Chemin relatif ou absolu illustrés par quelques commandes

- Objet : Exercices sur les chemins relatif ou absolu
- Niveau requis :
  débutants
- Commentaires : *les bases du bash, rappels et exercices*
- Débutant, à savoir :
  - Utiliser GNU/Linux en ligne de commande, tout commence là !.
  - sur le shell : shell
  - sur répertoires et chemins : chemins relatif
  - sur chemin relatif et absolu : repertoires
  - sur système droits:fichiers : droits-unix-bis
  - sur la commande cd : doc:programmation shell:cd
  - sur les différentes commandes : gnu_linux
- Suivi :
  - Création par Hypathie 24/01/2014
  - Testé par <...> le <...> Fix Me!
- Commentaires sur le forum : Lien vers le forum concernant ce tutoial [1)]

Nota :

Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Introduction

Objectifs

S'exercer sur quelques commandes en vue d'assimiler les notions de **chemin relatif** et de **chemin absolu**.
Ou se servir de différentes expressions syntaxiques pour écrire des commandes équivalentes.

À savoir

- **Le répertoire courant** est le répertoire dans lequel on se trouve.
- Attention à la confusion entraînée par l'utilisation de l'expression “son home”, préférerez plutôt “**son répertoire personnel**”.
- À l'ouverture d'une session de terminal, le répertoire courant est notre répertoire personnel,

comme nous l'indique le prompt de l'utilisateur `toto` ci-dessous:

```
toto@debian:~$
```

Mais ici :

```
toto@debian:~/Documents$
```

le prompt indique que le répertoire courant est le fichier de type répertoire “Documents” situé dans le répertoire personnel de l'utilisateur “toto” :

Ainsi par exemple :

⇒ Après l'ouverture du terminal entrez la commande **ls** puis validez votre commande en tapant sur la touche “entrée” : vous obtiendrez la liste des fichiers contenus dans votre répertoire personnel.

Puis en entrant: `cd “Documents”` et en validant votre saisie par la touche “entrée”, le **répertoire courant** sera alors le répertoire “Documents”.

* **Le tilde : ~/** est une variable de l'interpréteur de commande **bash** qui sera interprété, s'il n'est pas entouré de guillemets ou d'apostrophes, comme le répertoire personnel de l'utilisateur qui a lancé la commande. Ainsi, par exemple, l'utilisateur `toto` pourra utiliser le caractère tilde dans une ligne de commande en remplacement du chemin absolu de son répertoire personnel : **/home/toto**

Il permet donc d'écrire dans les commandes le chemin relatif à son répertoire personnel d'un fichier de type-répertoire ou d'un fichier-ordinaire [2)]

L'utilisateur `toto` peut ainsi écrire :

en utilisant un chemin absolu (pour un utilisateur “toto”)

```
cd /home/toto/Documents
```

ou

```
 cd ~/Documents
```

```
echo ~
```

```
/home/toto
```

(*Pour aller plus loin, voir* tilde)

Autres précisions :

1) Le point “.” représente le répertoire courant (où l'on se trouve).
Et deux points “**..**” indiquent le répertoire parent de celui dans lequel on se trouve.

2) Le slash “/“ représente la racine mais il sert aussi à séparer les composants d'un nom de fichier (comme dans /home/toto/Documents/fiche-anglais)[3)]

Ainsi par exemple:

cd .. : permet de retourner dans le répertoire parent du répertoire courant ou actuel (depuis le répertoire “Documents” par exemple **les deux points** signifient remonte au répertoire parent du

répertoire “Documents”)

cd . : permet de rester dans le répertoire courant (**le point** indiquera le répertoire où l'on se trouve)

cd et **cd ~** : permettent de retourner dans son répertoire personnel, depuis n'importe quel répertoire.

```
          ( le tilde ''~'' étant interprété comme le chemin du répertoire
personnel de l'utilisateur).
```

cd / : permet d'aller à la racine du système de fichier.



