

Présentation des autorisations POSIX

Les permissions POSIX (*Portable Operating System Interface*) sous Linux vous permettent d'attribuer des droits d'accès aux utilisateurs ainsi qu'aux groupes sur les dossiers et les fichiers. Elles sont identiques aux permissions traditionnelles Unix.

Le système de sécurité POSIX s'applique en toutes circonstances, que l'utilisateur accède à un fichier ou un dossier à partir localement de son poste ou d'un ordinateur sur le réseau.



NOTE: POSIX est régit par le standard IEEE 1003.1. Pour plus d'informations sur POSIX, consulter le site du IEEE au <http://posixcertified.ieee.org>.

Les permissions POSIX

Avant de débiter, voyons un peu de théorie sur l'application des permissions POSIX. Car il bien beau de savoir tous ces permissions, mais si vous ne pouvez pas donner les autorisations correctement, ces dernières ne servent plus vraiment.

L'application des permissions devrait toujours se faire au niveau des dossiers et que très rarement au niveau des fichiers, ce pour différentes raisons, soit:

- Pour faciliter la gestion des permissions,
- Permettre la diminution du TCO,
- Permettre d'éviter les erreurs qui causeraient des gains d'autorités, ce qui entraînerait des accès illicites à des données sensibles.

Si pour une raison quelconque vous êtes obligé de mettre des permissions sur des fichiers, dites-vous que votre structure de répertoire est, soit mal conçu ou est rendu inadéquate pour vos besoins et qu'il est temps de penser à la refaire avant que cela devienne trop complexe à gérer (et non ce n'est pas juste théorique).

NOTES



Il existe cependant des cas d'exceptions en Linux, où il vous pourriez mettre des permissions différentes sur un fichier, qui sont:

- Lorsque vous avez un besoin vraiment spécifique en terme de permissions (mais ne vous servez-vous pas de ce prétexte pour le faire);
- Lorsque vous avez une application qui nécessite les permissions *SUID*, *SGID* ou *Sticky*. Et encore, il est fortement recommandé de ne plus utiliser la permission sticky.

En prenant le temps de bien planifier vos autorisations POSIX et en observant quelques règles élémentaires, vous devriez arriver à une structure de répertoire *viable* et *extensible*. Nous vous conseillons de bien comprendre les concepts des permissions POSIX de base, décrite dans cette unité, mais d'y appliquer les extensions d'attributs d'ACL, ce qui vous simplifiera amplement la tâche.

Voici quelques règles d'administration qui saurons vous aider et dont vous devriez déroger le moins possible.

- Pour simplifier l'administration des permissions, regroupez vos fichiers dans des répertoires d'accès commun;
- Créer des groupes en fonction du type d'accès nécessaire par les utilisateurs qui doivent accéder aux ressources, puis octroyez leurs des permissions appropriées à ces groupes. Attribuez le moins possible des autorisations à des comptes utilisateurs que si cela s'avère vraiment nécessaire;
- Donner aux utilisateurs que les permissions minimales nécessaires (et non ce qu'ils demandent). Si un utilisateur a le besoin de lire un fichier, intégrez le dans un groupe et accordez lui l'autorisation *lecture* seulement.
- Et enfin, accorder toujours les permissions à un niveau des répertoires à la place des fichiers.

NOTES



Les permissions au niveau dossier

Pour contrôler l'accès des utilisateurs aux dossiers, aux fichiers et aux sous-dossiers, vous pouvez attribuer des permissions au niveau dossier.

