TP n°1
Répertoires et Fichiers sous Unix

Ce TP est destiné à vous familiariser avec les fichiers sous l'environnement UNIX. Fortement axé sur la pratique, il aborde la création, la manipulation et la suppression des répertoires et fichiers.

1 Environnement de travail

Apprenant l'informatique, vous allez commencer à arrêter de cliquer et vous allez utiliser un interpréteur de commandes à l'aide d'un terminal.

1.1 Interpréteur de commandes

Mais qu'est-ce que c'est un interpréteur de commandes ? Ce sont des outils pour exécuter des commandes tapées au clavier par un utilisateur dans un terminal. Les interpréteurs de commandes sont aussi appelées « shells » (coquilles).

Les commandes sont tapées dans un terminal en mode texte, constitué par une fenêtre dans un environnement graphique, ou par une console sur un écran en texte seul (sans environnement graphique).

Les résultats sont aussi affichés sur le terminal. Une sortie graphique n'est pas obligatoire (pour la plupart des commandes la sortie est du texte). Les interpréteurs de commandes peuvent être programmables : ils fournissent toutes les ressources nécessaires pour l'écriture de programmes complexes (variables, conditions, boucles, ... mais nous verrons cela beaucoup plus tard dans ce cours « Environnement Informatique 1 »).

1.2 Commande

Une commande Unix est un mot ou une phrase à la syntaxe bien particulière entré dans un éditeur ou interpréteur de commandes et donnant l'ordre d'actions à exécuter par l'ordinateur.

Une commande UNIX se décompose en trois parties :

- la commande elle-même.
- des options, qui comme son nom l'indique, sont optionnelles (zéro, une ou plusieurs options).
- des arguments : zéro, un nombre fixe ou variable d'arguments.

Par exemple, si vous voulez connaître une date, tapez dans un terminal la commande *date* et le système vous re- tournera un texte contenant la date du jour et l'heure.
Si nous prenons l'exemple de la commande *cal*, suivant son utilisation, vous pourrez avoir plusieurs résultats à partir de cette même commande :

- *cal* : imprimera le calendrier du mois courant
- *cal -3* : imprimera les calendriers du mois précédent, courant et suivant.
- *cal 2012* : imprimera le calendrier de l'année 2012

Dans les exemples précédents, *cal* est toujours la commande, *-3* est une option (commence par un -) et *2012* est un argument.

Mais pourquoi utiliser un terminal et taper des commandes ? Vous verrez un peu plus tard tout l'intérêt d'un interpréteur de commandes. En effet, l'énorme avantage, quand on le maitrise, est que l'on peut faire des traite- ments sur un nombre très important de fichiers en une seule commande ou un enchainement de commandes. En effet, certaines opérations ne sont pas faisables à l'aide de la souris (ou alors vous perdez votre temps).

1.3 Aide sur les commandes

Il est impossible de connaître toutes les commandes par cœur, ni toutes les options et arguments possibles pour chacune des commandes. Heureusement, UNIX est bien fait et à tout prévu.

TP n°1
Répertoires et Fichiers sous Unix

1.3.1 À propos

Si vous désirez connaître les commandes qui concernent un thème particulier, vous pouvez utiliser la commande *apropos*. Par exemple, si vous souhaitez connaître toutes les commandes qui traitent du calendrier :

$ apropos calendar

1.3.2 Manuel des commandes

Pour tout connaître sur une commande, une page de manuel est fournie avec le système (ces pages sont souvent disponibles dans plusieurs langues, mais la page en anglais est la référence). Pour tout savoir sur une commande :

$ man commande

Ce n'est pas toujours drôle de devoir lire le descriptif, mais c'est le passage obligé, même pour les plus chevronnés. Une expression anglaise résume bien la situation : RTFM (*Read The Fucking Manual*) ! Vous ne pourrez pas y échapper. Et maintenant que vous avez les informations minimum, commençons !

