TP OS n°2

Sommaire

Les objectifs de ce tp sont d'être capable, en utilisant des commandes de base sous GNU/-Linux, de manipuler des fichiers « binaire » et d'en comprendre leur contenu.

Introduction

Environnement de travail

Il vous faut ouvrir une **session** sur votre poste de travail. Vous pouvez utiliser soit le mode console soit l'interface graphique. Dans les deux cas, vous allez travailler **« en ligne de commande »**.

Pour ce TP, il vous faudra préalablement créer un répertoire de travail `tpos2` dans `$HOME/tpos` en tapant la commande suivante :

```
$ mkdir -p $HOME/tpos/tpos2
```

Pour se déplacer dans l'arborescence de travail, il faut taper la commande suivante :

```
$ cd $HOME/tpos/tpos2
```

Rappels : les Fichiers « texte »

On distingue en général deux types de fichiers : **texte** et **binaire**.

Les fichiers texte ont un contenu pouvant être interprété directement comme du texte (une suite de bits représentant un caractère), la plupart du temps en codage **ASCII** (*American Standard Code for Information Interchange*).

On utilise généralement un **éditeur de texte** (vi, **vim**, emacs, kwrite, kate, Notepad, Notepad++, UltraEdit, ...) pour manipuler ce type de fichiers.

Quelques exemples de fichiers textes : code source d'un programme, fichiers de configuration, etc .

Autres termes : fichier texte ou fichier texte brut ou fichier texte simple ou fichier ASCII.

Les Fichiers « binaire »

> "Un fichier binaire est un fichier informatique qui n'est pas assimilable à un fichier texte."

Donc, tout ce qui n'est pas un fichier texte est un fichier binaire (source wikipedia).

*Remarque : Le contenu d'un fichier binaire correspond souvent à un **format** précis lié à l'usage d'un logiciel applicatif spécifique.*

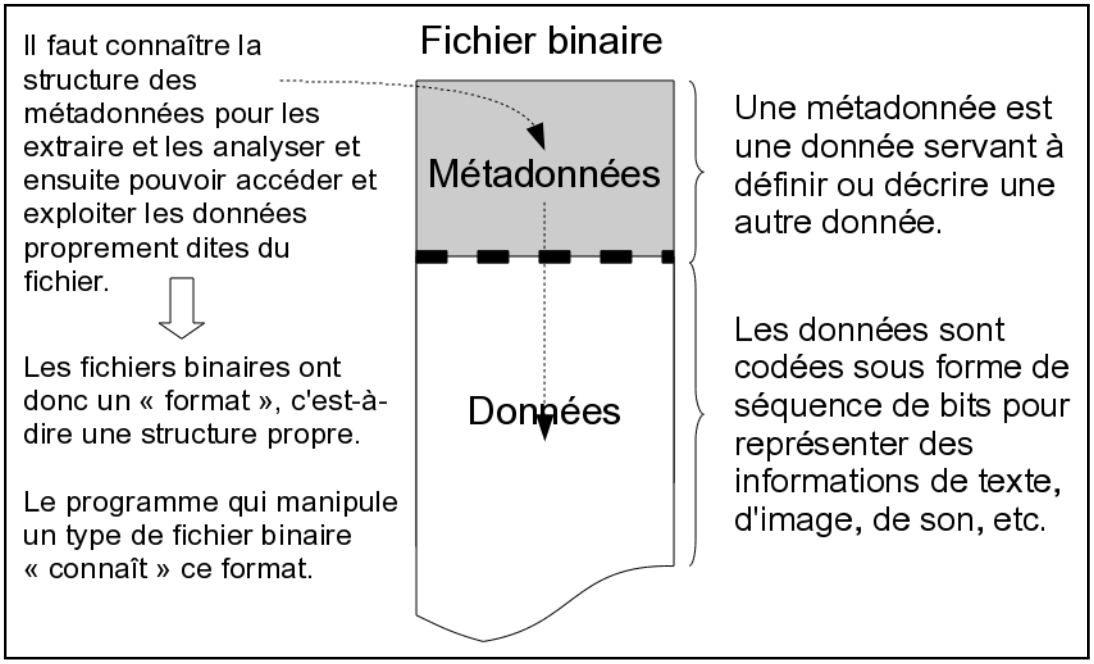
Voici quelques exemples de formats binaires usuels : code machine (exécutable), fichier de base de données, fichiers multimédias : images, sons, vidéos, traitement de texte, etc.

Le format de fichier est la convention selon laquelle les informations ainsi que les métadonnées sont numérisées et organisées dans le fichier.

Remarque : Le format du fichier est propriétaire lorsque la convention n'est connue que de son auteur et n'a jamais été publiée.

Selon la nature et le format du contenu, les fichiers peuvent être qualifiés d'exécutables, de compressés, de textes, d'archives, de documents, d'images, d'audio ou de vidéos.

Remarque : Il existe des centaines, voire des milliers de types de fichiers, qui se différencient par la nature du contenu, le format, le logiciel utilisé pour manipuler le contenu, et l'usage qu'en fait l'ordinateur.



Format "classique" d'un fichier binaire

Exemple : un fichier audio ne contiendra pas seulement la musique (les échantillons de sons = les données) mais pourra également apporter des informations sur celles-ci (telles que l'interprète, le titre, le nom de l'album, etc ...) et ces informations (les métadonnées) sont stockées dans le fichier audio lui-même.

L'**extension** est un suffixe ajouté au nom du fichier, et destiné à renseigner sur le format du fichier et le logiciel à utiliser pour le manipuler.

Remarque : certains logiciels vérifient l'extension d'un fichier mais ce n'est pas systématique. Ils préfèrent vérifier le type de fichier grâce à une signature (un code) inscrite à l'intérieur même du fichier.

