User and Permission Management

Foreword

This document describes user and permission management on openEuler. After learning this document, you will understand the basic concepts of users and user groups on openEuler, and be familiar with commands for creating, deleting, and modifying files and directories. You will also learn about file permission configuration and specific operation commands.

Objectives

- After completing this course, you will understand:
  - Basic concepts about users and user groups
  - Command line operations related to files and directories
  - File permission configuration and related command operations
  - Special control methods for file access

Contents

Basic Concepts About Users

- Linux is a multi-user operating system.
  - All users who wish to use system resources must apply for an account from a system administrator, and then use it to log in.
  - Multiple users can be created on the system, and can log in to the same system at the same time to perform different tasks without affecting each other.
- User
  - A user can be viewed as a set of permissions to obtain system resources.
  - Each user is assigned a unique user ID (UID).

UID

- UID is a unique user identification that can also identify the user type. When a user logs in to a system, the UID (not the user name) is used to identify the user type.
  - Superuser: also called a **root** user whose UID is **0**. A superuser has full control over a system, and can modify and delete files, as well as running commands. As such, exercise caution when delegating the **root** user.
  - Common user: also called a normal or regular user whose UID ranges from 1000 to 60000. Common users can access and modify files in their own directories, and access authorized files.
  - Virtual user: also called a system user whose UID ranges from 1 to 999. Virtual users can log in to a system without passwords, facilitating system management.

Distinguishing User Types

- You can determine whether a user is the superuser, a common user, or a virtual user by viewing the UID.
- Run the **id** [*option*] [*user_name*] command to view a UID.
- Main options:
  - **-u**, **-user**: displays only a valid UID.
  - **-n**, **-name**: displays a name instead of a number for **-ugG**.
  - **-r**, **-real**: displays a real ID instead of a valid ID for **-ugG**.
- UID **0** indicates the superuser (**root** user). UIDs from 1000 to 60000 indicate common users. UIDs from 1 to 999 indicate virtual users (system users).

Contents

Managing Users

- On Linux, each common user has an account containing information such as a user name, password, and home directory. In addition, there are some special users created by the system. The most important user has the administrator account, and is the superuser **root** by default.
- Users can run commands to create, modify, and delete user files, as well as changing passwords.

Creating a User - useradd

- The **useradd** command is used to create a user account and save it in the **/etc/passwd** file.
- Syntax: **useradd** [*options*] *user_name*

  Main options:

  - **-u** specifies the user's UID.
  - **-o** is used together with **-u** to allow duplicate UIDs.
  - **-g** specifies a basic group to which the user belongs. It can be a user group name or a group ID (GID) if the group exists.
  - **-d** specifies the home directory for the user and automatically creates the home directory.
  - **-s** specifies the default shell program of a user.
  - **-D** displays or changes the default configuration.

Creating a User - Example

- Create a user named **user**.
  - The command is **useradd user**.

```
[root@localhost ~]# useradd user
```

  - Run the **cat /etc/passwd** command to check whether a user has been successfully created. The command output shows that the user has been created.

```
[root@localhost ~]# cat /etc/passwd

dbus:x:980:980:System Message Bus:/:/usr/sbin/nologin
test:x:1000:1000::/home/test:/bin/bash
test02:x:1001:1001::/home/test02:/bin/bash
user:x:1002:1004::/home/user:/bin/bash
```

Modifying a User - usermod

- The **usermod** command is used to modify the information about a user account.
- Syntax: **usermod** [*options*] *user_name*

  Main options:

  - **-u** modifies the user's UID.
  - **-g** modifies the user group to which the user belongs.
  - **-l** modifies the user account name.
  - **-d** modifies the user's home directory.
  - **-s** modifies the user's default shell program.

Modifying a User - Example

- Modify a user named **user**:
  - Run the **id user** command to view the user's UID before modification.

```
[root@localhost ~]# id user

uid=1002(user) gid=1004(user) groups=1004(user)
```

  - Run the **usermod -u 1003 user** command to modify the UID to **1003**. The command output shows that the UID has been modified.

```
[root@localhost ~]# usermod -u 1003 user

[root@localhost ~]# id user
uid=1003(user) gid=1004(user) groups=1004(user)
```

Deleting a User - userdel

- The **userdel** command is used to delete a specified user and related files.
- Syntax: **userdel** [*options*] *user_name*

  Main options:

  - **-f** forcibly deletes a user account even if the user is logged in.
  - **-r** deletes a user and all related files.
  - **-h** displays the help information about a command.

  (Using the **userdel** command to delete a specified user and user-related files modifies the user account system files.)

Deleting a User - Example

- Delete a test user:
  - Run the **cat /etc/passwd** command to view a user before deletion.

```
[root@localhost ~]# cat /etc/passwd

dbus:x:980:980:System Message Bus:/:/usr/sbin/nologin
test:x:1000:1000::/home/test:/bin/bash
test02:x:1001:1001::/home/test02:/bin/bash
user:x:1002:1004::/home/user:/bin/bash
```

  - Run the **userdel user** command to delete a test user. The command output shows that the user has been deleted.

```
[root@localhost ~]# userdel user

[root@localhost ~]# cat /etc/passwd
dbus:x:980:980:System Message Bus:/:/usr/sbin/nologin
test:x:1000:1000::/home/test:/bin/bash
test02:x:1001:1001::/home/test02:/bin/bash
```

Changing a User Password - passwd

- The **passwd** command is used to change a user password.
- Syntax: **passwd** [*OPTION*...] *user_name*

  Main options:

  - **-n** sets the minimum number of days between password change.
  - **-x** sets the maximum number of days between password change.
  - **-w** sets the number of days of warning before password expires.
  - **-i** sets the number of days before an account is disabled once a password expires.
  - **-d** deletes the user password.
  - **-S** displays the user password information.

