

► 指导练习

从命令行管理文件系统权限

在本练习中，您将使用文件系统权限来创建一个目录，其中特定组的所有成员都可以添加和删除文件。

成果

您应该能够创建可由特定组的所有成员访问的协作目录。

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 perms-cli start** 命令。此起始脚本将创建一个名为 consultants 的组，以及名为 consultant1 和 consultant2 的两个用户。

```
[student@workstation ~]$ lab perms-cli start
```

- 1. 从 workstation，使用 ssh 命令以 student 用户身份登录 servera。

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

- 2. 将 redhat 用作密码，切换到 root 用户。

```
[student@servera ~]$ su -  
Password: redhat  
[root@servera ~]#
```

- 3. 使用 mkdir 命令来创建 /home/consultants 目录。

```
[root@servera ~]# mkdir /home/consultants
```

- 4. 使用 chown 命令，将 consultants 目录的组所有权更改给 consultants。

```
[root@servera ~]# chown :consultants /home/consultants
```

- 5. 确保 consultants 组的权限允许组成员在 /home/consultants 目录中创建和删除文件。这些权限应禁止其他人访问文件。

- 5.1. 使用 ls 命令，确认 consultants 组的权限允许组成员在 /home/consultants 目录中创建和删除文件。

```
[root@servera ~]# ls -ld /home/consultants  
drwxr-xr-x. 2 root consultants 6 Feb 1 12:08 /home/consultants
```

注意 **consultants** 组当前没有写入权限。

5.2. 使用 **chmod** 命令，为 **consultants** 组添加写入权限。

```
[root@servera ~]# chmod g+w /home/consultants  
[root@servera ~]# ls -ld /home/consultants  
drwxrwxr-x. 2 root consultants 6 Feb 1 13:21 /home/consultants
```

5.3. 使用 **chmod** 命令，禁止其他人访问 **/home/consultants** 目录中的文件。

```
[root@servera ~]# chmod 770 /home/consultants  
[root@servera ~]# ls -ld /home/consultants  
drwxrwx---. 2 root consultants 6 Feb 1 12:08 /home/consultants/
```

► 6. 退出 root shell 并切换到 **consultant1** 用户。密码是 **redhat**。

```
[root@servera ~]# exit  
logout  
[student@servera ~]$  
[student@servera ~]$ su - consultant1  
Password: redhat
```

► 7. 前往 **/home/consultants** 目录，再创建一个名为 **consultant1.txt** 的文件。

7.1. 使用 **cd** 命令更改到 **/home/consultants** 目录。

```
[consultant1@servera ~]$ cd /home/consultants
```

7.2. 使用 **touch** 命令，创建一个名为 **consultant1.txt** 的空文件。

```
[consultant1@servera consultants]$ touch consultant1.txt
```

► 8. 使用 **ls -l** 命令，列出新文件的默认用户和组所有权及其权限。

```
[consultant1@servera consultants]$ ls -l consultant1.txt  
-rw-rw-r--. 1 consultant1 consultant1 0 Feb 1 12:53 consultant1.txt
```

► 9. 确保 **consultants** 组的所有成员都可以编辑 **consultant1.txt** 文件。将 **consultant1.txt** 文件的组所有权更改给 **consultants**。

9.1. 使用 **chown** 命令，将 **consultant1.txt** 文件的组所有权更改给 **consultants**。

```
[consultant1@servera consultants]$ chown :consultants consultant1.txt
```

9.2. 使用 **ls** 命令及 **-l** 选项，列出 **consultant1.txt** 文件的新所有权。

```
[consultant1@servera consultants]$ ls -l consultant1.txt  
-rw-rw-r--. 1 consultant1 consultants 0 Feb 1 12:53 consultant1.txt
```

- 10. 退出 shell 并切换到 consultant2 用户。密码是 redhat。

```
[consultant1@servera consultants]$ exit  
logout  
[student@servera ~]$ su - consultant2  
Password: redhat  
[consultant2@servera ~]$
```

- 11. 前往 /home/consultants 目录。确保 consultant2 用户可以向 **consultant1.txt** 文件添加内容。退出 shell。

11.1. 使用 **cd** 命令更改到 /home/consultants 目录。使用 **echo** 命令添加 **text** 到 **consultant1.txt** 文件中。

```
[consultant2@servera ~]$ cd /home/consultants/  
[consultant2@servera consultants]$ echo "text" >> consultant1.txt  
[consultant2@servera consultants]$
```

11.2. 使用 **cat** 命令验证该文本是否已添加到 **consultant1.txt** 文件中。

```
[consultant2@servera consultants]$ cat consultant1.txt  
text  
[consultant2@servera consultants]$
```

11.3. 退出 shell。

```
[consultant2@servera consultants]$ exit  
logout  
[student@servera ~]$
```

- 12. 从 servera 注销。

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

完成

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

```
[student@workstation ~]$ lab perms-cli finish
```

本引导式练习到此结束。

管理默认权限和文件访问

培训目标

学完本节后，学员应能够：

- 控制用户创建的新文件的默认权限。
- 解释特殊权限的影响。
- 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。

