LINUX

17 February 2025 15:30

Setting File Permissions in Linux

File permissions control access to files and directories, defining who can read, modify, or execute them.

User Categories:

- **Owner (User):** The individual who owns the file.
- **Group:** Users who belong to the same group as the owner.
- **Others:** All other users on the system.

Permission Types:

- **Read (r / 4):** Grants access to view file contents.
- **Write (w / 2):** Allows modifications or deletion.
- **Execute (x / 1):** Permits running the file as a script or program.

```
sandy@sandy:~$ cd ~
sandy@sandy:~$ mkdir new
sandy@sandy:~$ cd new
sandy@sandy:~/new$ cd ..
sandy@sandy:~$ ls -lrt
total 8
-rw-r--r-- 1 sandy sandy   88 Feb 18 04:45 cinema.txt
drwxr-xr-x 2 sandy sandy 4096 Feb 19 22:15 new
sandy@sandy:~$
```

Changing File/Folder Permissions with chmod

The chmod command is used to modify file or folder permissions in Linux.

Methods:

- **chmod 777 filename**
  - The first **7** sets permissions for the **owner (user)**
  - The second **7** sets permissions for the **group**
  - The third **7** sets permissions for **others**
- **chmod u+x filename**
  - **+** adds a permission (e.g., granting execute access to the user).
- **chmod u-x filename**
  - **-** removes a permission (e.g., revoking execute access from the user).

```
sandy@sandy:~/new$ cd ..
sandy@sandy:~$ ls -lrt
total 8
-rw-r--r-- 1 sandy sandy   88 Feb 18 04:45 cinema.txt
drwxr-xr-x 2 sandy sandy 4096 Feb 19 22:15 new
sandy@sandy:~$ chmod 777 new
sandy@sandy:~$ ls -lrt
total 8
-rw-r--r-- 1 sandy sandy   88 Feb 18 04:45 cinema.txt
drwxrwxrwx 2 sandy sandy 4096 Feb 19 22:15 new
sandy@sandy:~$ chmod 744  cinema.txt
sandy@sandy:~$ ls -lrt
total 8
-rwxr--r-- 1 sandy sandy   88 Feb 18 04:45 cinema.txt
drwxrwxrwx 2 sandy sandy 4096 Feb 19 22:15 new
sandy@sandy:~$
```

Creating Directories (mkdir)

- **mkdir dir_name** → Creates a single directory.
- **mkdir folder1 folder2 folder3** → Creates multiple directories at once.
- **mkdir -p parent/child/grandchild** → Creates nested directories, ensuring parent directories exist.
- **[-d my_folder] || mkdir my_folder** → Checks if the directory exists; if not, it creates one.

Removing Files or Directories (rm -rf, rmdir)

The rm -rf command is used to delete files and directories permanently.

Flags:

- **-r (Recursive)** → Deletes directories along with their contents.
- **-f (Force)** → Forces deletion without confirmation.

Warning:

- rm -rf **permanently deletes files** without sending them to the trash.
- **Avoid running** rm -rf / or rm -rf /*, as it can erase your entire system.
- Always double-check before using wildcards (*).

- **rmdir directory_name** → Removes the specified empty directory.
- **rmdir -p dir1/dir2/dir3** → Removes dir3, then dir2, and finally dir1, but only if each is empty after the previous one is deleted.

```
sandy@sandy:~$ cd new
sandy@sandy:~/new$ mkdir -p a/b/c/d A/B/C/D
sandy@sandy:~/new$ ls
A  a
sandy@sandy:~/new$ cd a
sandy@sandy:~/new/a$ touch new_file.txt
sandy@sandy:~/new/a$ ls
b  new_file.txt
sandy@sandy:~/new/a$ rm -rf new_file.txt
sandy@sandy:~/new/a$ ls
b
sandy@sandy:~/new/a$
```

```
 76  mkdir -p a/b/c/test.txt
 77  ls
 78  cd a
 79  cd b
 80  ls
 81  cd c
 82  cd ..
 83  cd .. ..
 84  cd ..
 85  cd ..
 86  mkdir d e f
 87  ls
 88  cd d
 89  cd ..
 90  touch {1..5}.txt
 91  ls
 92  ls -lrt
 93  rm a
 94  rmdir a
 95  rmdir -rf a
 96  rmdir a
 97  rmdir -r a
 98  rmdir --help
 99  rmdir -p a
100  rmdir -p a/b/c
101  rm -rf* a
102  rm -rf a
```

Searching for Keywords in a File (grep)

The grep command is used to find specific text within files.

- **grep "search_term" filename** → Searches for "search_term" in the specified file.
- **grep -i "search_term" filename** → Case-insensitive search.
- **grep "error" file1.txt file2.txt** → Searches for "error" in multiple files.

Searching in Directories

- **grep -r "critical" /dir1/dir2** → Recursively searches for "critical" in all files under /dir1/dir2.

```
134  ls
135  vi 1.txt
136  cat 1.txt|grep "word"
137  vi 5.txt
138  grep -r "ing" /testPractice/
139  grep -r "ing"
140  cd d
141  vi 6.txt
142  cd ..
143  grep -r "ing"
144  grep -rl "ing"
145  grep -rn "ing"
146  vi 1.txt
147  grep -E "wo|ing" 1.txt
148  grep -E "wo|ing" 1.txt 5.txt
149  grep -rE "wo|ing"
150  grep -rE "wo&ing"
151  grep -rE "wo & ing"
152  grep -rE "wo && ing"
153  grep -rE ^wo
154  grep -rx "word"
155  grep -rx "word wording word"
```

Copying Files (cp)