Le tableau suivant décrit les autorisations POSIX que l'on peut attribuer au niveau des dossiers, ainsi que le type d'accès permis aux intervenants:

| Autorisation POSIX au niveau dossier | Permet à l'utilisateur |
|--------------------------------------|---|
| Lecture (r) | Permet d'afficher les fichiers et sous-dossiers du dossier ainsi qu'afficher le propriétaire, les permissions et les attributs du dossier. |
| Écriture (w) | Permet de créer et supprimer de nouveaux sous-dossiers et fichiers dans le dossier. |
| Exécution (x) | Permet de se déplacer d'un dossier à l'autre pour accéder à d'autres fichiers et sous-dossiers, ce même si l'utilisateur ne dispose d'aucune permission de lecture ou d'écriture pour ces dossiers. |
| Setuid (s) | Permet d'ouvrir le répertoire sous l'utilisateur du fichier. |
| Setguid (g) | Permet d'ouvrir le répertoire sous le groupe propriétaire du fichier. Très utile lors de l'utilisation avec les ACL(EA). |
| Sticky (t) | N'utilisez pas cet attribut. |
| Refus (-) | Indique un refus d'autorisation sur l'une des permissions r,w,x,s,g,t. |

Un exemple de ce que ressemblerait les permissions au niveau d'un dossier.

```
drwxr-xr-x 2 imurphy users 4096 May 15 07:37 cybiolab
```

NOTES



Caractéristiques des permissions au niveau dossier

À la création d'un répertoire, ce dernier:

- Par défaut, obtient les permissions du masque par défaut.
- Le créateur du répertoire devient alors le propriétaire du répertoire et son groupe principal devient le groupe propriétaire.
- Pour qu'un utilisateur puisse accéder à un fichier au sein du répertoire, l'intervenant doit avoir les droits d'accès à toute la hiérarchie des répertoires qui mène au fichier. Le concept des chemins UNC ne s'applique pas sous Linux.

Aussi, l'application des permissions sur les répertoires se font:

- L'application des autorisations sont linéaires et se font dans l'ordre: utilisateur propriétaire, groupe propriétaire et autres.
- Les autorisations ne sont pas cumulatives.
- Enfin, il n'existe aucun concept d'héritage de permissions similaire à celui du système de fichier NTFS avec les permissions de base POSIX.

Les permissions au niveau fichier

Pour contrôler l'accès des utilisateurs aux fichiers, vous pouvez attribuer des permissions au niveau fichier.

Le tableau suivant décrit les autorisations POSIX que l'on peut attribuer au niveau des fichiers, ainsi que le type d'accès permis aux intervenants:

| Autorisation POSIX au niveau fichier | Permet à l'utilisateur |
|--------------------------------------|---|
| Lecture (r) | Permet de lire le contenu d'un fichier. Contrairement aux permissions sous Windows, c'est la permission exécuter (x) sur le dossier qui permet de d'afficher les attributs, le propriétaire et les autorisations du fichier. |
| Écriture (w) | Permet de modifier le contenu d'un fichier. |

NOTES



| | |
|---------------|--|
| Exécution (x) | Cette permission permet de rendre un fichier exécutable. |
| Setuid (S) | Permet d'exécuter l'application sous le compte du propriétaire du fichier. A utiliser avec précaution sur un fichier. |
| Setgid (S) | Permet d'exécuter l'application sous le groupe propriétaire du fichier. A utiliser avec précaution sur un fichier. |
| Sticky (T) | Elle permet d'exécuter une application avec les permissions de l'utilisateur propriétaire. Cette permission ne devrait plus être utilisée. |
| Refus (-) | Indique un refus d'autorisation sur l'une des permissions r,w,x,S,T. |

Un exemple de ce que ressemblerait les permissions au niveau d'un fichier.

```
-rw-r--r-- 1 imurphy root 42 Jul 3 19:43 test.txt
```

Caractéristiques des permissions au niveau fichier

À la création d'un fichier, ce dernier:

- Par défaut, obtient les permissions du masque par défaut.
- Le créateur du répertoire devient alors le propriétaire du répertoire et son groupe principal devient le groupe propriétaire.

Lors de la modification d'un fichier par un utilisateur différent de l'actuel propriétaire, ce fichier:

- Conserve les permissions actives du fichier.
- L'utilisateur propriétaire et son groupe propriétaire est remplacé par celui de l'utilisateur qui a apporté les modifications.

