

DEV1 – ENVL – Laboratoires d'environnement**TD 1 – Introduction****Table des matières****1 Rechercher (dans) des fichiers**

Lorsqu'on commence à avoir beaucoup de fichiers, il arrive qu'on ne sache plus trop bien où se trouve un fichier ou quel fichier contient un texte particulier. Les commandes **find** et **grep** sont là pour nous aider.

1.1 Rechercher un fichier

La commande **find** permet de chercher des fichiers dans le système de fichiers en fonction de critères : son nom, sa taille, sa date de création...

find

`find dossierOùChercher critèreDeRecherche...` cherche dans le dossier indiqué (et tous les sous dossiers) les fichiers qui respectent tous les critères indiqués. Par défaut, la commande affiche le nom des fichiers trouvés.

Parmi les critères citons :

▷ `-name nomfichier`

2 Les permissions

Comme Linux est un système partagé, il est important de parler de sécurité. Ici, on va se concentrer sur la sécurité au niveau des fichiers et répondre aux questions suivantes :

- ▷ À qui **appartient** un fichier ?
- ▷ **Qui peut faire** quoi avec un fichier ?

Mais pour commencer il faut d'abord comprendre la notion de *propriétaire* et celle de *groupe*.

2.1 Les groupes d'utilisateurs

Les groupes

Les utilisateurs d'un système Linux sont placés dans des **groupes**. Un groupe contient un ou plusieurs utilisateur(s) et un utilisateur appartient à un ou plusieurs groupe(s).

Exemple : Dans l'exemple ci-contre on peut constater que :

- ▷ l'utilisateur **g12345** appartient aux groupes **users** et **etudiants**;
- ▷ l'utilisateur **g54321** appartient aux mêmes groupes;
- ▷ l'utilisateur **mcd** appartient aux groupes **users** et **enseignants**.

À quoi servent les groupes ? À gérer les permissions. On va pouvoir dire un truc du genre : « Seuls les professeurs peuvent voir le contenu de ce fichier ».

image/groupes.pdf

Visualiser les groupes

La commande **groups** renseigne sur les groupes.

- ▷ `groups` : affiche les groupes de l'utilisateur.
- ▷ `groups loginUtilisateur` : affiche les groupes de l'utilisateur donné.

Le premier groupe de la liste est le **groupe principal**. C'est celui qui sera utilisé par défaut (par exemple, lorsqu'on crée un fichier).

Exercice 1

Visualiser les groupes

- ▷ Visualiser les groupes auxquels vous appartenez.
- ▷ Quel est votre groupe principal ?
- ▷ Quels sont les groupes auxquels appartient votre professeur ?
- ▷ Avez-vous un groupe en commun avec lui ?
- ▷ Quel(s) groupe(s) Linux avez-vous en commun avec les autres étudiants de votre groupe ÉSI ?

Les groupes qui existent concrètement sur une machine sont définis par l'administrateur ^a de la machine. Sur **linux1**, il y a 3 groupes pour les utilisateurs.

- ▷ **users** : tous les utilisateurs sont dans ce groupe
- ▷ **etudiants** : tous les étudiants sont dans ce groupe
- ▷ **enseignants** : tous les professeurs sont dans ce groupe

^a. L'administrateur est la personne qui installe un système d'exploitation et le gère au quotidien : installation de logiciels, gestion des comptes utilisateurs et des groupes, sauvegardes, gestion des panne... Sur Linux, on dit aussi le « super user », le « compte root » ou le « root ».

Attention, on remarque que vous confondez souvent la notion de **groupe d'étudiants** à l'école et les **groupes sur Linux**. C'est un même mot qui recouvre 2 **concepts** complètement **différents**.

2.2 Propriétaire d'un fichier

« La propriété est un piège : ce que nous croyons posséder nous possède. » –

Alphonse Karr

« La propriété c'est le vol. » – Pierre Joseph Proudhon

Propriétaire

Chaque fichier/dossier **appartient** à une personne, son **propriétaire**.

Pour visualiser le propriétaire d'un fichier, on peut utiliser la commande `ls -l`.

On pourra donner des permissions spécifiques au propriétaire du fichier.