Les métadonnées

Voici quelques exemples de métadonnées utilisées en informatique :
- Les systèmes de fichiers disposent de quelques informations de base sur les fichiers, qui sont à ce titre des métadonnées. Les principales sont le nom du fichier, sa taille, la date de création et de dernière modification.
- Dans les systèmes UNIX, les droits d'accès (en lecture, écriture et exécution selon l'utilisateur, le groupe, ou les autres) sont des métadonnées sensibles. Ces droits d'accès sont décrits dans ce que l'on appelle les inodes (contraction d'index-node, noeud d'index).
- Les formats de fichiers tels que PDF, Word, Excel, ou OpenOffice.org utilisent des métadonnées. Elles sont visibles et peuvent être complétées à partir du menu Fichier > Propriétés du document dans l'interface des applications correspondantes. Le format de document OpenDocument (ou ODF), ouvert, contient des métadonnées dans le fichier meta.xml.
- Chaque format d'image numérique implémente une façon spécifique de stocker les métadonnées, mais il existe certaines normes communes à plusieurs types, par exemple : le standard EXIF (JPEG ou TIFF), le standard XMP est intégrable à une douzaine de types de fichier différents (JPEG, TIFF, GIF, PNG, etc.).
- Les métadonnées sont utilisées par le format MP3 dans les tags ID3. On peut en effet y insérer des informations comme le nom de la chanson, de l'interprète, ou encore la date de sortie.
- Les métadonnées sont coeur de l'architecture Web. Le contenu des pages web est structuré à l'aide de balises meta, en langage HTML.

Manipulations

Objectifs

L'objectif de cette partie est de s'initier à la manipulation et à la compréhension des fichiers binaire sous GNU/Linux.

Étape n°0 : L'éditeur de texte vim

La version incluse dans **Linux-Mandriva** est en fait `vim` (*vi improved*) qui comporte quelques différences avec `vi`. `vi/vim` comprend deux modes de fonctionnement : le mode **commande** et le mode **insertion**. Après le lancement de `vi/vim`, c'est le mode commande qui est actif. Pour passer en mode insertion (de texte évidemment) il faut appuyer sur la touche `i` ou `o`. On sait que l'on est en mode insertion par l'affichage de `INSERT` en bas de la fenêtre. Pour revenir au mode commande, il faut appuyer sur la touche `Echap` et l'affichage de `INSERT` en bas de la fenêtre disparaît.

Quelques commandes intéressantes :

```
:q! : sortie sans sauvegarde
:wq : sortie avec sauvegarde
:x  : sortie avec sauvegarde
$   : se déplacer sur le dernier caractère de la ligne
ZZ  : sortie avec sauvegarde
:w  : sauvegarde sans sortie
Ctrl f : afficher la page suivante
Ctrl b : afficher la page précédente
Ctrl d : afficher la demi-page suivante
Ctrl u : afficher la demi-page précédente
e  : se déplacer à la fin du mot
b  : se déplacer au début du mot
w  : se déplacer au début du mot suivant
H  : se déplacer en haut de l'écran
L  : se déplacer en bas de l'écran
M  : se déplacer au milieu de l'écran
z. : décaler l'affichage avec la ligne courante au centre
z (return) : décaler l'affichage avec la ligne courante en haut
z- : décaler l'affichage pour que la ligne courante
:num_ligne : se déplacer à la ligne num_ligne
G (ou :$)  : aller à la fin du fichier
u   : annulation de la dernière modification
dd  : suppression de la ligne courante
2dd : suppression des deux lignes suivantes
D   : suppression de la fin de la ligne à partir du curseur
:3,7 d : suppression des lignes 3 à 7
:3,7 t 10 : copie des lignes 3 à 7 après la ligne 10
:3,7 m 10 : transfert des lignes 3 à 7 après la ligne 10
yy  : mémorisation de la ligne courante (copier)
3yy : mémorisation des 3 lignes suivantes (copier)
p   : copie ce qui a été mémorisé après le curseur
P   : copie ce qui a été mémorisé avant le curseur
:set nu : affichage des numéros de ligne
/mot : recherche le mot mot (on se déplace avec n ou N ou *)
```


Étape n°1 : Les fichiers exécutables

Un fichier exécutable est un fichier contenant un programme (des instructions en langage machine) et identifié par le système d'exploitation en tant que tel. Le chargement d'un tel fichier entraîne la création d'un processus dans le système, et l'exécution du programme.

Nous utiliserons `gcc` sous Linux pour produire un fichier binaire exécutable. Par défaut, `gcc` fabrique un fichier nommé `a.out`. On peut aussi préciser le nom de l'exécutable avec l'option `-o`. Il n'y a pas d'extension spécifique pour les exécutables sous UNIX/Linux. Par défaut, `gcc` utilise une édition de liens dynamique.

Tout d'abord, il faut créer un fichier source avec l'éditeur `vim` :

```
$ vim hello.c
```

Il faut taper `i` pour passer en mode insertion, puis éditer le fichier avec le contenu ci-dessous. Pour finir, on enregistre et on quitte (touche `Echap` pour revenir en mode commande puis `:x` ou `:wq`).

```c
// Affiche à l'écran le message Hello world

#include <stdio.h>

#define TEXTE "Hello world"

int main ()
{
     printf("%s\n", TEXTE);
     return 0;
}
```

Le fichier source hello.c

Remarque : hello world (« bonjour le monde ») sont les mots traditionnellement écrits par un programme informatique simple dont le but est de faire la démonstration rapide d'un langage de programmation (par exemple à but pédagogique) ou le test d'un compilateur. La tradition d'utiliser « hello world » comme message de test a été initiée par le livre The C Programming Language de Brian Kernighan et Dennis Ritchie. Le premier exemple de ce livre affiche "hello, world". Au XXIe siècle, les programmes affichent plus souvent "Hello world !".

