Architecture des ordinateurs
Format ELF

Compiler le programme suivant pour obtenir un fichier objet `file.o` :

```
int main(void) {
  return 0;
}
```

Lancer la commande `$ readelf -h file.o` pour obtenir differentes informations sur l'entête du fichier. On y lit notamment que la taille de cette entête est de 64 octets. On va retrouver ces informations directement dans le fichier binaire avec `hexdump`. Par exemple la commande `echo "Helloworld" | hexdump -C` affiche `00000000  48 65 6c 6c 6f 77 6f 72  6c 64 0a     |Helloworld.|`. Dans ce cas le premier ensemble de zéros est un compteur en octet, suivent ensuite le code ASCII de la chaîne en hexadécimal et enfin la chaîne correspondante, qui est affichée entre `|  |` où un point remplace un caractère non affichable (nouvelle ligne ici, de code ascii `0a`). L'option `-n` suivi d'un chiffre permet de n'afficher que les n premiers caractères. Afficher l'entête du fichier avec la commande `$ hexdump -C -n 64 file.o`. Retrouver le plus d'informations données par la commande `readelf` (c'est donné dans l'ordre et le type `REL` est 01).

On lance la commande `$ readelf -S file.o` pour la table des sections. Regarder l'ensemble des sections décrites dans l'entête de sections. L'adresse de décalage est l'adresse du code correspondant dans le fichier. Vous connaissez le nombre de section qui s'y trouve et la taille de chacune, vous connaissez donc la taille entière de l'entête. Vous connaissez aussi son adresse (de décalage), vous connaissez donc l'adresse (de décalage) à la fin de cette entête. Vérifier que ça correspond à la fin du fichier à l'aide de la commande `$ wc file.o`. Remarque : vous pouvez aussi regarder ce que donne `$ objdump -h file.o`. Utiliser `$ hexdump -C -s N1 -n N2 file.o` en remplaçant `N1` par l'adresse de décalage de l'entête de section et `N2` par le nombre d'octet de cette entête pour afficher le code hexadécimal de cette entête (Attention : `hexdump` prend ses options en décimal, pas en hexadécimal). Identifier chaque section, à l'aide de leur taille. Prenez la section `.bss` (elles sont dans l'ordre). Les quatres premiers octets correspondent au nom de la section (en fait un index lié à la section `.shstrtab`), les quatres suivants à son type (par exemple 8 pour `NOBITS`), puis le fanion (1 pour `WRITE`, 2 pour `ALLOC`), ainsi que son adresse de décalage et sa taille. Remarque : dans un fichier objet les sections sont toutes à l'adresse 0.

Les différentes sections se trouvent donc entre l'adresse `0x40` et `N1`. On peut les afficher avec `$ hexdump -C -s 64 -n N3 file.o`, où `N3` = `N1-64`. Chercher dans les données précédentes l'adresse (de décalage) `N4` et la taille `N5` de la section `.comment` et afficher la avec `$ hexdump -C -s N4 -n N5 file.o`. Examiner la section `.text` (récupérez là avec son adresse de décalage et sa taille avec `hexdump`). Désassembler le fichier `file.o` avec `$ objdump -S file.o`. Normalement vous trouvez le code binaire correspondant au code assembleur de votre fonction `main`. Afficher la table des chaînes de l'entête des sections avec `$ readelf -x .shstrtab file.o` et retrouver y l'index de la section `.bss`.

Recompiler le programme pour obtenir un exécutable (noté `a.out`). Regarder l'entête du fichier et identifier les differences : le type est différent (03 pour `DYN`), l'adresse du point d'entrée, le début des entêtes de programme et de section. Il y a désormais des données dans l'entête de programme (noter leur nombre et leur taille) et le nombre de sections a changé.

Afficher l'entête du programme avec la commande `$ readelf -l a.out` (remarquer qu'il suit l'entête du fichier). Vous avez la liste des segments et leur caractéristiques, ainsi que la correspondance entre segments et sections (qui n'est pas stockée dans le fichier mais calculée par `readelf`). Remarquez que ce ne sont pas les mêmes sections que dans `file.o`. Vous connaissez le nombre d'entêtes du programme (segments) et leur taille, multiplier les pour obtenir le nombre `N6` d'octet de cette section. Afficher toute l'entête avec la commande `$ hexdump -C -s 64 -n N6 a.out` en identifiant chaque segment.

Les 4 premiers octets décrivent le type de segment (6 pour `PHDR`, 1 pour `LOAD`,..). Les quatres suivants le fanion pour les droits d'accès (`R` $= 4$, `W` $= 2$, `E` $= 1$). On a ensuite le décalage, adresse virtuelles et physique, puis la taille fichier, la taille mémoire et l'alignement. On rappelle que le décalage est l'adresse calculée par rapport au début du fichier. Par exemple le segment `PHDR` qui correspond à cette entête est à l'adresse (de décalage) `0x40`. Vérifier que la taille donnée correspond à celle que vous avez calculé (`N6`). Vérifier aussi que le segment `INTERP` correspond à des sections situées juste après cette entête, puis identifiez l'adresse (de décalage) du début et de la fin des sections du segment `NOTE`. Afficher les sections de ce dernier avec `hexdump`. Vérifier que les informations obtenues correspondent à celles obtenues avec `$ readelf -n a.out`.

Retrouver l'entête des sections : `$ readelf -S a.out`. Vous avez le nombre de sections et la taille de chacune dans cette entête, vous avez donc sa taille. Vérifier à l'aide de son adresse de décalage que cette entête est à la fin du code exécutable (en comparant avec `$ wc a.out`). L'emplacement de chaque section dans le fichier exécutable est donné par l'adresse de décalage, qui se situe entre la fin de l'entête des programmes et le début de l'entête des sections. Chercher celle de la section `.eh_frame_hdr` puis celle du segment où cette section intervient et vérifier que ces adresses de décalage coïncident. Regarder les droits d'écriture et d'exécution des segments `LOAD` et vérifier qu'ils sont cohérents avec les sections correspondantes. Etudier maintenant l'adresse des différentes sections (qui pour rappel étaient à 0 dans le fichier objet) : les premières sections ont des adresses croissantes correspondant à l'adresse précédente plus la taille de la section. A un moment donné, il y a un gros trou dans la mémoire, puis on reprend de nouveau les adresses croissantes correspondant à l'adresse précédente plus la taille (remarque : il peut y avoir un petit décalage). Remarquer que la section `.text` a grossi et retrouver votre fonction main à l'aide de la commande `$ objdump -S a.out | grep -A8 main.:` (noter le `.` au lieu de `>`).

Nous allons terminer ce TP en étudiant le comportement de la commande `$ strip a.out` qui a été vue à la fin de la partie compilation et édition des liens (qui permettait de réduire la taille du code exécutable). Vérifier dans l'entête du fichier que le nombre de section a baissé et que l'entête des programmes n'a pas changé. Identifier les sections manquantes et calculer la place gagnée en sommant leur taille, ainsi que la place gagnée correspondant dans l'entête des section. Vérifier que ça correspond à ce qui était attendu si on utilise la commande `wc` (il peut y avoir une légère différence, du à des décalages).