|  |  |
| --- | --- |
| Special Permissions (SetUID, SetGID and Sticky Bit) in Linux - Cyber Sophia | Permissions spéciales |

Table des matières

[Introduction 2](#_Toc84170688)

[Sticky Bit 2](#_Toc84170689)

[Set GID 3](#_Toc84170690)

[Set UID 4](#_Toc84170691)

# Introduction

Outre les permissions de lecture, d’écriture et d’exécution pour les utilisateurs, les groupes et les autres, chaque fichier peut avoir **trois autres permissions spéciales** qui peuvent **modifier le fonctionnement d’un répertoire ou l’exécution d’un programme**.

Les autorisations spéciales constituent un **quatrième niveau d'accès en plus de l'utilisateur**, **du groupe** et **des autres**. Les autorisations spéciales permettent des **privilèges supplémentaires sur les ensembles d'autorisations standard**.

Il existe une option d'autorisation spéciale pour chaque niveau d'accès.

Elles peuvent être **spécifiées en mode symbolique ou numérique**.

# Sticky Bit

Le sticky bit (bit collant), également appelé **étiquette de suppression restreinte**, possède …

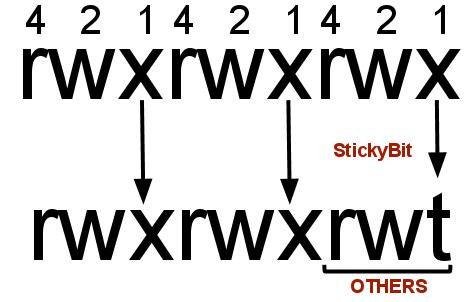
* la **valeur octale 1**   
  et
* en **mode symbolique est représenté par un t**

pour les permissions des autres (*others*).

Le Sticky bit est un bit d'autorisation défini sur un fichier ou un répertoire qui **permet uniquement au propriétaire du fichier/répertoire** ou à l'utilisateur **root de supprimer ou de renommer le fichier**.  
Aucun autre utilisateur n'est autorisé à supprimer le fichier créé par un autre utilisateur.

Cela permet de créer un répertoire que tout le monde (et les processus qu'ils lancent) peut utiliser comme stockage de fichiers partagé.   
Il est utile pour les répertoires partagés tels que /var/tmp et /tmp car les utilisateurs peuvent créer des fichiers, lire et exécuter des fichiers appartenant à d'autres utilisateurs, mais ne sont pas autorisés à supprimer des fichiers appartenant à d'autres utilisateurs.

De manière générale, le **sticky bit ne s’applique qu’aux répertoires**.



Les répertoires avec le sticky bit activé montrent un **t remplaçant le x sur les permissions des autres** …  
**>> ls -ld travaux/**  
**drwxr-xr-t 2 tux tux 4096 Oct 20 18:46 travaux/**

**Remarque** …  
Dans la commande précédente -d (ou --directory) permet d’afficher les noms de répertoires et non pas leur contenu.

En mode numérique, les permissions spéciales sont **spécifiées à l’aide d’une notation à 4 chiffres**, le **premier chiffre représentant la permission spéciale** sur laquelle agir.   
Par exemple, pour définir le sticky bit (valeur 1) pour le répertoire travaux/ en mode numérique, avec les permissions 755, la commande est …  
**>> chmod 1755 travaux/  
>> ls -ld travaux/**  
**drwxr-xr-t 2 tux tux 4096** **Oct 20 16:46 travaux/**

Exemple …

**>> sudo mkdir --verbose /data**  
**mkdir: création du répertoire '/data'  
>> sudo chmod 777 /data  
>> ls -ld /data**  
**drwxrwxrwx 2 root root 4096 Oct 20 16:46 /data**

En tant qu’utilisateur tux …  
**>> touch /data/rapport.texte  
>> ls -ld /data  
drwxrwxrwx 2 root root 4096 Oct 20 16:46 /data**

Le fichier rapport.texte est créé par un utilisateur différent mais dispose d'un accès en lecture-écriture-exécution pour tous les utilisateurs.  
Cela signifie que n’importe quel utilisateur peut supprimer ou renommer ce fichier.