1) Est-ce q'un fichier source est un fichier binaire ? Non !

```
$ file hello.c
hello.c: ASCII C program text
```

2) Est-ce q'un fichier en-tête (*header*) est un fichier binaire ? Non !

```
$ file /usr/include/stdio.h
/usr/include/stdio.h: ASCII C program text
```

Pour fabriquer un fichier binaire (avec une édition de liens dynamique par défaut), il faut utiliser `gcc` :

```
$ gcc hello.c

$ ls -hl a.out
-rwxr-xr-x 1 tv tv 5,7K 2010-07-18 11:52 a.out
```


3) Est-ce qu'un fichier fabriqué par gcc est un fichier binaire ? Oui !

```
// Exécuter le fichier binaire
$ ./a.out
Hello world

$ file a.out
a.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses
    shared libs), for GNU/Linux 2.6.9, not stripped
```

Remarque : ELF (Executable and Linking Format) est un format de fichier informatique binaire utilisé pour l'enregistrement de code compilé (objets, exécutables, bibliothèques de fonctions). La commande ***man elf*** *donnera de plus amples informations. Aujourd'hui, ce format est utilisé dans la plupart des systèmes d'exploitation Unix (GNU/Linux, Solaris, Irix, System V, BSD), à l'exception de Mac OS X qui utilise le format Mach-O. Windows utilise le format PE (Portable Executable). Par son format, un fichier exécutable sera déjà exclusivement utilisable sur un OS donné.*

4) Un fichier binaire peut-il contenir aussi du texte ? Oui !

Il est plus sage d'utiliser la commande `strings` (mais vous pouvez essayer avec la commande cat ou vim !) :

```
$ strings a.out
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
puts
__libc_start_main
GLIBC_2.0
PTRh
[^_]
Hello world
```

On retrouve bien notre chaîne de caractères "Hello world". On peut aussi modifier le contenu d'un fichier binaire, par exemple le 'w' en 'W' :

```
$ hexedit a.out

$ ./a.out
Hello World
```

5) Que contient un fichier binaire exécutable ? Une structure (un format) et des instructions machines

Les fichiers exécutable possèdent un format relativement complexe qui dépasse l'objet de ce TP.

```
// Visualiser la structure d'un fichier exécutable
$ objdump -s a.out

// Visualiser les instructions machines d'un fichier exécutable
$ objdump -d a.out
```

On trouve par exemple une métadonnée dans la section `.comment` : `GCC : (GNU) 4.4.3`. On peut vérifier en demandant la version de gcc :

```
$ gcc -v
```

Étape n°2 : Les fichiers "traitement de texte"

Rappel : Le contenu d'un fichier binaire correspond souvent à un format précis lié à l'usage d'un logiciel applicatif spécifique.

On va créer un document texte avec **OpenOffice** :

```
$ ooffice -writer
// ou :
$ ooffice3.2 -writer
```

On saisit le texte "Hello wordl" puis on sauvegarde le fichier (`Ctrl-s`) sous le nom `hello.odt` puis on quitte (`Alt-F4`).

6) Les traitements de texte produisent-ils des fichiers texte ? Non !

Vérifions tout de même :

```
$ cat -v hello.odt
// Aie ! apparemment non ! Cela s'apparente plus à des fichiers "binaires"

$ cat -v hello.odt | grep wordl
$ cat -v hello.odt | grep -i hello
// Rien ! Alors : quel est le type de ce fichier ?

$ file hello.odt
hello.odt: OpenDocument Text
```

OpenDocument est un format ouvert de données pour les applications bureautiques : traitements de texte, tableurs, présentations, diagrammes, dessins et base de données bureautique. OpenDocument est la désignation d'usage d'une norme dont l'appellation officielle est OASIS Open Document Format for Office Applications, également abrégée par le sigle ODF. La norme ISO 26300 publiée en 2006 fixe un format de document ouvert pour les applications de bureau (OpenDocument) v1.0.

Remarque : Office Open XML est une norme ISO/IEC (IS 29500) créée par Microsoft, destinée à répondre à la demande d'interopérabilité dans les environnements de bureautique et à concurrencer la solution d'interopérabilité OpenDocument. Ce format (dont les suffixes sont .docx, .xlsx, et .pptx) est utilisé par Microsoft Office 2007 ainsi que par Microsoft Office 2008 pour Mac, en remplacement des précédents formats Microsoft (reconnus à leurs suffixes tels que : .doc, .xls, .ppt)

Les deux formats sont structurés en **XML** et **Zip** :
- XML (*Extensible Markup Language*) ou langage extensible de balisage est un langage informatique de balisage générique. XML sert essentiellement à stocker/transférer des données de type texte Unicode structurées en champs arborescents. L'objectif initial est de faciliter l'échange automatisé de contenus entre systèmes d'informations hétérogènes (interopérabilité). Ce langage est qualifié d'extensible, car il permet à l'utilisateur de définir ses propres balises.
- Le ZIP est un format de fichier permettant l'archivage (utilisation d'un seul fichier pour stocker plusieurs fichiers) et la compression de données (diminution de l'espace occupé sur le support numérique) sans perte de qualité. Le format a été inventé par Phil Katz pour le logiciel PKZIP.

Remarque : Le format JAR (Java Archive), l'ODT (OpenDocument) et l'Open XML (OOXML) sont basés sur le format ZIP.

Les spécifications du format zip se trouvent à l'adresse suivante :
www.pkware.com/documents/casestudies/APPNOTE.TXT.

La **signature d'en-tête** de ce type de fichier est la séquence `0x04034b50` (en *little-endian*, soit `0x504b0304` en *big-endian*).

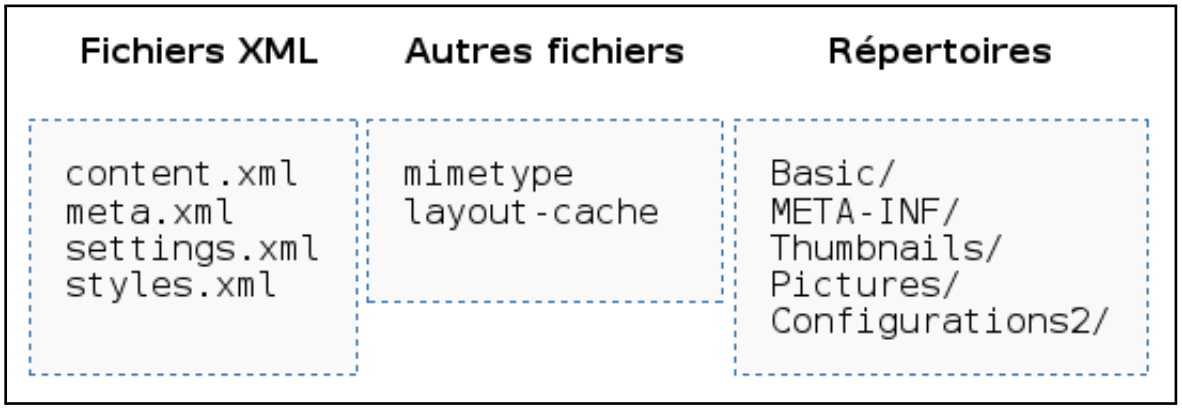
Remarque : En informatique, certaines données telles que les nombres entiers peuvent être représentées sur plusieurs octets. L'ordre dans lequel ces octets sont organisés en mémoire ou dans une communication est appelé endianness (mot anglais traduit par « boutisme »). De la même manière que certains langages humains s'écrivent de gauche à droite, et d'autres s'écrivent de droite à gauche, il existe une alternative majeure à l'organisation des octets représentant une donnée : l'orientation big-endian (dans l'ordre, octet de poids le plus fort vers l'octet de poids le plus faible) et l'orientation little-endian (dans l'ordre, octet de poids le plus faible vers l'octet de poids le plus fort).

```
$ hexdump -C hello.odt | head -1
00000000 50 4b 03 04 14 00 00 08 00 00 06 58 f3 40 5e c6 |PK.........X.@^.|
```

