Compte-rendu

Introduction :

L'objectif de ce projet est de créer un programme capable de réaliser la phase de fusion d'un éditeur de lien (Linker). Cela nous permet de fusionner deux fichiers au format ELF dans un seul fichier. Le projet est composé en 2 phases :
— Phase 1 : Lecture d'un fichier ELF32, el l'affichage
— Phase 2 : Fusion des données et modification

Mode d'emploi :

Compilation

Pour compiler les fichiers C de notre projet, on doit tapez ,/configure, et apres make, aprés qu'on soir dans le bon répertoire. Apres on aura le programme executable readelf.

execution :

./readelf [option] fichierELF

en lancant ce programme on obtiendra on obtiendra l'affichage

Les options sont :
-h : affichage du header (l'en-tête du fichier ELF)
-S : affichage de la table des sections
-x :affichage du contenu d'une section en tapant son numero ou son nom
-s : affichage de table de symboles
-r : affichage de relocations
-T : Obtenir le nombre de Section de fichier elf indiqué

./readelf fichier1 fichier2 -f

pour fusionner fichier1 et fichier2 au format ELF en un seul fichier

bash AutoTest.sh

pour test automatique

Description du code

Programme principal(readelf.c)

Le programme est réparti en plusieurs en fichier où readelf.c en est le principal. Il contient simplement le main qui est divisé en deux partie : l'une comprend des différentes **affichages** dans lesquels on peut choisir ce qu'on veut afficher selon un

fichier fourni au format ELF, l'autre est pour **fusionner** deux fichiers donnés en argument au format ELF si on a 4 arguments sur la commande.

all_elf_func.c

Ces fonctions correspondent à toutes les déclarations des structures, l'allocation de leur mémoire pour mémoriser temporairement le header, sections, symboles, relocations et deux strings tables(.shstrtab et .strtab). Et changement en little endian eventuellement.
En fin on a une fonction pour **liberer** la memoire des informations du fichier elf.

void initElf(Elf32_info *elf,FILE *file);
+ Initialise une structure Elf32_info contenant les information relative au fichier ELF (header, section, table des symboles et leurs noms ainsi que la table des relocations).
+ Appelle toutes les fonctions nécessaires qui récupèrent les données.
+ Alloue la mémoire nécéssaire pour les informations du header ainsi que les sections.
+ Convertit les valeurs en little endian si elles sont en big endian

void liberer(Elf32_info elf);
+libérer la mémoire qui est alloué par la fonction initElf

void getNbSection(Elf32_info elf);
+Écrire le nombre de section dans un fichier temporaire

Et les fonctions suivantes sont pour la partie 2

void read_progbits(Elf32_info *elf,FILE *file);
+lire le contenu du chaque section du ficher elf, en format binaire et stock les bits dans la structure Elf_info.progbits

void insert_bit(FILE *f3, Elf32_info *elf2, int dec_from_be, int size_of_fo, int size2 ,int nr_sec);
+ la funtion pour concatener les bits

void insert_bit_end(FILE *f3, Elf32_info *elf2, int nr_sec, int size);
+la function pour ecrir les bits a la fin du fichier

void concatener(Elf32_info *elf1, Elf32_info *elf2, Elf32_info *elf3 ,FILE *f3, FILE *f2) ;
+la functions qui parcours les deux ELF files et les concatenne

header.c sections.c symbole.c relocation.c

pour implémenter le **lecture** et **l'affichage** des différentes parties du fichier ELF.

header.c

```
//affichage de l'en -tête
void affiche_Magic(Elf32_info elf);
void affiche_Classe(Elf32_info elf);
void affiche_Type(Elf32_info elf);
void affiche_DATA(Elf32_info elf);
void affiche_Version(Elf32_info elf);
void affiche_Machine(Elf32_info elf);
// si coder en big endian, transformer en little endian pour l'affichage correcte
void setto_little_endian(Elf32_info *elf);
//etape 1 Affichage de l'en - tête
void afficheHeader(Elf32_info elf);
```

section.c

```
// Stocke les informations relatives aux sections dans la structure elf.
void lire_Section_table(Elf32_info *elf,FILE *file);
// Alloue la mémoire pour stocker la table contenant les noms des sections.
void init_strtable(Elf32_info *elf,FILE *file);
// Liste de type de section
char * getSectionType(Elf32_Shdr sh);
// Liste de flags de section
char * get_elf_section_flags(Elf32_Word sh_flags);
// si coder en big endian, transformer en little endian
void sectionto_little_endian(Elf32_info *elf,int i);
//etape 2 Affichage de la table des sections
void afficheTableSection(Elf32_info elf,FILE *file);
//etape 3 Affichage du contenu d’une section
void afficher_contenu_section(Elf32_info elf,FILE* fsource);
```

symbole.c

```
//Stocke les informations relatives à la table des symboles dans la structure elf.
void read_Sym(Elf32_info *elf,FILE *file);
//Alloue la mémoire pour stocker la table contenant les noms des symboles.
void read_SymTable(Elf32_info *elf,FILE *file);
//Affichage la valeur de symbole
uint16_t print_val_sym(Elf32_info elf,FILE *file,int k);

//etape 4 Affichage de la table des symboles et des détails relatifs à chaque symbole
void aff_s(Elf32_info elf,FILE *file);
```

relocation.c

```
//Stocke les informations relatives à la table des relocations dans la structure elf.
void read_Rel(Elf32_info *elf,FILE *file);
//etape 5 Affichage des tables de réimplantation et des d´etails relatifs à chaque entré.
void aff_r(Elf32_info elf,FILE *file) ;
```

swap.c

Des fonctions de transformer entre BIG ENDIAN et LITTLE ENDIAN.

Definitions et structures des codes

-elf.h : des types de fichier elf en 32 bits.
-baseElf32.h : un structure de type Elf32_info qui contient header,sections,strtable,symtab,reltab.