TD de Systèmes d'exploitation
François Cayre / françois.cayre@grenoble-inp.fr
Timothée Gerber / timothee.gerber@gipsa-lab.fr

Utilisation du shell et filtre de déchiffrage

Notions abordées : filtre, entrée et sortie standards, shell.

Symboles du shell utilisés : chaînage ou *pipe* (`|`), redirection (`<`, `>`, `>>`, `2>`, `2>>`, `2>&1`)

1 Rappels

Lorsqu'un processus est lancé par le système d'exploitation, celui-ci ouvre par défaut les fichiers associés à l'entrée standard (`stdin`, lecture seule), à la sortie standard (`stdout`, écriture seule tamponnée) et à la sortie d'erreur standard (`stderr`, écriture seule non tamponnée). Historiquement, `stderr` correspondait à l'imprimante, mais elle est maintenant redirigée vers l'écran, tout comme `stdout`. L'entrée standard `stdin` correspond au clavier.

1.1 Redirection

Dans le shell, il est possible de rediriger la sortie standard d'un processus vers un fichier. Par exemple, pour stocker la sortie de `ls` dans un fichier `catalogue.txt`, on écrira :

```
$ ls > catalogue.txt
```

Cette redirection a cependant pour effet de supprimer le contenu du fichier s'il existe. Si l'on souhaite conserver le contenu du fichier et écrire à la fin de ce dernier, on doublera le symbole '`>>`'.

De la même manière, la sortie d'erreur standard peut se rediriger vers un fichier à l'aide des opérateurs '`2>`' et '`2>>`'.

```
$ ls > catalogue.txt 2> erreurs.txt
```

La sortie d'erreur standard peut être redirigée vers le même fichier que la sortie standard et dans ce cas, pour éviter d'entrer deux fois le nom du fichier, on utilisera le raccourci '`2>&1`'. Les deux lignes suivantes sont donc équivalentes :

```
$ ls > catalogue.txt 2> catalogue.txt
```

```
$ ls > catalogue.txt 2>&1
```

Si l'on ne souhaite pas afficher les erreurs ou le résultat d'une commande, ni à l'écran, ni dans un fichier, il existe un puits sans fond nommé '`/dev/null`' dans lequel on peut rediriger les sorties standard pour détruire immédiatement toute information. Faites le test :

```
$ ls trouNoir_2.0_leRetourDeLaVengeance
```

```
$ ls trouNoir_2.0_leRetourDeLaVengeance 2> /dev/null
```

Il est également possible de rediriger l'entrée standard depuis un fichier, on utilisera le symbole '<'.

1.2 Chaînage

Si l'on veut chaîner la sortie standard d'un processus vers l'entrée standard d'un autre processus, on utilise le symbole '|' (*pipe* en anglais). Par exemple, pour compter le nombre de fichiers dans le répertoire courant, on utilisera :

```
$ ls -l | wc -l
```

Ici, la sortie standard de `ls` est redirigée vers l'entrée standard de `wc` (qui sert, entre autres, à compter les lignes et les caractères des fichiers).

2 Qu'est-ce qu'un filtre ?

Un filtre est un programme qui effectue un traitement sur les données qu'il reçoit depuis son entrée standard, et en donne le résultat sur sa sortie standard. Sauf en cas d'erreur (sur `stderr` !), il n'affiche rien d'autre que le résultat de son traitement. On peut ainsi chaîner plusieurs commandes simples et spécialisées pour réaliser des tâches complexes. On forme alors ce que les Anglais nomment un *pipeline*.

Les filtres les plus connus sous Unix sont `cat`, `grep`, `sort`, `uniq`, `head`, `tail`, `tee`, `tr` et `wc`. Référez-vous à leur page du manuel (`man commande`) pour connaître leur fonction.

Il est en général plus sage de prévoir qu'un programme devra pouvoir fonctionner comme un filtre. On pourra alors le combiner avec d'autres programmes très facilement, par exemple dans un script shell.

3 Utilisation du shell

Chacune des opérations suivantes peut s’exécuter en une seule ligne de commande en combinant des filtres simples.

1. Créez un fichier `liste` contenant la liste des dossiers et fichiers présents dans votre répertoire personnel.
2. Ajoutez la date (`date`) à la fin du fichier `liste`.
3. Dans le répertoire `FruitsEtLegumes`, listez tous les documents ayant un nom de fruit dans un fichier `fruits`.
4. Listez à présent l'ensemble des légumes dans un fichier `légumes`. Ayez à l'esprit qu'un légume n'est pas un fruit.
5. Dans le répertoire `Tri`, listez les documents par ordre alphabétique inverse et ne gardez que les 5 premiers résultats.
6. Comptez le nombre d'éléments uniques présents dans le fichier `colors`.

À savoir : le résultat d'une commande peut servir d'argument à une autre commande grâce à l'utilisation de l'accent grave '`' (Alt Gr + 7). Par exemple, pour afficher tous les fichiers dont le nom contient l'année en cours :

```
$ ls | grep `date +%Y`
```

4 Déchiffrage

Le fichier `messageChiffre` est illisible car il a été chiffré à l'aide d'une clé. Le chiffrage est très simple : à chaque symbole a été ajouté la valeur de la clé.

4.1 Récupération de la clé

La clé de déchiffrement a été cachée d'une manière un peu particulière dans le répertoire `Cle`. Pour la trouver, il faut compter le nombre de lignes contenant le terme 'KEY' (sans les guillemets) parmi les 3 derniers fichiers (par ordre alphabétique) dont le nom se termine par la lettre `k`.

4.2 Filtre de déchiffrage

Maintenant que vous possédez la valeur `X` de la clé, vous allez construire un filtre en C pour déchiffrer le message en une simple commande :

```
$ cat messageChiffre | ./dechiffrage X
```

Votre filtre lira un à un les caractères sur l'entrée standard, soustraira la valeur de la clé et écrira les caractères ainsi obtenus sur la sortie standard.

Enfin, plutôt que d'entrer la valeur de la clé à la main, écrivez en une ligne la commande permettant de récupérer la clé et de déchiffrer le message.

Il ne vous aura pas échappé que ce filtre vous permettra également de chiffrer un message avec une clé que vous aurez vous même choisie.

Bon travail !

```
1) ls > liste

2) ls | date "+DATE: %Y-%m-%d" >> liste

3) ls | grep -e .fruit > liste_fruit

4) ls | grep -v .fruit > liste_legume

5) ls | sort -r | head -5 >> liste_tri

6) ls | sort colors -d | uniq > list_colors
   ls | sort colors -d | uniq | wc -l

7) ls | grep KEY  `sort -r | grep k$ | head -3` | wc -l

8) int main ( int argc, char ** argv ) {
        char message[64];
        int i = 0;
        fgets(message, sizeof(message), stdin);
        for (i = 0; i < strlen(message); i++) {
            fprintf(stdout, "%c", message[i] - 4);
        }
        fprintf(stdout, "\n");
        exit( EXIT_SUCCESS );
   }
```