TD2 : Les fichiers Linux
V1.2.0

Document en ligne : www.mickael-martin-nevot.com

1 Généralités

Sans mention contraire, vous vous positionnerez dans votre (sous-) répertoire `tp2` durant l'ensemble de ce TD.

N'oubliez pas de consulter le manuel à chaque fois que cela est nécessaire, en particulier à chaque découverte d'une nouvelle commande. Vous pouvez aussi faire des recherches sur le Web en prenant soin de vérifier que les informations trouvées soient correctes.

2 Création de fichiers

2.1 Première approche

Rappelez-vous que Linux est sensible à la casse.

Créez un fichier vide `welcome.txt` à l'aide de la commande `touch`. Vérifiez ensuite que le fichier créé est bien vide (à l'aide de la commande `cat`).

Modifiez le fichier `welcome.txt`. Pour ce faire :
- exécutez la commande `cat > welcome.txt` ;
- écrivez la phrase « Salut ! » ;
- terminez l'édition du fichier en pressant `Ctrl+D` ;
- vérifiez que tout s'est bien passé en affichant de nouveau le contenu du fichier `welcome.txt`.

En utilisant cette fois-ci la commande `cat >> welcome.txt`, ajoutez à la fin du fichier `welcome.txt` la phrase « Ça va ? ». Vérifiez une nouvelle fois que tout s'est bien passé en affichant le contenu du fichier `welcome.txt`.

À l'aide de la commande `cp`, copiez le fichier `welcome.txt` dans un nouveau fichier `hello`. Vérifiez que le contenu du fichier `hello` soit identique à celui du fichier `welcome.txt` en utilisant la commande `diff`.

Enfin, entrez une nouvelle phrase de votre choix dans le fichier `hello` en utilisant la commande

`cat > hello`. Affichez le contenu de `hello`.

Concluez sur la différence qu'il y a, a priori, entre > et >>.

Notes
Notez qu'il n'est pas nécessaire d'avoir une extension à un fichier : en particulier, il n'est pas nécessaire qu'un fichier texte ait l'extension `.txt` (comme `hello`) mais c'est une convention.

De même, une extension donne le plus souvent une indication sur le format du fichier mais ne le contraint pas. Il arrive même qu'une extension puisse renvoyer à plusieurs formats (pour les vidéos notamment).

2.2 Introduction aux caractères de redirection

Les caractères (ou symboles) de redirection permettent de rediriger l'entrée standard (généralement le clavier) et la sortie standard (généralement l'écran) d'une commande vers des fichiers. Ainsi, la commande utilisera les données du fichier plutôt que celles provenant du clavier, et enregistrera le résultat dans un fichier (et le créera au besoin) plutôt que de l'afficher à l'écran. > et >> sont les principaux caractères de redirection (`cmd` étant une commande) :

- `cmd > file` : permet d'enregistrer le résultat de `cmd` dans le fichier `file` (le contenu de `file` étant écrasé) ;
- `cmd >> file` : rajoute à la fin du fichier `file` le résultat de `cmd`.

À l'aide d'une redirection et de la ligne de commande `ls -l`, enregistrez le résultat de cette commande dans le fichier `foo` (rappelez-vous que le fichier sera créé automatiquement au besoin).

À l'aide d'une redirection (non écrasante), enregistrez la liste des processus du *shell* à la fin du fichier `foo`. Vérifiez que tout s'est bien passé en affichant le contenu du fichier `foo`.

Note
La variable `foo` est une **variable métasyntaxique**, c.-à-d. un terme générique utilisé dans les exemples pour se concentrer sur le fond plutôt que sur la forme, et dont le nom est choisi pour être tacitement reconnu comme tel.

Voici quelques autres variables métasyntaxiques : `toto` (la plus utilisée en France), `tata`, `titi`, `tutu`, `tyty`, `bar`, `willi`, etc.

3 Filtres de fichiers

3.1 Code ASCII

Le jeu de caractères codés (ou code) **ASCII** est une norme de codage de caractères en informatique, ancienne et connue pour son influence incontournable sur les codages de caractères qui lui ont succédé (comme UTF-8 par exemple). On y retrouve des caractères correspondant aux lettres (sans accentuation), signes de ponctuation, chiffres et même des caractères « spéciaux » de base. À chaque caractère correspond un code numérique permettant de l'identifier en informatique (cela est en outre utile pour le tri des répertoires, des fichiers, etc.).