Exercice 2

Propriétaire

Visualisez le propriétaire des fichiers de votre dossier personnel.

FAQ

Les fichiers dans mon dossier personnel ne sont pas automatiquement à moi ?

Non. En pratique c'est généralement le cas, mais on peut très bien trouver dans un dossier personnel un fichier qui appartient à quelqu'un d'autre.

2.3 Groupes d'un fichier

Groupe d'un fichier

Chaque fichier/dossier **appartient** à **un et un seul groupe**.

Ces groupes sont les mêmes que ceux utilisés pour grouper les utilisateurs. On pourra donner des permissions spécifiques à toutes les personnes qui sont dans le même groupe que le fichier.

Changer le groupe

`chgrp nouveauGroupe nomFichier` place le fichier indiqué dans le groupe donné.

Par défaut, un fichier est placé dans le groupe principal de celui qui le crée.

Exercice 3

Groupe

- ▷ Visualisez vos fichiers et déterminez à quel groupe ils appartiennent.
- ▷ Créez un fichier de test et modifiez le groupe auquel il appartient.

2.4 Les 3 catégories de personnes

Ce qu'on a déjà vu

Un fichier a un propriétaire et appartient à un groupe.

Restez concentré !

D'expérience nous constatons que vous avez du mal avec les notions qui vont suivre. Restez concentrés et vérifiez que vous avez bien compris.

Lorsque nous allons donner des permissions sur un fichier/dossier, nous allons pouvoir donner des droits différents à trois catégories de personnes : le propriétaire, les utilisateurs qui sont dans le même groupe que le fichier et tous les « autres » (ceux qui ne sont pas dans les deux premières catégories).

2.5 Les permissions sur un fichier

Nous avons vu que nous pouvons spécifier des permissions différentes à trois catégories de personnes. Bien ! Mais quelles permissions pouvons-nous donner ? À quoi correspondent les lettres **r**, **w** et **x** vues dans la capture d'écran précédente ? Commençons par les permissions pour un fichier.

Les permissions sur un fichier

Il existe 3 types de permissions pour un fichier : en lecture, en écriture et en exécution.

r (pour *Read*, la lecture)

Avec ce droit, un utilisateur peut *lire* un fichier, il peut en voir le contenu. Par exemple, avec **cat**.

w (pour *Write*, l'écriture)

Un utilisateur a le droit en écriture sur un fichier, il peut en modifier le contenu. Par exemple, avec **nano**.

x (pour *eXecute*, l'exécution)

Cette permission concerne les exécutables. Un exécutable est un fichier qui contient un programme en langage machine directement exécutable par le processeur. Par exemple, il existe quelque part sur le disque un fichier **nano** qui contient le programme de la commande nano.

Les permissions sont toujours placées dans le même ordre (**rwX**) et un tiret (-) indique que la permission n'est pas accordée.

Exemple. Reprenons la capture d'écran précédente et interprétons les permissions sur le fichier **welcome**.

On y lit que :

- ▷ Les permissions pour le propriétaire sont **rw-**. Donc, (**g12345**) peut voir (**r**) le contenu et modifier (**w**) le fichier **welcome**. Par contre, il ne peut pas l'exécuter.

- ▷ Idem pour les utilisateurs qui sont dans le groupe **users**.
- ▷ Les autres utilisateurs (ceux qui ne sont ni le propriétaire, ni un utilisateur du groupe **users**) peuvent lire le fichier mais pas le modifier ni l'exécuter.

Exercice 4

Déterminez les bonnes permissions

Imaginons que vous écrivez un programme JAVA sur **linux1**. Vous ne voulez pas que quelqu'un copie sur vous. Par contre, vous désirez qu'il puisse exécuter la version compilée. Quelles permissions proposez-vous pour les fichiers :

- ▷ **.java**?
- ▷ **.class**?

Experience 1

Permissions par défaut

Quelle est la situation de départ quand on crée un fichier ? Faisons une petite expérience pour le découvrir.

- ✍ Si ce n'est pas encore fait, créez un dossier **dev1/td2**.
- ✍ Créez-y un fichier vide.
- ✍ Demandez les détails du fichier (propriétaire, groupe, permission)

On constate qu'un nouveau fichier appartient à celui qui l'a créé (on s'en doute) et au groupe principal du créateur. Il y a aussi des permissions par défaut (plutôt permissives dans notre cas).

