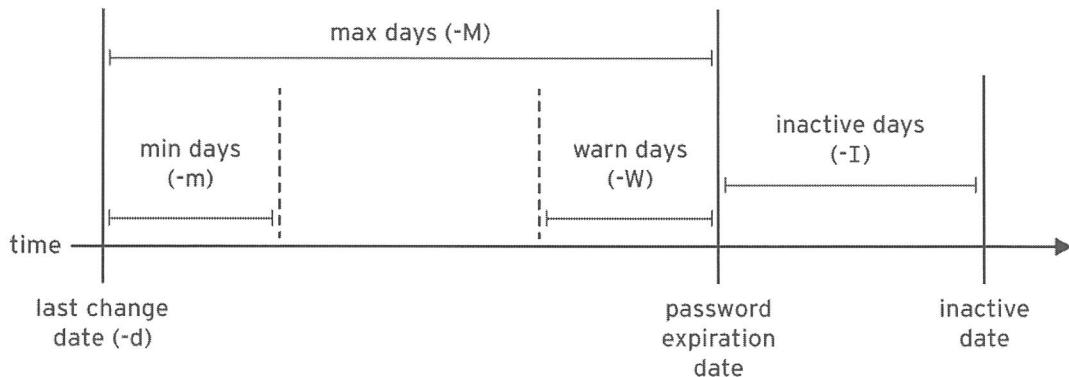


果与已加密密码不符，则用户键入了错误的密码，登录尝试也会失败。这种方式允许系统判断用户是否键入了正确的密码，同时又不以用于登录的密码形式来存储密码。

配置密码期限

下图显示了相关的密码期限参数，可以通过 **chage** 命令对其调整来实施密码期限策略。



```
[user01@host ~]$ sudo chage -m 0 -M 90 -W 7 -I 14 user03
```

前面的 **chage** 命令使用了 **-m**、**-M**、**-W** 和 **-I** 选项，它们分别用于设置用户密码的最短期限、最长期限、警告周期和失效期限。

chage -d 0 user03 命令强制 user03 用户在下一次登录时更新其密码。

chage -l user03 命令显示 user03 的密码期限详情。

chage -E 2019-08-05 user03 命令使 user03 用户的帐户于 2019-08-05 到期（YYYY-MM-DD 格式）。



注意

date 命令可用于计算未来的日期。**-u** 选项报告 UTC 时间。

```
[user01@host ~]$ date -d "+45 days" -u
Thu May 23 17:01:20 UTC 2019
```

编辑 **/etc/login.defs** 文件中的密码期限配置项，以设置默认的密码期限策略。PASS_MAX_DAYS 设置密码的默认最长期限。PASS_MIN_DAYS 设置密码的默认最短期限。PASS_WARN_AGE 设置密码的默认警告周期。默认密码期限策略的任何更改都仅对新用户有效。现有用户将继续使用旧密码期限设置，而非新密码设置。

限制访问

您可以使用 **chage** 命令来设置帐户到期日期。到了该日期时，用户无法以交互方式登录系统。**usermod** 命令可以通过 **-L** 选项锁定帐户。

```
[user01@host ~]$ sudo usermod -L user03
[user01@host ~]$ su - user03
Password: redhat
su: Authentication failure
```

如果用户离开公司，管理员可以通过一个 **usermod** 命令锁定帐户并使其到期。该日期必须使用距离 1970-01-01 的天数来指定，或者以 YYYY-MM-DD 格式指定。

```
[user01@host ~]$ sudo usermod -L -e 2019-10-05 user03
```

前面的 **usermod** 命令使用 **-e** 选项来设置给定用户帐户的帐户到期日期。**-L** 选项锁定用户的密码。

锁定帐户可防止用户使用密码向系统进行验证。这是预防已离开公司的员工访问某一帐户的推荐方式。如果以后该员工返回，其帐户可通过 **usermod -U** 解锁。如果帐户也已到期，务必也要更改到期日期。

nologin Shell

nologin shell 用作不打算以交互方式登录系统的用户帐户的替代 shell。从安全角度来看，如果用户帐户担当的职责不需要用户登录系统，则禁止用户帐户登录系统是明智的。例如，邮件服务器可能需要帐户来存储邮件，需要密码供用户通过检索邮件所用的邮件客户端进行身份验证。用户不需要直接登录该系统。

这种情况的常用解决方案是将用户的登录 shell 设为 **/sbin/nologin**。如果用户试图直接登录系统，**nologin** shell 将关闭该连接。

```
[user01@host ~]$ usermod -s /sbin/nologin user03
[user01@host ~]$ su - user03
Last login: Wed Feb  6 17:03:06 IST 2019 on pts/0
This account is currently not available.
```