  (The **root** user can change any user's password. Common users can change only their own passwords.)

Changing a User Password - Example

- Change the password of a test user:

the password can't be seen in the /etc/shadow file !!!

  - Run the **/etc/shadow** command to check the user password before the change. The output shows that the password has not set and is displayed as **!**.

```
[root@localhost ~]# cat /etc/shadow

user:!:18421:0:99999:7:::
```

  - Run the **passwd user** command to change the user password.

```
[root@localhost ~]# passwd user
Changing password for user user.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

  - Run the **/etc/shadow** command again to check whether the password has been changed successfully. The command output shows that the password has been changed successfully.

```
[root@localhost ~]# cat /etc/shadow

user:$6$KOrFTTStwbMS0eIG$3peFd8yIgxPyaSYi8TG8XFNUdYUdeMd60lR2hvRC6zx3dAdbEqQcnQuDoWT7ocu3Ss.zzWSrEb6cZ6Ae6b2EN/:18421:0:99999:7:::
```

Contents

Basic Concepts About User Groups

- User group:
  - A user group is a logical set of users with the same features. These users in one group can have the same permissions, which facilitates management.
  - Each user has a private group.
  - All users in the same group can share files in the group.
  - Each user group is assigned a unique group ID (GID).

GID

- A GID is similar to a UID, and is a unique identifier of a user group in a system.
  - When an account is added, a group with the same name as the user is created by default. The UID and GID are the same.
  - UID 0 is assigned to the superuser and GID 0 is assigned to a user group that has the superuser (that is, the **root** user group).
  - The system reserves some small GIDs for virtual users (also called system users).
- You can run the **id** [*option*] [*user_name*] command to view the GID and the number of users in a group.

User Group Classification

- Common user group: Multiple users can be added to a common user group.
- System group: Some system users are added to the system group.
- Private group (also called basic group): When a user is created, if a group to which the user belongs is not specified, a user private group is defined for the user. The group name is the same as the user name.

Relationships Between Users and User Groups

- One-to-one: A user is the only member in one group.
- One-to-many: A user belongs to multiple user groups and has their permissions.
- Many-to-one: Multiple users belong to a user group, and have that group's permissions.
- Many-to-many: Multiple users belong to multiple user groups, and have relevant permissions based on the above rules.

Contents

Managing User Groups

- As the number of users increases, user permission control and system security management are becoming more complex. But having user groups means each user is associated with at least one user group, which facilitates permission management.
- User and user group management is an important part of system security management. You can run commands to create, modify, and delete user group files, and associate users with user groups.

Creating a Group - groupadd

- The **groupadd** command is used to create a user group and add the new group information to a system file.
- Syntax: **groupadd** [*options*] *group_name*

  Main options:

  - **-f** exits if the group already exists.
  - **-g** assigns a GID to the new user group.
  - **-h** shows help information and exits.
  - **-o** allows duplicate GIDs.
  - **-p** assigns an encrypted password for the new user group.
  - **-r** creates a system account.

Creating a Group - Example

- Create a test user group:
  - Run the **groupadd usergroup** command to create a test user group.

```
[root@localhost ~]# groupadd usergroup
```

  - Run the **cat /etc/group** command to check whether the user group has been created successfully. The command output shows that a new test group has been successfully created.

```
[root@localhost ~]# cat /etc/group

test04:x:1003:
user:x:1004:
usergroup:x:1005:
```

Modifying a Group - groupmod

- The **groupmod** command is used to change a group identifier or group name.
- Syntax: **groupmod** [*options*] *group_name*
- Main options:
  - **-g** sets a new GID.
  - **-h** shows help information and exits.
  - **-n** sets a new group name.
  - **-o** allows duplicate GIDs.
  - **-p** modifies the encrypted password.

Modifying a Group - Example

- Modify a test user group GID:
  - Run the **id user** command to view the original test user GID.

```
[root@localhost ~]# id user

uid=1002(user) gid=1004(user) groups=1004(user)
```

  - Run the **groupmod –g 1006 user** command to modify the GID of the test user group to **1002**. The command output shows that the GID has been successfully modified.

```
[root@localhost ~]# groupmod -g 1006 user

[root@localhost ~]# id user
uid=1002(user) gid=1006(user) groups=1006(user)
```

Deleting a Group - groupdel

- The **groupdel** command is used to delete a user group from the system. If a user group contains users, delete users before deleting the group.
- Syntax: **groupdel** [*options*] *group_name*
- Main options:
  - **-f** deletes a group even if it is the primary group of a user.
  - **-h** shows help information and exits.

Deleting a Group - Example

- Delete a test user group:
  - Run the **cat /etc/group** command to view a test user group.

```
[root@localhost ~]# cat /etc/group

test04:x:1003:
user:x:1006:
usergroup:x:1005:
```

  - Run the **groupdel usergroup** command to delete the test user group. Then, run the **cat /etc/group** command to check whether the test user group has been deleted.

