Umask

Related articles

File permissions and attributes

The *umask* utility is used to control the file-creation mode mask, which determines the initial value of file permission bits for newly created files. The behaviour of this utility is standardized by **POSIX** and described in the **POSIX Programmer's Manual (http://pubs.opengroup.org/onlinepubs/9699919799/utilities/umask.html)**. Because *umask* affects the current shell execution environment, it is usually implemented as built-in command of a shell.

Contents

Meaning of the mode mask

The mode mask contains the permission bits that should **not** be set on a newly created file, hence it is the **logical complement** of the permission bits set on a newly created file. If some bit in the mask is set to `1`, the corresponding permission for the newly created file will be disabled. Hence the mask acts as a filter to strip away permission bits and helps with setting default access to files.

The resulting value for permission bits to be set on a newly created file is calculated using bitwise **material nonimplication** (also known as abjunction), which can be expressed in logical notation:

```
R: (D & (~M))
```

That is, the resulting permissions `R` are the result of **bitwise conjunction** of default permissions `D` and the **bitwise negation** of file-creation mode mask `M`.

Note:

- Linux does not allow a file to be created with execution permissions, in fact the default creation permissions are `777` for directories, but only `666` for files.
- On Linux, only the file permission bits of the mask are used - see **`umask(2) (https://jlk.fjfi.cvut.cz/arch/manpages/man/umask.2)`**. The *suid*, *sgid* and *sticky* bits of the mask are ignored.

For example, let us assume that the file-creation mode mask is 027. Here the bitwise representation of each digit represents:

- `0` stands for the *user* permission bits not set on a newly created file
- `2` stands for the *group* permission bits not set on a newly created file
- `7` stands for the *other* permission bits not set on a newly created file

With the information provided by the table below this means that for a newly created file, for example owned by `User1` user and `Group1` group, `User1` has all the possible permissions (octal value `7`) for the newly created file, other users of the `Group1` group do not have write permissions (octal value `5`), and any other user does not have any permissions (octal value `0`) to the newly created file. So with the `027` mask taken for this example, files will be created with `750` permissions.

Octal	Binary	Meaning
`0`	`000`	no permissions
`1`	`001`	execute only
`2`	`010`	write only
`3`	`011`	write and execute
`4`	`100`	read only
`5`	`101`	read and execute
`6`	`110`	read and write
`7`	`111`	read, write and execute

Display the current mask value

To display the current mask, simply invoke `umask` without specifying any arguments. The default output style depends on implementation, but it is usually octal:

```
$ umask
```

```
0027
```

When the `-S` option, standardized by POSIX, is used, the mask will be displayed using symbolic notation. However, the **symbolic notation value will always be the logical complement of the octal value**, i.e. the permission bits to be set on the newly created file:

```
$ umask -S
```

```
u=rwx,g=rx,o=
```

Set the mask value

Note: Umask values can be set on a case-by-case basis. For example, desktop users may find the restricted permissions on their home folder (`chmod 700` , as applied by `useradd -m`) sufficient, as they make all files within unaccessible to other users. Should this not be practical (for example when using **Apache**), and public files are stored amongst private ones, then consider restricting the umask instead.

You can set the umask value through the *umask* command. The string specifying the mode mask follows the same syntactic rules as the mode argument of **chmod** (see the **POSIX Programmer's Manual (http://pubs.opengroup.org/onlinepubs/9699919799/utilities/chmod.html#tag_20_17_13)** for details).

System-wide umask value can be set in `/etc/profile` or in the default **shell** configuration files, e.g. `/etc/bash.bashrc`. Most Linux distributions, including Arch **[1] (https://projects.archlinux.org/svntogit/packages.git/tree/trunk/profile?h=packages/filesystem)** set a default value of `022`. You can also set umask with `pam_umask.so` but it may be overridden by `/etc/profile` or similar.

If you need to set a different value, you can either directly edit such file, thus affecting all users, or call `umask` from your shell's user configuration file, e.g. `~/.bashrc` to only change your umask, however these changes will only take effect after the next login. To change your umask during your current session only, simply run `umask` and type your desired value. For example, running `umask 077` will give you read and write permissions for new files, and read, write and execute permissions for new folders.

See also

- POSIX Programmer's Manual:
  - **umask (http://pubs.opengroup.org/onlinepubs/9699919799/utilities/umask.html)** (also available as **`umask(1p) (https://jlk.fjfi.cvut.cz/arch/manpages/man/um`**

ask.1p))
  - **chmod (extended description) (http://pubs.opengroup.org/onlinepubs/9699919799/utilities/chmod.html#tag_20_17_13)** (also available as `chmod(1p) (https://jlk.fjfi.cvut.cz/arch/manpages/man/chmod.1p)`)
- **wikipedia:umask**
- **027 umask: a compromise (https://blogs.gentoo.org/mgorny/2011/10/18/027-umask-a-compromise-between-security-and-simplicity/)**

Retrieved from "https://wiki.archlinux.org/index.php?title=Umask&oldid=504065"