重要

nologin shell 可以防止以交互方式使用系统，但不会阻止所有访问。如果用户使用用户密码进行身份验证，他们有时可以通过身份验证，并使用 Web 应用、文件传输程序或邮件读取程序等应用上传或检索文件。



参考文献

chage(1)、**usermod(8)**、**shadow(5)**、**crypt(3)** man page

► 指导练习

管理用户密码

在本练习中，您将为几个用户设置密码策略。

成果

您应能够：

- 强制用户在第一次登录系统时更改密码。
- 强制每 90 天更改一次密码。
- 将帐户设置为从当天起 180 天后过期。

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 users-pw-manage start` 脚本来开始本练习。此脚本会创建必要的用户帐户和文件，以确保环境设置正确无误。

```
[student@workstation ~]$ lab users-pw-manage start
```

- 1. 从 `workstation`，以 `student` 用户身份打开连接 `servera` 的 SSH 会话。

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

- 2. 在 `servera` 上，以 `student` 身份探索用户帐户的锁定和解锁。

2.1. 以 `student` 身份，使用管理权限锁定 `operator1` 帐户。

```
[student@servera ~]$ sudo usermod -L operator1
[sudo] password for student:
```

2.2. 尝试以 `operator1` 身份登录。这应该会失败。

```
[student@servera ~]$ su - operator1
Password: redhat
su: Authentication failure
```

2.3. 解锁 `operator1` 帐户。

```
[student@servera ~]$ sudo usermod -U operator1
```

2.4. 再次尝试以 `operator1` 身份登录。这应该会成功。

```
[student@servera ~]$ su - operator1  
Password: redhat  
...output omitted...  
[operator1@servera ~]$
```

2.5. 从 operator1 用户的 shell 退出，以返回到 student 用户的 shell。

```
[operator1@servera ~]$ exit  
logout
```

► 3. 更改 operator1 的密码策略，使其每 90 天要求创建新密码。确认已成功设置密码期限。

3.1. 将 operator1 用户的密码的最长期限设为 90 天。

```
[student@servera ~]$ sudo chage -M 90 operator1
```

3.2. 验证 operator1 用户的密码是否在更改后 90 天过期。

```
[student@servera ~]$ sudo chage -l operator1  
Last password change : Jan 25, 2019  
Password expires : Apr 25, 2019  
Password inactive : never  
Account expires : never  
Minimum number of days between password change : 0  
Maximum number of days between password change : 90  
Number of days of warning before password expires : 7
```

► 4. 强制 operator1 帐户在第一次登录时更改密码。

```
[student@servera ~]$ sudo chage -d 0 operator1
```

► 5. 以 operator1 身份登录，并将密码更改为 forsooth123。设置密码后，返回到 student 用户的 shell。

5.1. 以 operator1 身份登录，并在提示时将密码更改为 forsooth123。

```
[student@servera ~]$ su - operator1  
Password: redhat  
You are required to change your password immediately (administrator enforced)  
Current password: redhat  
New password: forsooth123  
Retype new password: forsooth123  
...output omitted...  
[operator1@servera ~]$
```

5.2. 从 operator1 用户的 shell 退出，以返回到 student 用户的 shell。

```
[operator1@servera ~]$ exit  
logout
```

- 6. 将 operator1 帐户设置为从当天起 180 天后过期。提示：**date -d "+180 days"** 为您提供从当前日期和时间起 180 天的日期和时间。

6.1. 确定未来 180 天后的日期。将 %F 格式用于 **date** 命令以获取确切的值。

```
[student@servera ~]$ date -d "+180 days" +%F  
2019-07-24
```

根据系统中的当前日期和时间，您可能会获得不同的值供以下步骤中使用。

6.2. 将帐户设置为在上一步中显示的日期到期。

```
[student@servera ~]$ sudo chage -E 2019-07-24 operator1
```

6.3. 验证是否已成功设置帐户到期日期。

```
[student@servera ~]$ sudo chage -l operator1  
Last password change : Jan 25, 2019  
Password expires : Apr 25, 2019  
Password inactive : never  
Account expires : Jul 24, 2019  
Minimum number of days between password change : 0  
Maximum number of days between password change : 90  
Number of days of warning before password expires : 7
```