```
[root@localhost ~]# groupdel usergroup

[root@localhost ~]# cat /etc/group
test04:x:1003:
user:x:1006:
```

Associating a User with a Group - gpasswd

- The **gpasswd** command is used to add or delete a user from a group.
- Syntax: **gpasswd** [*option*] *group_name*
- Main options:
  - **-a** adds a user to a group.
  - **-d** deletes a user from a group.
  - **-M** sets a group member list.
  - **-A** sets a group administrator list.
  - **-r** removes the group password.
  - **-R** restricts group access to members only.
  - **-Q** indicates a directory specified by **chroot**.

Associating a User with a Group - Example

- Associate a test user with a test group.
  - Run the **gpasswd –a user usergroup** command to add a user to a user group.

```
[root@localhost ~]# gpasswd -a user usergroup

Adding user user to group usergroup
```

  - Run the **groups user** command. The command output shows that the user has been added to the group.

```
[root@localhost ~]# groups user

user : user usergroup
```

Contents

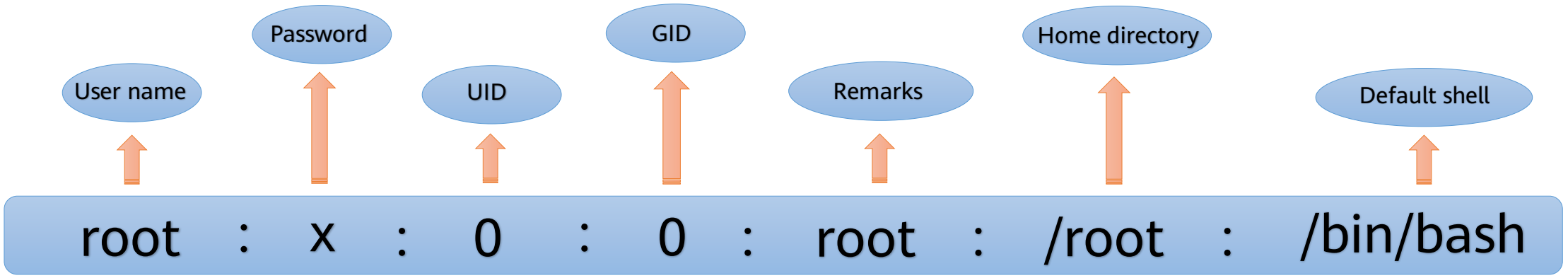
Files Associated with Users on openEuler

- In the openEuler directory, there are two types of files related to user information management:
- **/etc/passwd**: user account information file.
  - This file saves the main information about all users in a system. Each line represents a record.
  - Each record defines various user attributes.
- **/etc/shadow**: encrypted user account information file (also called shadow file).
  - This file stores user passwords in a system.
  - All users can read the **/etc/passwd** file, which may cause password leakage. Therefore, the password information is separated from the **/etc/passwd** file and stored in the **/etc/shadow** file.

/etc/passwd File

the order of each user information is important !!!

- Each line in the **/etc/passwd** file consists of seven fields separated by colons (:).



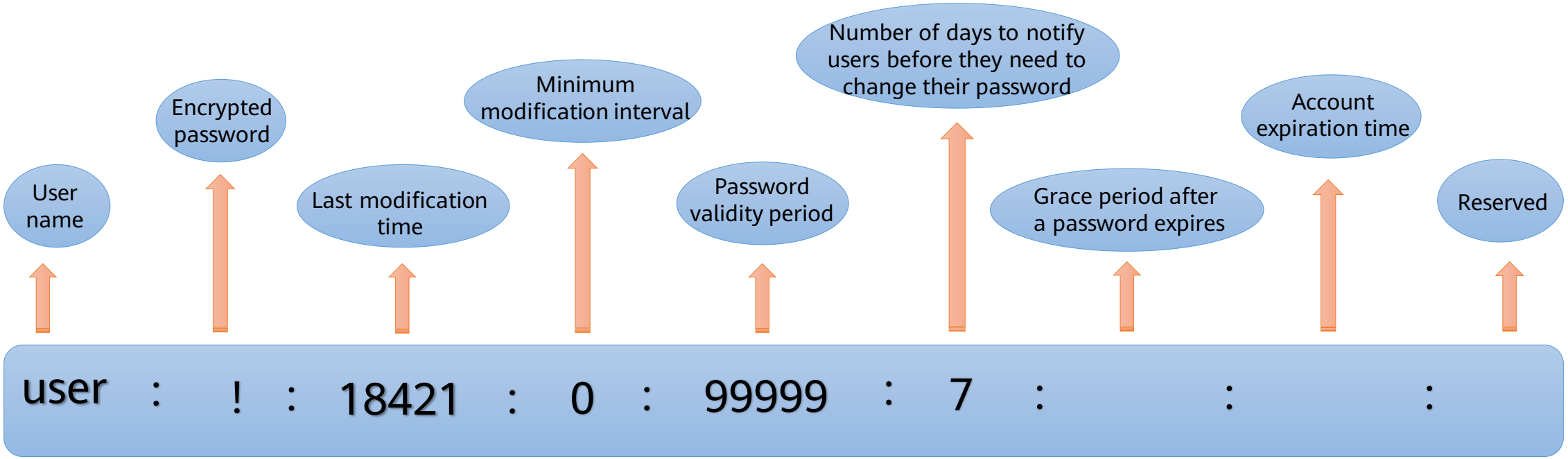
- The Linux shell is a kernel-based command line interpreter (CLI) for users, which is presented as an interface for user login. The Linux shells include sh, csh, ksh, tcsh, and bash.

/etc/passwd File-related Parameters !!!

No.	Description
1	A username, which can contain uppercase letters, lowercase letters, digits, minus signs (–), periods (.), and underscores (_). Other characters are invalid.
2	An encrypted password, which is represented by **X** in this field. For details about the password, see the **/etc/shadow** file.
3	A UID, which is used to identify a user and determine the user type.
4	A GID, which corresponds to the group information in **/etc/group** and is used to manage users by group.
5	Remarks, which contain attributes such as the user information.
6	Home directory, where a user is located during login. This field can be customized.
7	Default shell, which is used to transfer instructions issued by users to a kernel.

/etc/shadow File

- Only the superuser (**root**) has read permission on the **/etc/shadow** file, which ensures user password security.
- A password protected by **/etc/shadow** is displayed as **X** in a user record of the **/etc/passwd** file.
- In the **/etc/shadow** file, each line of record also indicates a user and there are 9 fields separated by colons (:).



/etc/shadow File-related Parameters

No.	Description
1	A user name, which means the same as the user name in the **/etc/passwd** file.
2	An encrypted password. If the value of this field is ***** or **!**, the account cannot be used to log in to the system.
3	Time when the password was last changed.
4	Minimum modification interval. If this field is set to **0**, a password can be changed at any time.
5	Validity period of a password. Users must periodically change their passwords to improve system security.