特殊权限

特殊权限构成了除了基本用户、组和其他类型之外的第四种权限类型。顾名思义，这些权限提供了额外的访问相关功能，超出了基本权限类型允许的范畴。本节详细介绍了特殊权限的影响，具体参见下表中的总结。

特殊权限对文件和目录的影响

特殊权限	对文件的影响	对目录的影响
u+s (suid)	以拥有文件的用户身份，而不是以运行文件的用户身份执行文件。	无影响。
g+s (sgid)	以拥有文件的组身份执行文件。	在目录中最新创建的文件将其组所有者设置为与目录的组所有者相匹配。
o+t (sticky)	无影响。	对目录具有写入访问权限的用户仅可以删除其所拥有的文件，而无法删除或强制保存到其他用户所拥有的文件。

对可执行文件的 setuid 权限表示将以拥有该文件的用户的身份运行命令，而不是以运行命令的用户身份。以 **passwd** 命令为例：

```
[user@host ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 35504 Jul 16 2010 /usr/bin/passwd
```

在长列表中，您可以通过小写的 **s** 辨别出 setuid 权限，该处通常是 **x**（所有者执行权限）。如果所有者不具有执行权限，这将由大写的 **S** 取代。

对某目录的特殊权限 setgid 表示在该目录中创建的文件将继承该目录的组所有权，而不是继承自创建用户。这通常用于组协作目录，将文件从默认的专有组自动更改为共享组，或者当目录中的文件始终都应由特定的组所有时使用。**/run/log/journal** 目录就是这样的一个示例：

```
[user@host ~]$ ls -ld /run/log/journal
drwxr-sr-x. 3 root systemd-journal 60 May 18 09:15 /run/log/journal
```

如果对可执行文件设置了 setgid，则命令以拥有该文件的组运行，而不是以运行命令的用户身份运行，其方式与 setuid 类似。以 **locate** 命令为例：

```
[user@host ~]$ ls -ld /usr/bin/locate  
-rwx--s--x. 1 root slocate 47128 Aug 12 17:17 /usr/bin/locate
```

在长列表中，您可以通过小写的 **s** 辨别出 setgid 权限，该处通常是 **x**（组执行权限）。如果组不具有执行权限，这将会由大写 **S** 取代。

最后，针对目录的粘滞位将对文件删除设置特殊限制。只有文件的所有者（及 **root**）才能删除该目录中的文件。例如，**/tmp**：

```
[user@host ~]$ ls -ld /tmp  
drwxrwxrwt. 39 root root 4096 Feb 8 20:52 /tmp
```

在长列表中，您可以通过小写的 **t** 辨别出 sticky 权限，该处通常是 **x**（其他执行权限）。如果其他不具有执行权限，这将会由大写 **T** 取代。

设置特殊权限

- 用符号表示：setuid = **u+s**; setgid = **g+s**; sticky = **o+t**
- 用数值表示（第四位）：setuid = 4; setgid = 2; sticky = 1

示例

- 在 **directory** 目录上添加 setgid 位：

```
[user@host ~]# chmod g+s directory
```

- 为 **directory** 设置 setgid 位，并且为用户和组添加读取/写入/执行权限，但其他不具有任何访问权限：

```
[user@host ~]# chmod 2770 directory
```

默认文件权限

创建新文件或目录时，会为其分配初始权限。有两个因素会影响这些初始权限。首先是您要创建常规文件还是目录。其次是当前的 umask。

如果是创建新目录，操作系统首先会为其分配八进制权限 0777 (**drwxrwxrwx**)。如果是创建新的常规文件，操作系统则为其分配八进制权限 0666 (**-rw-rw-rw-**)。必须始终为常规文件显式添加执行权限。这样攻击者就难以破坏网络服务，从而可以创建新文件并立即作为程序执行。

不过，shell 会话还会设置一个 umask，以进一步限制初始设置的权限。这是一个八进制位掩码，用于清除由该进程创建的新文件和目录的权限。如果在 umask 中设置了一个位，则新文件中的对应的权限将被清除。例如，umask 0002 可清除其他用户的写入位。前导零表示特殊的用户和组权限未被清除。umask 为 0077 时，清除新创建文件的所有组和其他权限。

不带参数运行 **umask** 命令将显示 shell 的 umask 的当前值：

```
[user@host ~]$ umask  
0002
```

通过一个数字参数使用 **umask** 命令，可以更改当前 shell 的 umask。该数字参数应当是与新 umask 值对应的八进制值。umask 中的任何前导零均可以省略。

Bash shell 用户的系统默认 umask 值在 **/etc/profile** 和 **/etc/bashrc** 文件中定义。用户可以在其主目录的 **.bash_profile** 和 **.bashrc** 文件中覆盖系统默认值。

umask 示例

以下示例说明了 umask 如何影响文件和目录的权限。查看当前 shell 中文件和目录的默认 umask 权限。所有者和组都对文件具有读写权限，其他设置为读取。所有者和组都对目录具有读取、写入和执行权限。其他的唯一权限是读取。

```
[user@host ~]$ umask  
0002  
[user@host ~]$ touch default  
[user@host ~]$ ls -l default.txt  
-rw-rw-r--. 1 user user 0 May  9 01:54 default.txt  
[user@host ~]$ mkdir default  
[user@host ~]$ ls -ld default  
drwxrwxr-x. 2 user user 0 May  9 01:54 default
```