2.6 Modifier les permissions

Faisons le point !

Nous savons qu'un fichier a un propriétaire et un groupe. Nous savons que nous allons pouvoir donner des permissions différentes au propriétaire, au membres du groupe et à tous les autres utilisateurs. Nous avons aussi appris à comprendre les permissions données sur un fichier.

Très bien mais comment modifier ces permissions ?

Modifier les permissions

`chmod permissions fichier` modifie les permissions du fichier.

Il existe deux façons d'indiquer les permissions : via des nombres ou via des lettres.

2.6.1 Donner les permissions avec des nombres

Pour indiquer les permissions avec un nombre, on considère que $r = 4$, $w = 2$ et $x = 1$. Et on additionne les valeurs ainsi obtenues.

	lettres	nombre	signification
Exemples :	r--	4 (4+0+0)	droit en lecture uniquement
	rw-	6 (4+2+0)	droit en lecture et en écriture
	--x	1 (0+0+1)	droit d'exécution uniquement


On combine alors les nombres obtenus pour les 3 sortes d'utilisateurs.

Exemple : `chmod 640 welcome` donne au propriétaire le droit en lecture/écriture, aux utilisateurs dans le même groupe que le fichier le droit en lecture uniquement et aucun droit aux autres.

Exercice 5

Convertir les permissions en nombres

Reprenez les permissions affichées dans la capture d'écran ci-dessous et exprimez-les avec un nombre de 3 chiffres.



image/perm-nb.pdf

Exercice 6

Modifier les permissions

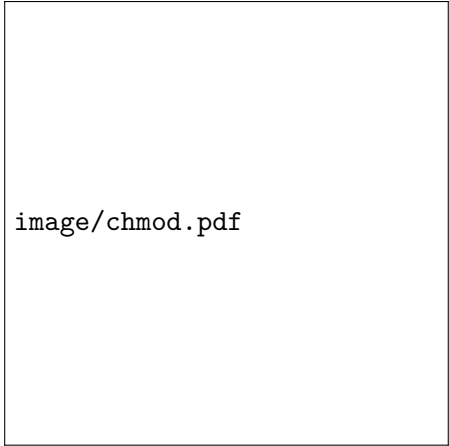
Créez un fichier `dev1/td2/bro1` contenant un petit texte de votre choix. Faites en sorte qu'il soit lisible par tout le monde mais modifiable uniquement par vous.

2.6.2 Donner les permissions avec des lettres

Avec la notation précédente, on spécifie toutes les permissions. Ici, on va aussi pouvoir indiquer uniquement ce qui doit changer : ajouter/enlever un droit sans toucher aux autres droits.

Le format de la commande est assez complexe.

- ▷ On indique d'abord les utilisateurs concernés par la modification des droits :
 - ▷ « u » pour le propriétaire (user),
 - ▷ « g » pour le groupe,
 - ▷ « o » pour les autres (other) et
 - ▷ « a » pour tout le monde (all)
- ▷ On indique ensuite si on veut ajouter (« + ») ou enlever (« - ») un droit



image/chmod.pdf

- ▷ On indique enfin quel(s) droit(s) on ajoute ou enlève (« r », « w » et/ou « x »)

Consultez à la page de manuel pour découvrir toutes les possibilités.

Exercice 7

Modifier les permissions

Créez un fichier `dev1/td2/bro12` contenant un petit texte de votre choix. Faites en sorte qu'il soit lisible par tout le monde mais modifiable uniquement par vous.

2.7 Différencier les étudiants des enseignants

Nous n'allons pas ici introduire un concept nouveau mais appliquer tout ce qui a été vu dans un exemple qui, par expérience, vous pose souvent un problème.

Tutoriel 1

Droit de lecture pour les enseignants uniquement

Supposons que vous avez un fichier qui ne doit être lu que par vous-même et les enseignants (mais pas les autres étudiants).