Afin d'éviter cela, le sticky bit peut être défini sur le répertoire data/.

On active le sticky bit collant sur le répertoire en utilisant l'indicateur +t de la commande chmod …  
**>> sudo chmod +t /data  
>> ls -ld /data  
drwxrwxrwt 2 root root 4096 oct 15:55 /data**

Comme on peut le constater, un bit de permission t est placé dans les bits de permission du répertoire.

Maintenant, si un utilisateur essaie de renommer le fichier rapport.texte …  
**>> cd /data  
>> mv /data/rapport.texte tableau.data  
mv: impossible de déplacer '/data/rapport.texte' vers 'tableau.data': Permission non accordée**

On voit donc que l'opération n'est pas autorisée.

**Remarque** …  
Après avoir défini le Sticky Bit sur un fichier/répertoire, si on voit T dans la zone d'autorisation de fichier, cela indique que le fichier/répertoire n'a pas d'autorisations exécutables pour tous les utilisateurs de ce fichier/répertoire particulier.

# Set UID

Communément appelée SUID, l'autorisation spéciale pour le niveau d'accès utilisateur possède une seule fonction …  
un **fichier avec SUID s'exécute toujours en tant qu'utilisateur propriétaire du fichier**, **quel que soit l'utilisateur qui passe la commande**.

Le SUID, également connu sous le nom de Set User ID, possède la **valeur octale 4 et est représenté par un s** sur les permissions de l'utilisateur en mode symbolique.  
Il **s’applique uniquement aux fichiers** et son **comportement est similaire au bit SGID**, mais le processus s’exécutera avec les privilèges de l'utilisateur qui possède le fichier.

Si le **propriétaire du fichier n'a pas d'autorisations d'exécution**, on utilise un **S majuscule**.

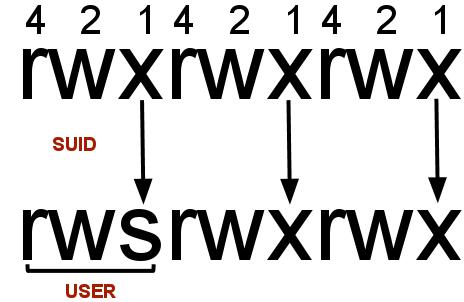
# Set UID

Communément appelée SUID, l'autorisation spéciale pour le niveau d'accès utilisateur possède une seule fonction …  
un **fichier avec SUID s'exécute toujours en tant qu'utilisateur propriétaire du fichier**, **quel que soit l'utilisateur qui passe la commande**.

Le SUID, également connu sous le nom de Set User ID, possède la **valeur octale 4 et est représenté par un s** sur les permissions de l'utilisateur en mode symbolique.  
Il **s’applique uniquement aux fichiers** et son **comportement est similaire au bit SGID**, mais le processus s’exécutera avec les privilèges de l'utilisateur qui possède le fichier.

Normalement, sous Linux/Unix, lorsqu'un programme s'exécute, il hérite des autorisations d'accès de l'utilisateur connecté. SUID est défini comme l'octroi d'autorisations temporaires à un utilisateur pour exécuter un programme/fichier avec les autorisations du propriétaire du fichier plutôt que l'utilisateur qui l'exécute .

Si le **propriétaire du fichier n'a pas d'autorisations d'exécution**, on utilise un **S majuscule**.