通过将 umask 值设置为 0，其他的文件权限将从读取改为读取和写入。其他的目录权限将从读取和执行改为读取、写入和执行。

```
[user@host ~]$ umask 0  
[user@host ~]$ touch zero.txt  
[user@host ~]$ ls -l zero.txt  
-rw-rw-rw-. 1 user user 0 May  9 01:54 zero.txt  
[user@host ~]$ mkdir zero  
[user@host ~]$ ls -ld zero  
drwxrwxrwx. 2 user user 0 May  9 01:54 zero
```

要屏蔽其他的所有文件和目录权限，请将 umask 值设置为 007。

```
[user@host ~]$ umask 007  
[user@host ~]$ touch seven.txt  
[user@host ~]$ ls -l seven.txt  
-rw-rw----. 1 user user 0 May  9 01:55 seven.txt  
[user@host ~]$ mkdir seven  
[user@host ~]$ ls -ld seven  
drwxrwx---. 2 user user 0 May  9 01:54 seven
```

umask 为 027 可确保新文件的用户具有读写权限，并且组具有读取权限。新目录的组具有读取和写入权限，而其他则没有访问权限。

```
[user@host ~]$ umask 027
[user@host ~]$ touch two-seven.txt
[user@host ~]$ ls -l two-seven.txt
-rw-r----. 1 user user 0 May  9 01:55 two-seven.txt
[user@host ~]$ mkdir two-seven
[user@host ~]$ ls -ld two-seven
drwxr-x---. 2 user user 0 May  9 01:54 two-seven
```

用户的默认 umask 由 shell 启动脚本设置。默认情况下，如果您帐户的 UID 为 200 或以上，并且您的用户名和主要组名相同，就会向您分配一个值为 002 的 umask。否则，您的 umask 将为 022。

作为 root 用户，您可以通过添加名为 **/etc/profile.d/local-umask.sh** 的 shell 启动脚本来更改此设置，该 shell 启动脚本的输出如下例所示：

```
[root@host ~]# cat /etc/profile.d/local-umask.sh
# Overrides default umask configuration
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

对于 UID 大于 199 且用户名和主要组名相匹配的用户，上面的示例会将 umask 设为 007，其他人的则设为 022。如果只想将每个人的 umask 都设为 022，可以仅使用以下内容创建该文件：

```
# Overrides default umask configuration
umask 022
```

要确保全局 umask 更改生效，您必须注销 shell 并重新登录。在此之前，当前 shell 中配置的 umask 仍然有效。



参考文献

bash(1)、**ls(1)**、**chmod(1)** 和 **umask(1)** man page

► 指导练习

管理默认权限和文件访问

在本练习中，您将使用 umask 设置和 setgid 权限来控制在目录中创建的新文件的权限。

成果

您应能够：

- 创建一个共享目录，其中新文件将自动由 operators 组拥有。
- 试验各种 umask 设置。
- 调整特定用户的默认权限。
- 确认您的调整正确无误。

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 perms-default start** 命令。该命令将运行一个起始脚本，它将确定 servera 是否可从网络访问。该脚本还会在 servera 上创建 operators 组和 operator1 用户。

```
[student@workstation ~]$ lab perms-default start
```

- 1. 使用 ssh 命令，以 student 用户身份登录 servera。

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

- 2. 使用 su 命令，并将 redhat 用作密码，切换到 operator1 用户。

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

- 3. 使用 umask 命令，列出 operator1 用户的默认 umask 值。

```
[operator1@servera ~]$ umask  
0002
```

- 4. 创建一个名为 /tmp/shared 的新目录。在 /tmp/shared 目录中，创建一个名为 defaults 的文件。查看其默认权限。

4.1. 使用 mkdir 命令来创建 /tmp/shared 目录。使用 ls -l 命令列出新目录的权限。

```
[operator1@servera ~]$ mkdir /tmp/shared  
[operator1@servera ~]$ ls -ld /tmp/shared  
drwxrwxr-x. 2 operator1 operator1 6 Feb 4 14:06 /tmp/shared
```

4.2. 使用 **touch** 命令，在 **/tmp/shared** 目录中创建一个名为 **defaults** 的文件。

```
[operator1@servera ~]$ touch /tmp/shared/defaults
```

4.3. 使用 **ls -l** 命令列出新文件的权限。

```
[operator1@servera ~]$ ls -l /tmp/shared/defaults  
-rw-rw-r--. 1 operator1 operator1 0 Feb 4 14:09 /tmp/shared/defaults
```

► 5. 将 **/tmp/shared** 的组所有权更改为 **operators**。确认新的所有权和权限。

5.1. 使用 **chown** 命令，将 **/tmp/shared** 目录的组所有权更改为 **operators**。

```
[operator1@servera ~]$ chown :operators /tmp/shared
```

5.2. 使用 **ls -ld** 命令列出 **/tmp/shared** 目录的权限。