Table 1 – Code ASCII

0 NUL	32 espace	64 @	96 `
1 SOH	33 !	65 A	97 a
2 STX	34 "	66 B	98 b
3 ETX	35 #	67 C	99 c
4 EOT	36 $	68 D	100 d
5 ENQ	37 %	69 E	101 e
6 ACK	38 &	70 F	102 f
7 BEL	39 '	71 G	103 g
8 BS	40 (	72 H	104 h
9 HT	41)	73 I	105 i
10 LF	42 *	74 J	106 j
11 VT	43 +	75 K	107 k
12 FF	44 ,	76 L	108 l
13 CR	45 -	77 M	109 m
14 SO	46 .	78 N	110 n
15 SI	47 /	79 O	111 o
16 SLE	48 0	80 P	112 p
17 CS1	49 1	81 Q	113 q
18 DC2	50 2	82 R	114 r
19 DC3	51 3	83 S	115 s
20 DC4	52 4	84 T	116 t
21 NAK	53 5	85 U	117 u
22 SYN	54 6	86 V	118 v
23 ETB	55 7	87 W	119 w
24 CAN	56 8	88 X	120 x
25 EM	57 9	89 Y	121 y
26 SIB	58 :	90 Z	122 z
27 ESC	59 ;	91 [	123 {
28 FS	60 <	92 \	124 |
29 GS	61 =	93]	125 }
30 RS	62 >	94 ^	126 ~
31 US	63 ?	95 _	127 ■

Parmi les particularités du code ASCII, on remarque :

- des caractères « spéciaux » dans la première colonne (les 31 premiers caractères) ; il ne s'agit pas de lettres ou de ponctuation, ni même de chiffres : ils ne sont pas visibles (même si leurs noms sont des chaînes de caractères, ce ne sont pourtant que des caractères) mais ont un rôle de mise en page ou autre ;
- que les chiffres ont un codage inférieur aux lettres ;
- que les majuscules ont un codage inférieur aux minuscules.

3.2 Mise en pratique

Un filtre permet de définir un ensemble de fichiers à manipuler. Il est constitué de caractères « normaux » et de caractères « spéciaux » appelés métacaractères de substitution ou caractères de filtrage. Les caractères de filtrage sont `?`, `[`, `]`, `^`, et `*`. Dans une chaîne de caractères :

- `?` : représente exactement un caractère quelconque ;
- `[adfg]` : représente un caractère au choix entre « a », « d », « f » ou « g » ;
- `[!adfg]` : représente un caractère au choix qui n'est pas « a », « d », « f » ou « g » ;
- `[a-f]` : représente un caractère au choix compris entre « a » et « f » (suivant le code ASCII) ;
- `[9-A]` : représente un caractère au choix entre « 9 », « : », « ; », « < », « = », « > », « ? », « @ » ou « A » (majuscule) ;
- `[!a-f]` : représente un caractère au choix non compris entre « a » et « f » ;
- `*` : représente n'importe quelle chaîne de caractères.

Par exemple, pour lister tous les fichiers du répertoire courant (ou de travail, celui où vous êtes) dont le nom commence par « *file* » suivi d'un caractère quelconque, il faut utiliser la ligne de commande `ls file?`. Lorsque cette commande est interprétée, `?` est remplacé par n'importe quel caractère.

Trouvez la ligne de commande permettant d'afficher tous les fichiers dont le nom commence par une chaîne de caractères quelconque, suivi d'un « a » ou d'un « m », suivi de nouveau d'un « a » ou d'un « m », et terminé par une suite de caractères quelconque. Créez dans le répertoire `tp2` un ensemble de fichiers vous permettant de vérifier votre réponse.

Positionnez-vous dans le répertoire `/bin` (en une seule ligne de commande). Tout d'abord, affichez l'ensemble du contenu de ce répertoire. Ensuite, listez seulement les fichiers :

- dont le nom commence par « b » ;
- dont le nom commence par une voyelle ;
- dont le nom contient la chaîne de caractères « sh » ;
- de trois lettres qui se terminent par la chaîne de caractères « sh ».

Ensuite, placez-vous dans le répertoire `/sbin` (en une seule ligne de commande) puis affichez les noms des fichiers dont la 4e lettre est « a », « b », « c » ou « d ».

4 Caractères « spéciaux » et noms de répertoires ou de fichiers

Les caractères « spéciaux » peuvent être utilisés pour nommer des répertoires ou des fichiers, mais cela n'est pas sans poser quelques soucis.