2 Répertoire

Comme sous Windows, un dossier ou répertoire est un objet informatique qui contient des fichiers. Et même les répertoires sont des fichiers, contenant une liste de fichiers et de sous-répertoires.

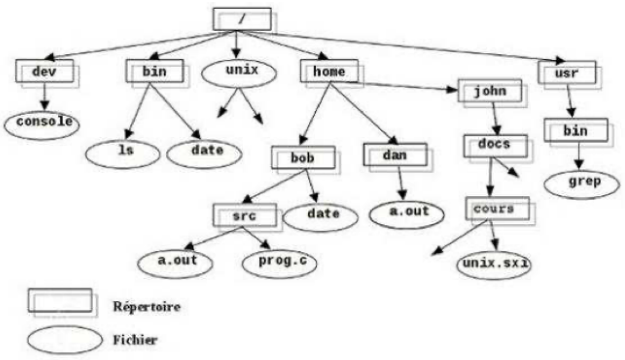
Imaginez une grande commode qui est le système de fichiers de votre disque dur. Il contient des tiroirs dans lesquels pourraient se trouver des fichiers mais aussi d'autres tiroirs ... un répertoire peut donc contenir :

- des fichiers;
- d'autres répertoires.

Si l'on reprend notre exemple de la commode, la plus grande entité contenant d'autres entités est la commode : elle ne peut pas se trouver dans un tiroir !

Dans le cas de l'informatique, on appelle cette entité le répertoire racine (appelée parfois tout simplement « *racine* » en anglais « *root directory* ») : il s'agit de l'entité de plus bas niveau, car elle peut contenir des fichiers ou des répertoires mais ne peut pas se trouver elle-même dans un répertoire !

Donc, si nous retournons dans le système informatique : le système de fichiers est organisé en une unique arborescence, les répertoires sont les nœuds de l'arborescence, la racine est unique (contrairement à Windows) et notée /, les périphériques sont cachés (contrairement à Windows) et les fichiers sont cités dans les répertoires.



Donc en résumé :

Arborescence : Structure hiérarchisée des répertoires et des sous répertoires sur un disque dur, par exemple. Le tronc étant représenté par la racine du disque et les branches par les répertoires, etc. Il existe plusieurs chemins (sous UNIX) pour se rendre de la racine au répertoire voulu.

Racine : Point de départ d'une arborescence (notée / sous UNIX).

Périphériques : Une pièce de matériel qui peut effectuer une fonction particulière (souris, clavier, écran, ...).

Homedir (HOME) : C'est VOTRE répertoire personnel, noté ~ ou encore */home/helene* (nous reviendrons un peu plus loin sur cette double notation).

Répertoire : Une liste de descriptions de fichiers. Du point de vue du système de fichiers, il est traité comme un fichier dont le contenu est la liste des fichiers référencés. Un répertoire a donc les mêmes types de propriétés qu'un fichier comme le nom, la taille, la date, les droits d'accès et les divers autres attributs. Lorsque d'un répertoire, on veut aller au répertoire parent, celui-ci est désigné par « .. » sur la plupart des systèmes (on tapera donc « cd ..» sous UNIX pour accéder à un répertoire parent).

Chemin (« path ») : Séquence de répertoires imbriqués, avec un fichier ou un répertoire à la fin, séparés par le caractère /

Il y a une différence entre *chemin relatif* et *chemin absolu*. Vous allez me dire : « j'y comprend rien » et vous avez raison !
Par exemple, on se trouve dans le répertoire « helene » donc :

- le chemin relatif est : *docs/cours/unix.txt*, i.e. relatif au répertoire courant.
- le chemin absolu est : */home/helene/docs/cours/unix.txt*, i.e. le chemin depuis le répertoire racine du système (/).

2.1 Les différents types de répertoires basiques et spéciaux

Depuis le début d'Unix, aucune limitation majeure n'est faite quant à la longueur d'un nom de fichier. Tout caractère (les espaces en particulier) peut être utilisé dans le nom, et les extensions sont facultatives. Les différences de *majuscules* ou de *minuscules* constituent des fichiers distincts.

