PROJET - PROG5 2024-25 **Groupe 6**
AMARCHA Mohamed
DABOUSSI Akram
D'HERIN Arthur
MAKOUNDIKA KIDZOUNOU Ifrel Rinel
NTYAM Kevin
ROUSSEAU Lenaïc
SCHWARTZMANN Victor

Phase de réimplantation d'un éditeur de liens : Descriptif de la structure du code

Pour obtenir toutes les données du fichier dans les différents programmes que nous avons implémentés, nous appelons la fonction *read_file*. Cette fonction permet de remplir une structure **ELF32_FILE** contenant toutes les structures nécessaires pour tous les affichages.

Les fonctions d'affichage sont implémentées dans les fichiers **display.c/.h**, qui contiennent les fonctions suivantes :

- *display_ELF_header* : renvoie une erreur si le fichier donné en argument n'est pas un fichier ELF, appelle la fonction *print_ELF_header* sinon.
- *print_ELF_header* : affiche le contenu de la structure remplie **ELF32_Ehdr** donnée en paramètre de la fonction.
- *display_ELF_section_table* : affiche la table des sections de la structure remplie **ELF32_FILE** donnée en paramètre de la fonction.
- *display_ELF_section_content* : affiche le contenu de la section de nom *section_name* se trouvant dans la structure remplie **ELF32_FILE,** donnés en paramètre de la fonction.
- *display_symbol_table* : affiche la table des symboles de la structure remplie **ELF32_FILE** donnée en paramètre de la fonction.
- *display_relocation_table* : affiche la table de réimplantation, si elle existe, de la structure remplie **ELF32_FILE** donnée en paramètre de la fonction.

Afin de pouvoir identifier et renvoyer de manière simple et lisible les différentes erreurs que nous pouvons rencontrer lors d'une lecture, nous avons mis au point la structure **ERR_CODE**, présente dans **status.h**, contenant les macros d'erreur suivantes:

- ***OK*** : aucune erreur n'a été relevée
- ***ERR_ALLOC*** : une allocation mémoire a échouée
- ***ERR_READ*** : une erreur a eu lieu lors de la lecture d'octets
- ***ERR_WRITE*** : une erreur a eu lieu lors de l'écriture d'octets
- ***ERR_MAGIC_BYTES*** : les "magic bytes" typiques d'un fichier ELF sont absents du fichier.

La structure **ELF32_FILE** contenant toutes les données nécessaires à l'affichage de données et à la réimplantation d'un fichier ELF est définie dans **structure.h** :
Cette structure contient en réalité plusieurs structures récupérées ou inspirées par **elf.h**. ***Elf32_Ehdr*** et ***Elf32_Sym*** viennent de **elf.h** et contiennent respectivement les données de l'entête de fichier ELF et les données de la table des symboles. Cependant, pour la table des sections, nous utilisons la structure ***Elf32_ShdrFull*** qui contient l'entête et le contenu

d'une section. De plus, pour les données de la table de réimplantation, nous avons défini une structure ***Elf32_Relocation*** contenant plus de données que la structure ***Elf32_Rel*** de **elf.h** : chaque section de réadressage est stockée dans un tableau de type ***Elf32Rel***, et *len_tab* contient la longueur totale de la table de réimplantation.
La structure ***Elf32Rel*** contient toutes les données d'une section de réadressage : son nom, son offset, son nombre d'entrées, son index, sa taille, et enfin un tableau de type ***__Elf32Rel*** contenant toutes les entrées de la section de réadressage.
Enfin, la structure ***__Elf32Rel*** contient toutes les données d'une entrée présente dans une section de réadressage (cf. ***Elf32_Rel*** dans **elf.h**), ainsi que des informations supplémentaires permettant de faire plus facilement le lien avec la table des symboles.

Les fonctions permettant d'effectuer des lectures ou écritures d'octets se situent dans les fichiers **io_fun.c/.h** :

- *get_u8*, *get_u16* et *get_u32* : permettent de lire respectivement 1, 2, et 4 octets dans le fichier. Les données sont récupérées dans le buffer donné en paramètre.
- *operationHex* : permet de soustraire 4 octets char1, char2, char3 et char4 séparés à la fois par la même valeur. Les différents résultats sont stockés dans les pointeurs add1, add2, add3 et add4.

Les fonctions permettant de remplir toutes ces sous-structures et donc la structure **ELF32_FILE** sont implémentées dans structure.c, et sont les suivantes :

- *read_file* : permet de récupérer toutes les informations de la structure **ELF32_FILE** à l'aide des fonctions *read_elf_header* (remplit la structure ***Elf32_Ehdr***), *read_section_tab* (remplit la structure ***Elf32_ShdrFull***), *read_symbol_tab* (remplit la structure ***Elf32_Sym***) et *read_reimp_tab* (remplit la structure ***Elf32_Relocation***).
- *free_file* : permet de libérer la structure **ELF32_FILE**
- *len_sections* et *offset_sections* : donnent respectivement la longueur et l'offset de la table des sections
- *write_file* : permet d'écrire toutes les informations de la structure **ELF32_FILE** à l'aide des fonctions *write_elf_header* (pour la structure ***Elf32_Ehdr***), *write_section_tab_header* (pour la structure ***Elf32_ShdrFull***), *write_section_content* (pour écrire le contenu des sections) et *write_symbol_tab* (pour la structure ***Elf32_Sym***).
- *symbols_table_index* et *len_symbols* : renvoient respectivement l'index et la longueur de la table des symboles
- *__read_reimp_tab_rel* : permet de remplir la structure ***__Elf32Rel*** contenant les données d'une entrée présente dans une section de réadressage.
- *restructuration_section* : permet d'effectuer la réimplantation complète d'un fichier ELF donné.
- *__remov_rel* : permet de renuméroter les sections
- *__adrresse_absolue_sections* : met à jour l'adresse des symboles

- *__correction_value_symbols* : permet de corriger les symboles
- *__reimplantation_R_ARM_ABS* : permet d'effectuer les réimplantations de type R_ARM_ABS32, R_ARM_ABS16 et R_ARM_ABS8.

- *find_section* : permet de récupérer une section dans la table des sections à partir d'un nom donné.
- *get_section_by_index* : permet de récupérer une section dans la table des sections à partir d'un index donné.

Nous avons également modifié le programme **ARM_runner_example** que l'on a renommé en **ARM_runner** afin de pouvoir prendre en argument un fichier ELF réimplanté que nous voulons tester sur le simulateur ARM. Pour cela, nous nous servons de la fonction *find_section* permettant de définir les adresses respectives des sections *.text* et *.data*, ainsi que la section de code.