Voici quelques cas illustrant certains problèmes pouvant se produire :

- dans le répertoire `tp2`, créez trois fichiers de nom `a*b`, `acb` et `addb`, puis exécutez la commande `ls a*b` ; expliquez le résultat obtenu ;
- exécutez les commandes `ls a\*b`, `ls a"*"b` et `ls a'*'b` ; expliquez le résultat obtenu et déduisez-en la différence d'utilisation entre `"` et `'` ;
- en utilisant la commande `mkdir` (qui permet de créer des répertoires), trouvez trois façons de créer un répertoire dont le nom est `hel\lo` ; demandez-vous s'il existe encore d'autres façons de le faire ;
- faites de même pour un répertoire dont le nom est `hel\lo*lo` ;
- créez un fichier dont le nom est `my file` ;
- effacez les répertoires inutiles avec la commande `rm`.

Déduisez-en l'utilité de `\`, de `"` et de `'`.

5 Compression/Décompression de fichiers

5.1 Archivage et compression

5.1.1 Archivage

L'archivage consiste en l'utilisation d'un seul fichier pour en stocker plusieurs (autrement dit, une concaténation de fichiers). Le plus souvent, le processus d'archivage est « intelligent » et préserve notamment les droits, le propriétaire ainsi que le groupe des répertoires et des fichiers, permettant de reproduire le plus fidèlement possible une arborescence de travail lors d'une copie par exemple. Sous Linux, les archives ont généralement l'extension `.tar`.

5.1.2 Compression

La compression consiste en la diminution de la taille d'un fichier (pouvant être une archive). Il existe deux types de compression :

- compression **sans perte** (d'information) : utilisée pour les documents, les archives, les fichiers textes, les exécutables, etc. ;
- compression **avec perte** (d'information, même si elle reste sensiblement la même) : utilisée pour les images, le son, la vidéo, etc.

L'intérêt d'une diminution de taille réside dans un gain de place de stockage, une plus grande vitesse de téléchargement, etc. Les fichiers compressés ont très souvent l'extension `.zip` sous Windows et Linux ainsi que `.gzip` sous Linux.

Généralement, une archive est compressée.

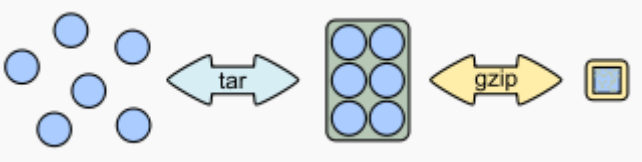

Figure 1 – Principe de combinaison de tar et de gzip

5.2 Mise en pratique

La commande pour archiver est : `tar`.

Voici les commandes pour compresser/décompresser :

- des fichiers au format ZIP : `zip`/`unzip` ;
- des fichiers au format gzip : `gzip`/`gunzip`.

Exécutez un navigateur Web (Chromium par exemple), puis téléchargez dans votre répertoire `tp2` le fichier `vade-mecum des commandes UNIX` disponible sur le site Web de l'enseignant.

Le fichier téléchargé est au format **PDF**, tout comme ce TD. Ce format est utilisé dans la majorité des transmissions de documents car il préserve la mise en forme d'un fichier telle qu'elle a été définie par son auteur, et cela quels que soient le logiciel, le système d'exploitation et l'ordinateur utilisés pour l'imprimer ou le visualiser.

Déterminez le poids du fichier téléchargé avant et après l'avoir compressé au format ZIP. Faites ensuite de même avec le format gzip.

Archivez votre répertoire `tp2` (`tar`) puis affichez le contenu du fichier obtenu (sans désarchiver). Enfin, restituez-le (désarchiver-le sans écraser le contenu initial du répertoire).

