|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Droits Fichiers  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  Droits Fichiers (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  Droits Fichiers (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  Droits Fichiers (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  Droits Fichiers (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  Droits Fichiers (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png  Droits Fichiers (suite)  http://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/512px-Tux.svg.png |  | Droits – Fichiers Linux    **Droits du système de fichiers**  Les **droits du système de fichiers** d’un système basé sur UNIX sont définis pour trois catégories d’utilisateurs …   * l’utilisateur qui possède le fichier ou propriétaire (u pour *uaer*) ; * les autres utilisateurs du groupe à qui appartient le fichier (g pour *group*) ; * tous les autres utilisateurs (o pour other) dont on parle aussi en tant que tous les autres.   Pour les **fichiers**, chaque droit correspondant permet les actions suivantes :   * le **droit en lecture** (r pour *read*) permet à son propriétaire de voir le contenu du fichier ; * le **droit en écriture** (w pour *write*) permet à son propriétaire de modifier et supprimer le fichier ; * le **droit d’exécution** (x pour *execute*) permet à son propriétaire de lancer le fichier comme une commande.   Pour les **répertoires**, chaque droit correspondant permet les actions suivantes :   * le **droit en lecture** (r pour *read*) permet à son propriétaire d’afficher le contenu du répertoire ; * le **droit en écriture** (w pour *write*) permet à son propriétaire d’ajouter ou supprimer des fichiers de ce répertoires ; * le **droit d’exécution** (x pour *execute* ou accès) permet à son propriétaire d’accéder aux fichiers du répertoire.   Ici, le droit en exécution sur un répertoire ne signifie pas uniquement l’autorisation de lire des fichiers dans ce répertoire mais aussi l’autorisation de voir leurs attributs, tels que leur taille et l’heure de modification.  La commande **ls** est utilisée afin d’afficher les informations sur les droits (et davantage) des fichiers et répertoires.  Lorsque cette commande est passée avec l’option **-l**, elle affiche les informations suivantes (dans l’ordre) …   * le **type de fichier** (premier caractère) ; * l’**autorisation d’accès au fichier** (neuf caractères, constitués de trois caractères pour l’utilisateur, le groupe et les autres) ; * le **nombre de liens physiques** vers le fichier ; * le **nom de l’utilisateur propriétaire** du fichier ; * le **nom du groupe** **propriétaire** à qui appartient le fichier ; * la **taille du fichier** en caractères (octets) ; * la **date et heure** du fichier (mtime) ; * le **nom du fichier**.  |  |  | | --- | --- | | Caractère | Signification | | - | fichier normal | | d | répertoire | | l | lien symbolique | | *c* | nœud de périphérique en mode caractère | | *b* | nœud de périphérique en mode bloc | | *F* | tube nommé | | *s* | socket |   *Liste des premiers caractères de la sortie de ls -l*  On utilise les commandes suivantes pour modifier les droits pour un objet d’un système de fichiers …   * Un administrateur peut utiliser la commande **chown** afin de modifier le propriétaire d’un fichier ; * La commande **chgrp** est, quant à elle, utilisée par le propriétaire du fichier ou par l’administrateur pour modifier le groupe propriétaire d’un objet ; * La commande **chmod** est utilisée par le propriétaire du fichier ou par l’administrateur pour modifier les droits d’accès à un fichier ou un répertoire.   La syntaxe de base pour manipuler le fichier travail est la suivante …  **>> chown <Nouvel utilisateur> <Objet>**  **>> chgrp <Nouveau groupe> <Objet>**  **>> chmod [ugoa][+-=][rwxXst][,...] <Objet>**  Il est, par exemple, possible de faire qu’une arborescence de répertoires soit la propriété de l’utilisateur tux et partagée par le groupe tuxo …  **>> cd <Répertooire>**  **>> chown -R tux:gestionnaires .**  **>> chmod -R ug+rwX,o=rX .**  Il existe un bit particulier qui donne des droits particuliers …  Il s’agit du bit collant ***sticky bit*** (t ou T situé à la place du x) des droits pour les autres (o).  