Voici quelques exemples …

**Commande passwd**Lors de la modification d’un mot de passe, on utilise la commande passwd, commande dont le propriétaire est root.   
La commande passwd essaiera de modifier certains fichiers de configuration système tels que /etc/passwd, /etc/shadow, …. lors de la demande de modification du mot de passe.   
Certains de ces fichiers ne peuvent pas être ouverts ou affichés par un utilisateur normal, seul l'utilisateur root possédant cette autorisation.   
La commande passwd est donc définie avec SUID afin d’attribuer des autorisations d'utilisateur root à l'utilisateur normal afin qu'il puisse mettre à jour /etc/shadow et d'autres fichiers.

**Commande crontab**Lors de la planification des tâches à l'aide de crontab ou de la commande at, l’utilitaire doit modifier certains des fichiers de configuration liés à crontab situés dans /etc qui ne sont pas accessibles en écriture pour les utilisateurs normaux. Ainsi, les commandes crontab/at sont définies avec SUID afin d'écrire des données.

Les fichiers avec le bit SUID montrent donc un **s remplaçant le x sur les permissions de l’utilisateur** …  
**>> ls -ld test.sh  
-rwsr-xr-x 1 tux tux 0 Oct 20 16:46 test.sh**

Il est possible de combiner plusieurs permissions spéciales dans un même paramètre en les additionnant.   
Ainsi, pour mettre SGID (valeur 2) et SUID (valeur 4) en mode numérique pour le script test.sh   
avec les permissions 755…  
**>> chmod 6755 test.sh  
-rwsr-xr-x 1 tux tux 0 Oct 20 16:46 test.sh**

Si le terminal prend en charge la couleur, ce qui est le cas de la plupart des terminaux actuels, on peut constater si ces permissions spéciales sont définies en jetant un coup d’œil à la sortie de ls -l.

Les fichiers avec le bit SUID montrent donc un **s remplaçant le x sur les permissions de l’utilisateur** …  
**>> ls -ld test.sh  
-rwsr-xr-x 1 tux tux 0 Oct 20 16:46 test.sh**

Il est possible de combiner plusieurs permissions spéciales dans un même paramètre en les additionnant.   
Ainsi, pour mettre SGID (valeur 2) et SUID (valeur 4) en mode numérique pour le script test.sh   
avec les permissions 755…  
**>> chmod 6755 test.sh  
-rwsr-xr-x 1 tux tux 0 Oct 20 16:46 test.sh**

Si le terminal prend en charge la couleur, ce qui est le cas de la plupart des terminaux actuels, on peut constater si ces permissions spéciales sont définies en jetant un coup d’œil à la sortie de ls -l.

Les utilitaires (les plus utilisés) qui possèdent la permission SUID sont …

* /usr/bin/chsh  
  Mondifier l'interpréteur de commandes initial ;
* /usr/bin/chfn  
  Modifier le nom complet et les informations associées à un utilisateur ;
* /usr/bin/passwd  
  Modification d’un mot de passe d’un utilisateur ;
* /usr/bin/umount  
  Démonter un système de fichier ;
* /usr/bin/newgrp  
  Se connecter avec un nouveau groupe ;
* /usr/bin/su  
  Se connecter comme l,utilisateur root ;
* /usr/bin/mount  
  Monter un système de fichier ;
* /usr/bin/sudo  
  Exécuter une commande au nom d’un autre utilisateur ;
* /usr/bin/fusermount   
  Démonter un système de fichier FUSE;
* /usr/bin/gpasswd  
  Modifier le mot de passe d’un groupe.

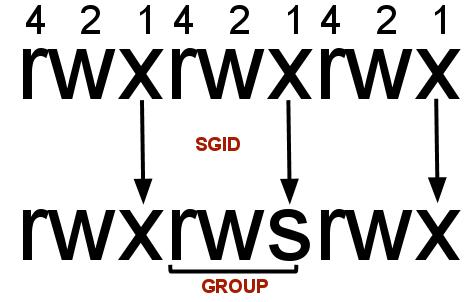
**Remarque** …  
Il existe un mécanisme de contrôle qui empêche un utilisateur de modifier le mot de passe d’un autre utilisateur.  
Ce mécanisme est codé au sein de l’utilitaire passwd, et non au niveau du système d'exploitation et du schéma SUID.