Comment donner des droits différents aux enseignants et aux étudiants ? La seule solution passe par une modification du groupe du fichier. S'il appartient au groupe `etudiants` par exemple et plus au groupe `users`, les permissions de groupe s'appliqueront aux étudiants et les permissions des *autres* concerneront les enseignants¹

- ✍ Entrez `cd /dev1/td2` pour vous placer dans le dossier indiqué.
- ✍ Entrez `touch examen` pour créer le fichier (vide) indiqué.
- ✍ Entrez `chgrp etudiants examen` pour changer son groupe.
- ✍ Entrez `ls -l` pour visualiser la modification.
- ✍ Entrez `chmod 604 examen` pour changer les permissions associées au fichier.
- ✍ Entrez `ls -l` pour visualiser le résultat final.

Exercice 8

Un fichier réservé aux étudiants

Créez un fichier `dev1/td2/gossip` pour partager des rumeurs entre élèves. On voudrait que tous les étudiants puissent le lire et le modifier mais que les enseignants ne puissent ni le lire, ni l'écrire.

2.8 Les permissions sur les dossiers

Tout comme pour les fichiers, on peut donner des permissions au dossier. Ce qui change, c'est la signification des permissions. Que veut dire « exécuter » un dossier par exemple ?

Pour bien comprendre, prenons une image. Imaginons qu'un dossier possède sur sa « couverture » une liste de son contenu. Par exemple, on voit ci-contre que le dossier `td2` contient 3 fichiers.

image/perm-dossier.pdf

1. Pour être plus précis, tout utilisateur qui n'est pas dans le groupe `etudiants`.

Permissions sur un dossier

Explicitons à présent le sens de chaque droit pour un dossier :

r (lecture)

On a le droit de lire ce qui est écrit sur la « couverture » du dossier. On peut par exemple faire `ls td2`.

w (écriture)

On peut modifier la « couverture » du dossier. On peut ainsi effacer un fichier (`rm td1/bro12`) ou en créer un (`touch td1/bro13`).

x (ouverture)

On peut « ouvrir » le dossier / entrer dedans. On peut ainsi en faire son dossier courant (`cd td2`) ou le traverser dans un chemin (`cat td2/gossip`).

Pour effacer un fichier il ne faut pas de droit en écriture sur le fichier mais bien sur le dossier qui le contient.

Exercice 9

Les permissions sur un dossier (I)

Dans un dossier `dir1` dans votre dossier `td2`. Dans ce dossier, créez un fichier `file`. Faites en sorte que tout le monde puisse voir quels fichiers se trouvent dans `dir1` mais sans pouvoir lire le contenu de ces fichiers.

Exercice 10

Les permissions sur un dossier (II)

Dans un dossier `dir2` dans votre dossier `td2`. Dans ce dossier, créez un fichier `file`. Faites en sorte que tout le monde puisse modifier ce fichier mais sans pouvoir le supprimer.

2.9 Les derniers détails

Voyons quelques détails mis de coté pendant notre apprentissage.

image/reste.pdf

FAQ

Vous dites que dans un affichage en format long, le premier caractère indique si c'est un fichier simple ('-') ou un dossier ('d'). Pourtant j'ai déjà vu d'autres symboles. C'était quoi ?

Il existe d'autres types de fichiers que les deux que nous avons vus. Ils se rencontrent moins souvent et sont surtout utilisés par le système. Par exemple, certains définissent des *pilotes* vers le matériel. Si vous voulez en savoir plus, vous pouvez lire ceci (en.wikipedia.org/wiki/Unix_file_types).

Vous avez mentionné les permissions 'r', 'w' et 'x'. Pourtant j'ai déjà vu d'autres lettres dans la zone réservée aux permissions. C'était quoi ?

Il y a 3 permissions dont nous n'avons pas parlé parce qu'elles sont moins courantes : le *suid* (set user id), le *sgid* (set group id) et le *sticky*. Si vous voulez en savoir plus, vous pouvez lire ceci (fr.wikipedia.org/wiki/Permissions_UNIX).

J'ai un Linux à la maison et les groupes ne sont pas les mêmes. C'est normal ?

Oui ! Les groupes dépendent à la fois de la distribution particulière utilisée et de la façon dont l'administrateur (le root) a configuré le système.