```
[operator1@servera ~]$ ls -ld /tmp/shared  
drwxrwxr-x. 2 operator1 operators 22 Feb 4 14:09 /tmp/shared
```

5.3. 使用 **touch** 命令，在 **/tmp/shared** 目录中创建一个名为 **group** 的文件。使用 **ls -l** 命令列出其文件权限。

```
[operator1@servera ~]$ touch /tmp/shared/group  
[operator1@servera ~]$ ls -l /tmp/shared/group  
-rw-rw-r--. 1 operator1 operator1 0 Feb 4 17:00 /tmp/shared/group
```



注意

/tmp/shared/group 文件的组所有者不是 **operators**，而是 **operator1**。

► 6. 确保在 **/tmp/shared** 目录中创建的文件归 **operators** 组所有。

6.1. 使用 **chmod** 命令，将 **/tmp/shared** 目录的组 ID 设置为 **operators** 组。

```
[operator1@servera ~]$ chmod g+s /tmp/shared
```

6.2. 使用 **touch** 命令，在 **/tmp/shared** 目录中创建一个名为 **operations_database.txt** 的新文件。

```
[operator1@servera ~]$ touch /tmp/shared/operations_database.txt
```

6.3. 使用 **ls -l** 命令，验证 **operators** 组是新文件的所有者。

```
[operator1@servera ~]$ ls -l /tmp/shared/operations_database.txt  
-rw-rw-r--. 1 operator1 operators 0 Feb 4 16:11 /tmp/shared/  
operations_database.txt
```

- 7. 在 **/tmp/shared** 目录中创建一个新文件，取名为 **operations_network.txt**。记录其所有权和权限。更改 **operator1** 的 umask。创建一个名为 **operations_production.txt** 的新文件。记录 **operations_production.txt** 文件的所有权和权限。

- 7.1. 使用 **echo** 命令，在 **/tmp/shared** 目录中创建一个名为 **operations_network.txt** 的文件。

```
[operator1@servera ~]$ echo text >> /tmp/shared/operations_network.txt
```

- 7.2. 使用 **ls -l** 命令，列出 **operations_network.txt** 文件的权限。

```
[operator1@servera ~]$ ls -l /tmp/shared/operations_network.txt  
-rw-rw-r--. 1 operator1 operators 5 Feb 4 15:43 /tmp/shared/  
operations_network.txt
```

- 7.3. 使用 **umask** 命令，将 **operator1** 用户的 umask 更改为 027。使用 **umask** 命令确认更改。

```
[operator1@servera ~]$ umask 027  
[operator1@servera ~]$ umask  
0027
```

- 7.4. 使用 **touch** 命令，在 **/tmp/shared/** 目录中创建一个名为 **operations_production.txt** 的新文件。使用 **ls -l** 命令，确保在创建新创建的文件时使 **operators** 组拥有只读访问权限，其他用户没有访问权限。

```
[operator1@servera ~]$ touch /tmp/shared/operations_production.txt  
[operator1@servera ~]$ ls -l /tmp/shared/operations_production.txt  
-rw-r-----. 1 operator1 operators 0 Feb 4 15:56 /tmp/shared/  
operations_production.txt
```

- 8. 打开一个新的终端窗口，并以 **operator1** 用户身份登录 **servera**。

```
[student@workstation ~]$ ssh operator1@servera  
...output omitted...  
[operator1@servera ~]$
```

- 9. 列出 **operator1** 的 umask 值。

```
[operator1@servera ~]$ umask  
0002
```

- 10. 更改 **operator1** 用户的默认 umask。新 umask 将阻止不属于其组的用户的访问权限。确认 umask 已被更改。

10.1. 使用 **echo** 命令，将 **operator1** 用户的 umask 更改为 007。

```
[operator1@servera ~]$ echo "umask 007" >> ~/.bashrc
[operator1@servera ~]$ cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
umask 007
```

10.2. 注销，然后重新以 **operator1** 用户身份登录。使用 **umask** 命令确认更改是否持久。

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh operator1@servera
...output omitted...
[operator1@servera ~]$ umask
0007
```

► 11. 在 **servera** 上，从所有 **operator1** 和 **student** 用户的 shell 退出。



警告

退出 **operator1** 打开的所有 shell。不从所有 shell 退出将导致完成脚本运行失败。

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

完成

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

```
[student@workstation ~]$ lab perms-default finish
```

本引导式练习到此结束。

▶ 开放研究实验

控制对文件的访问

任务执行清单

在本实验中，您将配置文件权限并设置一个目录，使得特定组中的用户可以使用该目录方便地共享本地文件系统上的文件。

成果

您应能够：

- 创建一个目录，让用户能够在该目录中协作处理文件。
- 创建自动分配组所有权的文件。
- 创建在组外无法访问的文件。

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 perms-review start` 命令。该命令将运行一个起始脚本，它将确定 `serverb` 是否可从网络访问。该脚本还会创建 `techdocs` 组，以及名为 `tech1`、`tech2` 和 `database1` 的三个用户。