# Set GID

Communément appelée SGID, cette autorisation spéciale possède plusieurs fonctions …

* Si elle est définie sur un **fichier**, elle **permet au fichier d'être exécuté en tant que groupe propriétaire** du fichier (similaire à SUID) ;
* Si elle est définie sur un **répertoire**, **tous les fichiers créés dans le répertoire auront leur propriété de groupe** **définie sur celle du propriétaire du répertoire**.

Cet ensemble d'autorisations est **noté par un s minuscule où le x indiquerait normalement les privilèges d'exécution** **pour le groupe**.   
Il est également particulièrement utile pour les répertoires qui sont souvent utilisés dans les **efforts de collaboration entre les membres d'un groupe**.   
**Tout membre du groupe peut accéder à tout nouveau fichier**.   
Cela s'applique également à l'exécution des fichiers.   
SGID est très puissant lorsqu'il est utilisé correctement.



Comme indiqué précédemment pour SUID, si le **groupe propriétaire n'a pas d'autorisations d'exécution**, un **S majuscule est utilisé**.

Le Set GID possède la **valeur octale 2 et en mode symbolique est représenté par un s** sur les **permissions du groupe**.

**SGID est similaire à SUID**.   
La différence entre les deux est que SUID assume le propriétaire des autorisations de fichier et **SGID assume les autorisations du groupe lors de l'exécution d'un fichier** au lieu des autorisations d'héritage de l'utilisateur connecté.

**En résumé** …  
Quand un utilisateur crée un fichier dans un répertoire dont il est membre du groupe, le **fichier prendra les permissions du groupe courant**.  
Quand le SGID est fixé sur un répertoire, le fichier créé par l’utilisateur prendra les droits du groupe du répertoire.

En conséquence, tous les fichiers créés **quel que soit l’utilisateur appartiendront au groupe du répertoire**.

Les fichiers et répertoires avec un bit SGID montrent un **s remplaçant le x sur les permissions du groupe** sur la sortie de ls -l …  
**>> ls -l test.sh  
-rwxr-sr-x 1 tux tux 33 Oct 20 16:46 test.sh**

Pour ajouter des permissions SGID à un fichier en mode symbolique …  
**>> chmod g+s test.sh  
>> ls -l test.sh  
-rwxr-sr-x 1 tux tux 33 Oct 20 16:46 test.sh**

L’exemple suivant permet de mieux visualiser les effets du SGID sur un répertoire.   
En supposant qu’il existe un répertoire appelé documents/, dont les propriétaires sont l’utilisateur tux et le groupe users, avec la structure de permission suivante …  
**>> ls -ldh documents/  
drwxr-xr-x 2 tux tux 4,0K Oct 20 16:46 documents/**

Maintenant, on se déplace dans le répertoire documents/ et, à l’aide de la commande touch, on crée un fichier vide à l’intérieur.   
Le résultat serait …  
**>> cd documents/  
>> touch rapport.texte  
>> ls -lh rapport.texte  
-rw-r--r-- 1 tux tux 0 Oct 20 16:46 rapport.texte**

Ce nouveau fichier est la propriété de l’utilisateur tux et du groupe tux.   
Toutefois, si le répertoire possédait les permissions SGID définies, le résultat serait différent.

Tout d’abord, on ajoute le bit SGID au répertoire documents/ …  
**>> sudo chmod g+s documents/  
>> sudo ls -ldh documents/  
drwxr-sr-x 2 tux gestionnaires 4,0K Oct 20 16:46 documents/**

Le s sur les permissions de groupe indique que le bit SGID est activé.   
Maintenant, on se déplace dans ce répertoire et, à nouveau, on crée un fichier vide à l’aide de la commande touch …  
**>> cd documents/  
>> touch tableau.texte  
>> ls -lh tableau.texte  
-rw-r--r-- 1 tux gestionnaires 0 Oct 20 16:46 tableau.texte**

Comme on peut le constater, le groupe qui possède le fichier est gestionnaires.   
Cela est dû au fait que le bit SGID implique que le nouveau fichier hérite du groupe propriétaire de son répertoire parent, qui est users.

