I106B - Linux 1

Ajouter le repertoire courant " . " au PATH, est-ce une bonne idée?

(cf. exercices 7 et 8 de la fiche 15)

Par facilité, certains utilisateurs de systèmes Unix/Linux ajoutent le répertoire courant à leur PATH afin de ne plus devoir systématiquement entrer le préfixe « ./ » pour exécuter un fichier du répertoire courant (ex : `./monscript.sh`). En effet si « . » fait partie du PATH, le shell recherchera également les commandes dans le répertoire courant (ex : on peut entrer `monscript.sh` sans spécifier que le fichier est dans le répertoire courant).

Si vous êtes le seul utilisateur de la machine, cette pratique est acceptable, tant que vous savez ce que vous faites !

Le problème est qu'en ayant votre répertoire actuel dans PATH, vous ne pouvez pas voir les commandes comme une liste constante. Vous aurez tous vos petits scripts partout dans votre système de fichiers qui seront exécutables dès que vous serez dans leur répertoire. Un jour, vous exécuterez certainement le mauvais script. Donc, avoir votre PATH comme une liste prédéfinie de chemins statiques (donc sans le répertoire courant) est une question d'ordre et de se prémunir contre un problème potentiel.

Soyez conscient qu'avoir « . » dans le PATH est un énorme risque de sécurité. En voici un cas simple. Si vous êtes *root* sur le système et que ce système contient les comptes d'autres utilisateurs, vous pouvez faire un `cd` dans le répertoire d'un utilisateur et y exécuter involontairement un script malveillant sans vous en rendre compte. Cela sera notamment le cas si vous avez fait une faute de frappe (ex : `command` au lieu de `commande`) ou que le script a le même nom qu'un binaire du système[1] (rien n'empêche un utilisateur de définir un script se nommant `ls`).

> *Exemple* : Imaginez que quelqu'un place un script appelé "`ls`" chez lui (ou dans n'importe quel répertoire) et que ce script appelle les commandes "`cd ~ ; rm -f`" qui effacerait tous vos fichiers (ou tout autre fichier sensible) !

Ainsi, ajouter le répertoire de travail actuel à votre chemin est considéré comme très dangereux. Vous ne pouvez plus vous déplacer en toute sécurité dans un système de fichiers.

[1] Remarquez que ce risque s'applique aux commandes dites « externes », qui ne font pas partie des *builtin commands* (les commandes « internes » du shell). En effet, le shell vérifie d'abord si la commande entrée fait partie de ses commandes internes (comme *cd*). Ce n'est que dans le cas où il ne la trouve pas qu'il parcourra l'ensemble des répertoires définis dans la variable d'environnement PATH à la recherche d'un fichier exécutable qui a le nom indiqué.

Si vous avez besoin d'exécuter un script ou un programme à partir de votre répertoire courant, une meilleure pratique est simplement de toujours l'exécuter en ajoutant explicitement le préfixe `./` à son nom (vous dites au système "Je veux exécuter ce fichier à partir de mon répertoire courant").

Cependant, si vous voulez absolument ajouter « . » à votre PATH, il vaut mieux l'ajouter à la fin de la liste plutôt qu'au début :

```
export PATH=$PATH:.
```

En effet, de cette manière, le shell cherchera d'abord une commande dans les répertoires *bin* du système avant de la chercher dans votre répertoire courant. Cela empêche le risque d'exécuter un script malveillant ayant le même nom qu'une commande shell.

Solution recommandée

Pour pouvoir accéder à vos scripts de n'importe quel répertoire (tout en évitant de devoir spécifier le chemin du script ou d'ajouter le repertoire courant au PATH), placez tous les scripts dans un répertoire local caché de votre *home directory* (par exemple: `~/.local/bin`) et ajoutez ce répertoire à la fin du PATH :

```
export PATH=$PATH:~/.local/bin
```

Cette configuration empêche bien le risque d'une redéfinition malicieuse d'une commande système car, même en voyageant dans le compte d'un autre utilisateur, ~ dans PATH reste lié à mon propre *home directory* (contrairement à . et .. qui sont par conséquent dangereux). Il s'agit donc de la manière la plus sûre d'enrichir les commandes du shell avec vos propres commandes (i.e. vos programmes et scripts exécutables).