Si l'on lance une commande en root, le chemin relatif du répertoire personnel de l'utilisateur toto (représenté par ~) ne fonctionnera pas. Il faudra alors utiliser le chemin absolu.
Et cela, parce que le répertoire “Documents” fait partie du compte de l'utilisateur toto, mais ne fait pas partie au compte root.
Ainsi par exemple, pour déplacer un fichier-ordinaire du fichier-répertoire “Bureau” au fichier-répertoire “Documents”, la commande suivante fonctionnera :

```
mv                /home/toto/Bureau/fichier-ordinaire
/home/toto/Documents/fichier-ordinaire
```

Et la commande suivante ne fonctionnera pas en root, mais seulement en user:

```
 mv ~/Bureau/fichier-ordinaire ~/Documents/fichier-ordinaire
```

Énoncé des questions ci-dessous

Soit le répertoire personnel d'un utilisateur "toto" (également appelé "home" de "toto")

le prompt à l'ouverture du terminal est alors :

```
 toto@debian:~$
```

```
pwd
```

```
/home/toto
```



vocabulaire : Ce n'est pas le prompt qui répond, il ne sert que d'indicateur. C'est le programme de la commande qui retourne le texte affiché.
⇒ **Entrainez-vous en entrant pwd, après avoir rentré chacune les trois utilisations de cd indiquées ci-dessus. Puis en les entrant après avoir avancé jusqu'où vous le souhaitez dans l'arborescence des fichiers !**

Soit la création dans ce répertoire personnel d'un fichier de type répertoire nommé essai1

```
mkdir essai1
```

Soit la création d'un autre fichier de type répertoire nommé « essai2 »

après les commandes suivantes :

1) pour avancer dans l'arborescence et créer « essai2 » dans « essai1 »

```
cd essai1
```



À remarquer : le prompt est alors

```
toto@debian:~/essai1$
```

après **pwd** le lieu indiqué est bien :

```
/home/toto/essai1
```

2) Pour créer un fichier de type répertoire « essai2 » depuis son répertoire parent (“essai1”)

Entrez alors :

```
mkdir essai2
```



Depuis “essai1” on aurait pu créer “essai2” en faisant

```
mkdir /home/toto/essai1/essai2
```

3) Pour créer le fichier normal “lili” dans le fichier de type répertoire “essai2”
(reconnu aussi sous l'écriture ”/home/toto/essai1/essai2” ou “~/essai1/essai2”) Utilisons la commande **echo** pour y inscrire le mot « bonjour » comme ceci :

depuis le répertoire parent « essai1 » du fichier “lili” on peut faire :

```
cd essai2/
```

depuis le répertoire parent « essai1 » du fichier “lili” on aurait pu faire aussi :

```
cd /home/toto/essai1/essai2/
```



Remarquez le changement du prompt qui indique le répertoire actuel comme étant “essai2”

```
toto@debian:~/essai1/essai2$
```

Pour créer le fichier ordinaire “lili”, dans le répertoire “essai2” :



utilisez la commande echo selon la syntaxe suivante
(avec ou sans les guillemets sur le mot “bonjour”) :

```
 echo "texte à écrire dans un fichier" > /chemin-absolu-du-fichier-à-créer/nom-du-fichier-créé
```

On notera que le texte peut contenir des retours à la ligne, il doit alors impérativement être entouré de guillemets.

```
echo bonjour > lili
```

Ou encore

```
echo bonjour > /home/hypathie/essai1/essai2/lili
```

4) Retour au niveau dans son répertoire personnel en faisant :

```
cd ~
```

À la place de ce long énoncé

On aurait pu créer en une seule commande les deux répertoires “essai1” et “essai2” ainsi

```
mkdir -p /home/toto/essai1/essai2
```

puis pour créer le fichier ordinaire “lili”

```
echo "bonjour" > /home/toto/essai1/essai2/lili
```

Questions sur cette base :

Afin d'assimiler les notions de chemin relatif et absolu, je vous propose d'essayer de répondre de vous-mêmes aux questions avant d'aller regarder la réponse.