# Un peu plus …

**Commande stat**La commande stat affiche des informations précises sur un fichier …  
**>> stat $(which passwd)  
 Fichier : /usr/bin/passwd  
 Taille : 63960 Blocs : 128 Blocs d'E/S : 4096 fichier  
Périphérique : 801h/2049d Inœud : 1177437 Liens : 1  
 Accès : (4755/-rwsr-xr-x) UID : ( 0/ root) GID : ( 0/ root)  
 Accès : 2021-09-08 08:32:44.000000000 -0400  
Modif. : 2020-02-07 09:54:14.000000000 -0500  
Changt : 2021-09-08 08:32:52.229140474 -0400  
 Créé : 2021-09-08 08:32:52.057140480 -0400**

**Recherche – Permissions spéciales**Afin de rechercher les fichiers et répertoires ayant des permissions spéciales …  
**>> sudo find / -perm /4000 2> /dev/null #SUID  
>> sudo find / -perm /2000 2> /dev/null #SGID  
>> sudo find / -perm /1000 2> /dev/null #Sticky bit**

# Commande newgrp

# La commande newgrp est utilisée afin de modifier le GID actuel (ID de groupe) lors d'une session de connexion.

# Si un tiret (-) est inclus comme argument, l'environnement de l'utilisateur est initialisé comme s'il s'était connecté ; sinon, l'environnement de travail actuel demeure inchangé.

# La commande newgrp modifie le groupe courant réel ID au spécifié groupe, ou, si aucun groupe est spécifié, au groupe par défaut répertorié dans le fichier /etc/passwd.

# Si l'utilisateur est root, il n'est pas invité à saisir un mot de passe. Si l'utilisateur n'est pas root, il est invité à saisir un mot de passe de groupe si …

# l'utilisateur n'a pas de mot de passe, mais le groupe en possède un, ou si

# l'utilisateur n'est pas répertorié en tant que membre du groupe et le groupe possède un mot de passe.

# Si aucun mot de passe de groupe n'est défini et que l'utilisateur n'est pas répertorié comme membre du groupe, l'accès lui est refusé.

# S'il existe une entrée pour le groupe dans le fichier de mot de passe du groupe masqué, /etc/gshadow, la liste des membres et le mot de passe de ce groupe sont extraits de ce fichier. Sinon, l'entrée de groupe dans /etc/group est utilisée.

## Syntaxe

# newgrp [-] [ groupe ]

# La commande newgrp utilise les fichiers suivants …

* **/etc/passwd** Informations sur le compte utilisateur ;
* **/etc/shadow** Informations sécurisées du compte utilisateur ;
* **/etc/group** Informations sur le compte de groupe ;
* **/etc/gshadow** Informations de compte de groupe sécurisées.

Exemples …

**>> sudo groupadd développeurs   
>> newgrp développeurs**Tente de se connecter au groupe de développeurs  
**>> newgrp - développeurs**Tente de se connecter au groupe développeurs et, en cas de succès, réinitialise l'environnement utilisateur.

# Commande gpasswd

La commande gpasswd modifie les mots de passe des groupes.

Les mots de passe de groupe sont stockés dans les fichiers /etc/group et /etc/gshadow.   
/etc/group contient des informations de groupe et /etc/gshadow contient des versions chiffrées des informations de groupe.