6	Number of days to notify users before they need to change their password. When a user password is about to expire, a warning message is displayed to prompt the user to change the password.
7	Grace period after a password expires. If a password is not changed within the grace period, the user is disabled.
8	User expiration date. After a password expires, the user is no longer a valid user and cannot log in to the system.
9	This field is reserved and is left blank for future use.

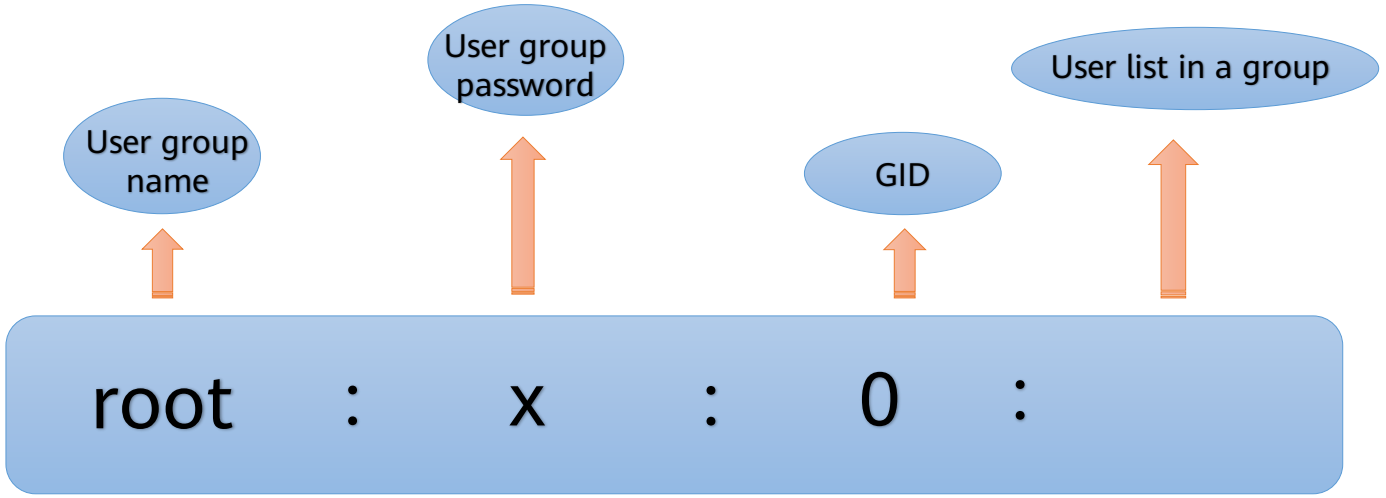
(Note: The **/etc/shadow** file is automatically generated by the **pwconv** command based on data in the **/etc/passwd** file.)

Files Associated with User Groups on openEuler

- In the openEuler directory, there are two types of files related to user group information management:
- **/etc/group** : group information file
  - This file saves all information about user groups. Each line represents a user group.
  - Grouping users is a method of managing users and controlling access permissions. Each user belongs to one or more user group and a group can contain multiple users.
- **/etc/gshadow**: encrypted group information file
  - This file saves the encryption information of a user group. For example, this file contains the user group management password (similar to the **/etc/shadow** file).
  - This file is complementary to the **/etc/group** file. For a large-scale server, there are many users and user groups, generating complex permission models. Therefore, it is important to set and manage passwords.

/etc/group File

- Each line in the **/etc/group** file consists of four fields separated by colons (:).
- The **root** user is used as an example to describe fields in a user group record in the **/etc/group** file.

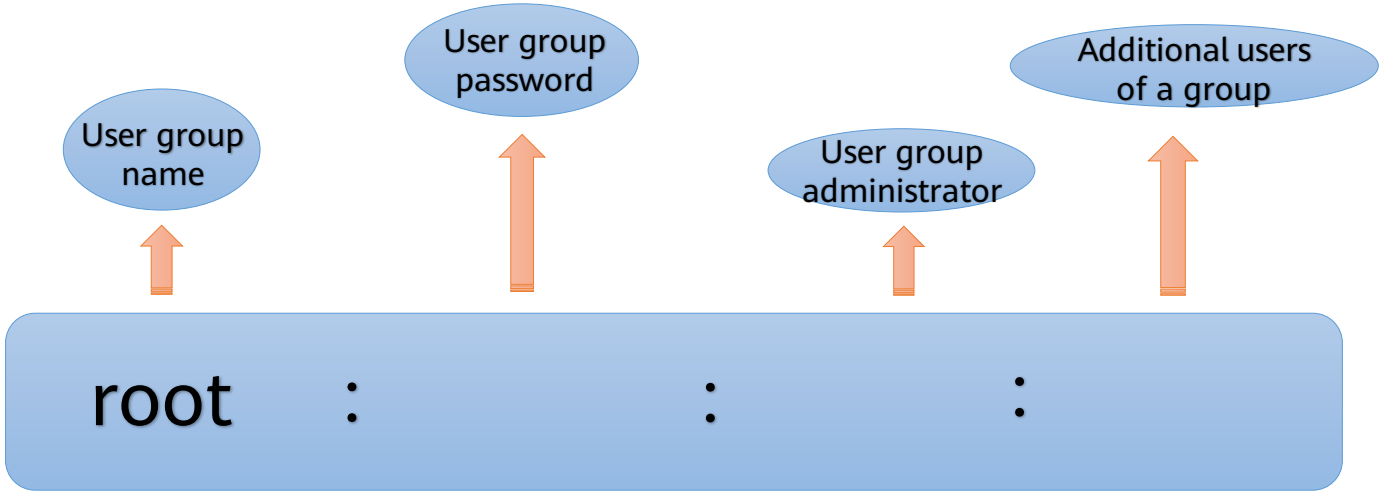


/etc/group File-related Parameters

No.	Description
1	A user group name, which has the same meaning as that in the **/etc/passwd** file.
2	A user group password. To ensure security, this field is represented by **X**. The user group password is stored in the **/etc/gshadow** file.
3	A GID, which is used to identify a user group and determine a user group type.
4	A list of users in a user group. This field lists all users in a user group.

/etc/gshadow File

- Each line in the **/etc/gshadow** file consists of four fields separated by colons (:).
- The **root** user is used as an example to describe fields in the **/etc/gshadow** file.



/etc/gshadow File-related Parameters

No.	Description
1	A user group name, which has the same meaning as the user name in the **/etc/group** file.
2	A user group password. Generally, a user group password is not set, and this field can be left empty or set to an exclamation mark (!).
3	A user group administrator. This field can be left empty. If there are multiple user administrators, separate them with commas (,).
4	Additional user group users. This field lists additional users in a user group.

Note (USER PASSWORD) : If password is not set for user, it can be either "*" or "!"

Contents

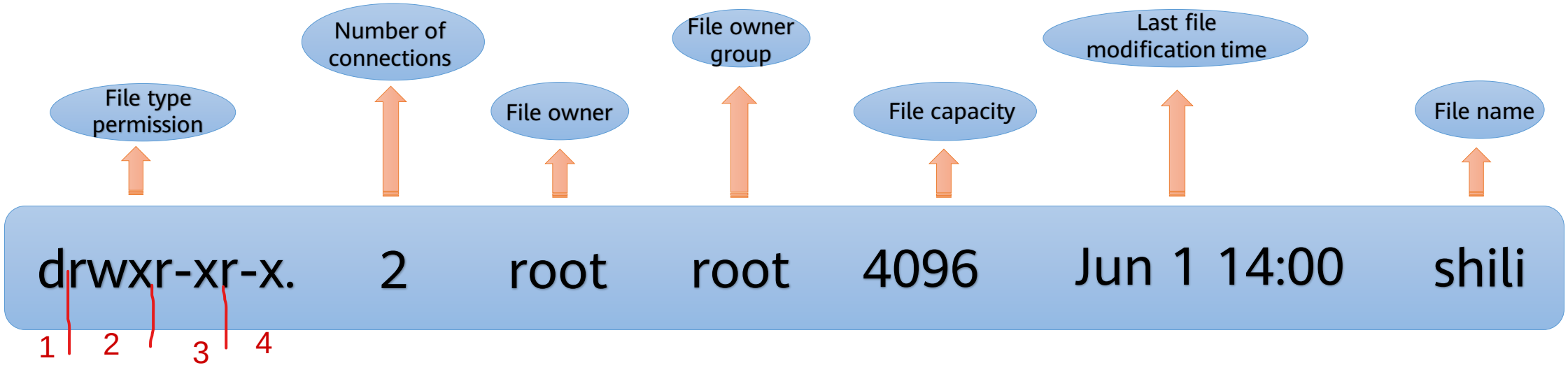
Permission Overview

- Permissions are used by an operating system to control resource access, and are classified into read, write, and execute permissions.
- Different users have different levels of permissions on a Linux system. To ensure system security, a Linux system defines rules for each user permission.
- Each file or directory has a specific access permission, owner user, and owner group. These rules can be used to specify which user and which group can perform what operations on a specific file.

Permission Example

- You can run the **ls –l** command to display detailed information about file permissions.
  - The file information of the **root** user is used here as an example.

drwxr-xr-x. 2 root root 4096 Jun 1 14:00 shili



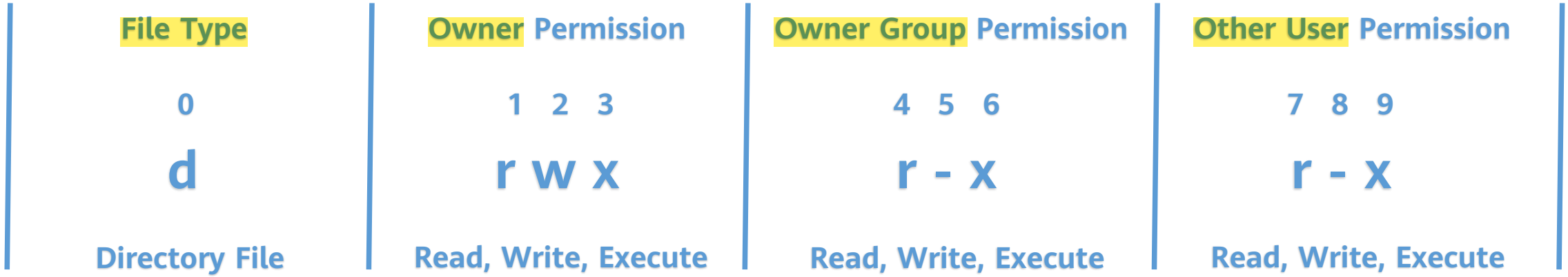
