Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2020 – 2021
DEV1 – ENVL

DEV1 – ENVL – Laboratoire d'environnement système

TD 2 – GNU/Linux (partie II)

Table des matières

Dans votre répertoire `~/dev1`, créez un répertoire `td2`. Il contiendra tous les fichiers que vous allez créer aujourd'hui.

Version du document
12 septembre 2020

1 Système de fichiers

Pour l'instant vous êtes resté dans votre dossier personnel mais le monde est plus vaste. Explorons le système de fichiers.

1.1 Racine du système de fichiers

Rappelez-vous que les fichiers sont organisés en hiérarchie : un dossier contient d'autres dossiers qui lui-même...

On a vu que chaque utilisateur dispose d'un dossier personnel. Où se trouve-il ? Dans un dossier appelé `home`. Et où se trouve ce dossier `home` ? Dans un dossier appelé « `/` » (la barre oblique). Et où se trouve ce dossier « `/` » ? Nulle part ;) C'est le dossier principal (la *racine*).

La racine du système de fichiers

La *racine* est le dossier principal, tout en haut de la hiérarchie des dossiers.

1.2 Chemin absolu et relatif

« Tout est relatif, et cela seul est absolu. » – Auguste Comte

Régulièrement, il faut indiquer un endroit du système de fichiers (par exemple pour y aller). Désigner simplement le nom du dossier ne suffit pas. Nous avons déjà abordé la notion de *chemin* ; Précisons.

Chemin absolu et relatif

Chemin

Un *chemin* est une suite de dossiers à traverser pour arriver au dossier ou au fichier qui nous intéresse. On les sépare par « `/` ».

Par exemple : `home/g12345/dev1/td1/test` indique que dans le dossier `home` il y a un dossier `g12345` qui contient un dossier `dev1` qui contient un dossier `td1` qui contient un fichier (ou un dossier) `test`.

Chemin absolu

Un chemin est *absolu* s'il commence à partir de la *racine*. Il commence donc par « `/` ». Par exemple : `/home/g12345/td1/test`

Chemin absolu

Dans le cas contraire il est *relatif*. Il s'agit d'un chemin à suivre à partir du *dossier courant*.

Un chemin *absolu* désigne toujours le même endroit. Un chemin *relatif* non ! ça dépend d'où on est, du dossier courant.

Exercice 1

pwd

Entrez la commande `pwd` (que fait-elle encore ?). Comprenez-vous la notation qu'elle utilise pour la réponse ? Est-ce une chemin absolu ou relatif ?

Rappel : dans une commande, un nom de fichier (ou de dossier) désigne toujours un fichier (ou un dossier) se trouvant dans le dossier courant
Pour indiquer un *fichier* (ou un dossier) se trouvant *ailleurs*, il faut donner son *chemin* (absolu ou relatif).

Par exemple : `ls /home/g12345` permet de visualiser le contenu de la home de `g12345` (quel que soit l'endroit où on se trouve puisque c'est un chemin absolu).

Exercice 2 **Utilisation d'un chemin**

Supposons que vous êtes dans votre home et que vous ne pouvez pas vous déplacer (la commande `cd` est interdite !).

1. Affichez le contenu du fichier `readme` (qui se trouve dans le dossier `td1`) en utilisant un chemin *relatif*.
2. Faites la même chose en utilisant un chemin *absolu*.
3. Créez une copie du fichier `welcome` dans le dossier `dev1/td2` en utilisant des chemins absolus.

1.3 Raccourcis pour des chemins

Raccourcis

Dans un chemin :

- ▷ `~g12345` : désigne la home de l'utilisateur `g12345` ;
- ▷ `~` : désigne la home de l'utilisateur qui entre la commande ;
- ▷ `..` : désigne le dossier parent du dossier courant ;
- ▷ `.` : désigne le dossier courant.

Exercice 3 **Se familiariser avec les raccourcis**

Placez-vous dans votre dossier personnel. En utilisant les raccourcis afin de trouver la solution la plus courte possible :

1. Affichez le contenu de votre dossier personnel en utilisant un chemin absolu.
2. Affichez le contenu de la home de votre professeur en utilisant un chemin absolu.
3. Refaites la même chose mais en utilisant un chemin relatif.