## Syntaxe

**gpasswd [ choix ] groupe**

## Options

Les principales options de la commande gpasswd sont …  
(Sauf pour les options -A et -M, les options suivantes ne peuvent pas être combinées.)

|  |  |
| --- | --- |
| Option | Fonction |
| a ou -- add <Utilisateur> | Ajout d’un utilisateur au groupe nommé |
| d ou -- delete <Utilisateur> | Suppression de l'utilisateur du groupe nommé |
| h ou --help | Affichage du message d'aide |
| r ou --remove <Mot de passe> <groupe> | Suppression du mot de passe du groupe nommé Le mot de passe du groupe sera vide Seuls les membres du groupe seront autorisés à utiliser newgrp pour rejoindre le groupe nommé |
| R ou restrict <Groupe> | Restriction de l'accès au groupe nommé Le mot de passe du groupe est défini sur ! Seuls les membres du groupe avec un mot de passe seront autorisés à utiliser newgrp pour rejoindre le groupe nommé |
| A ou --administrators <Utilisateur, …> | Définition de la liste des utilisateurs administratifs |
| M ou --members <Utilisateur, …> | Définition de la liste des membres du groupe. |

**Examples**

**>> sudo gpasswd -a mozart musicien**Ajoute l'utilisateur mozart au groupe musiciens

**>> sudo gpasswd -A tux mathématiciens**Accorde à l'utilisateur tux les droits d'administration au groupe mathématiciens

**>> sudo gpasswd -d tux expos**Supprime l'utilisateur tux du groupe expos

# Commande umask

Sur Linux, de nouveaux fichiers sont créés avec un jeu par défaut pour les autorisations.

Plus précisément, les autorisations d'un nouveau fichier peuvent être restreintes d'une manière spécifique en appliquant un masque d'autorisations appelé umask.   
La commande umask est utilisée pour définir ce masque, ou pour afficher sa valeur actuelle.

## Syntaxe

**umask [-S] [ masque ]**

**Options** …

* **-S**Acceptation d’une représentation symbolique d'un masque ou son renvoie ;
* **masque**  
  Si un masque valide est spécifié, le masque est défini sur cette valeur.   
  Si aucun masque n'est spécifié, la valeur du masque actuelle est renvoyée.

## Rappel

Chaque fichier d’un système est associé à un ensemble d'autorisations utilisées afin de protéger les fichiers : les autorisations d'un fichier déterminent quels utilisateurs peuvent accéder à un fichier/répertoire et quel type d'accès ils y ont.

Il existe trois catégories générales d'utilisateurs …

* L'utilisateur qui possède le fichier/répertoire (Utilisateur ou *User*) ;
* Utilisateurs appartenant au groupe de propriété défini pour un fichier/répertoire (Groupe ou *Group*) ;
* Tout le monde (Autre ou *Others*).

À son tour, pour chacune de ces classes d'utilisateurs, il existe trois types d'accès aux fichiers …

* Possibilité de consulter (lire) le contenu du fichier/répertoire (Read) ;
* Possibilité de modifier le contenu du fichier/répertoire (Write) ;
* Possibilité d'exécuter un fichier ou accéder à un répertoire (Exécuter).

Ainsi, pour chacune des trois classes d'utilisateurs, il existe trois types d'accès.   
Ensemble, ces informations constituent les autorisations du fichier

## Spécification du masque de création de fichier à l'aide de symboles

La représentation d’une classe d'utilisateurs peut être un ou plusieurs des suivantes …

* u Utilisateur (le propriétaire du fichier/répertoire) ;
* u Groupe (tout membre du groupe défini du fichier/répertoire) ;
* o N'importe qui d'autre ;
* a Tous (équivalent à ugo).

L'opérateur d'autorisations peut être l'un des suivants …

* + Autoriser l'activation des autorisations de fichier spécifiées pour les classes d'utilisateurs spécifiées   
   (les autorisations qui ne sont pas spécifiées sont inchangées dans le masque) ;
* - interdire l'activation des autorisations de fichier spécifiées pour les classes d'utilisateurs spécifiées   
   (les autorisations qui ne sont pas spécifiées demeurent inchangées dans le masque) ;
* = autoriser l'activation des autorisations de fichier spécifiées pour les classes d'utilisateurs spécifiées   
   (les autorisations non spécifiées seront interdites par le masque lors de la création du fichier).

