Comprehensive Notes on File and Directory Permissions in Linux

Understanding Linux Permissions

Basics of File Permissions

Linux file system permissions are a core concept that governs who can do what with a file or directory. There are three types of permissions:

1. **Read (r):** Permission to read the file or directory contents.
2. **Write (w):** Permission to modify the file or directory contents.
3. **Execute (x):** Permission to execute a file or view the contents of a directory.

Permissions are divided among three categories:

- **User (u):** The owner of the file.
- **Group (g):** The group associated with the file.
- **Others (o):** Everyone else with access to the system.

Permissions are represented by a three-digit octal number. Each digit represents a group of permissions:

- **4 for read**
- **2 for write**
- **1 for execute**

For example, 755 means:

- 7 (4+2+1): Read, write, execute for the user
- 5 (4+1): Read, execute for the group
- 5 (4+1): Read, execute for others【4:7†source】【4:8†source】.

- **`chown` :** This command changes the ownership of a file or directory. Format: `chown user:group filename` .
- **`chmod` :** Used to change the permissions of a file or directory. Numerical values represent permissions (e.g., `chmod 755 filename`)【4:5†source】.

Example Scenario Using Permissions

In a school-like analogy, consider files representing resources (like basketballs and footballs):

1. **Creating Files and Groups:**
   - Create files: basketball, football, tennis.
   - Set up user groups like basketball_group and football_group 【4:11†source】.
2. **Assigning Permissions:**
   - Set ownership with `chown` , e.g., `chown root:basketball_group basketball` .
   - Modify permissions with `chmod` , e.g., `chmod 760 basketball` means the teacher (root) has all permissions, the basketball group can read and write, and others have no permissions【4:2†source】【4:5†source】.

Special Concepts

- **Access Control Lists (ACLs):** For more granular permission management, ACLs can be used. They allow specifying permissions for individual users beyond the standard user-group-others model. Commands like `setfacl` and `getfacl` are used to set and view ACLs respectively【4:18†source】【4:17†source】.
- **Redirection and Pipes:**
   - **Redirection:** Redirect output using `>` for overwriting and `>>` for appending.
   - **Pipe (`|`):** Chains the output of one command to another【4:6†source】【4:15†source】.

- **`usermod`:** Add users to groups, e.g., `usermod -aG groupname username` 【4:3†source】.
- **Wildcards:** Used in command line to represent one or more characters in filenames (e.g., `*` for multiple characters)【4:16†source】.

Closing Summary

Permissions in Linux provide a powerful way to control file access and modification. Remembering the numeric representation and mastering commands like `chmod` and `chown` are crucial for system management. ACLs extend these capabilities, offering even finer control when needed【4:17†source】【4:18†source】.

These tools and concepts collectively ensure that files and directories are accessed and modified securely, maintaining system integrity.