2.1.1 Le répertoire relatif courant : .

C'est le *répertoire courant*. Il est utilisé pour les commandes qui ont un répertoire comme argument. Il est également utilisé parfois pour lancer des commandes dans le répertoire courant.

2.1.2 Le répertoire relatif parent : ..

C'est le répertoire parent. Il fait toujours partie du répertoire. Il est l'unique référence au répertoire parent. L'utilisation la plus courante est :

$ cd ..

2.1.3 Le répertoire absolu homedir : ~

Il n'est pas vraiment un répertoire spécial. Les interpréteurs de commande le remplacent juste par le « *homedir* » de l'utilisateur courant et il ne peut pas être utilisé dans la plupart des programmes, car il n'est pas un vrai répertoire.

2.1.4 Le répertoire absolu homedir : ~helene (ou plus généralement ~login)

De façon analogue, il est remplacé par les shells par le répertoire utilisateur de l'utilisateur *helene*.

Si vous ne savez plus dans quel répertoire vous êtes, tapez la commande pwd et Linux vous l'indiquera! La commande mkdir sert quant à elle à créer un répertoire.

2.2 Manipulation des répertoires

Nous allons apprendre à « *lire* » le contenu d'un système de fichiers, ainsi qu'à consulter et comprendre tous les types de fichiers d'Unix.

La commande *ls* (abréviation du mot *list*) est prévue à cet effet. Elle affiche les fichiers et les sous-répertoires qui se trouvent dans un répertoire.

Les options de la commande *ls*

Option	Objectif
-a (abréviation de mot « all » : tous)	Affiche tous les fichiers en incluant ceux qui commencent par un point (i.e. les fichiers cachés)
-c	Affiche la dernière fois où le fichier a été modifié
--color	Affiche en couleur suivant le type des éléments
-l (abréviation de mot « long »)	Affiche en format long (type, date, taille, propriétaire, permissions)
-R	Affiche les contenus des répertoires du répertoire en cours
-S (abréviation de mot « size » : taille)	Liste les fichiers par taille (les fichiers les plus gros en premier)
-t (abréviation de mot « time » : temps)	Liste les fichiers selon la date de la dernière modification
-u	Liste les fichiers selon la date du dernier accès
-r (abréviation de mot « reverse » : inversé)	Affiche les fichiers en ordre inverse

Ces options peuvent être utilisées séparément ou combinées.

Par exemple :

-ltr	Format long, les fichiers les plus récents à la fin

Parmi les options de ce tableau, les options -a et -l sont les plus utilisées. Et la commande ls possède une cinquantaine d'options !

2.3 Comment se déplacer dans les répertoires ?

Pour se déplacer à l'intérieur de l'arborescence, il y a *cd* qui est une commande intégrée à bash, le shell par défaut sur les systèmes GNU/Linux et Unix plus généralement. Elle s'utilise en lui donnant le répertoire qui doit devenir celui courant pour les actions suivantes.

$ cd répertoire

Le répertoire peut être indiqué de manière absolue (par rapport à la racine en commençant le chemin par /) ou relative (par rapport au répertoire courant). On peut aussi utiliser .. qui désigne le répertoire parent d'un répertoire ou . qui désigne le répertoire courant. Par exemple pour remonter dans l'arborescence, on utilise :

$ cd ..

Faites bien attention de séparer par un espace « cd » et « .. », UNIX exige une grande précision dans la syntaxe des commandes. Soumettez la commande au système grâce à la touche « Entrée », évidemment !

À la suite de cd on peut aussi utiliser - (tiret) qui désigne le répertoire précédent. Cela permet de revenir au répertoire précédemment visité. Mais cela ne se fait que sur un seul niveau. Une fois revenu dans le répertoire précédent, - désigne celui d'où on vient.

$ cd -

3 Fichier

Un fichier est une suite d'informations binaires, c'est-à-dire une suite de 0 et de 1. Ce fichier peut être stocké pour garder une trace de ces informations. Un fichier texte est un fichier composé de caractères stockés sous la forme d'octets.