- 7. 将所有用户的密码设置为自当前日期起 180 天后过期。使用管理权限编辑配置文件。

7.1. 在 **/etc/login.defs** 中，将 **PASS_MAX_DAYS** 设为 **180**。通过文本编辑器打开文件时应使用管理权限。您可以使用 **sudo vim /etc/login.defs** 命令执行此步骤。

```
...output omitted...  
# Password aging controls:  
#  
#      PASS_MAX_DAYS    Maximum number of days a password may be  
#      used.  
#      PASS_MIN_DAYS    Minimum number of days allowed between  
#      password changes.  
#      PASS_MIN_LEN     Minimum acceptable password length.  
#      PASS_WARN_AGE    Number of days warning given before a  
#      password expires.  
  
PASS_MAX_DAYS 180  
PASS_MIN_DAYS 0  
PASS_MIN_LEN 5  
PASS_WARN_AGE 7  
...output omitted...
```



重要

默认密码和帐户到期设置对新用户有效，但对现有用户无效。

7.2. 从 servera 注销。

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

完成

在 workstation 上，运行 **lab users-pw-manage finish** 来完成本练习。此脚本将删除在练习开始时创建的用户帐户和文件，以确保环境清理干净。

```
[student@workstation ~]$ lab users-pw-manage finish
```

本引导式练习到此结束。

▶ 开放研究实验

管理本地用户和组

任务执行清单

在本实验中，您将设置一个默认本地密码策略，创建一个包含三个用户的补充组，并允许该组使用 **sudo** 以 **root** 身份运行命令，然后修改一个用户的密码策略。

成果

您应能够：

- 为本地用户的密码设置默认密码期限策略。
- 创建一个组，并将该组用作新用户的补充组。
- 创建三个新用户，并使这个新组作为其补充组。
- 配置补充组的组成员，使其能使用 **sudo** 以任何用户身份运行任何命令。
- 设置一个特定于用户的密码期限策略。

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 users-review start** 脚本来开始本练习。此脚本会创建必要的文件，以确保环境设置正确无误。

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

1. 从 **workstation**，以 **student** 用户身份打开连接 **serverb** 的 SSH 会话。
2. 在 **serverb** 上，确保新建的用户具有必须每 30 天更改一次的密码。
3. 创建新组 **consultants**，GID 设为 **35000**。
4. 为 **consultants** 的所有成员配置管理权限，使其能够以任何用户身份执行任何命令。
5. 创建 **consultant1**、**consultant2** 和 **consultant3** 用户，并使 **consultants** 作为它们的补充组。
6. 将 **consultant1**、**consultant2** 和 **consultant3** 帐户设为从当天起 90 天后过期。
7. 更改 **consultant2** 帐户的密码策略，使其每 15 天要求创建新密码。
8. 另外，强制 **consultant1**、**consultant2** 和 **consultant3** 用户在第一次登录时更改密码。

评估

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

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

完成

在 workstation 上，运行 **lab users-review finish** 脚本来完成本实验。此脚本将删除在实验过程中创建的用户帐户和文件，以确保环境清理干净。

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

本实验到此结束。

► 解决方案

管理本地用户和组

任务执行清单

在本实验中，您将设置一个默认本地密码策略，创建一个包含三个用户的补充组，并允许该组使用 **sudo** 以 **root** 身份运行命令，然后修改一个用户的密码策略。

成果

您应能够：

- 为本地用户的密码设置默认密码期限策略。
- 创建一个组，并将该组用作新用户的补充组。
- 创建三个新用户，并使这个新组作为其补充组。
- 配置补充组的组成员，使其能使用 **sudo** 以任何用户身份运行任何命令。
- 设置一个特定于用户的密码期限策略。

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 users-review start** 脚本来开始本练习。此脚本会创建必要的文件，以确保环境设置正确无误。

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

1. 从 **workstation**，以 **student** 用户身份打开连接 **serverb** 的 SSH 会话。

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

2. 在 **serverb** 上，确保新建的用户具有必须每 30 天更改一次的密码。

2.1. 在 **/etc/login.defs** 中，将 **PASS_MAX_DAYS** 设为 **30**。通过文本编辑器打开文件时应使用管理权限。您可以使用 **sudo vim /etc/login.defs** 命令执行此步骤。当 **sudo** 提示您输入 **student** 用户的密码时，将 **student** 用作密码。

```
...output omitted...
# Password aging controls:
#
#      PASS_MAX_DAYS    Maximum number of days a password may be
#      used.
#      PASS_MIN_DAYS    Minimum number of days allowed between
#      password changes.
```

```
#      PASS_MIN_LEN      Minimum acceptable password length.
#      PASS_WARN_AGE      Number of days warning given before a
#                          password expires.
#
PASS_MAX_DAYS    30
PASS_MIN_DAYS    0
PASS_MIN_LEN     5
PASS_WARN_AGE    7
...output omitted...
```