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

1. 使用 `ssh` 命令，以 `student` 用户身份登录 `serverb`。将 `redhat` 用作密码，在 `serverb` 上切换到 `root`。
2. 创建名为 `/home/techdocs` 的目录。
3. 将 `/home/techdocs` 目录的组所有权更改为 `techdocs` 组。
4. 验证 `techdocs` 组中的用户是否能够在 `/home/techdocs` 目录中创建和编辑文件。
5. 设置 `/home/techdocs` 目录的权限。对 `/home/techdocs` 目录配置 `setgid(2)`，所有者/用户名和组具有读取/写入/执行权限 (7)，其他用户则没有权限 (0)。
6. 验证是否已正确设置权限。
7. 确认 `techdocs` 组中的用户能够在 `/home/techdocs` 目录中创建和编辑文件。不属于 `techdocs` 组的用户无法在 `/home/techdocs` 目录中创建和编辑文件。用户 `tech1` 和 `tech2` 在 `techdocs` 组中。用户 `database1` 不在该组中。
8. 修改全局登录脚本。普通用户应具有一个 `umask` 设置，该设置将阻止其他人查看或修改新的文件和目录。
9. 从 `serverb` 注销。

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

评估

在 workstation 上，运行 **lab perms-review grade** 脚本来确认是否成功完成本练习。

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

完成

在 workstation 上，运行 **lab perms-review finish** 脚本来完成实验。

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

本实验到此结束。

► 解决方案

控制对文件的访问

任务执行清单

在本实验中，您将配置文件权限并设置一个目录，使得特定组中的用户可以使用该目录方便地共享本地文件系统上的文件。

成果

您应能够：

- 创建一个目录，让用户能够在该目录中协作处理文件。
- 创建自动分配组所有权的文件。
- 创建在组外无法访问的文件。

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 perms-review start** 命令。该命令将运行一个起始脚本，它将确定 serverb 是否可从网络访问。该脚本还会创建 techdocs 组，以及名为 tech1、tech2 和 database1 的三个用户。

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

1. 使用 **ssh** 命令，以 student 用户身份登录 serverb。将 redhat 用作密码，在 serverb 上切换到 root。

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ su -
Password: redhat
[root@serverb ~]#
```

2. 创建名为 **/home/techdocs** 的目录。

- 2.1. 使用 **mkdir** 命令，创建一个名为 **/home/techdocs** 的目录。

```
[root@serverb ~]# mkdir /home/techdocs
```

3. 将 **/home/techdocs** 目录的组所有权更改为 techdocs 组。

- 3.1. 使用 **chown** 命令，将 **/home/techdocs** 目录的组所有权更改给 techdocs 组。

```
[root@serverb ~]# chown :techdocs /home/techdocs
```

4. 验证 **techdocs** 组中的用户是否能够在 **/home/techdocs** 目录中创建和编辑文件。

4.1. 使用 **su** 命令，切换为 **tech1** 用户。

```
[root@serverb ~]# su - tech1  
[tech1@serverb ~]$
```

4.2. 使用 **touch**，在 **/home/techdocs** 目录中创建一个名为 **techdoc1.txt** 的文件。

```
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt  
touch: cannot touch '/home/techdocs/techdoc1.txt': Permission denied
```



注意

请注意，即使 **/home/techdocs** 目录归 **techdocs** 所有，并且 **tech1** 是 **techdocs** 组的成员，也无法在该目录中创建新文件。这是因为 **techdocs** 组没有写入权限。使用 **ls -ld** 命令显示其权限。

```
[tech1@serverb ~]$ ls -ld /home/techdocs/  
drwxr-xr-x. 2 root techdocs 6 Feb 5 16:05 /home/techdocs/
```

5. 设置 **/home/techdocs** 目录的权限。对 **/home/techdocs** 目录配置 **setgid** (2)，所有者/用户和组具有读取/写入/执行权限 (7)，其他用户则没有权限 (0)。

5.1. 从 **tech1** 用户 shell 退出。

```
[tech1@serverb ~]$ exit  
logout  
[root@serverb ~]#
```

5.2. 使用 **chmod** 命令设置 **/home/techdocs** 目录的组权限。对 **/home/techdocs** 目录配置 **setgid** (2)，所有者/用户和组具有读取/写入/执行权限 (7)，其他用户则没有权限 (0)。

```
[root@serverb ~]$ chmod 2770 /home/techdocs
```

6. 验证是否已正确设置权限。

```
[root@serverb ~]$ ls -ld /home/techdocs  
drwxrws---. 2 root techdocs 6 Feb 4 18:12 /home/techdocs/
```

注意 **techdocs** 组现在具有写入权限。

7. 确认 **techdocs** 组中的用户能够在 **/home/techdocs** 目录中创建和编辑文件。不属于 **techdocs** 组的用户无法在 **/home/techdocs** 目录中创建和编辑文件。用户 **tech1** 和 **tech2** 在 **techdocs** 组中。用户 **database1** 不在该组中。

7.1. 切换到 **tech1** 用户。使用 **touch**，在 **/home/techdocs** 目录中创建一个名为 **techdoc1.txt** 的文件。从 **tech1** 用户 shell 退出。