Vous n'y comprenez rien ?? C'est normal !

Un fichier est enregistré sur le disque dur sous la forme « *nom_du_fichier.ext* ». « *.ext* » représente l'extension c'est un moyen de reconnaître le type de programme avec lequel ce fichier peut être ouvert (attention cela ne garantit pas le type de fichier : lorsque l'on change l'extension on ne change pas le type de fichier !).

Lorsque vous faites la commande ls avec l'option -l dans un répertoire, vous obtenez la liste détaillée des fichiers contenus dans ce répertoire. Pour savoir de quel est type est un fichier, il suffit de regarder la première lettre de la ligne. Pour les autres données, nous verrons cela ultérieurement.

Les types de fichiers Linux

Désignation	Type	Description
-	Fichier standard	C'est un fichier ordinaire tel qu'un fichier texte ou un programme
b	Périphérique bloc	L'élément est un pilote (programme de contrôle) pour un support tel qu'un disque dur ou un lecteur de CD-ROM.
c	Périphérique caractère	L'élément est un pilote (programme de contrôle) pour un matériel qui transmet des données, tel un modem.
d	Répertoire	C'est l'élément qui contient les fichiers, en référence au dossier d'un système d'exploitation.
l	Lien	Un élément qui est soit un lien hard, soit un lien soft.

Il faut avoir créé un répertoire pour y ranger un fichier !!

3.1.1 La commande « *touch* »

La commande « touch » fixe la date de dernière modification du fichier au moment présent. Si le fichier n'existe pas, la commande crée un fichier vide.

L'autre méthode dépend du type de fichier à créer. Si c'est un fichier texte, utilisez un des éditeurs de texte, par exemple *gedit*.

3.1.2 Editer le contenu d'un fichier texte : gedit

Vous pouvez créer ou éditer un fichier texte grâce à la commande *gedit*. Cette application est aussi disponible dans le Menu **Application -> Accessoires**.

3.1.3 La commande cp ou copier des fichiers et des répertoires

La commande cp (abréviation du mot « copy ») copie un fichier existant vers un autre. Elle a deux arguments :

- le fichier à copier, qui doit exister et pouvoir être lu,
- le fichier résultat, qui doit être placé dans un répertoire où on a le droit d'écriture.

Si le fichier existe déjà, alors il est remplacé par un autre nom, par exemple :

$ cp VariableAléatoire Laplace

Si on met un nom de répertoire comme destination, le fichier copié aura le même nom que le fichier à copier.

$ cp VariableAléatoire khi2/
$ ls khi2/
VariableAléatoire
ou :
$ cd khi2/
$ cp ../VariableAléatoire .
$ls
VariableAléatoire

C'est la même chose au répertoire près !

La commande cp peut avoir plusieurs arguments :
$cp -r VariableAléatoire khi2/ LoiCauchy LoiBernoulli Probabilités/
(cp -r : copie récursive qui prend le répertoire et tout ce qu'il contient, y compris les autres répertoires)

En gros, on copie toute la branche d'un arbre, i.e. tout un tiroir !
$ ls Probabilités/
VariableAléatoire khi2/ LoiCauchy LoiBernoulli

ou :
$ cd Probabilités/
$cp -r ../VariableAléatoire ../khi2/ ../LoiCauchy ../LoiBernoulli .

(cp -r : copie récursive qui prend le répertoire et tout ce qu'il contient, y compris les autres répertoires))
$ ls
VariableAléatoire khi2/ LoiCauchy LoiBernoulli

En règle générale, la commande cp a au moins deux arguments :

- n-1 premiers sont les noms de fichiers (ou de répertoires) à copier,
- le dernier est le nom du répertoire où copier,
- les noms sont systématiquement les mêmes.

Nous venons de voir que l'utilisation de cp est dangereuse et l'on risque parfois d'effacer des fichiers importants. Les options de cp peuvent vous éviter des situations fâcheuses.