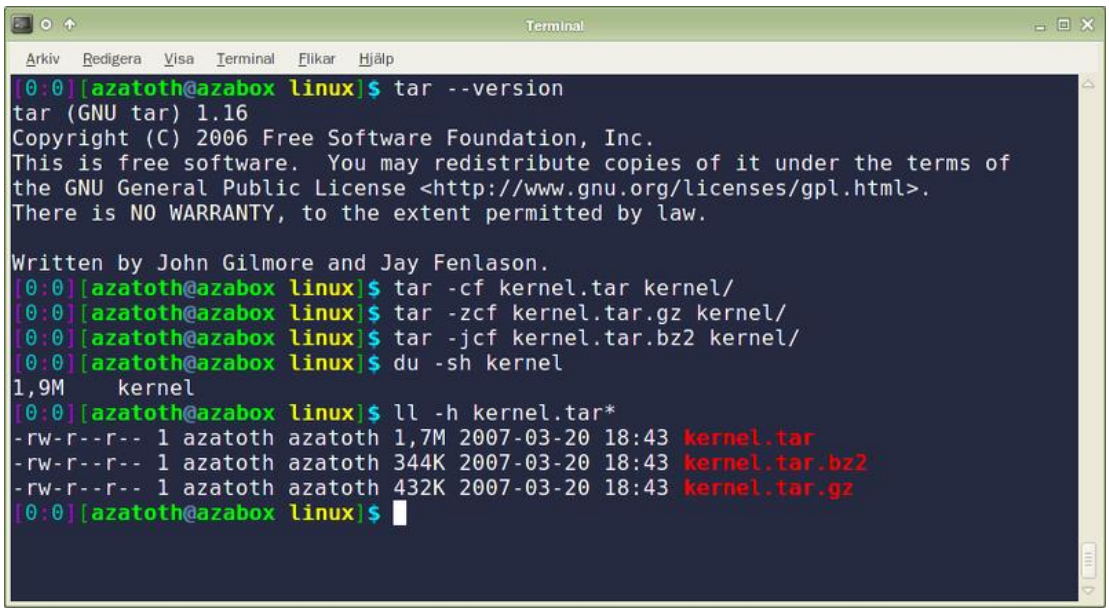

Figure 2 – Utilisation de la commande tar

6 Droits utilisateurs sous Linux

6.1 Fichier Linux

Sous Linux, tout est fichier (les répertoires ne sont en fait que des fichiers particuliers).

6.2 Informations sur les fichiers

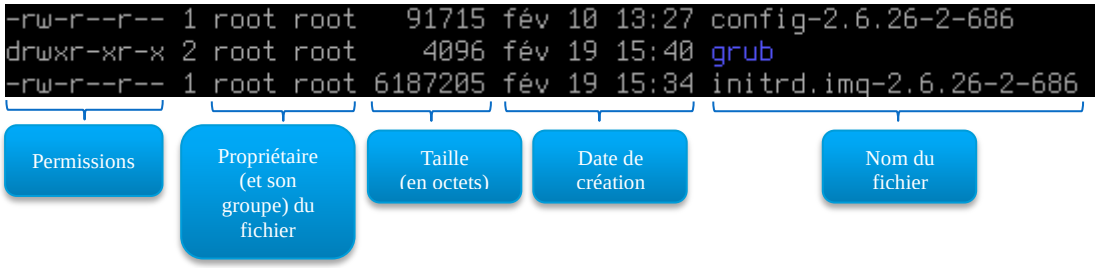

Figure 3 – Informations sur les fichiers

6.3 Système binaire (base 2)

À la place du système décimal (base 10) utilisé usuellement, un ordinateur utilise un système binaire (base 2). Au lieu de 10 chiffres (0, 1, 2, 3, 4, 5, 6, 7, 8 et 9), le binaire n'en compte que 2 (0 et 1). Ainsi 10 en binaire est égal à 2 en décimal, 11 est égal à 3, 100 à 4, 10000 à 16, etc.

Outre pour le fondement même d'un ordinateur, ce système est utilisé pour la gestion des droits d'accès sous Linux.

6.4 Gestion des droits utilisateurs

Contrairement aux systèmes de type MS-DOS, les systèmes de type UNIX comme Linux disposent d'un système flexible de droits (d'accès) aux ressources (fichiers). Le système prévoit que chaque

utilisateur appartienne à un **groupe**. Les permissions effectives d'un utilisateur peuvent provenir des permissions accordées à son groupe d'appartenance (et donc à tous les membres qui le composent) ou de ses permissions exclusives.