Permission Example - File Types

- The first character in each line indicates a file type. There are seven file types on Linux:

File Type	Description
-	Files excluding the other six file types
d	Directory
b	Block device file, which is a random access device
c	Character device file, which is a one-time read device such as keyboard and mouse
l	Symbolic link file, which points to another file
p	Named pipe file
s	Socket file

Permission Bits

- Linux file or directory access permission is controlled by nine permission bits. Every three bits form a group, which is a combination of read (r), write (w), and execute (x) permissions. The order of the three permission bits remain unchanged in a file or directory. A hyphen (-) indicates that a user does not have the permission.



Bit 0 indicates a file type. Bits 1 to 3 indicate the permissions of a file owner. Bits 4 to 6 indicate the permissions of other users in the same group of the file owner. Bits 7 to 9 indicate the permissions of other users on the file.

Permission Example - Access

- The following is displayed in the file information:
  - **-r** allows users to read files or all contents in a directory.
  - **-w** allows users to write files, or to create or delete files in a directory.
  - **-x** allows users to execute files or go to a directory.
  - **-** specifies no permissions and is displayed in the **r**, **w**, and **x** positions.
- The following uses the **usertxt** file as an example: **drwxr-xr-x. 2 root root 4096 Jun 1 14:00 usertxt**

Position	Permission	Binary	Decimal	Permission Details
Bits 1, 2, and 3	**rwx**	2^1 111 2^2 2^0	4+2+1=7	The file owner has the read, write, and execute permissions.
Bits 4, 5, and 6	**r-x**	101	4+1=5	Users in the same group have the read and execute permissions, but do not have the write permission.
Bits 7, 8, and 9	**r-x**	101	4+1=5	Other users have the read and execute permissions, but do not have the write permission.

Contents

Main Permission Setting Commands

- **chmod**: modifies a file permission.
  - The visitors to a Linux file is divided into three categories: file owner, group, and others. The **chmod** command specifies who can visit a file.
  - Usage permission: file owner
- **chown**: modifies a file owner and owner group.
  - Linux is a multi-user and multi-task system, where all files have owners. You can run the **chown** command to change an owner of a specific file to a specified user or group.
  - Usage permission: administrator (**root**)
- **chgrp**: modifies a file owner group.
  - You can run the **chgrp** command to change a group to which a file or directory belongs.
  - Usage permission: administrator (**root**)
- **umask**: sets the mask code.
  - You can run the **umask** command to preset the permission mask when creating a file
  - Usage permissions: administrator and common user

Modifying File Permissions - chmod

- The **chmod** command specifies who (file owner, group, and others) can visit a file.
- Syntax: **chmod** [*OPTION*]... **MODE**[,*MODE*]... *FILE*...
- Main options:
  - Operation objects:
    - **u**: user, indicating an owner of a file or directory
    - **g**: user group, indicating a group to which a file or directory belongs
    - **o**: other users
    - **a**: all users
  - Operators:
    - **+**: adds permissions.
    - **-**: reduces permissions.
    - **=**: provides specified permissions.
  - Permissions:
    - **r**: read
    - **w**: write
    - **x**: execute
- Based on the configuration scenario, you can modify a group of permissions on a file at the same time or modify a single permission on the file.

Modifying File Permissions - Example

- Modify all permissions of the test file **usertxt**:
  - Run the **ls –l** command to check the permissions of the **usertxt** file before modification.