```
[root@serverb ~]# su - tech1
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
-rw-rw-r--. 1 tech1 techdocs 0 Feb  5 16:42 /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 7.2. 切换到 tech2 用户。使用 echo 命令，在 /home/techdocs/techdoc1.txt 文件中添加一些内容。从 tech2 用户 shell 退出。

```
[root@serverb ~]# su - tech2
[tech2@serverb ~]$ cd /home/techdocs
[tech2@serverb techdocs]$ echo "This is the first tech doc." > techdoc1.txt
[tech2@serverb techdocs]$ exit
logout
[root@serverb ~]#
```

- 7.3. 切换到 database1 用户。使用 echo 命令，在 /home/techdocs/techdoc1.txt 文件中附加一些内容。请注意，您会收到 **Permission Denied** 消息。使用 ls -l 命令，确认 database1 没有该文件的访问权限。从 database1 用户 shell 退出。

```
[root@serverb ~]# su - database1
[database1@serverb ~]$ echo "This is the first tech doc." \
>> /home/techdocs/techdoc1.txt
-bash: /home/techdocs/techdoc1.txt: Permission denied
[database1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
ls: cannot access '/home/techdocs/techdoc1.txt': Permission denied
[database1@serverb ~]$ exit
logout
[root@serverb ~]#
```

8. 修改全局登录脚本。普通用户应具有一个 umask 设置，该设置将阻止其他人查看或修改新的文件和目录。

- 8.1. 确定 student 用户的 umask。使用 su - student 命令切换到 student 的登录 shell。完成后，退出 shell。

```
[root@serverb ~]# su - student
[student@serverb ~]$ umask
0002
[student@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 8.2. 对于 UID 大于 199 且用户名和主要组名相匹配的用户，使用以下内容创建 /etc/profile.d/local-umask.sh 文件，从而将 umask 设为 007，其他人的则设为 022。

```
[root@serverb ~]# cat /etc/profile.d/local-umask.sh
# Overrides default umask configuration
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