Vous n'avez pas expliqué le sens de la 2^e colonne fournie par la commande ls (juste avant le propriétaire) ?

C'est vrai mais c'est moins utile et plus lié à la structure interne du système de fichier. Je veux bien vous dire qu'il s'agit du nombre de liens physiques sur le fichier mais je sens que vous commencez déjà à regretter d'avoir posé la question ;)

Nous avons vu qu'un fichier est créé avec des permissions par défaut. C'est configurable ?

Oui. Voyez la commande `umask`.

3 Les variables d'environnement

3.1 Introduction

Variable d'environnement

Une **variable d'environnement** est une variable associée à votre shell contenant un texte qui est accessible par toutes les applications que vous lancez. Généralement, elle permet de configurer certaines applications, d'en modifier le comportement.

Manipuler une variable d'environnement

Si VARENV est une variable d'environnement :

- ▷ `export VARENV=valeur` crée une nouvelle variable à la valeur donnée.
- ▷ `VARENV=valeur` modifie la valeur d'une variable existante.
- ▷ `$VARENV` est remplacé par la valeur de la variable.

Experience 2

Manipuler une variable

- ✍ Entrez `VAR=12` pour créer la variable.
- ✍ Entrez `echo "Bonjour !"`.
Cette commande affiche ce qu'on lui donne en paramètre.
- ✍ Entrez `echo $VAR` pour afficher le contenu de la variable.

Exercice 11

Comprendre la signification du \$

Supposons que la variable d'environnement VAR vaut 12. Que va faire chacune des commandes suivantes ? Une fois que vous pensez le savoir, vérifiez-le le tapant sur la machine.

- ▷ `echo VAR`
- ▷ `$VAR=12`
- ▷ `VAR=$VAR`
- ▷ `echo $VAR + 30`
- ▷ `VAR=$VAR+30`
- ▷ `VAR=$VAR + 30`
- ▷ `VAR="$VAR + 30"`
- ▷ `VAR=VAR`

3.2 Le prompt

Prompt

Le **prompt** (ou *invite* en français) est le texte qui apparaît à gauche de ce que vous tapez dans votre shell. Il est déterminé par la variable d'environnement PS1.

Experience 3

Le prompt

Affichez la valeur de votre prompt. Vous remarquerez qu'il contient des codes qui seront remplacés par certaines valeurs. Par exemple, `\w` indique le dossier courant.

Exercice 12

Modifier le prompt

Modifiez la valeur de votre prompt. Par exemple, modifiez l'invite en "Bonjour! ".

Tutoriel 2

Durée de vie de la variable

✍ Déconnectez-vous de `linux1`, puis reconnectez-vous.

Surprise, votre prompt a repris sa valeur d'origine! Pour rendre une modification permanente, il faut ajouter la commande à votre fichier `.bashrc`. C'est un fichier caché qui est exécuté par votre shell lors de votre connexion.

3.3 Pour aller plus loin : l'éditeur `vi`

Si vous avez fini, vous serez peut-être curieux d'apprendre à utiliser `vi` plutôt que `nano`.

Quels sont les avantages de `vi` ?

- ▷ Certaines manipulations des fichiers sont plus simples : copier, supprimer, déplacer des lignes par exemple.
- ▷ Il est facile d'indenter proprement un programme Java qui ne l'est pas.