```
[root@localhost ~]# ls -l

drwx------. 2 root    root      4096 Jun  8 11:10 usertxt
```

  - Run the **chmod 644 usertxt** command to modify permissions. Run the **ls –l** command to check whether permissions have been modified.

```
[root@localhost ~]# chmod 644 usertxt

[root@localhost ~]# ls –l
drw-r--r--. 2 root    root      4096 Jun  8 11:10 usertxt
```

Modifying File Permissions - chown

- The **chown** command is used to change the owner of a specified file to a particular user or group.
- Syntax: **chown** [*OPTION*]... [*OWNER*][:[*GROUP*]] *FILE*...
- Main options:
  - **-c**: displays the modified information.
  - **-f**: ignores error information.
  - **-h**: restores a symbolic link.
  - **-v**: displays detailed processing information.
  - **-R**: processes all files in a specified directory and its subdirectories.
- Based on the configuration scenario, you can change an owner or owner group, or change both at the same time.

Modifying File Permissions - Example

- Modify both owner and owner group:
  - Run the **ls –l** command to check the owner and owner group of the **usertxt** file before modification.

```
[root@localhost ~]# ls –l

drw-r--r--. 2 root   root      4096 Jun  8 11:10 usertxt
```

  - Run the **chown user: usergroup usertxt** command to modify permissions, and run the **ls –l** command to check whether permissions have been modified.