3. 创建新组 **consultants**, GID 设为 **35000**。

```
[student@serverb ~]$ sudo groupadd -g 35000 consultants
```

4. 为 **consultants** 的所有成员配置管理权限，使其能够以任何用户身份执行任何命令。

4.1. 创建新文件 **/etc/sudoers.d/consultants**，并添加以下内容。您可以使用 **sudo vim /etc/sudoers.d/consultants** 命令执行此步骤。

```
%consultants  ALL=(ALL) ALL
```

5. 创建 **consultant1**、**consultant2** 和 **consultant3** 用户，并使 **consultants** 作为它们的补充组。

```
[student@serverb ~]$ sudo useradd -G consultants consultant1
[student@serverb ~]$ sudo useradd -G consultants consultant2
[student@serverb ~]$ sudo useradd -G consultants consultant3
```

6. 将 **consultant1**、**consultant2** 和 **consultant3** 帐户设为从当天起 90 天后过期。

6.1. 确定未来 90 天后的日期。根据系统中的当前日期和时间，您可能会获得与以下输出不同的值。

```
[student@serverb ~]$ date -d "+90 days" +%F
2019-04-28
```

6.2. 设置 **consultant1**、**consultant2** 和 **consultant3** 帐户的帐户到期日期，使其与上一步中确定的值相同。

```
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant1
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant2
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant3
```

7. 更改 **consultant2** 帐户的密码策略，使其每 15 天要求创建新密码。

```
[student@serverb ~]$ sudo chage -M 15 consultant2
```

8. 另外，强制 **consultant1**、**consultant2** 和 **consultant3** 用户在第一次登录时更改密码。

8.1. 将密码更改的最后一天设置为 **0**，这样每当用户在第一次登录系统时都必须更改密码。

```
[student@serverb ~]$ sudo chage -d 0 consultant1  
[student@serverb ~]$ sudo chage -d 0 consultant2  
[student@serverb ~]$ sudo chage -d 0 consultant3
```

8.2. 从 **serverb** 注销。

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

评估

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

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

完成

在 **workstation** 上，运行 **lab users-review finish** 脚本来完成本实验。此脚本将删除在实验过程中创建的用户帐户和文件，以确保环境清理干净。

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

本实验到此结束。

总结

在本章中，您学到了：

- 用户帐户有三种主要类型：超级用户、系统用户和普通用户。
- 用户必须有主要组，并且可以是一个或多个补充组的成员。
- 包含用户和组信息的三个关键文件是 **/etc/passwd**、**/etc/group** 和 **/etc/shadow**。
- **su** 和 **sudo** 命令可用于以超级用户身份运行命令。
- **useradd**、**usermod** 和 **userdel** 命令可用于管理用户。
- **groupadd**、**groupmod** 和 **groupdel** 命令可用于管理组。
- **chage** 命令可用于配置和查看用户的密码到期设置。

章 7

控制对文件的访问

目标

设置文件的 Linux 文件系统权限，并解释不同权限设置的安全效果。

培训目标

- 列出文件和目录的文件系统权限，并解释这些权限对用户和组访问权限的影响。
- 利用命令行工具更改文件的权限和所有权。
- 控制用户创建的新文件的默认权限，解释特殊权限的影响，并使用特殊权限和默认权限设置在特定目录中创建的文件的组所有者。

章节

- 解释 Linux 文件系统权限（及测验）
- 从命令行管理文件系统权限（及引导式练习）
- 管理默认权限和文件访问（及引导式练习）

实验

控制对文件的访问

解释 LINUX 文件系统权限

培训目标

学完本节后，您应能够列出文件和目录的文件系统权限，并解释这些权限对用户和组访问权限的影响。

LINUX 文件系统权限

文件权限控制对文件的访问。Linux 文件权限简单而又灵活，易于理解和应用，又仍然可以轻松地处理最常见的权限情况。

文件具有三个应用权限的用户类别。文件归用户所有，通常是创建文件的用户。文件还归单个组所有，通常是创建该文件的主要用户组所有，但是可以进行更改。可以为所属用户、所属组和系统上的非用户和非所属组成员的其他用户设置不同的权限。