Exercice 4 **Comprendre un chemin**

Que désigne le chemin suivant : `~mcd/../../home` ?

Exercice 5 **Chemins absolus et relatifs**

Parmi tous les chemins suivants, quels sont ceux qui sont *relatifs* ?

- ☐ `~/../g12345/td2`
- ☐ `/home/g12345/../g54321/Hello.java`
- ☐ `./tds/td2`
- ☐ `tds/td2/Hello.java`
- ☐ `~g12345/tds/td2`

Exercice 6 **Utilisation d'un chemin (2)**

On voudrait créer une copie du fichier `welcome` (vu ci-dessus) dans le dossier `~/dev1/td2`.

1. Quelle est la commande pour le faire en utilisant des chemins *absolus* si le dossier courant est le répertoire personnel ?
2. Idem mais avec des chemins *relatifs*.
3. Avec des chemins *relatifs* si le répertoire courant est `~/dev1/`.
4. Idem mais avec des chemins *absolus*.
5. Avec des chemins *relatifs* si le répertoire courant est `~/dev1/td2/`.
6. Idem mais avec des chemins *absolus*.

2 Ligne de commandes

Nous avons déjà vu quelques règles concernant la ligne de commande. Voyons quelques notions supplémentaires qui vont vous être utiles.

2.1 Complétion de commande

Parfois, vous ne vous rappelez plus exactement d'une commande. Le shell peut vous aider.

Expérience 1

La complétion de la commande

Supposons que vous ne vous rappeliez plus très bien de la commande qui permet de modifier le mot de passe. Vous vous rappelez juste qu'elle commence par `pas`.

- Tapez `pas` sans appuyer sur `ENTER`.
- Appuyez 2× sur la touche `TAB`. Le shell affiche la liste des commandes commençant ainsi.
- Tapez sur `s` puis à nouveau 2× sur la touche `TAB`. Comme il n'y a plus qu'une seule possibilité, le shell complète la commande.

Exercice 7

La complétion des noms de fichiers

La touche de tabulation permet également de compléter un nom de fichier.

1. Copiez chez vous le fichier
   `monfichieraunomtellementlongquilmeparaitpeuprobabledeletaper2xsanserreur`
   qui se trouve dans le dossier `/eCours/dev1/envl/`.
2. Affichez le contenu de ce fichier en évitant de retaper son nom.

2.2 Historique

Il arrive souvent qu'on oublie une commande qu'on a déjà utilisée avant. Le shell retient un historique des commandes tapées par un utilisateur et il est possible de l'exploiter.

Historique

- ▷ Les flèches *haut* et *bas* permettent de se déplacer dans les commandes déjà entrées.
- ▷ `history` affiche la liste (numérotée) des dernières commandes tapées.
- ▷ `!n` réexécute la commande numéro *n*.
- ▷ `!mot` réexécute la commande la plus récente *commençant* par *mot*.
- ▷ `!?mot` réexécute la commande la plus récente *contenant mot*.
- ▷ `^chaine1^chaine2^` réexécute la dernière commande en remplaçant *chaine1* par *chaine2*.

2.3 Options

La plupart des commandes possèdent des *options*.

Les options

Une *option modifie le sens* d'une commande ;

- ▷ Elle commence par le signe `-` suivi d'une seule lettre ;
- ▷ ou encore par le double tiret `--` suivi d'un nom d'option.
- ▷ Elle est placée n'importe où après le nom de la commande.

Expérience 2 **Expérimenter les options**

- ✎ Tapez la commande `ls -l`. Vous constatez que le résultat obtenu est beaucoup plus verbeux[1] que celui obtenu sans l'option.
- ✎ Entrez la commande `cat -n welcome`. L'option demande de numéroter les lignes
- ✎ Essayez `cat --number welcome`, la version *longue* équivalente à la précédente.

3 Fichiers cachés

Nous avons vu que le shell retient les dernières commandes entrées. Où exactement est ce que c'est retenu ? Dans un fichier qui s'appelle `.bash_history` dans votre *home*.

Expérience 3 **Un fichier caché**

Vérifions ce qu'on vient d'affirmer.

- ✎ Entrez `cd` pour vous assurer d'être dans votre *home*.
- ✎ Entrez `ls` pour en voir le contenu.
  Le fichier `.bash_history` n'a pas l'air d'être présent ! ?
- ✎ Entrez `cat .bash_history` pour quand même en afficher son contenu.
  Le shell vous affiche bien vos anciennes commandes sans afficher d'erreur ! ?