- **cp file.txt newfile.txt** → Copies file.txt and renames it to newfile.txt.
- **cp file1.txt /dir1/dir2/dir3/** → Copies file1.txt to the specified directory.
- **cp -rf my_folder /dir1/dir2/** → Copies my_folder and all its contents to /dir1/dir2/ (-r for recursive copying).

```
102  rm -rf a
103  ls
104  cp 1.txt d
105  cd d
106  ls
107  cd ..
108  ls
109  mv 2.txt d
110  ls
111  mv 3.txt,4.txt d
112  mv 3.txt 4.txt d
113  cd d
114  ls
115  cd ..
116  cp -rf 5.txt /mnt/c
117  cp -rf 5.txt /mnt/c/Users/user
118  cd ..
119  ls
120  ls -lrt
121  cd Desktop
122  ls
123  cd testPractice
124  ls
125  cp -rd f /mnt/c/Users/user/Desktop
```

To move files
mv source destination

```
105  cd d
106  ls
107  cd ..
108  ls
109  mv 2.txt d
110  ls
111  mv 3.txt,4.txt d
112  mv 3.txt 4.txt d
```

man Command
The man command in Linux displays documentation for commands, programs, and system calls.

- **man ls** → Shows details about ls, including available flags and usage.

Finding Files in Linux

1. **Using locate**
   - locate filename → Quickly finds a file by name.
   - locate -i document → Case-insensitive search for "document".
2. **Using whereis**
   - whereis program_name → Finds the binary, source, and manual files of a program.
3. **Using find**
   - find /home -name "file.txt" → Searches for "file.txt" in the /home directory.

```
sandy@sandy:/mnt/e/test/new$ locate file1.txt
/mnt/c/Windows.old/Users/user/AppData/Local/npm-cache/_npx/86fd6f984ad48a35/node_modules/@pnpm/network.ca-file/dist/fixtures/ca-file1.txt
/mnt/c/Windows.old/Users/user/AppData/Local/npm-cache/_npx/86fd6f984ad48a35/node_modules/@pnpm/network.ca-file/fixtures/ca-file1.txt
/mnt/c/Windows.old/Users/user/AppData/Roaming/npm/node_modules/thirdweb/node_modules/@pnpm/network.ca-file/dist/fixtures/ca-file1.txt
/mnt/c/Windows.old/Users/user/AppData/Roaming/npm/node_modules/thirdweb/node_modules/@pnpm/network.ca-file/fixtures/ca-file1.txt
sandy@sandy:/mnt/e/test/new$
```

```
346  find -name "*.txt"
347  ls
348  find -type d
349  ls
350  mkdir new
351  find -type d
352  find type d
```

```
sandy@sandy:/mnt/e/test$ whereis python
python:
sandy@sandy:/mnt/e/test$ whereis java
java: /usr/share/java
sandy@sandy:/mnt/e/test$ whereis mysql
mysql: /usr/bin/mysql /usr/lib/mysql /etc/mysql /usr/share/mysql /usr/share/man/man1/mysql.1.gz
```

Tar and Gunzip Commands in Linux

1. **Creating an Archive:**
   - tar -cvf archive.tar file1 file2 directory/
     - -c → Create archive
     - -v → Show progress
     - -f → Specify archive file
2. **Extracting an Archive:**
   - tar -xvf archive.tar → Extracts files in the current directory
   - tar -xvf archive.tar -C /dir1/dir2 → Extracts to a specific directory
3. **Creating a Compressed Archive:**
   - tar -czvf archive.tar.gz file1 file2 directory/

4. **Extracting a Compressed Archive:**
   - tar -xzvf archive.tar.gz

Task	Command
Create .tar archive	tar -cvf archive.tar files/
Extract .tar archive	tar -xvf archive.tar
Create .tar.gz archive	tar -czvf archive.tar.gz files/
Extract .tar.gz archive	tar -xzvf archive.tar.gz
Compress a file with gzip	gzip file.txt → file.txt.gz
Decompress a .gz file	gunzip file.txt.gz

```
sandy@sandy:/mnt/e/test$ ls
basicCalculator.sh  code.sh   file.txt  new.sh   sample-logs.md
case.sh             data.txt  new       new1.sh  test1.sh
sandy@sandy:/mnt/e/test$ tar -cvf new.tar file.txt
file.txt
sandy@sandy:/mnt/e/test$ ls
basicCalculator.sh  code.sh   file.txt  new.sh   new1.sh         test1.sh
case.sh             data.txt  new       new.tar  sample-logs.md
sandy@sandy:/mnt/e/test$ cat new.tar
file.txt0000777000000000000000000000022147546045510112 50 0ustar  rootrootnew file
new data
sandy@sandy:/mnt/e/test$
```

SHELL SCRIPTING

Variable declaration can happen in both local and global ways

1) local

```
var1="sindhuja" var2="gudla" echo "my name is $var1 $var2"
var1="sindhuja"
var2="gudla"
echo "my name is $var1 $var2"
unset var1
echo $var1
```

2) Global

```
name="sindhu"
echo "my name is $name"
~
```

```
sandy@sandy:/mnt/e/test$ vi sample.sh
sandy@sandy:/mnt/e/test$ ./sample.sh
my name is sindhu
```

IF - ELSE

```
sandy@sandy: /mnt/e/test
num=10

if [ $num -gt 5 ]; then
    echo "Number is greater than 5"
else
    echo "Number is 5 or less"
fi
~
~
~
```

```
sandy@sandy:/mnt/e/test$ vi sample1.sh
sandy@sandy:/mnt/e/test$ ./sample1.sh
Number is greater than 5
```