3.1.4 La commande mv ou déplacer, renommer des fichiers et des répertoires

La commande *mv* (abréviation du mot « *move* ») déplace un fichier existant ou le renomme. Sous sa forme la plus simple, elle a 2 arguments :

- le fichier source, qui doit exister et pouvoir être lu et supprimé,
- le fichier destination, qui doit être dans un répertoire où on a le droit d'écriture.

Renommer un fichier, c'est le déplacer !

Si le fichier destination existe déjà, il est remplacé; si les 2 fichiers sont dans le même répertoire, cela revient à en changer le nom (renommer). Et comme pour la commande *cp*, il peut y avoir n arguments, le dernier étant un répertoire.

Soyez prudent si vous utilisez le compte root (nous le verrons dans gestion des utilisateurs à la fin du TP). Renommer, déplacer ou supprimer des fichiers système peut entraîner de très sérieux dégâts !

3.1.5 La commande rm ou supprimer des fichiers et des répertoires

Quel que soit votre système d'exploitation, nettoyez de temps en temps le système de fichiers. Si vous ne le faites pas, vous risquez de ne plus avoir d'espace disque disponible. Linux possède deux commandes pour supprimer les fichiers et les répertoires. La première est la commande rm (abréviation du mot « remove »), utilisée sans option et supprimant des fichiers spécifiés.

Les options de la commande rm

Option	Description
-f	Supprime des éléments sans demander de confirmation
-i	Demande confirmation avant de supprimer quoi que ce soit
-r	Accède aux sous-répertoires et supprime les éléments qui s'y trouvent
-v	Affiche des informations au cours du processus de suppression

Ces options peuvent être utilisées séparément ou combinées.

La commande rm -rf est peut-être la plus dangereuse des commandes Linux, en particulier si vous êtes loggé sous root. Les fichiers supprimés ne pourront pas être restaurés !

Pour supprimer des répertoires, vous utilisez la commande rmdir qui se contente de supprimer les répertoires vides. De nombreux utilisateurs préfèrent utiliser une combinaison de commandes rm pour supprimer les fichiers, puis la commande rmdir pour supprimer les répertoires, les erreurs étant ainsi moindres.

Si vous tapez rm -rf * vous supprimez sans confirmation tout ce que se trouve à l'intérieur de ce répertoire, y compris les sous-répertoires et leur contenu. C'est une commande très dangereuse lorsqu'elle est exécutée sous root, elle risque de générer des dégâts irréversibles !

3.2 Références à plusieurs fichiers - Caractères génériques

Beaucoup de commandes acceptent une liste de fichiers dont les noms sont séparés par des espaces et constituent autant d'arguments.

Le shell fournit une notation pour citer plusieurs fichiers à la fois :

- le joker « * » représente une chaîne quelconque, éventuellement vide,
  **.c* *doc*unix.html*

- le joker « ? » dénote un caractère quelconque,
  ?nix **.s??*

- les caractères « . » (en début de nom de fichier) et « / » ne peuvent pas être reconnus par un joker.

- [...] représente un caractère appartenant à un ensemble
  [Uu]nix **.sx[ci]*

- [^...] dénote un caractère n'appartenant pas à un ensemble
  .[^co] *[^[:upper:]]**

La notation « *~nom d'utilisateur* » désigne le répertoire personnel de l'utilisateur cité. Par exemple :
~/helene $

La notation « ~ » représente le répertoire personnel de l'utilisateur courant. Par exemple :
~$

Le shell possède la caractéristique de terminer les noms de fichiers à votre place. Si vous écrivez un nom de fichier long sur la ligne de commandes, entrez les premières lettres, puis appuyez sur la touche Tab, le shell complète le reste ! C'est pareil pour les commandes.