On reconnaît le début caractéristique (`"PK"`) des fichiers **zip** ! On peut donc le décompresser :

```
$ unzip hello.odt
Archive: hello.odt
 extracting: mimetype
   creating: Configurations2/statusbar/
  inflating: Configurations2/accelerator/current.xml
   creating: Configurations2/floater/
   creating: Configurations2/popupmenu/
   creating: Configurations2/progressbar/
   creating: Configurations2/menubar/
   creating: Configurations2/toolbar/
   creating: Configurations2/images/Bitmaps/
  inflating: content.xml
  inflating: styles.xml
 extracting: meta.xml
  inflating: Thumbnails/thumbnail.png
  inflating: settings.xml
  inflating: META-INF/manifest.xml
```


Le format **OpenDocument** soutient une forte séparation entre contenu, mise en page et métadonnées (fr.wikipedia.org/wiki/OpenDocument).



Le fichiers contenus dans une archive compressée au format OpenDocument

7) Où se trouve notre texte ? dans content.xml !

`content.xml` est le fichier le plus important, il contient le contenu réel du document (excepté le contenu binaire telles les images qui sont stockées dans des fichiers séparés) :

```
$ cat content.xml | grep -i "Hello wordl"
// ou
$ vim content.xml
```

Remarque : le fichier meta.xml contient les métadonnées du document.

Questions de révision

L'idée de base des questions de révision est de vous donner une chance de voir si vous avez identifié et compris les points clés de ce TP.

Question 1. Que contient un fichier binaire ?

Question 2. Citer un exemple de métadonnée pour un fichier image ?

Question 3. Est-ce qu'un fichier binaire peut contenir du texte ASCII ?

Question 4. À quoi sert l'extension d'un fichier ?

Question 5. Est-ce que tous les fichiers image contiennent-ils les mêmes données ?

Question 6. Peut-on renommer l'extension d'un fichier PNG en GIF sans danger ?

Question 7. Lorsqu'on compresse un fichier, est-ce que l'on modifie son format ?

Question 8. Un fichier informatique exécutable est-il utilisable (interopérable) sur n'importe quel système d'exploitation ?

Question 9. Que se passe-t-il lorsqu'on édite (en texte) un fichier binaire ?

Question 10. La durée en secondes contenue dans un fichier audio, est-ce une donnée ou une métadonnée ?

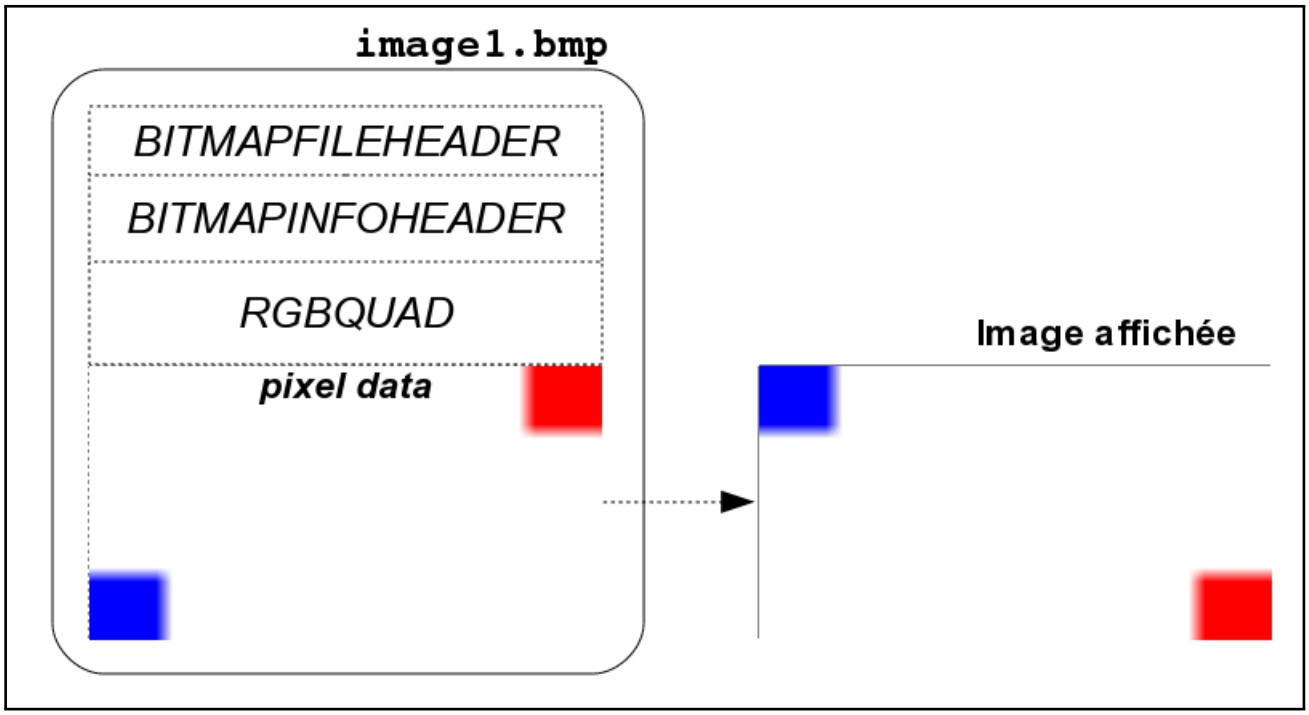
Travail demandé

Exercice 1 : le format image BMP

L'objectif de cet exercice est de manipuler et comprendre la structure d'un fichier image au format BMP.