8.3. 从 shell 注销并以 **student** 身份重新登录，验证全局 umask 是否更改为 **007**。

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ umask
0007
```

9. 从 **serverb** 注销。

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

评估

在 **workstation** 上，运行 **lab perms-review grade** 脚本来确认是否成功完成本练习。

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

完成

在 **workstation** 上，运行 **lab perms-review finish** 脚本来完成实验。

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

总结

在本章中，您学到了：

- 文件具有三个应用权限的类别。文件由用户、单个组和其他用户拥有。应用最具体的权限。用户权限覆盖组权限，组权限又覆盖其他权限。
- 使用 **-l** 选项运行 **ls** 命令将展开文件列表，以包括文件权限和所有权。
- **chmod** 命令可从命令行更改文件权限。有两种表示权限的方法：符号（字母）法和数值（数字）法。
- **chown** 命令可更改文件所有权。**-R** 选项可以递归更改目录树的所有权。
- 不带参数运行 **umask** 命令将显示 shell 的当前 umask 值。系统上的每个进程都有一个 umask。Bash 的默认 umask 在 **/etc/profile** 和 **/etc/bashrc** 文件中定义。

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

章 8

监控和管理 LINUX 进程

目标

评估和控制运行在红帽企业 Linux 系统上的进程。

培训目标

- 获取有关在系统上运行的程序的信息，以便您可以确定状态、资源使用情况和所有权，从而对它们进行控制。
- 使用 Bash 作业控制来管理从同一终端会话启动的多个进程。
- 控制和终止与 shell 无关的进程，以及强行结束用户会话和进程。
- 描述负载平均值的定义，并确定对服务器上高资源使用量负责的进程。

章节

- 列出进程（及测验）
- 控制作业（及引导式练习）
- 中断进程（及引导式练习）
- 监控进程活动（及引导式练习）

实验

监控和管理 Linux 进程

列出进程

培训目标

学完本节后，您应能够获取有关在系统上运行的程序的信息，以便确定状态、资源使用情况和所有权，从而对它们进行控制。

进程的定义

进程是已启动的可执行程序的运行中实例。进程由以下组成部分：

- 已分配内存的地址空间
- 安全属性，包括所有权凭据和特权
- 程序代码的一个或多个执行线程
- 进程状态

进程的环境包括：

- 本地和全局变量
- 当前调度上下文
- 分配的系统资源，如文件描述符和网络端口

现有的（父）进程复制自己的地址空间 (**fork**) 来创建一个新的（子）进程结构。每个新进程分配有一个唯一进程 ID (PID)，满足跟踪和安全性之需。PID 和父进程 ID (PPID) 是新进程环境的元素。任何进程可创建子进程。所有进程都是第一个系统进程的后代，在红帽企业 Linux 8 系统上，第一个系统进程是 **systemd**。

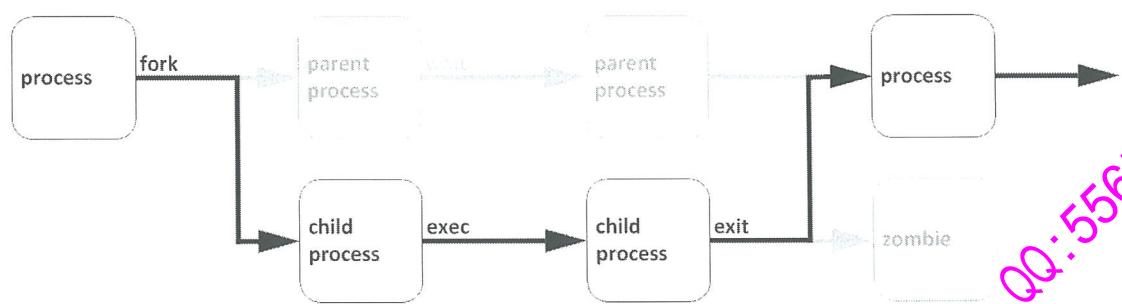


图 8.1: 进程生命周期

通过 **fork** 例程，子进程继承安全性身份、过去和当前的文件描述符、端口和资源特权、环境变量，以及程序代码。随后，子进程可能 **exec** 其自己的程序代码。通常，父进程在子进程运行期间处于睡眠状态，设置一个在子进程完成时发出信号的请求 (**wait**)。在退出时，子进程已经关闭或丢弃了其资源和环境。唯一剩下的资源称为僵停，是进程表中的一个条目。父进程在子进程退出时收到信号而被唤醒，清理子条目的进程表，由此释放子进程的最后一个资源。然后，父进程继续执行自己的程序代码。

描述进程状态

在多任务处理操作系统中，每个 CPU（或 CPU 核心）在一个时间点上处理一个进程。在进程运行时，它对 CPU 时间和资源分配的直接要求会有变化。进程分配有一个状态，它随着环境要求而改变。

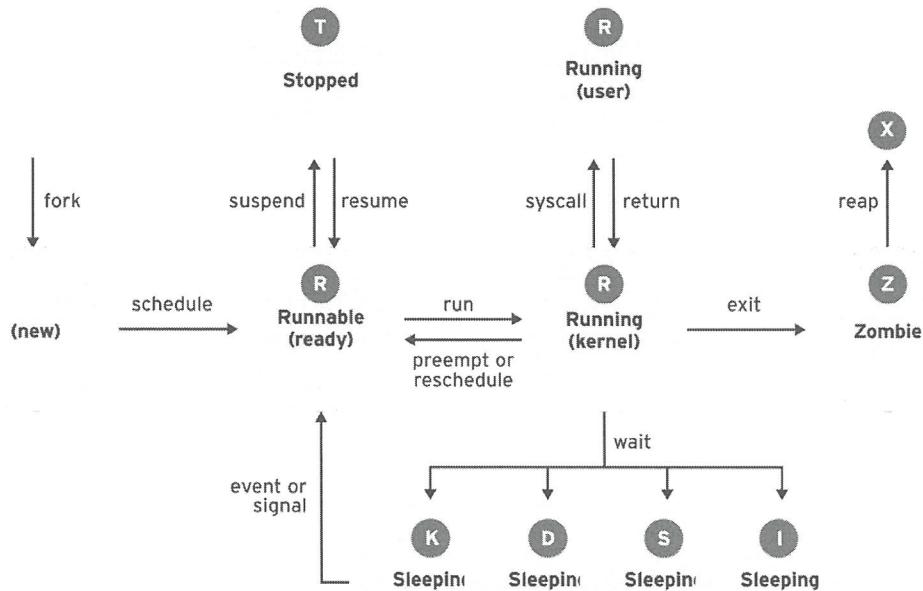


图 8.2: Linux 进程状态

Linux 进程状态在上图中进行了演示，下表也提供了说明。

Linux 进程状态

名称	标志	内核定义的状态名称和描述
运行	红	TASK_RUNNING：进程正在 CPU 上执行，或者正在等待运行。处于运行中（或可运行）状态时，进程可能正在执行用户例程或内核例程（系统调用），或者已排队并就绪。
睡眠	S	TASK_INTERRUPTIBLE：进程正在等待某一条件：硬件请求、系统资源访问或信号。当事件或信号满足该条件时，该进程将返回到运行中。
	D	TASK_UNINTERRUPTIBLE：此进程也在睡眠，但与 S 状态不同，不会响应信号。仅在进程中断可能会导致意外设备状态的情况下使用。
	K	TASK_KILLABLE：与不可中断的 D 状态相同，但有所修改，允许等待中的任务响应要被中断（彻底退出）的信号。实用程序通常将可中断的进程显示为 D 状态。

名称	标志	内核定义的状态名称和描述
	I	TASK_REPORT_IDLE: D 状态的一个子集。在计算负载平均值时，内核不会统计这些进程。用于内核线程。设置了 TASK_UNINTERRUPTABLE 和 TASK_NOLOAD 标志。类似于 TASK_KILLABLE，也是 D 状态的一个子集。它接受致命信号。
已停止	T	TASK_STOPPED: 进程已被停止（暂停），通常是通过用户或其他进程发出的信号。进程可以通过另一信号返回到运行中状态，继续执行（恢复）。
	T	TASK_TRACED: 正在被调试的进程也会临时停止，并且共享同一个 T 状态标志。
僵停	Z	EXIT_ZOMBIE: 子进程在退出时向父进程发出信号。除进程身份 (PID) 之外的所有资源都已释放。
	X	EXIT_DEAD: 当父进程清理（获取）剩余的子进程结构时，进程现在已彻底释放。此状态从不会在进程列出实用程序中看到。

进程状态的重要性

在对系统进行故障排除时，了解内核如何与进程通信以及进程如何相互通信非常重要。在创建进程时，系统会为进程分配一个状态。**top** 命令的 **S** 列或 **ps** 的 **STAT** 列显示每个进程的状态。在单 CPU 系统上，一次只能运行一个进程。可以看到多个状态为 **R** 的进程。然而，并非所有这些进程都在持续运行，其中一些将处于等待状态。