Voici à quoi ressemblent les droits d'accès d'un fichier Linux :

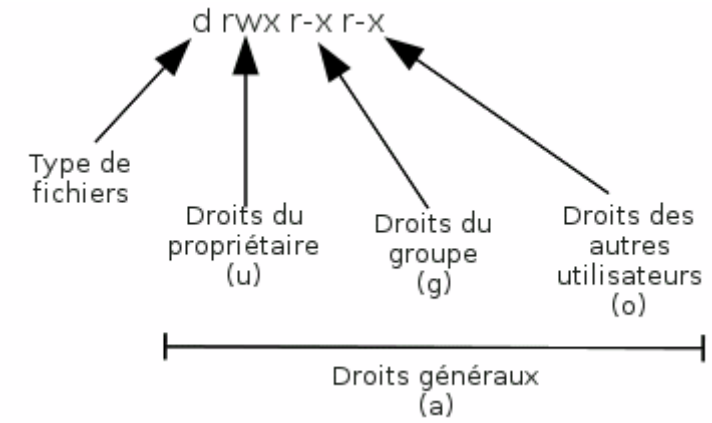

Figure 4 – Droits d'accès Linux

La colonne des droits est composée de quatre champs : le premier caractère correspond au type de fichier, suivi de trois champs de trois caractères correspondant respectivement aux droits du propriétaire, du groupe et des autres.

Si le type de fichier est `d`, alors il s'agit d'un répertoire, si c'est −, c'est un fichier normal.

Chaque champ de droit est conçu de la même façon. On trouvera les droits de lecture `r`, les droits d'écriture `w` et les droits d'exécution `x`. L'absence de droits est notée par −. La présence d'un droit donne différentes permissions selon qu'il s'agisse d'un fichier ou d'un répertoire :

Table 2 – Permissions fichier/répertoire

Droit	Fichier	Répertoire
r	Consulte le contenu	Consulte la liste des éléments du répertoire
w	Modifie le contenu	Modifie le répertoire
x	Exécute le contenu	Permet de naviguer dans le répertoire

6.5 Mise en pratique

La commande pour gérer les droits d'accès est `chmod`. La commande pour changer le propriétaire d'un fichier est `chown`.

Il existe deux utilisations équivalentes pour gérer les droits d'utilisation avec `chmod` :

- avec les droits explicites (attention à ne pas mettre d'espace autour des `,`) : `chmod u=rwx,g=rx,o=rx my-file` ;
- avec le système binaire (plus commode) : `chmod 755 my-file`.

Voici les combinaisons des droits possibles, transposées dans le système binaire :

Table 3 – Combinaisons possibles des droits utilisateur

Binaire r w x	Décimal	Droits correspondants
0 0 0	0	aucun
0 0 1	1	exécution
0 1 0	2	écriture
0 1 1	3	écriture & exécution
1 0 0	4	lecture
1 0 1	5	lecture & exécution
1 1 0	6	lecture & écriture
1 1 1	7	lecture, écriture, exécution

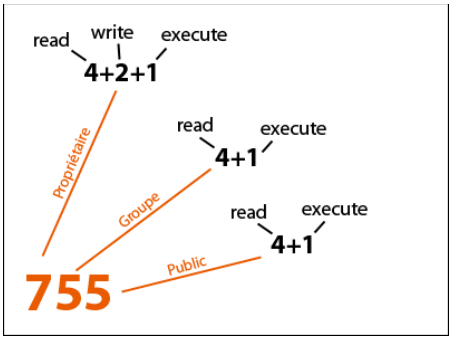

Figure 5 – Construction des droits utilisateur

Créez l'arborescence suivante en utilisant uniquement des commandes *shell* :

- `chmod-test-repository` (un répertoire avec les droits `rwx rwx rwx`)
  - `priv-file` (un fichier avec les droits `rw- rw- rw-`) contenant le texte « Secret »
  - `sub-repository-1` (un répertoire avec les droits `rw- rw- rw-`)
    - `file` (un fichier avec les droits `rw- rw- rw-`) contenant le texte « Salut »
  - `sub-repository-2` (un répertoire avec les droits `--x --x --x`)
    - `file` (un fichier avec les droits `rw- rw- rw-`) contenant le texte « Ciao »

Exécutez les lignes de commande suivantes et déterminez celles qui peuvent être utilisées à partir du répertoire `chmod-test-repository` :

- `ls ./sub-repository-1`
- `ls -l ./sub-repository-1`
- `cd ./sub-repository-1`
- `cat ./sub-repository-1/file`
- `touch ./sub-repository-1/file2`
- `rm ./sub-repository-1/file`
- `ls ./sub-repository-2`
- `cd ./sub-repository-2`
- `cat ./sub-repository-2/file`
- `touch ./sub-repository-2/file2`
- `rm ./sub-repository-2/file`
- `echo "Comment vas-tu" >> ./sub-repository-2/file`

Affectez les droits `--- --- ---` à `private-file` puis concluez.