Correction IN3R21-TP1

1) ~ et $HOME sont susbstitués par le répertoire par défaut de l'utilisateur logé. ~xxx, ~hilairex, et ~bin sont susbstitués par le répertoire par défaut des utiisateurs xxx, hilairex, et bin. "~hilairex" nest pas substitué à cause des double-quotes

2) Elles suppriment le dernier et le premier mot de la liste, respectivement.

3) Elle développe les fichiers dont le nom se termine par d, et qui résident dans un répertoire de 3 caractères rattaché à la racine et commençant par un e.

Oui: la deuxième forme impose que le nom de fichier comporte au moins 5 caractères.

4) -l est l'option 'long' (pour lister les droits, le compteur de liens, le propriétaire, la taille, la date de dernier accés). -a inclut les fichiers commençant par un point '.'. -L "suit" les liens symboliques, çàd n'affiche pas les liens mais les fichiers auxuquels ils mènent.

5) xterm& lance le terminal en arrière-plan. On peut le rattacher au shell propriétaire avec la commande fg.

6) Il doit lancer lui aussi cat acces.txt , et cat se verra refuser l'accès.

7) Parce que '.' ne fait normalement pas partie du PATH. Solution: PATH="$PATH:."

8) VAR=hello ./maint
- ou -
VAR=hello; export VAR
./maint

9) L'appel avec arguments a pour effet d'ouvrir le fichier maint.txt, ce que le binôme ne peut normalement pas faire avec les droits 0644 par défaut.

Deux solutions: i) changer les droits de maint.txt en 0664 (voire 0666 pour autoriser l'écriture à tout le monde) avec chown
ii) conserver les droits 0644 de maint.txt, mais rajouter un bit SetUID sur l'exécutable, soit chmod u+s maint

La deuxième est évidemment préférable, car l'écriture sur le fichier se fera via le programme, qui constitue le seule moyen de contrôle restant.

10) On édite en réalité répertoire/cible.ext, après avoir suivi successivement deux liens relatifs. Le fait de supprimer le fichier lui-même, n'enlève malheureusement pas liens symboliques, d'où erreur à l'ouverture. Une différence notable entre liens logiques et matériels (durs).

11) access.txt : le compteur de liens a augmenté d'une unité.

12) Les fichiers désignent désignent en le même fichier physique, ce qu'on peut vérifier immédiatement par une modification dans l'un qui se retrouve dans l'autre. ls -li liste le numéro d'inode en premier, on peut donc s'apercevoir qu'il s'agit du même fichier physique.

13) Le lien dur est impossible car il ne s'adresse plus au même système de fichiers. Certains systèmes transforment parfois le lien dur en simple copie dans

ce cas, ce qui peut être catrastrophique.

14) C'est un pipeline : il redirige la sortie standard d'un processus vers l'entrée standard d'un autre.
ls -l | wc comptera les lignes, mots, et nombre de caractères produits par la sortie de ls. ls -l | grep dr va lister les répertoires lisibles, ls -l | grep -v dr tous les autres fichiers, et ls -l | grep dr | wc -l va compter le nombre de répertoires lisibles.

15) cat /etc/passwd | grep -v \#

ou cat /etc/passwd | grep -v "#"

mais pas # seul qui marque le début de commentaire

16) La première ligne extrait le premier champs de /etc/passwd avec ':' comme délimiteur, la deuxième fait de même avec les champs 4 et 5.

17) Le résultat de l'affichage de ls, par redirection.

18) sortie ne contient plus rien, et c'est normal: le message d'erreur est sorti sur le canal d'erreur standard, soit le numéro 2. Méthodes pour le capturer:
- par redirection: ls -l /rien 2> sortie
- par unification: ls -l /rien > sortie 2>&1

19) aucune différence au niveau du résultat. Mais dans le 1er cas, on a 2 processus et un pipe, qui doit recopier régulièrement la sortie du 1er vers l'entrée du 2ème, alors que dans le deuxième forme on n'a qu'un seul processus qui, par redirection, lit un fichier au lieu du clavier.

20) echo "Le fichier $FIC comporte `cat $FIC | wc -l` lignes"