1) Reformulez l'énoncé au moyen du chemin relatif

En faisant :

```
mkdir -p ~/essai1/essai2
```

puis

```
echo "Bonjour" > ~/essai1/essai2/lili
```

2) Comment aller directement (sans une succession de cd, puis de cd ..) de son répertoire personnel à "lili", puis retourner du répertoire contenant "lili" à son répertoire personnel ?

a) si on connaît où est “lili” : utiliser l'autocomplétion (touche tabulation) :

```
cd essai1/essai2/
```



le prompt est maintenant:

```
toto@debian:~/essai1/essai2$
```

b) on a malheureusement oublié dans quel répertoire est rangé “lili” :

* **astuce pour que soit indiqué le chemin absolu d'un fichier:**



depuis n'importe où (car le programme “locate” n'utilise pas, pour ses recherches, l'arborescence du système de fichiers, mais une base de données, mise à jour régulièrement par le programme “updatedb”) :

```
updatedb
```

```
locate lili
```

```
/home/toto/essai1/essai2/lili
```

Puis du répertoire contenant “lili” à mon répertoire personnel ?

```
toto@debian:~$ cd essai1/essai2/
toto@debian:~/essai1/essai2$ cd ~
toto@debian:~$
```

3) Corriger la faute d'orthographe sur "bonjour" en écrasant "lili", depuis son home ?

Pour ce faire utiliser la commande echo.



On est au niveau de son répertoire personnel
le prompt est

```
toto@debian:~$
```

```
echo Bonjour > ~/essai1/essai2/lili
```

4) Trouvez deux façons d'écrire le chemin relatif de la commande ci-dessus ?

```
cd /home/toto/essaie1/essai2
```

⇒

```
cd ~/essaie1/essai2
```

```
cd ./essaie1/essai2
```

Écritures équivalentes d'une commande

1) Pour aller du home à la racine avec cd puis avec l'usage des points ?

```
cd /
```

puis

```
pwd
```

```
/
```



Et le prompt !

```
toto@debian:/$
```

ou encore

```
cd ../..
```

```
pwd
```

```
/
```

2) Retourner dans son home depuis la racine à l'aide du tilde ?

```
cd ~
```

ou

```
cd ~
```



le prompt !

```
toto@debian:~$
```

3) Utiliser le point avec une autre commande que cd :

Depuis son home : copier de trois façons différentes “lili” dans le fichier de type répertoires nommé « essai1 » ?

en absolu :

```
cp  /home/toto/essai1/essai2/lili /home/toto/essai1/lili
```

en relatif (depuis n'importe où) :

```
cp ~/essai1/essai2/lili ~/essai1/lili
```

en relatif (depuis son répertoire personnel) :

```
cp ./essai1/essai2/lili ./essai1/lili
```

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

2)

Cette distinction parce que sous Linux tout est “fichier”, mais en pratique, on peut dire fichier de type-répertoire pour désigner les fichiers qui peuvent contenir d'autres fichiers ; tandis que fichier-ordinaire renvoie aux fichiers dans lesquels on peut écrire du texte, par exemple. Pour une explication très approfondie de cette différence, voir http://www.linux-france.org/article/dalox/unix02.htm Et bien sûr en tapant “fichier” dans la fenêtre rechercher de la documentation de **Debian-Facile** vous aurez une idée sur les différentes sortes de texte que l'on peut inscrire sur un fichier.

3)

C'est donc un caractère qu'il est interdit d'utiliser dans les noms de fichiers de tout type (normaux, répertoires, de périphériques.

From:
http://debian-facile.org/ - **Documentation - Wiki**

Permanent link:
http://debian-facile.org/doc:programmation:shell:illustration-navigation-shell

Last update: **03/07/2014 00:06**