TP 01

Exercice 1 : Preparation de l'environnement du TP

Dans ce TP vous :
— devrez travailler dans un dossier nommé `R1.03/TP01` dans votre répertoire principal ;
— allez visualiser et manipuler des informations représentées en binaire. Ces informations, stockées sur le disque dur de votre machine, sont accessibles dans des fichiers.
Il existe plusieurs commandes utiles pour manipuler le contenu des fichiers comme une simple représentation binaire :
— `hd` est une commande permettant d'afficher les données binaires d'un fichier sous forme de nombres hexadecimaux.
— `od` est une commande permettant d'afficher les données binaires d'un fichier sous différentes formes.
— `hexedit` est une commande permettant de voir **et** de modifier interactivement le contenu d'un fichier sous forme hexadécimale ou ASCII.

Q1. En utilisant une ligne de commande, créez le répertoire de travail et déplacez-y vous.

Exercice 2 : Première visualisation

La commande `hd` permet d'afficher le contenu binaire d'un fichier. Son affichage, pour chaque ligne, est découpé en trois colonnes :
— La première colonne est l'**offset**. Elle est composée d'une valeur écrite en hexadécimale correspondant à combien d'octets ont déjà été lu avant le début de la ligne.
— La deuxième colonne est le **contenu** du fichier écrit en hexadécimal. Chaque nombre hexadécimal est composé de deux chiffres hexadécimaux correspondant à un octet. Il y a au plus 16 octets représentés sur une ligne.
— La troisième colonne, placée entre deux "`|`", correspond à une **transcription** des octets lus suivant la table ASCII (man ascii). Attention : certains caractères spéciaux sont remplacés par des "`.`".

Q1. Copiez le fichier `/home/public/baste/R1.03/TP01/exemple` dans votre dossier de travail.

Q2. En utilisant la commande `cat`, observez son contenu.

Q3. Observez à nouveau son contenu, mais cette fois-ci en utilisant la commande `hd`.

Q4. Dans un autre terminal, ouvrez la table ascii (`man ascii`) et vérifiez la correspondance pour chaque caractère du fichier. Quels caractères posent problème ? Pourquoi ?

Q5. En regardant uniquement la première colonne retournée par `hd`, déterminez la taille, en octet, du fichier.

Exercice 3 : Prise en main des outils

La commande `hd` est en réalité une version simplifiée de la commande `od` qui a elle beaucoup plus d'expressivité. Dans cet exercice vous allez devoir effectuer des conversions. Vous DEVEZ les effectuer à la main. Si vous ne vous sentez pas capable de le faire alors c'est que vous n'avez pas compris l'essence de la conversion entre binaire, octal et hexadécimal, car toutes ces conversions sont extrêmement simples. Dans ce cas, demandez à votre encadrant, ce TP est en particulier là pour cela.

Q1. Lisez la documentation de `od` et en particulier les sections NAME, SYNOPSIS, DESCRIPTION et EXAMPLES.

Q2. Par défaut, c'est-à-dire sans arguments autres que le nom du fichier à étudier :

— sous quelle forme est représenté l'offset ?

— sous quelle forme sont représentés les nombres ?

— combien d'octets représentent chaque nombre ?

— combien `od` affiche-t-il d'octets par ligne ?
En cas de difficulté sur cette question pensez à relire la documentation de `od`. Pour vous aider, on précise que `sizeof(short)` vaut 2

Q3. Comparez le résultat de la sortie standard de `od` par rapport à celle de `hd` appliqué au fichier `exemple`.

— Combien de colonnes sont affichées par `od` ? Lesquelles ?

— Pourquoi les nombres de la deuxième colonne affichés par `od` sont-ils plus grand ? À quoi correspondent-ils ?

— Combien y a-t-il de nombres dans chaque ligne produite par `od` ? Combien d'octets sont représentés par ligne ?

— En effectuant les conversions nécessaires, mettez explicitement en relation les trois premiers nombres de cette deuxième colonne d'`od` (donc les nombres `062503`, `063040` et `061551`) avec les nombres affichés par `hd`.