TP n°1
Répertoires et Fichiers sous Unix

Exercices

Exercice n°1:

Testez la commande *ls* en affichant, depuis votre répertoire personnel initial (*home directory*), la liste de tous vos fichiers et sous-répertoires :

1. sous un format *condensé*
2. sous un format long (donnant le propriétaire, les permissions, la taille, ...)
3. en affichant les fichiers cachés (dont le nom commence par un point)
4. en colorant le type des fichiers et en ordre inverse
5. avec un format long et en affichant les fichiers cachés, mais du plus récent au plus ancien
6. avec un format long et en affichant les fichiers cachés, mais du plus ancien au plus récent

Exercice n°2:

Où que vous soyez, quel est l'effet de la commande *cd* sans paramètre ?

Exercice n°3:

Dans votre répertoire courant, créez en une commande les fichiers suivants :

annee1 Annee2 annee4 annee45 annee41 annee510 banane annee_saucisse

Exercice n°4:

Créez le répertoire *Year* dans votre répertoire courant, en une commande déplacez les fichiers précédemment créés dans le répertoire *Year*.

Exercice n°5:

1. Créez un répertoire *system* sous votre répertoire de travail, puis un répertoire *tp1* sous *system*
2. Effacez le répertoire *system* avec la commande *rmdir*. Que constatez-vous ?
3. Après avoir effacé les répertoires *tp1* et *system*, créez à l'aide d'une seule commande les répertoires *system, system/tp1, system/tp2*
4. Renommez le répertoire *system* en *test*
5. Copiez un fichier de votre choix du répertoire */bin* dans le répertoire *test/tp1* :
    1. depuis le répertoire */bin*
    2. depuis le répertoire *test/tp1*
    3. depuis votre *homedir*, en utilisant des chemins absolus
    4. depuis votre *homedir*, en utilisant des chemins relatifs
6. Effacez à l'aide d'une seule commande les répertoires *test/tp1* et *test/tp2*

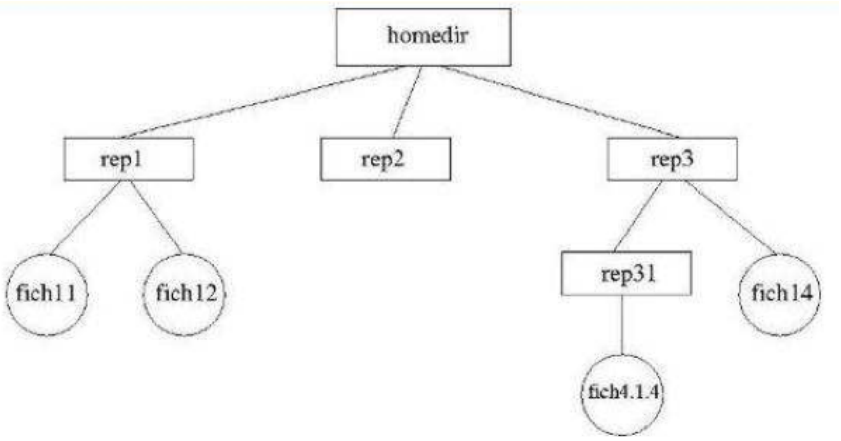
Exercice n°6:

1. Combien y a-t-il de noms de répertoires dans la racine ?
2. Donnez un exemple de nom de fichier se trouvant dans votre répertoire personnel :
    a. par un **chemin relatif**.
    b. par un **chemin absolu**.

TP n°1
Répertoires et Fichiers sous Unix

Exercice n°7:

Dans votre répertoire d'accueil (*/home/user* par exemple), créez l'arborescence suivante, en n'utilisant que des chemins *relatifs* :



puis, vérifiez.

Exercice n°8:

Comment déplacer toute l'arborescence *rep3* sous le répertoire *rep2* ? Vérifiez l'opération que vous avez faite en une seule commande.

Exercice n°9:

Copier les fichiers dont le dernier caractère est un 4 ou 1 dans le répertoire */tmp* en une seule commande. Supprimez tout sauf *rep1*, *fich11* et *fich12*.

Ce document a été réalisé à l'aide des supports suivants :
http://www.funix.org/fr/unix/exercices.htm
et
le livre « Linux pour les nuls »