最具体的权限具有优先权。用户权限覆盖组权限，后者又覆盖其他权限。

在图 7.1 中，`joshua` 是组 `joshua` 和 `web` 的成员，而 `allison` 则是 `allison`、`wheel` 和 `web` 的成员。当 `joshua` 和 `allison` 需要协作时，文件应当与 `web` 组关联，并且组权限应当允许所需的访问。

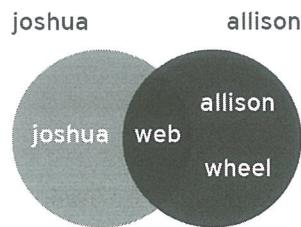


图 7.1: 可促进协作的组成员资格示例

有三种权限类别可应用：读取、写入和执行。下表说明了这些权限如何影响对文件和目录的访问。

权限对文件和目录的影响

| 权限 | 对文件的影响 | 对目录的影响 |
|---------------|-------------|--|
| r (读取) | 可以读取文件内容。 | 可以列出目录的内容（文件名）。 |
| w (写入) | 可以更改文件内容。 | 可以创建或删除目录中的任一文件。 |
| x (执行) | 可以作为命令执行文件。 | 目录可以成为当前工作目录。（您可以 <code>cd</code> 它，但还需要读取权限才能列出里面的文件） |

用户通常对只读目录具有读取和执行权限，因此他们可以列出该目录，并且对其内容具有完整的只读访问权限。如果用户仅对某目录具有读取访问权限，可以列出其中文件的名称，但是其他信息（包括权限或时间戳）都不可用，也不可访问。如果用户仅对目录具有执行权限，则无法列出目录中的文件名。如果他们知道自己有读取权限的文件的名称，可通过显式指定相对文件名，以便从目录外部访问该文件的内容。

在文件所在的目录中拥有所有权或写入权限的任何人都可以删除此文件，不论此文件本身的所有权或权限如何。（可以通过特殊权限粘滞位将其覆盖，本章后续部分将对此进行讨论。）



注意

Linux 文件权限的工作方式与 Microsoft Windows NTFS 文件系统所用的权限系统有所不同。

在 Linux 上，权限仅适用于设置它们的文件或目录。也就是说，目录中的子目录和文件不会自动继承目录的权限。但是，目录的权限可以阻止访问该目录的内容，具体取决于它们的限制程度。

Linux 中的目录的 **read** 权限大致等同于 Windows 中的 **List folder contents**。

Linux 中的目录的 **write** 权限等同于 Windows 中的 **Modify**；意味着可以删除文件和子目录。在 Linux 中，如果对某目录同时设置了 **write** 和 **sticky bit**，则只有文件或子目录的所有者可以删除该目录，这与 Windows 的 **Write** 权限行为相似。

Linux root 用户对所有文件具有与 Windows **Full Control** 等同的权限。但是，root 的访问权限可能会受到使用进程和文件安全性上下文的系统 SELinux 策略的限制。SELinux 将在后续课程中予以讨论。

查看文件和目录的权限及所有权

ls 命令的 **-l** 选项可显示有关权限和所有权的详细信息：

```
[user@host~]$ ls -l test
-rw-rw-r--. 1 student student 0 Feb  8 17:36 test
```

使用 **-d** 选项可显示有关目录本身（而非其内容）的详细信息。

```
[user@host ~]$ ls -ld /home
drwxr-xr-x. 5 root root 4096 Jan 31 22:00 /home
```

长列表的第一个字符表示文件类型，解译如下：

- **-** 是常规文件。
- **d** 是目录。
- **l** 是软链接。
- 其他字符代表硬件设备（**b** 和 **c**）或其他具有特殊用途的文件（**p** 和 **s**）。

接下来的九个字符是文件权限。它们分为三组，每组三个字符，分别对应：应用于拥有该文件的用户的权限、应用于拥有该文件的组的权限，以及应用于其他所有用户的权限。如果组中显示 **rwx**，说明该类别具有读取、写入和执行三种权限。如果其中一个字母被替换为 **-**，则表示该类别没有这个权限。

在链接数之后，第一个名称指定拥有该文件的用户，第二个名称指定拥有该文件的组。

因此，在上面的示例中，用户 **student** 的权限由第一组三个字符指定。用户 **student** 对 **test** 具有读写权限，但不具有执行权限。

组 **student** 的权限由第二组三个字符指定：它也对 **test** 具有读写权限，但不具有执行权限。

其他任何用户的权限由第三组三个字符指定：它们仅对 **test** 具有读取权限。

应用最具体的权限组。因此，如果用户 `student` 与组 `student` 具有不同的权限，而用户 `student` 也是该组的成员，那么用户权限将是最终应用的权限。