```
[user@host ~]$ top
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1 root 20 0 244344 13684 9024 S 0.0 0.7 0:02.46 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
...output omitted...
```

```
[user@host ~]$ ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
...output omitted...
root 2 0.0 0.0 0 0 ? S 11:57 0:00 [kthreadd]
student 3448 0.0 0.2 266904 3836 pts/0 R+ 18:07 0:00 ps aux
...output omitted...
```

可以使用信号暂停、停止、恢复、终止和中断进程。本章稍后将更为详细地探讨信号。信号可以由其他进程、内核本身或登录系统的用户使用。

列出进程

ps 命令用于列出当前的进程。它可以提供详细的进程信息，包括：

- 用户识别符 (UID)，它确定进程的特权
- 唯一进程识别符 (PID)
- CPU 和已经花费的实时时间
- 进程在各种位置上分配的内存数量
- 进程 **stdout** 的位置，称为控制终端

- 当前的进程状态



重要

Linux 版的 **ps** 命令支持三种选项格式：

- UNIX (POSIX) 选项，可以分组但必须以短划线开头
- BSD 选项，可以分组但不可与短划线同用
- GNU 长选项，以双短划线开头

例如，**ps -aux** 不同于 **ps aux**。

aux 或许是最常见的选项集，它可显示包括无控制终端的进程在内的所有进程。长列表（选项 **lax**）提供更多技术详细信息，但可通过避免查询用户名来加快显示。相似的 UNIX 语法使用选项 **-ef** 来显示所有进程。

```
[user@host ~]$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1  0.1  0.1  51648  7504 ?        Ss   17:45  0:03 /usr/lib/systemd/
syst
root      2  0.0  0.0     0     0 ?        S    17:45  0:00 [kthreadd]
root      3  0.0  0.0     0     0 ?        S    17:45  0:00 [ksoftirqd/0]
root      5  0.0  0.0     0     0 ?        S<  17:45  0:00 [kworker/0:0H]
root      7  0.0  0.0     0     0 ?        S    17:45  0:00 [migration/0]
...output omitted...
[user@host ~]$ ps lax
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY      TIME COMMAND
4   0     1     0   20   0  51648  7504 ep_pol Ss   ?      0:03 /usr/lib/
systemd/
1   0     2     0   20   0     0     0 kthrea S    ?      0:00 [kthreadd]
1   0     3     2   20   0     0     0 smpboo S    ?      0:00 [ksoftirqd/0]
1   0     5     2   0   -20    0     0 worker  S<  ?      0:00 [kworker/0:0H]
1   0     7     2  -100  -     0     0 smpboo S    ?      0:00 [migration/0]
...output omitted...
[user@host ~]$ ps -ef
UID      PID  PPID  C STIME TTY      TIME CMD
root      1     0  0 17:45 ?      00:00:03 /usr/lib/systemd/systemd --
switched-ro
root      2     0  0 17:45 ?      00:00:00 [kthreadd]
root      3     2  0 17:45 ?      00:00:00 [ksoftirqd/0]
root      5     2  0 17:45 ?      00:00:00 [kworker/0:0H]
root      7     2  0 17:45 ?      00:00:00 [migration/0]
...output omitted...
```

默认情况下，如果不使用选项而运行 **ps**，将选择具有与当前用户相同的有效用户 ID (EUID) 并与调用 **ps** 所处同一终端关联的所有进程。

- 方括号中的进程（通常位于列表顶部）为调度的内核线程。
- 僵停列为 **exiting** 或 **defunct**。
- **ps** 的输出显示一次。使用 **top** 来获得动态更新的进程显示。

- **ps** 可以采用树形显示格式，以便您可以查看父进程和子进程之间的关系。
- 默认输出按进程 ID 编号排序。第一眼可能看起来像是按时间顺序排列。但是，内核会重用进程 ID，因此其顺序不如显示的那样有序。若要排序，请使用 **-o** 或 **--sort** 选项。显示顺序与系统进程表的顺序匹配，在进程终止和新进程创建时重新使用列表行。输出可能会按照时间顺序显示，但不一定，除非使用明确的 **-o** 或 **--sort** 选项。



参考文献

info libc signal (GNU C 库参考手册)

- 第 24 节：信号处理

info libc processes (GNU C 库参考手册)

- 第 26 节：进程

ps(1) 和 **signal(7)** man page