L'application des permissions sur les fichiers se font:

- l'application des autorisations sont linéaires et se font dans l'ordre: utilisateur propriétaire, groupe propriétaire et autres.
- les autorisations ne sont pas cumulatives.

NOTES



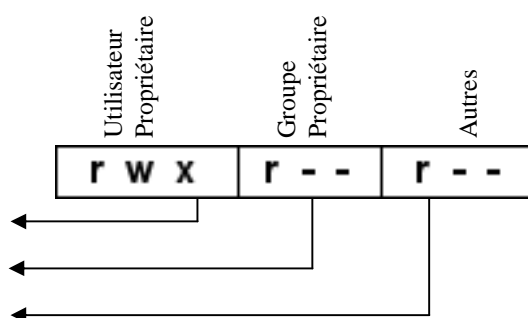
Valeur symbolique des permissions

Cette première méthode de décrire les permissions sur les objets est appelée symbolique car elle est représentée par la présence de la lettre r, w ou x. Celles-ci accordent différentes autorisations, tandis qu'un tiret '-' représente pour ça part un refus.

- r:** signifie possibilité de lire ce fichier / dans ce répertoire,
- w:** signifie possibilité d'écrire dans ce fichier / répertoire,
- x:** signifie possibilité d'exécuter ce fichier / d'aller dans ce répertoire.

Liste de contrôle d'accès POSIX

Utilisateur: r w x
Groupe: r - -
Autres: r - -



Valeur octale des permissions

Il existe une autre méthode de décrire les permissions sur les objets. Cette méthode assez simple, consiste à remplacer les lettres par une valeur octale.

| Octale | Symbole | Permission |
|--------|---------|------------|
| 1 | r | Lecture |
| 2 | w | Écriture |
| 4 | x | Exécution |

En additionnant tout ces chiffres, nous pouvons obtenir une valeur maximale de 7.

Donc pour une permission décrite en octale de 754, nous obtiendrons les permissions symboliques rwxr-xr--, soit que:

- l'utilisateur propriétaire a les droits de lecture, écriture et exécution.
- Le groupe propriétaire de lecture et exécution.
- Et enfin les autres ont le droit de lecture seulement.

NOTES



Aussi, il existe un quatrième octale, permettant de définir les permissions spéciales sur nos objets: SUID, SGID et Sticky.

| Quatrième Octale | Symbole | Permission |
|------------------|---------|------------|
| 1 | T | Sticky |
| 2 | S | SGID |
| 4 | S | SUID |

NOTES



Autorisations spéciales POSIX

Voici un tableau qui donne des explications plus détaillée concernant les permissions spéciales POSIX.

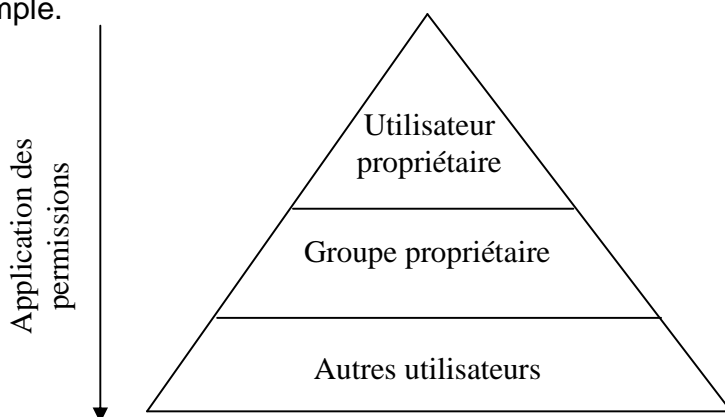
| Autorisation spéciale | Description |
|-----------------------|--|
| Set User ID (SUID) | <p>Utilisé pour les applications seulement, cette permission indique que l'application fonctionne comme sous les permissions du propriétaire du fichier et non comme l'utilisateur qui exécute l'application.</p> <p>Cette permission est représentée par le symbole s à la place du x dans la colonne de <i>l'utilisateur propriétaire</i>. Si l'utilisateur propriétaire n'a pas les droits d'exécution, le s sera affiché en majuscule.</p> |
| Set Group ID (SGID) | <p>Utilisé principalement pour les applications, cette permission indique que l'application fonctionne comme sous le groupe propriétaire du fichier et non comme le groupe exécutant l'application. Si cette permission est appliquée au répertoire, tous les fichiers que l'utilisateur va créer à l'intérieur de ce répertoire va être identifié au groupe propriétaire du répertoire, plutôt que par le groupe principal de l'utilisateur.</p> <p>Cette permission est représentée par le symbole s à la place du x dans la colonne du <i>groupe propriétaire</i>. Si le groupe propriétaire n'a pas les droits d'exécution, le s sera affiché en majuscule.</p> |
| Sticky | <p>Utilisé principalement sur les répertoires, ce bit indique que le fichier créé dans ce répertoire peut être enlevé uniquement par l'utilisateur qui a créé le fichier. Cette permission est représentée par le symbole t à la place du x dans la colonne '<i>autre</i>'.</p> |