权限影响示例

以下示例将帮助说明文件权限的影响。针对这些示例，我们假定有四个具有以下组成员身份的用户：

| 用户 | 组成员资格 |
|-------------|------------------------|
| operator1 | operator1, consultant1 |
| database1 | database1, consultant1 |
| database2 | database2, operator2 |
| contractor1 | contractor1, operator2 |

这些用户将使用 `dir` 目录中的文件。这是该目录中文件的长列表：

```
[database1@host dir]$ ls -la
total 24
drwxrwxr-x.  2 database1 consultant1  4096 Apr  4 10:23 .
drwxr-xr-x. 10 root      root       4096 Apr  1 17:34 ..
-rw-rw-r--.  1 operator1 operator1    1024 Apr  4 11:02 lfile1
-rw-r--rw-.  1 operator1 consultant1  3144 Apr  4 11:02 lfile2
-rw-rw-r--.  1 database1 consultant1 10234 Apr  4 10:14 rfile1
-rw-r-----. 1 database1 consultant1   2048 Apr  4 10:18 rfile2
```

`-a` 选项可显示隐藏文件的权限，包括用于表示目录及其父目录的特殊文件。在本例中，`.` 反映了 `dir` 本身的权限，`..` 则反映了其父目录的权限。

`rfile1` 具有哪些权限？拥有该文件的用户 (`database1`) 具有读写权限，但不具有执行权限。拥有该文件的组 (`consultant1`) 具有读写权限，但不具有执行权限。其他所有用户具有读取权限，但不具有写入或执行权限。

下表探讨了这组权限对这些用户的一些影响：

| 影响 | 为什么会这样？ |
|---|--|
| 用户 <code>operator1</code> 可以更改 <code>rfile1</code> 的内容。 | 用户 <code>operator1</code> 是 <code>consultant1</code> 组的成员，并且该组对 <code>rfile1</code> 具有读取和写入权限。 |
| 用户 <code>database1</code> 可以查看和修改 <code>rfile2</code> 的内容。 | 用户 <code>database1</code> 拥有 <code>rfile2</code> 文件，对其具有读取和写入访问权限。 |
| 用户 <code>operator1</code> 可以查看但不能修改 <code>rfile2</code> 的内容（不会删除和重新创建）。 | 用户 <code>operator1</code> 是 <code>consultant1</code> 组的成员，并且该组对 <code>rfile2</code> 具有读取访问权限。 |
| 用户 <code>database2</code> 和 <code>contractor1</code> 对 <code>rfile2</code> 的内容没有任何访问权限。 | 其他权限适用于用户 <code>database2</code> 和 <code>contractor1</code> ，并且这些权限不包括读取或写入权限。 |

| 影响 | 为什么会这样？ |
|---|--|
| operator1 是唯一可以更改 lfile1 内容的用户（不会删除和重新创建）。 | 用户和组 operator1 对文件具有写入权限，而其他用户没有该权限。但是，组 operator1 的唯一成员是用户 operator1。 |
| 用户 database2 可以更改 lfile2 的内容。 | 用户 database2 不是拥有该文件的用户，也不在组 consultant1 中，因此将应用其他权限。这样将会授予写入权限。 |
| 用户 database1 可以查看 lfile2 的内容，但不能修改 lfile2 的内容（不会删除和重新创建）。 | 用户 database1 是组 consultant1 的成员，并且该组仅对 lfile2 具有读取权限。即使其他具有写入权限，组权限具有优先权。 |
| 用户 database1 可以删除 lfile1 和 lfile2 。 | 用户 database1 对包含这两个文件的目录具有写入权限（如 . 所示），因此可以删除该目录中的任何文件。即使 database1 对文件本身没有写入权限也是如此。 |



参考文献

ls(1) man page

info coreutils (GNU Coreutils)

· 第 13 节：更改文件属性

► 小测验

解释 LINUX 文件系统权限

查看以下信息并使用它来回答测验问题。

系统有四个用户被分配给以下组：

- 用户 **consultant1** 位于组 **consultant1** 和 **database1** 中
- 用户 **operator1** 位于组 **operator1** 和 **database1** 中
- 用户 **contractor1** 位于组 **contractor1** 和 **contractor3** 中
- 用户 **operator2** 位于组 **operator2** 和 **contractor3** 中

当前目录(.) 包含四个具有以下权限信息的文件：

```
drwxrwxr-x. operator1 database1 .
-rw-rw-r--. consultant1 consultant1 lfile1
-rw-r--rw-. consultant1 database1 lfile2
-rw-rw-r--. operator1 database1 rfile1
-rw-r-----. operator1 database1 rfile2
```

► 1. 哪个常规文件归 **operator1** 所有并可被所有用户读取？

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 2. 哪个文件可以被 **contractor1** 用户修改？

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 3. 哪个文件无法被 **operator2** 用户读取？

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 4. 哪个文件的组所有者为 **consultant1**?