1. Vous apprendrez dans un prochain TD à comprendre ces informations supplémentaires.

Fichiers cachés

Un *fichier caché* est un fichier dont le nom commence par un point.

- ▷ Par défaut, il n'est pas montré par la commande `ls`.
- ▷ L'option `-a` demande de montrer les fichiers cachés.

Expérience 4

Voir les fichiers cachés

Testons ce qu'on vient d'apprendre.

✎ Entrez la commande `ls -a`.
La commande affiche beaucoup plus de fichiers qu'avant, dont celui qui nous intéresse.

4 Documentation

4.1 Recherche d'informations

Non seulement, il y a beaucoup de commandes à connaitre mais, en plus, chacune dispose d'une multitude d'options. Impossible de tout retenir ! Comment faire ?

Information

- ▷ `nomCommande --help` affiche une aide succincte sur la commande.
- ▷ `man nomCommande` affiche une documentation plus complète. (*q* pour quitter).

Si vous ne connaissez pas le nom de la commande, vous pouvez

- ▷ utiliser `man 1 intro` pour lire une introduction aux commandes les plus simples ;
- ▷ consulter les ressources que l'on met à votre disposition sur poESI ;
- ▷ utiliser `apropos` suivi d'une expression à chercher ;
- ▷ utiliser votre moteur de recherche favori, pour chercher par exemple une *GNU/Linux Cheat Sheet*.

Exercice 8

Trouver la bonne option

La commande `ls -l` affiche le contenu du dossier en format *long*. La 5e colonne donne la taille du fichier (en octets). Lorsque les nombres sont grands, ce n'est pas très lisible. Trouvez l'option qui permet d'afficher cette taille sous un format plus lisible.

De grâce, *cherchez* la réponse, ne la *demandez pas* à votre voisin. Le but de cet exercice n'est pas de connaitre l'option (elle n'est pas si utile que ça) mais d'apprendre à trouver soi-même l'information.

4.2 Parcours d'un long document

Lorsque vous avez demandé de l'aide, la commande `man` vous a affiché le contenu d'un fichier contenant cette aide. Comme il est long, il ne l'a pas affiché comme le ferait un `cat` mais au travers d'une commande qui permet d'afficher *page par page* un document.

less

`less nomDeFichier` permet de *consulter* le fichier en question *page par page*.

- ▷ Ce n'est pas un éditeur ; vous ne pouvez *pas modifier* le document.
- ▷ Les flèches et les touches `PGUP` et `PGDN` permettent de se déplacer dans le document.
- ▷ Appuyez sur `q` pour terminer la consultation.
- ▷ `/mot` cherche le mot donné dans le document.
- ▷ `n` montre l'occurrence suivante du mot recherché.

Toutes ces facilités peuvent être utilisées lorsque vous parcourez une aide [2].

Exercice 9

Trouver la bonne option (II)

Quelqu'un m'a dit que la commande `ls pouvait` trier l'affichage des noms fichiers dans l'ordre chronologique de leur dernière modification, et mettre le fichier le plus récemment modifié en dernier dans la liste. Aide : en anglais, trier se dit « sort ».

5 Conclusion

Notions importantes de ce TD

Voici les notions importantes que vous devez avoir assimilées à la fin de ce TD.

- ▷ Comprendre la notion de chemin absolu et relatif. Savoir les utiliser dans les commandes.
- ▷ Connaitre la signification dans un chemin de : « `~` », « `.` » et « `..` ».
- ▷ Pouvoir demander plus d'informations sur une commande, ses options. Comprendre les informations fournies.
- ▷ Comprendre la notion de fichier caché.
- ▷ Savoir utiliser à bon escient les facilités du shell pour simplifier l'entrée des commandes (historique, flèches...)

Félicitations !

Vous êtes arrivé au bout de ce TD. Avant de quitter le laboratoire, n'oubliez pas de quitter proprement la connexion avec `linux1` (`exit`) et d'éteindre l'ordinateur (ou au moins de vous déloguer).
À la semaine prochaine et soyez à l'heure !

2. Il en existe bien plus ! `man less` si vous êtes intéressé.