```
[root@localhost ~]# chown user:usergroup usertxt

[root@localhost ~]# ls -l
drw-r--r--. 2 user   usergroup    4096 Jun  8 11:10 usertxt
```

Modifying File Permissions - chgrp

- The **chgrp** command is used to modify a group to which a file or directory belongs.
- Syntax: **chgrp** [*OPTION*]... *GROUP FILE*...
- Main options:
  - **-v**: displays the command execution process.
  - **-c**: similar to the **-v** option, but returns only the modified part.
  - **-f**: does not display error information.
  - **-h**: modifies only symbolic link files.
  - **-R**: recursively processes all files and subdirectories in a specified directory.
- Modify a group to which the file belongs based on the configuration scenario.

Modifying File Permissions - Example

- Modify a file owner group:
  - Run the **ls –l** command to check the owner group of the **usertxt** file before modification.

```
[root@localhost ~]# ls –l

drw-r--r--. 2 user   usergroup    4096 Jun  8 11:10 usertxt
```

  - Run the **chgrp usergroup01 usertxt** command to modify permissions, and run the **ls –l** command to check whether permissions have been modified.

```
[root@localhost ~]# chgrp usergroup01 usertxt

[root@localhost ~]# ls –l
drw-r--r--. 2 user   usergroup01    4096 Jun  8 11:10 usertxt
```

Preset Permission Mask - umask

- The **umask** command is used to specify a preset permission mask when a file or directory is created.
- Syntax: **umask**: **umask** [**-p**] [**-S**] [*mode*]
- Main options:
  - **-p**: displays a command name.
  - **-S**: specifies a permission mask in text format.
- Main umask values and corresponding file or directory permissions are as follows:

umask Value	File Permission	Directory Permission
022	644	755
027	640	750
002	664	775
006	660	771
007	660	770

Preset Permission Mask - Example

- Modify a file permission mask:
  - Run the **umask** command to check a umask value before modification.

```
[root@localhost ~]# umask

0077
```

  - Run the **umask 022** command to modify a permission, and run the **umask** command to check whether a permission mask has been changed.

```
[root@localhost ~]# umask 022

[root@localhost ~]# umask
0022
```

Contents

Access Control List (ACL)

- The **chmod**, **chown**, **chgrp**, and **umask** commands for common permissions can be used to modify file permissions. Why do we use the Access Control List (ACL)?
  - Without using ACL, the visitors to the Linux system are divided into three categories: file owner, user group, and other users. However, as technology develops, traditional file permission control cannot meet the requirements of complex scenarios. For example, there are multiple employees (such as **user01** and **user02**) in a department (a user group), and different permissions are assigned to employees with different responsibilities. For example, the read and write permissions are assigned to **user01**, the read-only permission is assigned to **user02**, and no permissions are assigned to **user03**. As these employees belong to the same department, employee permissions cannot be further refined. The access control list (ACL) is developed to solve this problem. ACL provides permission settings in addition to common permissions (such as **rwx** and **ugo**), and you can set specific permissions for a single user or a user group.
- Main types:
  - Assign permissions to a file owner.
  - Assign permissions to a user group of a file.
  - Assign permissions to other users.

ACL - Related Commands

- On Linux, we can use ACL to manage a file and its specific user and user group permissions. Simply put, an ACL requires only three commands: **setfacl**, **getfacl**, and **chacl**.
- **setfacl**: sets a file ACL.
  - The **chmod** command assigns permissions based on a file owner, owner group, and other users. The **setfacl** command assigns permissions more precisely for each file or directory.
- **getfacl**: obtains a file ACL.
- **chacl**: changes an ACL of a file or directory.
  - The **chacl** command is similar to the **chmod** command, but is more powerful and precise. While the **chmod** command specifies who can invoke a file, if a file of a user needs to be viewed only by a specific user, the **chacl** command must be executed to meet the user's requirements.

Obtaining a File ACL - getfacl

- The **getfacl** command is used to obtain a file or directory ACL.
- Syntax: **getfacl** [**-aceEsRLPtpndvh**] *file*...
- Main options:
  - **-a**: displays only a file ACL.
  - **-d**: displays only the default ACL.
  - **-c**: does not display comment headers.
  - **-e**: displays all valid permissions.
  - **-E**: displays invalid permissions.
  - **-s**: skips files that contain only base entries.

Obtaining a File ACL - Example

- Run the **getfacl** command to view all valid permissions of the **usertxt** file.
  - Command: **getfacl –e usertxt**

```
[root@localhost ~]# getfacl -e usertxt

# file: usertxt
# owner: user
# group: usergroup01
user::rw-
group::r--
other::r--
```

Setting a File ACL - setfacl

- The **setfacl** command is used to set a file ACL.
- Syntax: **setfacl** [**-bkndRLP**] {**-m|-M|-x|-X**... } *file* ...
- Main options:
  - **-m**: modifies the ACL of a specified file. This parameter cannot be used together with **-x**.
  - **-x**: deletes subsequent parameters.
  - **-b**: deletes all ACL parameters.
  - **-k**: removes preset ACL parameters.
  - **-R**: recursively sets ACL parameters.
  - **-d**: presets the ACL parameters of a directory.

Setting a File ACL - Example

- Grant a user's read and write permissions.
  - Run the **getfacl usertxt** command to view permissions on the **usertxt** file.