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 5. 哪些文件可以被 **operator1** 用户删除?

- a. **rfile1**
- b. **rfile2**
- c. 以上都对。
- d. 以上都不对。

► 6. 哪些文件可以被 **operator2** 用户删除?

- a. **lfile1**
- b. **lfile2**
- c. 以上都对。
- d. 以上都不对。

► 解决方案

解释 LINUX 文件系统权限

查看以下信息并使用它来回答测验问题。

系统有四个用户被分配给以下组：

- 用户 **consultant1** 位于组 **consultant1** 和 **database1** 中
- 用户 **operator1** 位于组 **operator1** 和 **database1** 中
- 用户 **contractor1** 位于组 **contractor1** 和 **contractor3** 中
- 用户 **operator2** 位于组 **operator2** 和 **contractor3** 中

当前目录(.) 包含四个具有以下权限信息的文件：

```
drwxrwxr-x. operator1 database1 .
-rw-rw-r--. consultant1 consultant1 lfile1
-rw-r--rw-. consultant1 database1 lfile2
-rw-rw-r--. operator1 database1 rfile1
-rw-r-----. operator1 database1 rfile2
```

► 1. 哪个常规文件归 **operator1** 所有并可被所有用户读取？

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 2. 哪个文件可以被 **contractor1** 用户修改？

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 3. 哪个文件无法被 **operator2** 用户读取？

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 4. 哪个文件的组所有者为 **consultant1**?

- a. **lfile1**
- b. **lfile2**
- c. **rfile1**
- d. **rfile2**

► 5. 哪些文件可以被 **operator1** 用户删除?

- a. **rfile1**
- b. **rfile2**
- c. 以上都对。
- d. 以上都不对。

► 6. 哪些文件可以被 **operator2** 用户删除?

- a. **lfile1**
- b. **lfile2**
- c. 以上都对。
- d. 以上都不对。

从命令行管理文件系统权限

培训目标

学完本节后，您应能够使用命令行工具更改文件的权限和所有权。

更改文件和目录权限

用于从命令行更改权限的命令为 **chmod**，意为“change mode”（更改模式）（权限也称为文件的模式）。**chmod** 命令在要更改的文件或目录列表后面列出了权限说明。可使用符号（符号法）或数值（数值法）来发布此权限说明。

通过符号法更改权限

chmod WhoWhatWhich file|directory

- Who 是指 u、g、o、a（代表用户、组、其他、全部）
- What 是指 +、-、=（代表添加、删除、精确设置）
- Which 是指 r、w、x（代表读取、写入、执行）

更改文件权限的符号法使用字母代表不同的权限组：**u** 表示用户，**g** 表示组，**o** 表示其他，**a** 表示全部。

使用符号法时，不需要设置一组全新的权限。取而代之，您可以更改现有的一个或多个权限。使用 **+** 或 **-** 来分别添加或删除权限，或者使用 **=** 来替换一组权限的整个集合。

权限自身由单个字母来表示：**r** 表示读取，**w** 表示写入，**x** 表示执行。在使用 **chmod** 通过符号法来更改权限时，仅当文件是目录或者已经为用户、组或其他人设置了执行权限时，使用大写的 **X** 作为权限标志才会添加执行权限。



注意

chmod 命令支持 **-R** 选项以递归方式对整个目录树中的文件设置权限。在使用 **-R** 选项时，使用 **X** 选项以符号形式设置权限会非常有用。这将能够对目录设置执行（搜索）权限，以便在不更改大部分文件权限的情况下，访问这些目录的内容。不过，使用 **X** 选项时要谨慎，因为如果某个文件设置有任何执行权限，则 **X** 也将会对该文件设置指定的执行权限。例如，以下命令会以递归方式为组所有者设置对 **demodir** 及其所有子代的读、写访问权限，但将仅向已为用户、组或其他人设置了执行权限的目录和文件应用组执行权限。

```
[root@host opt]# chmod -R g+rwx demodir
```