Bitmap (BMP) est un format d'image matricielle ouvert développé par Microsoft et IBM. C'est un des formats d'images les plus simples à développer et à utiliser pour programmer. Il est lisible par quasiment tous les visualiseurs et éditeurs d'images.

Un fichier BMP se découpe en 4 zones :
– l'en-tête du fichier ;
– l'en-tête de l'image ;
– la palette de couleurs ;
– les données relatives à l'image.



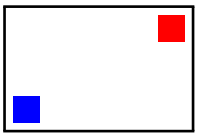
Structure d'un fichier BMP

Remarque : La palette
Les logiciels de lecture utilisent trois octets pour coder la couleur (système rouge vert bleu, RVB) :
– *en BMP 24 bits (1 octet bleu, 1 octet vert, 1 octet rouge par pixel), la palette n'est pas nécessaire (on peut représenter toutes les couleurs). Dans ce cas, la palette n'existe pas dans le fichier BMP ;*
– *en BMP 8 bits, on ne peut représenter que 256 couleurs (2^8), il faut définir une correspondance entre la couleur du pixel et les 3 composantes. Il a donc été ajouté une table juste après l'en-tête (octet 56), qui donne pour chaque valeur (de 0 à 255) les trois composantes RVB qui y correspondent).*

La profondeur de couleurs est le nombre de bits associés à chaque pixel pour en coder la couleur.

L'étude porte sur le fichier suivant :

```
$ ls -l image1.bmp
-rw-rw-r-- 1 tv tv 1974 2011-10-15 15:08 image1.bmp
```



Le fichier BMP à étudier

Question 11. Pour une image BMP 24 bits, combien de couleurs différentes pourra-t-on coder pour un pixel ?

Remarque : Le pixel (souvent abrégé px) est l'unité de base permettant de mesurer la définition d'une image numérique matricielle. Son nom provient de « picture element » qui signifie « élément d'image ».

Question 12. Sachant qu'un pixel, pour une image BMP 24 bits, sera codé par dans l'ordre : 1 octet pour le bleu, 1 octet pour le vert et 1 octet pour le rouge, compléter le tableau soit en indiquant la valeur en hexadécimale sur 3 octets soit la couleur correspondante.

Valeur (en hexadécimale)	Couleur d'un pixel
000000	Noire
00FF00	
	Bleue
	Rouge
	Blanche
00FFFF	

Il vous faut maintenant afficher le contenu en hexadécimal du fichier `image1.bmp` en tapant la commande suivante :

```
// Un extrait de l'affichage obtenu :
$ hexdump -C image1.bmp
00000000 42 4d b6 07 00 00 00 00 00 00 36 00 00 00 28 00 |BM........6...(.|
00000010 00 00 20 00 00 00 14 00 00 00 01 00 18 00 00 00 |.. .............|
00000020 00 00 80 07 00 00 13 0b 00 00 13 0b 00 00 00 00 |................|
00000030 00 00 00 00 00 00 ff 00 00 ff 00 00 ff 00 00 ff |................|
00000040 00 00 ff 00 00 ff ff ff ff ff ff ff ff ff ff ff |................|
00000050 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |................|
*
```

Remarque : la présence d'étoiles (*) indique qu'il y a plusieurs lignes identiques à la précédente.

Offset#	Taille	Valeur
0x0000	2 octets	le nombre magique correspondant à l'utilisation du fichier BMP • **BM** - Windows 3.1x, 95, NT, ... etc • **BA** - OS/2 Bitmap Array • **CI** - OS/2 Icône Couleur (Color Icon) • **CP** - OS/2 Pointer Couleur (Color Pointer) • **IC** - OS/2 Icône (Icon) • **PT** - OS/2 Pointer (Pointer)
0x0002	4 octets	la taille du fichier BMP en octets
0x0006	2 octets	réservé pour l'identifiant de l'application qui a créé le fichier
0x0008	2 octets	réservé pour l'identifiant de l'application qui a créé le fichier
0x000A	4 octets	l'offset (l'adresse de départ) du contenu du BMP

Le bloc d'entête BITMAPFILEHEADER

Attention : la convention utilisée dans les fichiers BMP pour l'organisation des octets représentant une donnée est l'orientation ***little-endian****.*

Question 13. Décoder le premier bloc d'entête (`BITMAPFILEHEADER`) en complétant le tableau ci-dessous.

Champ	Valeur en hexadécimale	Valeur décodée
Type (`bfType`) en ASCII		
Taille (`bfSize`) en octets		
Offset (`bfOffBits`) en octets		

Question 14. Quel est la longueur en octets de ce premier bloc d'entête (BITMAPFILEHEADER) ?

start	size	name	stdvalue	purpose
15	4	biSize	40	specifies the size of the BITMAPINFOHEADER structure, in bytes.
19	4	biWidth	100	specifies the width of the image, in pixels.
23	4	biHeight	100	specifies the height of the image, in pixels.
27	2	biPlanes	1	specifies the number of planes of the target device, must be set to zero.
29	2	biBitCount	8	specifies the number of bits per pixel.
31	4	biCompression	0	Specifies the type of compression, usually set to zero (no compression).
35	4	biSizeImage	0	specifies the size of the image data, in bytes. If there is no compression, it is valid to set this member to zero.
39	4	biXPelsPerMeter	0	specifies the the horizontal pixels per meter on the designated targer device, usually set to zero.
43	4	biYPelsPerMeter	0	specifies the the vertical pixels per meter on the designated targer device, usually set to zero.
47	4	biClrUsed	0	specifies the number of colors used in the bitmap, if set to zero the number of colors is calculated using the biBitCount member.
51	4	biClrImportant	0	specifies the number of color that are 'important' for the bitmap, if set to zero, all colors are important.

Le bloc d'entête BITMAPINFOHEADER

Question 15. Décoder le deuxième bloc d'entête (BITMAPINFOHEADER) en complétant le tableau ci-dessous.

Champ	Valeur en hexadécimale	Valeur décodée
Taille de cette entête (`biSize`) en octets		
Largeur (`biWidth`) en pixels		
Height (`biHeight`) en pixels		
Codage des couleurs (`biBitCount`) en bits		
Taille des données image (`biSizeImage`) en octets		
Nb pixels par mètre en X (`biXPelsPerMeter`)		
Nb pixels par mètre en Y (`biYPelsPerMeter`)		

Question 16. Ce fichier `image1.bmp` possède-t-il une palette de couleurs ? Pourquoi ?

Question 17. Calculer la taille en octets du fichier `image1.bmp` en utilisant la formule suivante : taille entête fichier + taille entête image + taille palette + taille données image.

Question 18. Calculer la taille en octets des données image (pixels) du fichier `image1.bmp` en utilisant la formule suivante : (Codage des couleurs en bits) x (Hauteur en pixels) x (Largeur en pixels).

Question 19. Calculer la proportion des métadonnées pour ce fichier `image1.bmp`.

Question 20. Calculer la proportion des données image (pixels) pour ce fichier `image1.bmp`.

Question 21. En décodant les données image contenues dans l'image image1.bmp, compléter le tableau ci-dessous.

Champ	Valeur en hexadécimale	Valeur décodée
Premier pixel des données image		
Dernier pixel des données image		

Question 22. Bonus : Le BMP est un format d'image matricielle. Connaissez-vous un type de format d'image autre que matricielle ? Pouvez-vous citer un format de fichier qui utiliserait cette autre technique ?

Exercice 2 : le format audio MP3

L'objectif de cet exercice est de manipuler et comprendre la structure d'un fichier audio au format MP3.

Un format de fichier audio est un format de données utilisé en informatique pour stocker des sons, de la musique ou des voix sous forme numérique. De nombreux standards existent ; certains s'appliquent à la production, au stockage et à la diffusion, d'autres (ceux qui utilisent des algorithmes de compression de données ou de débit), sont destinés, en principe, uniquement à la diffusion.

La compression audio avec perte (*lossy*) se base sur des algorithmes spécialisés pour déterminer quelles transformations simplifient la représentation du son tout en étant perçue quasiment de la même manière par l'oreille humaine. Elle diminue la taille du fichier en éliminant les nuances perçues comme les moins utiles et l'élimination est définitive.

Actuellement, le format le plus utilisé est de loin le mp3. MP3 (MPEG-1 Layer III) est l'abréviation de MPEG-1/2 Audio Layer 3. La couche (Layer) III est la couche la plus complexe. Ce format propose une qualité sonore très correcte pour un débit de 128 kbits/s. Il permet typiquement un gain d'un facteur 10 de taille du fichier. C'est ce format qui a été massivement utilisé pour transférer les musiques via internet

dès la fin des années 1990. Cet algorithme de compression avec perte prend naissance en 1987. L'ISO en fera un standard dans les années 92-93.

Remarque : Des royalties importantes sont à payer pour exploiter la licence MP3. Il convient de noter qu'aucune royalty n'est prélevée par ces ayants-droits sur un fichier au format MP3 contrairement à une pratique courante dans le monde de la propriété intellectuelle.

Un « taux d'échantillonnage fixe » (*constant bitrate* ou CBR) signifie en fait que cette piste vidéo et/ou audio utilise le même montant d'espace disque pour chaque seconde, peu importe sa position dans le temps.

Un « taux d'échantillonnage variable » (*variable bitrate* ou VBR) fera varier les données requises pour encoder chaque seconde d'un fichier multimédia tel qu'une vidéo ou une piste audio, et ce, basé sur la complexité de ces derniers. Le but est évidemment de réaliser le meilleur compromis possible entre la qualité finale du fichier multimédia et sa taille, en opposition au CBR qui garde constamment le même taux d'échantillonnage. Par exemple, les sons complexes sont codés à un débit élevé et les sons simples à un débit plus bas. La taille finale ne peut pas du tout être prédite contrairement aux CBR.

Remarque : Le CBR est bien adapté au streaming sur réseau. Le VBR quant à lui est bien adapté aux lecteurs MP3, lecteurs de DVD et émissions satellite.

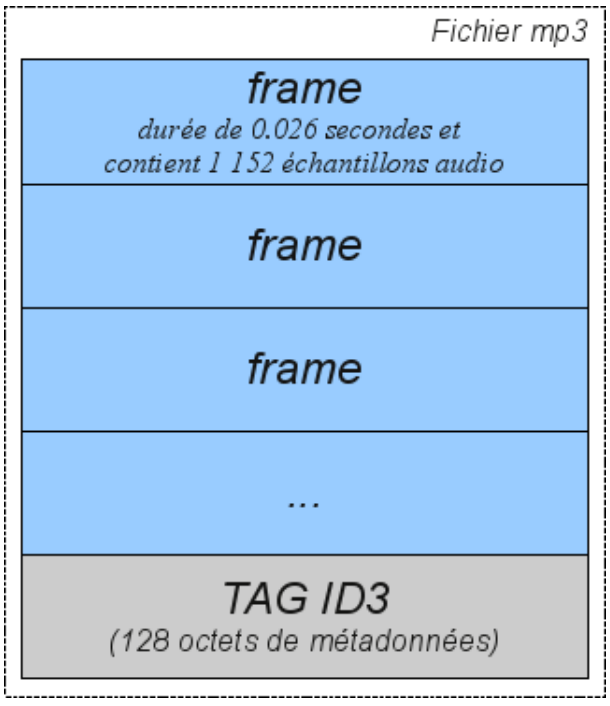
Calcul de la taille d'un fichier audio pour un format à débit constant (CBR) : taille (Ko) = temps (s) x débit (Kbit/s) / 8

Exemple pour un fichier MP3 encodé à 192 Kbit/s d'une durée de 3 minutes : taille = 3 x 60 x 192 / 8 = 4 320 Ko

Sur les CD vierges il est indiqué « 80 minutes » pour 737 Mo (702 Mio) soit 1,24 Mbit/s.

Remarque : les métadonnées augmentent la taille des fichiers audio, même si leur taille est négligeable.

Un fichier mp3 est constitué d'un ensemble de *frames*. Chaque *frame* possède sa propre en-tête (*header*) et contient 1 152 échantillons audio. Elle a une durée de 0,026 secondes. Les 128 derniers octets du fichiers sont utilisés pour les métadonnées ID3 (*tags*).



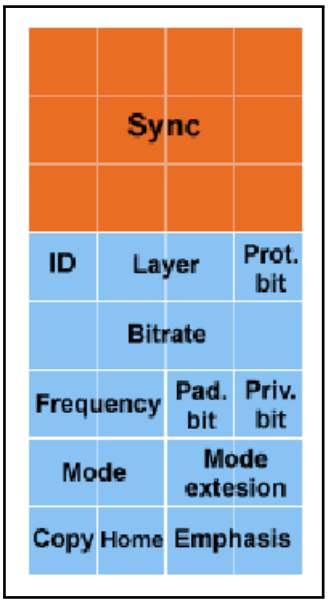
Structure d'un fichier MP3

Une *frame* est décomposée de la manière suivante :

Header	CRC	Side Information	Main Data	Ancillary Data

Structure d'une frame d'un fichier MP3

Chaque *frame* possède sa propre en-tête (*header*) qui commence par une séquence de 11 bits à 1 (synchronisation).



En-tête d'une frame d'un fichier MP3

Remarque : On pourra donc découper un fichier mp3 (ou l'endommager) tout en préservant normalement le contenu audio restant.

Le contenu détaillé d'un en-tête est disponible :
- www.mpgedit.org/mpgedit/mpeg_format/mpeghdr.htm
- www.mpgedit.org/mpgedit/mpeg_format/MP3Format.html
- www.mp3-tech.org/programmer/frame_header.html
- ou sur le serveur : `mpeghdr.htm`

Le MP3 apporte une fonctionnalité rarement présente sur les formats audio qui l'ont précédé : les métadonnées (des données sur les données). En clair, le fichier MP3 ne contient pas seulement la musique mais peut également apporter des informations sur celles-ci (telles que l'interprète, le titre, le nom de l'album, etc ...). Ces informations sont stockées sous forme de 'balises' (*tag*) dont il existe plusieurs versions. Les balises MP3 sont enregistrées au format ID3. Les caractères alphanumériques sont codés en code ASCII.

Elle consiste en un espace de 128 octets placés à la fin du fichier. Les 3 premiers octets commencent par la chaîne « TAG », cela permet de trouver le début des informations. Le reste des octets est partagé entre les différents champs d'informations. Les chaînes de caractères doivent être codées en ISO 8859-1, seuls les caractères de l'alphabet latin peuvent donc être utilisés.

Offset (en partant du début de la structure)	Taille (en octets)	Description
0	3	Identifiant "TAG"
3	30	Titre de la chanson
33	30	Nom de l'interprète
63	30	Nom de l'album
93	4	Année de parution
97	30	Commentaire sur la chanson
127	1	Genre musical

Les différents genre musicaux sont définis par une valeur numérique de 0 à 79, par exemple :

0	Blues
1	Classic rock
17	Rock
78	Rock & Roll
79	Hard Rock

ID3v1.1 est une extension de la version originale. Elle consiste en l'ajout d'un champ pour le numéro de la piste. Les octets utilisés par ce champ ont été pris sur le champ réservé aux commentaires. De cette manière, la longueur de 128 octets pour le tag ID3 est conservée et reste compatible avec les anciens lecteurs audio.

Offset (en partant du début de la structure)	Taille (en octets)	Description
0	3	Identifiant "TAG"
3	30	Titre de la chanson
33	30	Nom de l'interprète
63	30	Nom de l'album
93	4	Année de parution
97	28	Commentaire sur la chanson
125	1	Caractère null servant de séparateur
126	1	Numéro de la piste
127	1	Genre musical

ID3v2 a ajouté un certain nombre de champs d'informations pour y intégrer les paroles et même des images. Contrairement à la version 1, les informations sont placées au début du fichier et la taille des tags ID3 est variable. ID3v2 supporte les caractères Unicode.

Les étiquettes ID3v2 permettent donc de stocker quasiment n'importe quel type d'information et notamment : les paroles de la chanson, la pochette de l'album, l'auteur, le compositeur, le chef d'orchestre, etc.

ID3v2.4 est la dernière version du standard (1er novembre 2000).

Le site officiel de ID3 (*en*) : www.id3.org

Question 23. Afficher (avec `hexdump`) et décoder seulement les *tags* ID3 du fichier `music.mp3` fourni. Donner la commande exacte en vous adiant du manuel de `hexdump` et compléter la ligne ci-dessous.

```
$ hexdump -C -s _______ -n ___ music.mp3
```

Question 24. Donner (et décoder) le contenu des champs « numéro de piste » et « genre musical » (attention ces deux champs ne sont pas codés en ASCII mais numériquement).

Question 25. Vérifier votre décodage avec la commande `mp3info` :

```
$ mp3info music.mp3
```

La commande `mp3info` permet aussi d'afficher des informations techniques :

```
$ mp3info -x music.mp3
File: music.mp3
Media Type: MPEG 1.0 Layer III
Audio:      192 KB/s, 44 100 Hz (joint stereo)
Emphasis:   none
CRC:        No
Copyright:  No
Original:   Yes
Padding:    No
Length:     5:34

// Obtenir le nombre de frames :
$ mp3info -p "%u\n" music.mp3
12789
```

On vous donne la formule pour calculer la taille d'une frame : FrameLengthInBytes = 144 * BitRate / SampleRate + Padding

Et la formule pour calculer la durée d'un fichier audio mp3 : Durée (en s) = nombre de frames x taille d'une frame x 8 / bitrate (+0.5 pour arrondir)

Question 26. Calculer la taille en octets d'une frame.