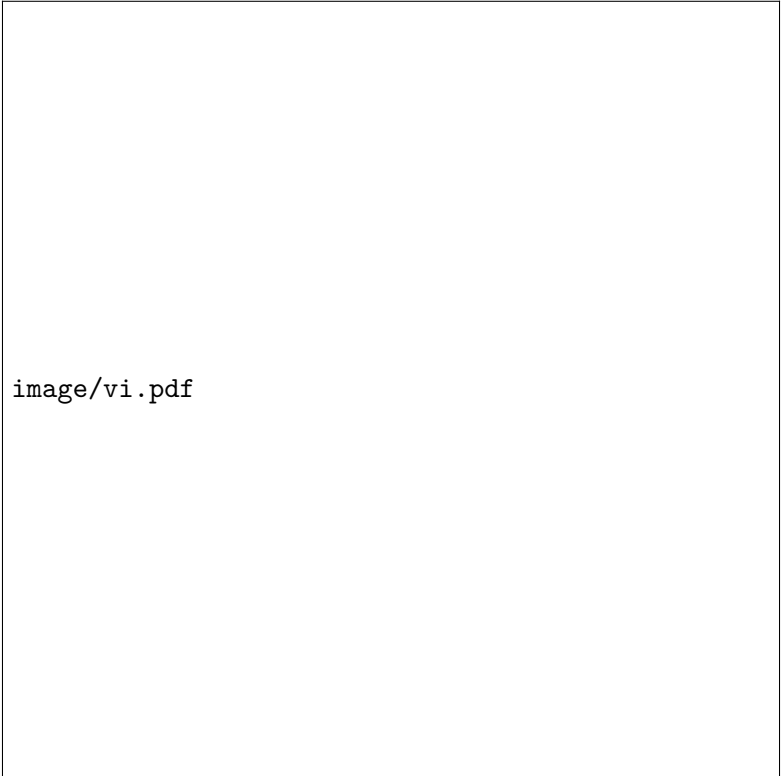
`vi` est plus puissant que `nano` mais il est moins intuitif quand on débute. Voici les bases à comprendre.

Démarrer

Pour éditer un fichier texte existant ou créer un nouveau fichier, il suffit de taper :

```
vi monFichier.
```

Les 3 modes de fonctionnement sous `vi(m)` :



image/vi.pdf

Le mode commande

C'est le mode dans lequel vous vous trouvez quand vous ouvrez `vi`. Dans ce mode, les touches sur lesquelles vous appuyez ne sont pas insérées dans le texte (comme dans nano) mais sont considérées comme des commandes. C'est le mode qui permet de **manipuler** le texte.

Exemples de commandes :

- ▷ `i` : passer en mode édition (cfr. infra) ;
- ▷ `yy` : copier la ligne sous le curseur (comme un `CTRL-C` sous Windows) ;
- ▷ `3yy` : copier 3 lignes ;
- ▷ `dd` : couper la ligne sous le curseur (`5yy` pour en couper 5) ;
- ▷ `dw` : couper le mot sur lequel se trouve le curseur ;
- ▷ `p` : coller (ce qui a été précédemment copié ou coupé) sous la ligne qui suit le curseur ;
- ▷ `u` : annuler la dernière modification. Vous pouvez appuyer plusieurs fois pour annuler les dernières modifications.

Le mode édition

C'est le mode dans lequel ce qu'on tape est ajouté au texte, comme dans nano. On y accède par la touche :

- ▷ `i` (insert) : on insère à l'endroit du curseur ;
- ▷ `a` (append) : on insère après le curseur ;
- ▷ `o` : on insère dans une nouvelle ligne créée sous le curseur ;

Dans tous les cas, l'indicateur `INSERT` apparaît alors en bas de l'écran.

Le mode exécution

On y accède à partir du mode commande en tapant `:`. Ce mode permet d'entrer d'autres types de commandes, plus riches que celles du mode commande. Voici les plus utilisées :

- ▷ `:h` pour accéder à l'aide (`:q` pour quitter l'aide) ;
- ▷ `:w` pour sauver le fichier ;
- ▷ `:w monFichier` pour enregistrer sous `monFichier`,
- ▷ `:q` pour quitter l'éditeur (sans sauver) ;
- ▷ `:x` pour enregistrer et quitter ;
- ▷ `:q!` pour quitter sans enregistrer les modifications ;
- ▷ `:set nu` pour afficher les numéro de ligne (`:set nonu` pour les retirer) ;
- ▷ `:numéroDeLigne` pour aller directement à cette ligne ;
- ▷ `:%s/old/new/g` pour remplacer toutes les occurrences de la chaîne de caractères `old` par la chaîne de caractère `new`.

3.4 Pour aller plus loin : le prompt

Exercice 13

Explorer les codes du prompt

Les possibilités de configuration du prompt sont nombreuses : afficher l'heure, le nom de la machine, votre login, utiliser des couleurs... Examinez la documentation pour configurer le prompt comme il vous plait (`man bash`, section `PROMPTING`). Rendez la modification permanente.