Attribution des permissions POSIX

L'attribution des permissions POSIX obéit à certain nombre de règles.

Ordre d'application des autorisations POSIX

L'ordre d'application des autorisations POSIX se font dans un ordre linéaire, soit utilisateur propriétaire, groupe propriétaire et enfin les autres utilisateurs. La première règle qui correspond sera appliquée laissant les suivantes à l'écart. En somme il n'y a pas de cumul de permissions comme le ferait les permissions NTFS de Windows par exemple.



Pour démontrer l'ordre d'application des autorisations POSIX, nous allons prendre un répertoire ayant les autorisations suivantes:

```
drwxr-xrwx 2  jdraper    users 4096  May 22  10:09  test1
```

Imaginons que l'utilisateur *jdraper* fait parti du *groupe users* et accède au *répertoire test1*. Le système Posix va vérifier chacune des autorisations, dans l'ordre respectif pour voir si ces dernières s'appliquent à l'utilisateur. Cette ordre sera alors de l'utilisateur *propriétaire*, *groupe propriétaire* et les *autres utilisateurs*.

Étant donné que *jdraper* est l'utilisateur propriétaire, cette autorisation va lui être appliquée, le système Posix arrêtera sa vérification et les autres permissions seront abandonnées.

NOTES



Commande umask

Affiche ou définit le masque de création des fichiers et répertoires.

Syntaxe:
`umask masque`

Exemple:
`umask 0022`

Donc lors de la création d'un fichier ou d'un répertoire, vous aurez par défaut les permissions la valeur maximale pour l'objet moins le masque. Par défaut ce masque est de 0022.

Pour un fichier nous obtiendrons: 0666 - 0022 soit 0644

Pour un répertoire nous obtiendrons: 0777 - 0022 soit 0755

Attribution des permissions spéciales POSIX

Pour attribuer des permissions spéciales pour un fichier ou un répertoire nous devons utiliser le quatrième octale disponible.

Par exemple, sur un répertoire ayant comme permission de base 755. Vous pouvez définir les permissions spéciales suivantes en utilisant les commandes,

| Permission spéciale Posix | Attribution en octale | Attribution en symbolique |
|---------------------------|-------------------------|---------------------------|
| SUID | <code>chmod 1755</code> | <code>chmod u+s</code> |
| SGID | <code>chmod 2755</code> | <code>chmod g+s</code> |
| Sticky | <code>chmod 4755</code> | <code>chmod o+t</code> |

Aussi vous pouvez effectuer des combinaisons, soit

Comme vous pouvez le constater, le x se retrouve caché par la lettre d'attribution de la permission spéciale Posix. La case de la lettre des permissions spéciales joueront un rôle important.