— Pourquoi l'offset de la deuxième ligne d'`od` est `0000020` alors que celui de la deuxième ligne de `hd` est `00000010` ? Combien d'octets y a-t-il sur la première ligne donnée par `od` et sur la première ligne donnée par `hd` ?

Q4. Déterminez les lignes de commandes permettant :

— de voir le contenu du fichier `exemple` sous forme de nombres **hexadécimaux** représentant **un octet chacun**. Vous devez alors retomber sur une représentation proche de celle donnée par `hd`.

— de voir le contenu du fichier `exemple` sous forme de nombres **décimaux** représentant **un octet chacun**

Q5. Observez le fichier `/etc/debian_version` en utilisant `cat` puis déterminer sa taille en utilisant la commande `stat`. Expliquez d'où vient le caractère supplémentaire en utilisant `od` ou `hd` et en utilisant la table ASCII (`man ascii`).

Q6. Après avoir lu la documentation de `hexedit` décrivez comment :
— quitter `hexedit` ?
— aller directement à la fin du fichier ?
— basculer le curseur de la représentation hexadécimale à la représentation ASCII ?
— sauvegarder les modifications faites ?
— sauvegarder puis quitter `hexedit` en une seule action ?

Exercice 4 : Modifier le contenu d'un fichier binaire

Q1. Copier le fichier `/home/public/baste/R1.03/TP01/Secret.class` dans votre dossier de travail[1].

Ce fichier est un programme écrit en Java dont on ne vous donne que la partie exécutable. Dans ce fichier est stocké un mot secret. Vous pouvez essayer de découvrir ce mot en exécutant le programme avec le mot que vous voulez tester. Par exemple pour tester le mot `toto`, vous devez saisir la ligne de commande suivante :

```
java Secret toto
```

Si le mot que vous avez testé est le bon alors le programme affiche le mot `Bravo`, sinon il affiche le mot `Perdu`.

Q2. Essayez effectivement le mot `toto` puis quelques autres mots de votre choix.

Q3. En analysant le fichier `Secret.class` avec `hexedit` ou `od`, essayez de déterminer quel est ce mot secret.

Q4. Modifiez le fichier `Secret.class` de façon à ce que le mot secret, qui déclenche l'affichage de `Bravo`, soit désormais le mot `public`.

Q5. Vérifiez que votre modification est effective.

Exercice 5 : Comprendre le stockage des octets dans les fichiers et en mémoire

Q1. Copier le fichier `/home/public/baste/R1.03/TP01/donnees.bin` dans votre dossier de travail.

Du fait de leur architecture, les ordinateurs que vous utilisez stockent les données en respectant la convention *little endian*. En mémoire et pour des nombres représentés sur plus d'un octet, les octets de poids faibles sont ainsi stockés en premier.

Par ailleurs la taille des nombres entiers courts (`short int`), définie par le langage C et donc utilisée par les commandes écrites avec ce langage (comme `od` ou `hexedit`), est de 2 octets.

Par défaut quand `od` lit un `short int` d'un fichier vers la mémoire, il lit donc en premier l'octet de poids faible **puis** l'octet de poids fort. Ensuite, quand il affiche un tel nombre et comme le spécifie l'option par défaut `-t oS`, il affiche donc un entier, représenté sous forme octale, codé sur 2 octets et dont l'octet de poids faible représente le premier octet du fichier et l'octet de poids fort le second octet du fichier.

Q2. Notez les 4 premiers nombres (c'est-à-dire une représentation des 8 premiers octets) affichées par `od donnees.bin`.

Q3. Convertissez **correctement** chaque **octet** lu en hexadécimal. (En cas de difficulté, pensez à repasser par le binaire.)

Q4. Trouvez la ligne de commande `od` qui affiche le résultat escompté exact permettant de vérifier votre conversion.

1. Pour mémoire ce doit être `~/R1.03/TP01` qui doit d'ailleurs être votre dossier courant