In Unix, altering the permissions of files and directories is a common task, executed using the `chmod` (change mode) command. There are two primary methods to modify permissions with `chmod`: the symbolic mode and the absolute (numeric) mode.

Using `chmod` in Symbolic Mode

Symbolic mode is user-friendly, especially for beginners. It involves using characters to add, remove, or set specific permissions. Here's a summary of the symbolic mode operators:

Symbol	Operation
`+`	Adds specified permissions
`-`	Removes specified permissions
`=`	Sets specified permissions exactly

Examples in Symbolic Mode

Consider a file named `testfile` with initial permissions as follows:

```
$ ls -l testfile
-rwxrwxr--  1 user  group 1024  Nov 2 00:10  testfile
```

Applying various `chmod` commands in symbolic mode:

1. **Add Write and Execute Permissions to Others**:

   ```
   $ chmod o+wx testfile
   $ ls -l testfile
   -rwxrwxrwx  1 user  group 1024  Nov 2 00:10  testfile
   ```

2. **Remove Execute Permission from the Owner**:

   ```
   $ chmod u-x testfile
   $ ls -l testfile
   -rw-rwxrwx  1 user  group 1024  Nov 2 00:10  testfile
   ```

3. **Set Group Permissions to Read and Execute Only**:

```
$ chmod g=rx testfile
$ ls -l testfile
-rw-r-xrwx  1 user  group 1024  Nov 2 00:10  testfile
```

Combining multiple changes:

```
$ chmod o+wx,u-x,g=rx testfile
$ ls -l testfile
-rw-r-xrwx  1 user  group 1024  Nov 2 00:10  testfile
```

Using `chmod` with Absolute Permissions

Absolute mode involves using numeric values to represent each set of permissions. This method is precise and often used in scripting.

Each permission is assigned a value:

Number	Octal Permission	Permissions
0	---	No permission
1	--x	Execute
2	-w-	Write
3	-wx	Write and Execute
4	r--	Read
5	r-x	Read and Execute
6	rw-	Read and Write
7	rwx	All permissions

Understanding the absolute (numeric) mode of the `chmod` command becomes more intuitive when you consider the numbers in binary form. Each permission in Unix (read, write, execute) can be represented as a binary digit, making it easier to remember and calculate the numeric values used in chmod.

In binary, each digit represents a state: `0` for "off" (no permission) and `1` for "on" (permission granted). Permissions are ordered as read (r), write (w), and execute (x), corresponding to the

binary places:

- **Read (r)**: 4 in decimal, represented as `100` in binary (the leftmost digit).
- **Write (w)**: 2 in decimal, represented as `010` in binary (the middle digit).
- **Execute (x)**: 1 in decimal, represented as `001` in binary (the rightmost digit).

Combining these binary digits for each set of permissions (owner, group, others) forms the octal (base-8) permission number used in `chmod`. For example:

- **Read and Write (rw-)**: In binary, read (`100`) + write (`010`) = `110`, which is `6` in decimal.
- **All Permissions (rwx)**: Read (`100`) + write (`010`) + execute (`001`) = `111`, which is `7` in decimal.
- **Read and Execute (r-x)**: Read (`100`) + execute (`001`) = `101`, which is `5` in decimal.

So, when you set permissions using `chmod 754`, for instance, you're effectively setting:

- `7` for the owner: `111` in binary, which is read (`4`), write (`2`), and execute (`1`).
- `5` for the group: `101` in binary, which is read (`4`) and execute (`1`).
- `4` for others: `100` in binary, which is read (`4`) only.

Thinking of permissions in terms of binary digits helps to conceptualize what each numeric value in the chmod command represents, making it easier to remember and use the correct numbers for setting desired permissions.

Examples in Absolute Mode

Starting with the same `testfile`:

```
$ ls -l testfile
-rwxrwxr--  1 user  group 1024  Nov 2 00:10  testfile
```

Applying `chmod` commands in absolute mode:

1. **Set Permissions to 755**:

   ```
   $ chmod 755 testfile
   $ ls -l testfile
   -rwxr-xr-x  1 user  group 1024  Nov 2 00:10  testfile
   ```

2. **Set Permissions to 743**:

```
$ chmod 743 testfile
$ ls -l testfile
-rwxr---wx  1 user  group 1024  Nov 2 00:10  testfile
```

3. **Set Permissions to 043**:

```
$ chmod 043 testfile
$ ls -l testfile
----r---wx  1 user  group 1024  Nov 2 00:10  testfile
```