Mini-projet C – Rapport

Par Simon Nicod et Julien Mourcely

1. Points forts des programmes

1.1. Commande xxd

Tout d'abord, le premier programme est très simple d'utilisation, en effet, il suffit de taper "./xxd fichier" dans le terminal afin de lire le fichier de la même manière que le ferait la commande originale.

De plus, l'un de ses avantages par rapport à l'originale, est qu'elle peut prendre en compte un paramètre ne correspondant à aucun fichier, dans ce cas, elle crée un fichier temporaire dans lequel elle stocke les arguments. Ensuite, elle renvoie ce que renverrait la commande xxd originale si elle était utilisée sur ce même fichier, avant de le supprimer.

Si aucun argument n'est entré, le programme attendra que l'utilisateur entre une chaîne de caractères en entrée, la commande aura alors le même effet que si l'entrée avait été donnée en argument. De ce fait, la commande xxd ne renverra jamais d'erreur car s'adaptera à la saisie de l'utilisateur. Pour finir, elle respecte les normes données.

1.2. Commande compare

Pour ce qui est de la seconde commande, la mémoire est allouée dynamiquement, on ne consomme donc pas de mémoire de manière excessive. Nous avons mis en place une gestion d’erreur concernant la syntaxe de l’entrée, l’allocation mémoire, l’ouverture de fichiers et la suppression de fichiers temporaires.

Pour finir, la commande respecte toutes les normes ayant été données dans le sujet, ce qui fait de cette commande une réussite selon nous.

2. Points faibles des programmes

2.1. Commande xxd

La commande que nous avons réalisé a bien évidemment quelques faiblesses, tout d'abord et surtout en terme d'utilité, elle n'a pas vraiment d'usage, étant donné que nous nous basions sur une commande déjà existante, de plus, la nôtre est bien moins ergonomique et est probablement bien moins rapide et efficace que la commande originale. Cela à part, notre commande fonctionne très bien, et elle n'a pas non plus énormément de points faibles.

2.2. Commande compare

Cette commande a également quelques faiblesses, notamment le fait que les deux commandes entrées ne sont pas exécutées exactement au même moment, ce qui peut donner des résultats contre-intuitifs. Cela se remarque directement lorsque l'on essaie la commande "./compare ls -l /tmp -- ls -l /tmp". En effet, cette commande nous donne alors deux sorties différentes, étant donnés que les numéros de processus ne seront pas les mêmes à deux moments différents, bien que la commande soit la même des deux côtés.

3. Interfaces utilisées

3.1. Commande xxd

Pour ce qui est de la première commande, nous avons utilisé des interfaces de haut-niveau pour la création, l'ouverture et la fermeture du fichier temporaire, en revanche, nous avons utilisé des interfaces de bas-niveau pour l'ouverture, l'obtention des caractères des fichiers à lire, et la fermeture du fichier déjà existant.

3.2. Commande compare

Dans ce programme, l'interface de bas-niveau a été utilisée avec l'utilisation de la commande "mkstemp", qui retourne un entier, afin d'ouvrir les fichiers.

4. Exécution des différentes commandes dans le deuxième exercice

Les commandes sont stockées dans un pointeur de tableau de caractères. Elles seront chacunes utilisées dans un processus avec la commande "exevp", "dup2" permettant de rediriger la sortie des commandes vers les fichiers temporaires. On utilisera ensuite la commande "execlp" dans un nouveau processus pour exécuter "diff -u".

5. Utilisation du deuxième programme pour vérifier le premier

Le programme du deuxième exercice peut être utilisé pour comparer la commande xxd originale et la commande créée pour le premier exercice, ainsi, on peut vérifier si la commande réalisée lors du premier exercice est correcte.

On utilisera donc la commande suivante : "compare xxd test.txt -- ./xxd test.txt".