Par exemple, la commande umask suivante …

**umask u+w**Cette commande définit le masque de sorte que lorsque les fichiers sont créés, ils disposent d'autorisations qui autorisent l'autorisation d'écriture pour l'utilisateur (propriétaire du fichier).   
Le reste des autorisations du fichier serait inchangé par rapport à la valeur par défaut du système d'exploitation.  
**>> umask   
0022  
>> umask u+w  
0022**

**umask u-x,g=r,o+w**Plusieurs modifications peuvent être spécifiées en séparant plusieurs ensembles de notations symboliques par des virgules (sans espace).  
Cette commande définit le masque de sorte que lorsque les fichiers suivants sont créés, ils disposent des autorisations suivantes …  
interdire l'autorisation d'exécution d'être définie pour le propriétaire du fichier (utilisateur), tout en laissant les autres autorisations du propriétaire inchangées ;  
activer l'autorisation de lecture pour le groupe, tout en interdisant l'autorisation d'écriture et d'exécution pour le groupe ;  
activer l'autorisation d'écriture pour les autres, tout en laissant le reste des autres autorisations inchangé.  
**>> umask   
0022  
>> umask u-x,g=r,o+w  
0130**

**umask a=**L’utilisation de l'opérateur égal (=), toutes les autorisations non spécifiées seront spécifiquement interdites.   
Par exemple, cette commande définit le masque de création de fichier afin que les nouveaux fichiers soient inaccessibles à tout le monde.  
**>> umask   
0022  
>> umask a=  
0777**

## Spécification du masque de création de fichier à l'aide d'une représentation numérique

Le masque de création de fichier peut également être représenté numériquement, en utilisant des valeurs octales   
(les chiffres de 0 à 7).   
Lors de l'utilisation de la représentation numérique octale, certains nombres représentent certaines autorisations, et ces nombres sont ajoutés ou soustraits les uns aux autres pour représenter la valeur d'autorisations combinée finale.

La commande umask masque les autorisations en les restreignant d'une certaine valeur.

Essentiellement, chaque chiffre du masque est soustrait de la valeur par défaut du système d'exploitation pour arriver à la valeur par défaut que l’on définit.   
Ce n'est pas vraiment une soustraction ; techniquement, le masque est nié (son compliment au niveau du bit est pris) et cette valeur est ensuite appliquée aux autorisations par défaut à l'aide d'une opération ET logique .   
Le résultat est que le masque indique au système d'exploitation quels bits d'autorisation désactiver lorsqu'il crée un fichier.

Sous Linux, la valeur des autorisations par défaut est 666 pour un fichier normal et 777 pour un répertoire.   
Lors de la création d'un nouveau fichier ou répertoire, le noyau prend cette valeur par défaut, soustrait la valeur du masque et donne aux nouveaux fichiers les autorisations résultantes.

Ce tableau indique comment chaque chiffre de la valeur du masque affecte les nouvelles autorisations de fichier et de répertoire.

|  |  |  |
| --- | --- | --- |
| Valeur du masque | Autorisations de fichier par défaut (666) | Autorisations de répertoire par défaut (777) |
| 0 | rw | rwx |
| 1 | rw | rw |
| 2 | r | rx |
| 3 | r | r |
| 4 | w | wx |
| 5 | w | w |
| 6 | x | x |
| 7 | Aucune autorisation n'est autorisée | Aucune autorisation n'est autorisée |

Donc, si la valeur du masque est 022 …

* Tous les nouveaux fichiers auront, par défaut, les autorisations 644 (666 - 022) ;
* Tout nouveau répertoire sera, par défaut, créé avec les autorisations 755 (777 - 022).