Les lettres s ou t en minuscules indiquent que l'utilisateur, le groupe ou les autres, selon la position, a la permission d'exécuter (x) en plus. Les lettres S ou T en majuscules indiquent que la permission exécuter (x) n'est pas attribué.

NOTES



Copie et déplacement de fichiers et des répertoires

Avant de débiter, voici un petit rappel. Un fichier peut avoir une permission octale maximale de 0666, ainsi qu'un répertoire peut avoir une permission octale maximale de 0777. Le masque par défaut est défini à 0022.

Donc par défaut, tout nouveau fichier créer aura les permissions 0644, car nous prenons les permissions maximales de 666 en lui soustrayant le masque par défaut de 022, ce qui nous donnera les permissions appliquées de 644.

$$\begin{array}{rcl} \text{Permission maximale} & - & \text{Masque par défaut} = \text{Permission appliquée} \\ \text{d'un fichier} & & \\ 0666 & - & 0022 & = & 0644 \end{array}$$

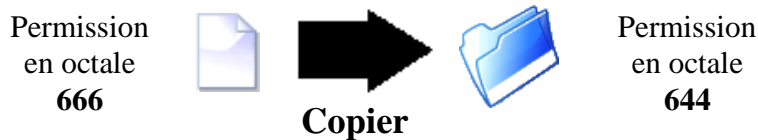
Il va de même pour les permissions sur les répertoires. Un répertoire nouvellement créer aura comme permissions 0755. En prenant les permissions maximales de 777 et en soustrayant le masque 022, nous obtiendrons des permissions appliquées de 755.

$$\begin{array}{rcl} \text{Permission maximale} & - & \text{Masque par défaut} = \text{Permission appliquée} \\ \text{d'un répertoire} & & \\ 0777 & - & 0022 & = & 0755 \end{array}$$

Copie de fichiers et de répertoires

Lorsque vous copiez des fichiers d'un répertoire ou d'un point de montage vers un autre, les permissions changent, et deviennent celle appliqué avec le masque.

Donc prenons un fichier ayant comme permission initiale de 666 et qui est copié dans un autre répertoire sur le même ordinateur. Il lui sera attribué les permissions 644.



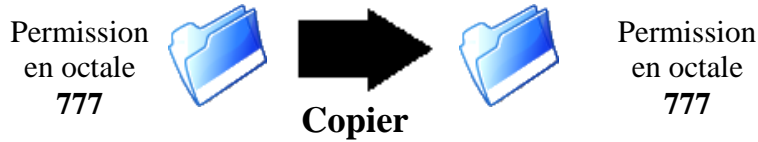
Cela est du au fait que le fichier copié est un nouveau fichier, et par conséquent doit appliquer la permission maximale moins le masque.

Si l'utilisateur propriétaire et le groupe propriétaire initiale sont respectivement drobbins et users assigné au fichier, ces derniers changeront lorsque le fichier est copié.

NOTES



Contrairement aux fichiers, les répertoires lorsqu'ils sont copiés conservent leurs permissions



L'utilisateur propriétaire et le groupe propriétaire change aussi lorsque le répertoire est copié d'un répertoire à un autre.



NOTE: Pour qu'une copie conserve ses permissions initiales, utiliser le commutateur **-p** avec la commande cp.

Déplacement de fichiers et de répertoires

Lorsque vous déplacez des fichiers d'un répertoire ou d'un point de montage vers un autre, les permissions ne changent pas.

Prenons un nouvel exemple avec fichier ayant comme permission initiale de 666. Ce dernier est alors déplacé dans un autre répertoire sur le même ordinateur. Il conservera ses permissions initiales. Cela est dû au fait qu'il s'agit du même fichier.



L'utilisateur propriétaire et le groupe propriétaire sont conservés lorsque le fichier est déplacé.

Tout comme les fichiers, les répertoires lorsqu'ils sont déplacés conservent leurs permissions initiales. Aussi, l'utilisateur propriétaire et le groupe propriétaire sont conservés lorsque le répertoire est déplacé.



NOTES

