ID 或 UID）来区分不同的用户帐户。人类使用用户帐户时，通常会为其分配一个密码，供用户用于在登录时证明他们是实际的授权用户。

用户帐户构成了系统安全的基础。系统中的每个进程（运行程序）都作为一个特定用户运行。每个文件都有一个特定用户作为其所有者。文件所有权有助于系统对文件用户实施访问控制。与运行进程相关联的用户可确定该进程可访问的文件和目录。

用户帐户有三种主要类型：超级用户、系统用户和普通用户。

- 超级用户帐户用于管理系统。超级用户的名称为 **root**，其帐户 UID 为 0。超级用户对系统具有完全访问权限。
- 系统的系统用户帐户供提供支持服务进程使用。这些进程（或守护进程）通常不需要以超级用户身份运行。系统会为它们分配非特权帐户，允许它们确保其文件和其他资源不受彼此以及系统上普通用户的影响。用户无法使用系统用户帐户以交互方式登录。
- 大多数用户都有用于处理日常工作的普通用户帐户。与系统用户一样，普通用户对系统具有有限的访问权限。

您可以使用 **id** 命令显示有关当前已登录用户的信息。

```
[user01@host ~]$ id  
uid=1000(user01) gid=1000(user01) groups=1000(user01)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

要查看有关其他用户的基本信息，请将用户名作为参数传递给 **id** 命令。

```
[user01@host]$ id user02  
uid=1002(user02) gid=1001(user02) groups=1001(user02)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

要查看文件的所有者，可使用 **ls -l** 命令。要查看目录的所有者，可使用 **ls -ld** 命令。在下列输出中，第三列显示用户名。

```
[user01@host ~]$ ls -l file1  
-rw-rw-r--. 1 user01 user01 0 Feb 5 11:10 file1  
[user01@host]$ ls -ld dir1  
drwxrwxr-x. 2 user01 user01 6 Feb 5 11:10 dir1
```

要查看进程信息，可使用 **ps** 命令。默认为仅显示当前 shell 中的进程。添加 **a** 选项可查看与某一终端相关的所有进程。若要查看与进程相关联的用户，请在命令中包含 **u** 选项。在下列输出中，第一列显示用户名。

```
[user01@host]$ ps -au
USER        PID %CPU %MEM      VSZ   RSS TTY      STAT START   TIME COMMAND
root        777  0.0  0.0 225752  1496  tty1      Ss+  11:03   0:00 /sbin/agetty -o -
p -- \u --noclear tty1 linux
root        780  0.0  0.1 225392  2064  ttyS0     Ss+  11:03   0:00 /sbin/agetty -o -
p -- \u --keep-baud 115200,38400,9600
user01     1207  0.0  0.2 234044  5104  pts/0      Ss   11:09   0:00 -bash
user01     1319  0.0  0.2 266904  3876  pts/0      R+   11:33   0:00 ps au
```

以上命令的输出按名称显示用户，但操作系统内部利用 UID 来跟踪用户。用户名到 UID 的映射在帐户信息数据库中定义。默认情况下，系统使用 **/etc/passwd** 文件存储有关本地用户的信息。

/etc/passwd 文件的每一行都包含了有关某个用户的信息。它分为七个以冒号分隔的字段。以下是 **/etc/passwd** 中某一行的示例：

① user01:②x:③1000:④1000:⑤User One:⑥/home/user01:⑦/bin/bash

- ① 该用户 (**user01**) 的用户名。
- ② 用户的密码曾以加密格式存储在此处。现在它已移至 **/etc/shadow** 文件，稍后我们将对此进行介绍。该字段始终应为 **x**。
- ③ 该用户帐户的 UID 号 (**1000**)。
- ④ 该用户帐户的主要组的 GID 号 (**1000**)。本节稍后将对组进行讨论。
- ⑤ 该用户的真实姓名 (**User One**)。
- ⑥ 该用户的主目录 (**/home/user01**)。这是 shell 启动时的初始工作目录，其中包含有用户数据和配置设置。
- ⑦ 该用户的默认 shell 程序，会在登录时运行 (**/bin/bash**)。对于普通用户，这通常是提供用户命令行提示符的程序。如果系统用户不允许进行交互式登录，该用户可能会使用 **/sbin/nologin**。