```
[root@localhost ~]# getfacl -e usertxt
# file: usertxt
# owner: user
# group: usergroup01
user::rw-
group::r--
other::r--
```

  - Run the **setfacl –m u:user:rw usertxt** command to add read and write permissions to the user, and then run the **getfacl** command to check permissions. The command output shows that permissions have been added.

```
[root@localhost ~]# getfacl -e usertxt
# file: usertxt
# owner: user
# group: usergroup01
user::rw-
user:user:rw-                   #effective:rw-
group::r--                      #effective:r--
mask::rw-
other::r--
```

Changing a File or Directory ACL - chacl

- The **chacl** command is used to set control permissions on files or directories.
- Syntax: **chacl** [**acl/R/D/B/l/r**] *pathname*... **/ chacl -b acl dacl** *pathname*... **/ chacl -d dacl** *pathname*...
- Main options:
  - **-b**: modifies file permissions and default directory permissions at the same time.
  - **-d**: sets default permissions on a directory.
  - **-R**: deletes only file permissions.
  - **-D**: deletes only directory permissions.
  - **-B**: deletes all permissions.
  - **-l**: lists all file and directory permissions.
  - **-r**: sets permissions on all directories and subdirectories.

Changing a File or Directory ACL - Example

- Clear ACL settings from the **usertxt** file:
  - Command: **chacl –B usertxt**

```
[root@localhost ~]# getfacl -e usertxt
# file: usertxt
# owner: user
# group: usergroup01
user::rw-
user:user:rw-                  #effective:rw-
group::r--                     #effective:r--
mask::rw-
other::r--

[root@localhost ~]# chacl -B usertxt
[root@localhost ~]# getfacl -e usertxt
# file: usertxt
# owner: user
# group: usergroup01
user::rw-
group::r--
other::r--
```

Contents

Other Management Permissions

- In the Linux operating system, the default account is a common user. However, only the **root** user can modify system files or run certain commands.
- The following commands are commonly used to switch to the **root** user:
  - **su**: switches a user identity to the **root** user. The user is still a common user in the shell environment.
  - **su –**: switches a user to the **root** user for both the user identity and shell environment.
  - **sudo**: allows common users to execute commands that can be executed only by administrator accounts.

Commands - su and su-

- The **su** command is used to change a user identity, but does not change the shell environment.
- Syntax: **su** [*options*] [-] [<*user*> [<*argument*>...]]
- Main options:
  - **-m, -p**: specifies that environment variables are not changed when the **su** command is executed.
  - **-s**: specifies the shell to be executed (such as **bash**, **csh**, and **tcsh**).
  - **-c**: switches to the account *user* and restores to the original user after the command is executed.
  - **-f**: does not need to read the boot file. This parameter is used only for **csh** or **tcsh**.

Example

- Change the user identity to **root** and display detailed information.
  - Command: **su –c ls root**

```
[root@localhost ~]# su -c ls root

anaconda-ks.cfg  createVM.sh    deleteVM.sh                  revertsnap.sh  test    usertxt
createsnap.sh    deletesnap.sh  httpd-2.4.34-15.oe1.aarch64.rpm  shili          test01
```

Command - sudo

- The **sudo** command allows common users to execute tasks that can be executed only by the **root** user.
- Syntax: **sudo –h | -K | -k | -V**
- Main options:
  - **-h**: displays the version number and instructions. !!!
  - **-k**: asks the user for a password upon the next run of the **sudo** command.
  - **-V**: displays the version number.
  - **-l**: displays user permissions.
  - **-L**: displays the **sudo** settings.

Example

- Run the **sudo** command to view the version number.
  - Command: **sudo -V**

```
[root@localhost ~]# sudo –V

Sudo version 1.8.27
Configure options: --build=aarch64-openEuler-linux-gnu --host=aarch64-openEuler-linux-gnu --program-prefix= --disable-dependency-tracking --prefix=/usr --exec-prefix=/usr --bindir=/usr/bin --sbindir=/usr/sbin --sysconfdir=/etc --datadir=/usr/share --includedir=/usr/include --libdir=/usr/lib64 --libexecdir=/usr/libexec --localstatedir=/var --sharedstatedir=/var/lib --mandir=/usr/share/man --infodir=/usr/share/info --prefix=/usr --sbindir=/usr/sbin --libdir=/usr/lib64 --docdir=/usr/share/doc/sudo --disable-root-mailer --with-logging=syslog --with-logfac=authpriv --with-pam --with-pam-login --with-editor=/bin/vi --with-env-editor --with-ignore-dot --with-tty-tickets --with-ldap --with-selinux --with-passprompt=[sudo] password for %p:  --with-linux-audit --with-sssd
Sudoers policy plugin version 1.8.27
Sudoers file grammar version 46
```

Quiz

1. On openEuler, which of the following UIDs belongs to a common user by default? ( )

   A. 0

   B. 200

   C. 800

   D. 1200

2. Which of the following commands can be used to view information in files associated with users and groups? ( )

   A. cat

   B. chmod

   C. clear

   D. chage

cat /etc/passwd
cat /etc/shadow
cat /etc/group
cat /etc/gshadow

Summary

- This document describes basic concepts related to users and user groups on openEuler, and provides commands and methods for adding users and user groups. It describes common file permissions, such as read, write, and execute, and special file permissions, such as **setfacl**, **getfacl**, and **chacl**, as well as the access control list (ACL). It shows how to modify file permissions by using related command parameters and examples, such as how to modify read, write, and execute permissions on files and directories.

More Information

openEuler open source community:

https://openeuler.org/en/docs/20.03_LTS/docs/Releasenotes/release_notes.html

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI