Etudiants : Guilhem CALONNE et Nathan LE MOAN

Réponses aux questions :

1. Si on déclare une fasse fréquence d'échantillonnage, la vitesse de lecture de la musique change. Si la fréquence déclarée est plus élevée que la fréquence réelle, la musique sera lue plus rapidement. Dans le cas contraire, la musique sera lue plus lentement.
   Pour expliquer ce phénomène, il faut comprendre ce qu'est la fréquence d'échantillonnage : c'est le nombre d'échantillons lu par seconde. Ainsi, si on lit moins d'échantillons de la musique chaque seconde, la musique sera lue plus lentement (et inversement si on lit un plus grand nombre d'échantillons par seconde).

2. Si on déclare le fichier en tant que fichier mono alors qu'il est en réalité en stéréo, la lecture sera altérée. Dans un fichier stéréo, deux échantillons successifs correspondent chacun à un canal et doivent être lus en même temps, chacun dans leur canal attitré (par exemple, le fichier contient les échantillons 1A et 1B qui doivent être lus en même temps, l'un dans le canal A et l'autre dans le canal B). Si le fichier est déclaré en tant que fichier mono, alors les échantillons 1A et 1B ne seront plus lus simultanément mais l'un après l'autre, ce qui change complètement la musique jouée.

3. Si on déclare la mauvaise taille d'échantillons, le son produit est très fortement altéré pour devenir un grésillement et non plus une musique. Pour lire la musique, l'onde sonore doit être « mathématisée ». Informatiquement, cette onde sonore est une suite de valeurs qui sont ensuite lues et interprétées pour retranscrire la musique originale. Si on change la taille des échantillons, on change donc la valeur lue et donc le son produit ne sera plus le même. Par exemple : le chiffre 7 s'écrit 0111 en binaire et doit donc être lu sur 4 bits. Si on décide de lire cette même valeur sur 2 bits seulement, l'ordinateur lira 1 puis 3. Les valeurs étant totalement différentes, le son produit le sera également.

Compte-Rendu :

Objectif : Le but de cette première partie du TP était de produire un lecteur audio qui serait capable de lire un fichier .wav donné.

Structure du projet :

Le projet est structuré en plusieurs dossiers contenant chacun un type de fichiers précis :

- bin : il contient les exécutables produits lors de la compilation
- src : il contient les fichiers sources en .C
- include : il contient les fichiers d'en-tête en .h
- obj : il contient les fichiers objets en .o
- data : il contient les fichiers .wav que l'on souhaite pouvoir lire avec notre programme

Les autres fichiers comme le Makefile et la documentation se trouvent à la racine du projet.

Production : Nous avons produit un fichier lecteur.c qui contient le code nécessaire à la lecture d'un fichier .wav dont le nom est passé en ligne de commandes.

Pour ce faire, il y a plusieurs étapes. La première est de récupérer le descripteur du fichier qu'on souhaite lire. Ce descripteur servira à indiquer le point de départ de la lecture de la musique. Pour ce faire, on utilise la fonction « aud_readinit() » comme suit :

```
int sample_rate = 0;
int sample_size = 0;
int channels = 0;

// Lecture des caractéristiques du fichier et récupération du descripteur du fichier
int file_descriptor = aud_readinit(filename, &sample_rate, &sample_size, &channels);

if (file_descriptor == -1){
   printf("Erreur du file_descriptor\n");
   return -1;
}
```

La fonction demande également de passer en paramètre trois entiers qui seront modifiés. Ces trois entiers nous donnent les informations nécessaires à la lecture du fichier (c'est-à-dire sa fréquence d'échantillonage, la taille des échantillons ainsi que le nombre de canaux de lecture).

Ensuite, nous devons récupérer le descripteur du lecteur audio. Cela sert à préparer le lecteur audio de l'ordinateur à la lecture du fichier. On utilise pour cela la fonction « aud_writeinit() » comme montré ci-dessous :

```
// Initialisation du descripteur du lecteur audio
int audio_descriptor = aud_writeinit(sample_rate, sample_size, channels);

if (audio_descriptor == -1){
   printf("Erreur du audio_descriptor\n");
   return -1;
}
```

Les paramètres passés à la fonction sont les informations sur le fichier récupérées grâce à la fonction « aud_readinit() » utilisée précédemment.

Avant de passer à la lecture du fichier, il est nécessaire de déclarer de nouvelles variables :

- bytes_lu : un tableau de char de taille « sample_size » (taille des échantillons). Il servira à stocker les bits composant l'échantillon lu afin de les écrire dans le lecteur.
- nbr_bytes_lu : un entier indiquant la taille de l'échantillon lu et initialisé à la même valeur que « sample_size »
- nbr_bytes_ecrits : un entier indiquant la taille de l'échantillon écrit dans le lecteur et initialisé à la même valeur que « sample_size »

```
char bytes_lus[sample_size];
ssize_t nbr_bytes_lu = sample_size, nbr_bytes_ecrits=sample_size;
```

La dernière étape consiste donc à lire le fichier. Pour cela, on utilise une boucle while dont on ne sort que lorsque les échantillons lus et écrits ne sont plus de la bonne taille (une fois que la lecture du fichier est arrivée à sa fin, les échantillons sont de taille 0). La boucle utilisée est la suivante :

```
while(nbr_bytes_lu == sample_size && nbr_bytes_ecrits == sample_size){
    // Lecture des octets dans le fichier.wav
    nbr_bytes_lu = read(file_descriptor, bytes_lus , sample_size);
    // Ecriture de ces octets dans le lecteur audio
    nbr_bytes_ecrits = write(audio_descriptor, bytes_lus, sample_size);
    // Nettoyage du tableau bytes_lus
    bzero(bytes_lus, sample_size);
}
```

La lecture se déroule en 3 étapes :

1. On lit un échantillon depuis « file_descriptor » de taille « sample_size » et on l'écrit dans le tableau « bytes_lu ». Le tableau contient donc la valeur de l'échantillon lu dans le fichier.
2. On écrit dans le lecteur indiqué par « audio_descriptor » un échantillon de taille « sample_size » à partir du tableau « bytes_lus ». L'échantillon lu lors de la première étape est écrit dans le lecteur et est donc joué par l'ordinateur.
3. On nettoie le tableau grâce à la fonction bzero afin d'être sûr que le tableau contiendra bien la valeur de l'échantillon lors des lectures suivantes.

Une fois la lecture terminée, on affiche un message dans le terminal et le programme s'arrête.

Améliorations possibles :

- Nous n'avons pas été en mesure de résoudre un problème nous empêchant d'ouvrir le fichier si le chemin jusqu'à celui-ci était un chemin relatif. Cela nous a obligé à entrer en dur un chemin jusqu'au dossier data contenant les fichiers .wav qui doit être changé manuellement selon la machine sur laquelle l'utilisateur exécute le programme.