示例

- 对 **file1** 中的组和其他，删除读取和写入权限：

```
[user@host ~]$ chmod go-rw file1
```

- 对 **file2** 中的每个人添加执行权限。

```
[user@host ~]$ chmod a+x file2
```

通过数值法更改权限

在下例中，#字符代表一个数字。

```
chmod ### file|directory
```

- 每个数字代表一个访问级别的权限：用户、组、其他。
- 数字的计算方法是：将所要添加的每个权限的数值加在一起，其中 4 代表读取，2 代表写入，1 代表执行。

如果使用数值法，权限由三位（或在设置高级权限时为四位）八进制数来表示。单个八进制数字可以表示 0-7 的任何单个值。

在权限的三位八进制（数值）表示法中，每个数字代表一个访问级别，从左至右为：用户、组和其他。要确定每个数字：

1. 从 0 开始。
2. 如果该访问级别应该具有读取权限，请加 4。
3. 如果应该具有写入权限，请加 2。
4. 如果应该具有执行权限，请加 1。

检查权限 **-rwxr-x---**。对于用户，**rwx** 计算为 $4+2+1=7$ 。对于组，**r-x** 计算为 $4+0+1=5$ ，对于其他用户 **---** 表示为 0。将这三个放在一起，这些权限的数值法表示为 750。

这一计算也可按相反的方向执行。我们来看一下权限 640。对于用户权限，6 表示读取 (4) 和写入 (2)，其显示为 **rw-**。对于组部分，4 仅包含读取 (4)，显示为 **r--**。对于其他用户使用的 0 表示无权限 (**---**)，因此这一文件最终的符号权限集为 **-rw-r-----**。

有经验的管理员通常使用数值权限，因为它们的输入和发音会比较短，而仍能完全控制所有权限。

示例

- 对用户设置读取和写入权限，对组设置读取权限，对 **samplefile** 中的其他设置读取权限：

```
[user@host ~]$ chmod 644 samplefile
```

- 对用户设置读取、写入和执行权限，对组设置读取和执行权限，而对 **sampledir** 中的其他则设置无权限：

```
[user@host ~]$ chmod 750 sampledir
```

更改文件和目录的用户或组所有权

新建的文件由创建该文件的用户所有。默认情况下，新文件的组所有权为创建该文件的主要用户组。在红帽企业 Linux 中，用户的主要组通常为仅有该用户作为成员的私有组。要根据组成员资格授予对文件的访问权限，可能需要更改拥有该文件的组。

只有 **root** 用户可以更改拥有文件的用户。不过，组所有权可以由 **root** 用户或文件的所有者来设置。**root** 用户可将文件所有权授予任何组，而普通用户仅可将文件所有权授予他们所属的组。

使用 **chown**（更改所有者）命令可更改文件所有权。例如，要将文件 **test_file** 的所有权授予 **student** 用户，可使用以下命令：

```
[root@host ~]# chown student test_file
```

chown 可与 **-R** 选项配合使用，以递归更改整个目录树的所有权。以下命令可将 **test_dir** 的所有权以及此目录树中的所有文件和子目录授予给 **student**：

```
[root@host ~]# chown -R student test_dir
```

chown 命令也可用于更改文件的组所有权，只需在组名称之前加上冒号 (**:**)。例如，以下命令将 **test_dir** 组更改为 **admins**：

```
[root@host ~]# chown :admins test_dir
```

chown 命令也可用于同时更改所有者和组，此时可使用语法 **owner:group**。例如，要将 **test_dir** 的所有权更改为 **visitor**，同时将组更改为 **guests**，可使用以下命令：

```
[root@host ~]# chown visitor:guests test_dir
```

有些用户并不使用 **chown**，而是使用 **chgrp** 命令来更改组所有权。该命令的作用与 **chown** 类似，不同之处在于它仅用于更改组所有权，而且不需要组名前的冒号 (**:**)。



重要

您可能会遇到 **chown** 命令使用替代语法的示例，该语法用句点而不是冒号分隔所有者和组：

```
[root@host ~]# chown owner.group filename
```

您不应该使用此语法。应始终使用冒号。

句点是用户名中的有效字符，但冒号不是。如果系统上存在用户 **enoch.root**、用户 **enoch** 和组 **root**，则 **chown enoch.root filename** 的结果是 **filename** 归用户 **enoch.root** 所有。您可能是在尝试将文件所有权设置为用户 **enoch** 和组 **root**。这可能会造成混淆。

如果您在同时设置用户和组时始终使用 **chown** 冒号语法，其结果总是能够轻松预测。



参考文献

ls(1)、**chmod(1)**、**chown(1)** 和 **chgrp(1)** man page