Question 27. Calculer la taille en octets des métadonnées (en-têtes des frames + ID3).

Question 28. Vérifier vos calculs et estimer en pourcentage la proportion des métadonnées pour ce fichier mp3.

Question 29. Calculer la durée en secondes de ce fichier mp3

Question 30. Convertir en binaire les quatre octets de l'en-tête de la première frame :

```
$ hexdump -C -n 4 music.mp3
00000000 ff fb b0 44                                       |...D|
```

Hexadécimal	Binaire
0xFF	
0xFB	
0xB0	
0x44	

Question 31. Décoder maintenant l'en-tête de la première frame :

Champs	Valeur	Décodage (signification)
Frame synchronisation		
MPEG Audio version ID		
Layer description		
Protection bit (crc)		
Bitrate index		
Sampling rate frequency index		
Padding bit		
Private bit (extension)		
`Channel mode`		
`Mode extension`		
`Copyright`		
`Original`		
`Emphasis`		

Question 32. Vérifier avec les commandes `mp3_check` ou `mpg123` ou avec le programme fourni `decodeheadermp3` (disponible sur le serveur).

```
$ mp3_check music.mp3
$ mpg123 -t -v music.mp3
$ ./decodeheadermp3 music.mp3
```

On va maintenant "découper" un fichier mp3 en enlevant 2000 frames :

```
$ START=$(( 626*2000 ))
$ dd if=./music.mp3 of=./music2.mp3 skip=$START ibs=1
```

Remarque : chaque frame d'un mp3 a une durée de 0.026 secondes.

Question 33. Calculer la nouvelle durée du fichier mp3. Vérifier votre calcul.

Question 34. Le nouveau fichier audio mp3 est-il audible ?

```
$ mpg123 music2.mp3
```

Exercice 3 : le format livre numérique EPUB

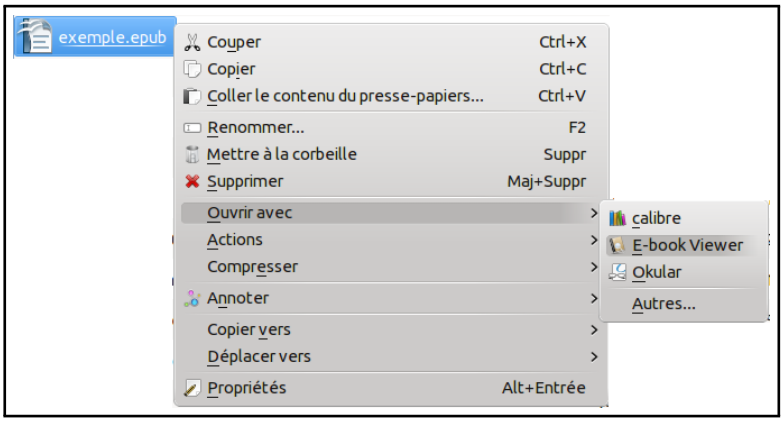
L'objectif de cet exercice est de manipuler et comprendre la structure d'un fichier livre numérique au format EPUB.

EPUB (« electronic publication » ou « publication électronique ») est un format ouvert standardisé pour les livres numériques. Proposé par l'*International Digital Publishing Forum* (IDPF), ces fichiers ont l'extension `.epub`.

L'étude portera sur le fichier exemple.epub fourni :

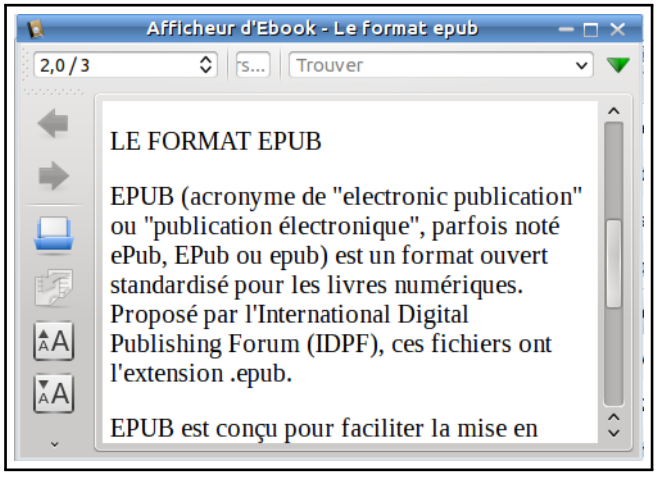
```
$ ls -l exemple.epub
-rw-rw-r-- 1 tv tv 65027 2012-07-20 10:06 exemple.epub
```

L'application multiplateforme (Windows, Linux et Mac) et libre, **Calibre**, permet la lecture et la gestion de livres numériques (ebooks). Elle offre une visionneuse :



Ouvrir un fichier EPUB avec Calibre ou directement avec la visionneuse

La première page du livre numérique au format EPUB fourni en exemple



La deuxième page du livre numérique au format EPUB fourni en exemple

Question 35. Tester le livre numérique `exemple.epub` en utilisant la visionneuse de **Calibre**.

```
$ ebook-viewer /home/tv/Téléchargements/exemple.epub
```

Question 36. Réaliser et rédiger une analyse technique de ce format de fichier.

Bilan

Ce qu'il faut retenir :

Un fichier binaire est donc un fichier informatique contenant des données sous forme d'octets qui n'ont de sens que pour le logiciel qui les utilise (et non pour les utilisateurs finaux).

Si par stricte définition tout fichier est binaire, l'usage veut que l'on qualifie un fichier de binaire pour indiquer qu'il ne s'agit pas d'un fichier texte.

Ces fichiers peuvent quand même être ouverts par des éditeurs de texte (déconseillé aux utilisateurs non expérimentés !), mais les données y seront mal représentées et surtout incompréhensibles.

La plupart des fichiers existants sont des fichiers binaires, il s'agit notamment des fichiers contenant du son, de la vidéo, des images, du langage machine, etc.

Pour décrire l'organisation des données dans un fichier, on parle de format de fichier.

Les fichiers informatiques (binaire et texte) contiennent le plus souvent des métadonnées (une donnée servant à définir ou décrire une autre donnée).