什么是组？

组是需要共享文件和其他系统资源访问权限的用户的集合。组可用于向一组用户授予文件访问权限，而非仅仅向一个用户授予访问权限。

与用户相似，组也有组名称以增加操作的便利性。在内部，系统通过分配的唯一标识号（组 ID 或 GID）来区分不同的组。

组名称到 GID 的映射在组帐户信息数据库中定义。默认情况下，系统使用 **/etc/group** 文件存储有关本地组的信息。

/etc/group 文件的每一行都包含了有关某个组的信息。每个组条目被分为四个以冒号分隔的字段。以下是 **/etc/group** 中某一行的示例：

① group01:②x:③10000:④user01,user02,user03

- ① 该组的组名称 (**group01**)。
- ② 过时的组密码字段。该字段始终应为 **x**。
- ③ 该组的 GID 号 (**10000**)。

- ④ 属于作为补充组的该组成员的用户列表（`user01`、`user02`、`user03`）。本节稍后将对主要组（或默认组）和补充组进行讨论。

主要组和补充组

每个用户有且只有一个主要组。对于本地用户而言，这是按 `/etc/passwd` 文件中的 GID 号列出的组。默认情况下，这是拥有用户创建的新文件的组。

通常，在创建新的普通用户时，会创建一个与该用户同名的新组。该组将用作新用户的主要组，而该用户是这一用户专用组的唯一成员。事实证明，这有助于简化文件权限的管理。本课程的后续部分将对此进行讨论。

用户也可以有补充组。补充组的成员资格由 `/etc/group` 文件确定。根据所在的组是否具有访问权限，将授予用户对文件的访问权限。具有访问权限的组是用户的主要组还是补充组无关紧要。

例如，如果用户 `user01` 有一个主要组 `user01` 以及两个补充组 `wheel` 和 `webadmin`，那么该用户就可以读取其中任何一个组可读的文件。

该 `id` 命令还可用于查找用户的组成员身份。

```
[user03@host ~]$ id  
uid=1003(user03) gid=1003(user03) groups=1003(user03),10(wheel),10000(group01)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

在上面的示例中，`user03` 将组 `user03` 作为自己的主要组 (**gid**)。**groups** 项目列出了该用户的所有组。除了主要组 `user03` 外，用户还有组 `wheel` 和 `group01` 作为补充组。



参考文献

`id(1)`、`passwd(5)` 和 `group(5)` man page

`info libc` (GNU C 库参考书册)

· 第 30 节：用户和组

(请注意，必须安装 glibc-devel 软件包，此 info 节点才可用。)

► 小测验

描述用户和组概念

选择以下问题的正确答案：

► 1. 哪一项表示在最基本的层面上标识用户的编号？

- a. 主用户
- b. UID
- c. GID
- d. 用户名

► 2. 哪一项表示提供用户命令行提示符的程序？

- a. 主 shell
- b. 主目录
- c. 登录 shell
- d. 命令名称

► 3. 哪一项目或文件表示本地组信息的位置？

- a. 主目录
- b. `/etc/passwd`
- c. `/etc/GID`
- d. `/etc/group`

► 4. 哪一项目或文件表示用户个人文件的位置？

- a. 主目录
- b. 登录 shell
- c. `/etc/passwd`
- d. `/etc/group`

► 5. 哪一项表示在最基本的层面上标识组的编号？

- a. 主组
- b. UID
- c. GID
- d. groupid

► 6. 哪一项目或文件表示本地用户帐户信息的位置？

- a. 主目录
- b. `/etc/passwd`
- c. `/etc/UID`
- d. `/etc/group`