En positionnant le *sticky* bit pour un répertoire empêche un fichier de ce répertoire d’être supprimé par un usager qui n’est pas le propriétaire du fichier.  Afin de sécuriser le contenu d’un fichier dans des répertoires pouvant être écrits par tout le monde tels que */tmp* ou dans des répertoires pouvant être écrits par le groupe, il ne faut pas uniquement supprimer le droit du fichier en écriture mais aussi positionner le *sticky* *bit* sur le répertoire.  Sinon, le fichier pourra être supprimé et un nouveau fichier créé avec le même nom par un utilisateur ayant accès en écriture au répertoire.  Voici quelques exemples intéressants de droits de fichiers.  **>> ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/exim4**  **crw------- 1 root root 108, 0 oct. 20 07:12 /dev/ppp**  **-rw-r--r-- 1 root root 2718 oct. 5 22:09 /etc/passwd**  **-rw-r----- 1 root shadow 1741 oct. 5 22:09 /etc/shadow**  **-rwsr-xr-x 1 root root 973824 sept. 23 07:04 /usr/sbin/exim4**  **>> ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src**  **drwxrwxrwt 17 root root 102400 oct. 20 16:42 /tmp**  **drwxrwsr-x 10 root staff 4096 nov. 29 2011 /usr/local**  **drwxr-xr-x 13 root root 4096 sept. 30 11:24 /usr/src**  **drwxrwsr-x 2 root mail 4096 sept. 27 11:07 /var/mail**  **drwxrwxrwt 5 root root 4096 oct. 20 09:14 /var/tmp**  **Mode numérique**  Il existe aussi un mode numérique pour décrire les droits des fichiers avec **chmod**.  Ce mode numérique utilise des nombres en base 8 (radix=8) codés sur 3 ou 4 chiffres.   |  |  | | --- | --- | | Chiffre | Signification | | 1ere valeur (optionnel) | somme de set user ID (=4), set group ID (=2), et sticky bit (=1) | | 2ème valeur | somme des droits de lecture (=4), écriture (=2), et exécution (=1) pour l’utilisateur | | 3ème valeur | identique pour groupe | | 4ème valeur | identique pour autres |   *Mode numérique des droits de fichiers dans les commandes chmod*  Cela peut sembler compliqué mais, en pratique, cela est assez simple.  Si on regarde les premières colonnes (2-10) de la sortie de la commande **ls -l** et que l’a déchiffre en représentation binaire (base 2) les droits des fichiers (le - représentant 0 et rwx représentant 1), les trois derniers chiffres de la valeur numérique du mode devraient donner la représentation des droits du fichier en octal (base 8).  Par exemple …  **>> touch tux.texte tux.document**  **>> chmod u=rw,go=r tux.texte**  **>> chmod 644 tux.document > ls -l tux.texte tux.document**  **-rw-r--r-- 1 root root 0 oct. 20 16:46 titi**  **-rw-r--r-- 1 root root 0 oct. 20 16:46 toto**  Les droits appliqués à un fichier ou à un répertoire venant d’être créé sont restreintes par la commande interne du shell **umask**.   |  |  |  |  | | --- | --- | --- | --- | | umask | Droits des fichiers créés | Droits des répertoires créés | Utilisation | | 0022 | -rw-r--r-- | -rwxr-xr-x | ne peut être écrit que par l’utilisateur | | 0002 | -rw-rw-r-- | -rwxrwxr-x | peut être écrit par le groupe |   *Exemples de valeurs de umask*  De manière générale, Linux utilise par défaut un schéma de groupe privé par utilisateur (UPG).  Un UPG est créé chaque fois qu’un utilisateur est ajouté au système.  Un UPG possède le même nom que l’utilisateur pour lequel il a été créé et cet utilisateur est le seul membre de l’UPG. Un principe d’UPG rend plus sécure le positionnement de umask à 0002 car chaque utilisateur possède son propre groupe privé.  **Astuce...**  Activez UPG en plaçant **umask 002** dans le fichier ~/.bashrc.  Droits pour les groupes d’utilisateurs (*group*)  Afin que des droits attribués à un groupe soient appliquées à un utilisateur particulier, il faut que cet utilisateur soit déclaré membre du groupe. La nouvelle configuration du groupe n’est effective qu’après une [re]connexion de l’utilisateur (ou l’exécution de **exec newgrp**).  Sous Linux, les périphériques ne sont qu’un autre type de fichier. Si vous avez des problèmes pour accéder à des périphériques tel qu’un CD-ROM ou une clé USB depuis le compte d’un utilisateur, il faudra rendre cet utilisateur membre du groupe concerné.  Certains groupes importants fournis par le système permettent à leurs membres l’accès à des fichiers et des périphériques particuliers sans avoir les privilèges de l’administrateur.   |  |  | | --- | --- | | Groupe | Description des fichiers et périphériques accessibles | | dialout | accès complet et direct aux ports série */dev/ttyS[0-3]* | | dip | accès limité aux ports série pour une connexion *dialup IP* (réseau commuté) vers des pairs de confiance | | cdrom | lecteurs et graveurs de CD-ROM, DVD+/-RW | | audio | périphérique audio | | video | périphérique vidéo | | scanner | dispositifs de numérisation (scanners) | | adm | journaux de surveillance du système | | staff | quelques répertoires où effectuer du travail d’administration de début : */usr/local*, */home* |   *Liste des groupes importants fournis par le système pour l’accès aux fichiers*    Astuce...  Certains groupes faisant partie du système, permettent à leurs membres d’exécuter des commandes particulières sans les privilèges de l’administrateur (root).  Horodatage  Il existe trois types d’horodatage pour un fichier GNU/Linux.   |  |  | | --- | --- | | Type | Signification | | mtime | date de modification du fichier (**ls -l**) | | ctime | date de changement d’état du fichier (**ls -lc**) | | atime | date de dernier accès au fichier (**ls -lu**) |   *Liste des types d’horodatage*    **Notes**...   * Écraser un fichier va modifier tous les attributs *mtime*, *ctime* et *atime* du fichier. * Modifier le propriétaire ou les droits d’un fichier va changer les attributs *ctime* et *atime* du fichier. * La lecture d’un fichier va modifier l’attribut *atime* du fichier.     Liens  Il existe deux méthodes pour associer le fichier « toto » avec un nom de fichier différent « titi » :   * Lien physique Nom dupliqué d’un fichier existant **>** **ln tux.texte tux.document** * Lien symbolique ou <symlink> Fichier spécial pointant vers un autre fichier par son nom **> ln -s tux.texte tux.document**   Consultez l’exemple suivant pour des modifications du nombre de liens et les subtiles différences dans le résultat de la commande **rm**.  **> umask 002**  **> echo "Contenu d'origine" > tux.texte**  **> ls -li tux.texte**  **1449840 -rw-rw-r-- 1 tux tux 18 oct. 20 16:50 tux.texte**  **> ln tux.texte tux.document # lien physique**  **> ln -s tux.texte tux.rapport # lien symbolique**  **> ls -li tux.texte tux.document tux.rapport**  **1449840 -rw-rw-r-- 2 tux tux 18 oct. 20 16:50 tux.texte**  **1449840 -rw-rw-r-- 1 tux tux 18 oct. 20 16:50 tux.document**  **1450180 lrwxrwxrwx 2 tux tux 18 oct. 20 16:55 tux.texte -> tux.document**  **> rm tux.texte**  **> echo "Nouveau contenu" > tux.texte**  **> ls -li tux.texte tux.document tux.rapport**  **1450183 -rw-rw-r-- 2 pingouin pingouin 18 oct. 20 16:56 tux.texte**  **1449840 -rw-rw-r-- 1 pingouin pingouin 18 oct. 20 16:50 tux.document**  **1450180 lrwxrwxrwx 2 pingouin pingouin 18 oct. 20 16:55 tux.texte -> tux.rapport**  **>** **cat tux.document**  Contenu d'origine  **> cat tux.texte**  Nouveau contenu  Un lien physique peut être mis en place à l’intérieur du même système de fichiers, il partage le même numéro d’inœud, ce que montre l’option *-i* de la commande **ls**.  Le lien symbolique a les permissions d’accès nominales au fichier de « *rwxrwxrwx* » comme il apparaît dans l’exemple ci-dessus, alors que les permissions d’accès effectives sont celles du fichier vers lequel il pointe.  Attention...  En règle générale — à moins d’avoir une très bonne raison pour cela — il faudrait s’abstenir de créer des liens physiques ou des liens symboliques compliqués. Cela peut provoquer des cauchemars lorsque la combinaison logique des liens symboliques crée une boucle dans le système de fichiers.  Note...  Il est généralement préférable d’utiliser des liens symboliques plutôt que des liens physiques à moins que vous n’ayez une bonne raison d’utiliser un lien physique.  Si vous venez de passer à Linux depuis Windows, la bonne conception d’un système de fichiers UNIX comparé à l’équivalent le plus proche que sont les « raccourcis Windows » deviendra vite claire. Parce qu’il est implémenté dans le système de fichiers, les applications ne voient pas de différence entre un fichier lié et son original. Dans le cas de liens physiques, il n’y a vraiment aucune différence.  **Adapté du document original**  <https://www.debian.org/doc/manuals/debian-reference/ch01.fr.html#_sudo_configuration> |
|  |  |  |