Compte rendu projet
PROG5

Réalisation d'un éditeur de liens

Groupe 1:

BARJON Clément
COMBE Gaétan
FAURIE Alban
GREGOIRE Thomas
POTELBERG Maxime
VIAL Fabien

Mode d'emploi:

1.1 Compilation:

make

1.2 AffichageELF :

Le programme fonctionne de la manière suivante :

./affichageElf [option] [fichier]

Option	Résultat
-a	Affiche toutes les informations du fichier ELF (option par défaut)
-h	Affiche le Header
-S	Affiche la Table des Sections
-x [i]	Affiche le contenu de la Section numéro **i**
-s	Affiche la Table des Symboles
-r	Affiche la Table des Réimplantations
-z --help	Affiche cette Aide, pas besoin de fichier en paramètre

1.3 Fusion:

Le programme fonctionne de la manière suivante:

./main [fichier1] [fichier2]

La fusion de ces fichiers sera écrite dans le fichier: *file_fusion.o*

1.4 Tests:

Mettre les fichiers de tests dans le dossier *./tests*

Lancement du shell en faisant : *./test.sh*

Descriptif de la structure du code développé :

1.1 Fichier reader_binaire.c:

Ce fichier nous permet de créer et utiliser un lecteur nous permetant de lire des octets présents dans un fichier (1, 2 ou 4 octets à la fois, little ou big endian).
Le lecteur est initialisé à partir d'un fichier ou vide de taille choisie, il peut ensuite être enregistré dans un fichier.
Ce fichier nous sert également à écrire le fichier "file_fusion.o" final, qui résulte de notre fusion des fichiers passés en paramètre.

1.2 Fichier lecteur_fichier.c:

Dans ce fichier sont décrites les structures utilisées tout au long du projet: header, section header, table de symbole et table des réimplantation.
Ce fichier nous permet de remplir toutes les structures ci-dessus à partir d'un lecteur et de remplir le magic number.
Additionnellement, il nous permet de récupérer un string à partir d'un lecteur et de sa position dans ce dit lecteur.

1.3 Fichier afficheur.c:

Ce fichier permet d'afficher les structures définies plus haut ainsi que leurs composants individuels.

1.4 Fichier affichageElfMain.c:

Ce fichier nous permet d'utiliser les commandes d'affichage d'un fichier ELF décrit dans le mode d'emploi plus haut.

1.5 Fichier fusion.c:

Ce fichier nous permet de fusionner, en 3 étapes, deux fichiers ELF: fusion_section(...), fusion_symbol(...) et fusion_relocation(...) qui permettent de fusionner respectivement les sections, les tables de symboles et les tables de réintégration

1.6 Fichier main.c:

Ce fichier nous permet de fusionner 2 fichiers entrés en paramètre et qu'il enregistrera dans un fichier "file_fusion.o"

Liste des fonctionnalités implémentées et manquantes:

Fonctionnalités implémentées:

- Affichage de l'en-tête
- Affichage de la table des sections
- Affichage du contenu d'une section
- Affichage de la table des symboles
- Affichage de la table des réimplantation
- Fusion et renumérotation des sections
- Fusion, renumérotation et correction des symboles
- Fusion et correction des tables de réimplantation
- Production d'un fichier résultat au format ELF

Aucune fonctionnalité manquante

Liste des éventuels bogues connus non résolus:

- Le start of section header n’est pas le même entre notre fusion et celle réalisé par arm-none-eabi
- Dans certains cas précis, le calcul du sh_info et sh_link est incorrecte -> Cela se produit dans le cas où l’index indiquée n’a pas de correspondance dans le premier fichier (ce bug peut également se produire dans la table des symboles pour les mêmes raisons)

Liste et description des tests effectués:

Nos tests sont effectués automatiquement en lançant le fichier ./test.sh

Affichage:

- Le programme parcourt tous les fichiers du répertoire ./tests et compare leur affichage avec notre programme, avec leur affichage par la commande *arm-none-eabi-readelf*
- - S'il n'y a pas de différence le message suivant est affiché : " ['SECTION de l'elf'] OK pour 'fichier' "
- - Sinon il affiche : " ['SECTION de l'elf'] ECHEC pour 'fichier' à la ligne x " ou juste " ['SECTION de l'elf'] ECHEC pour 'fichier' "
- - - Dans ce cas-là, il indique l'endroit où il y a une différence. ATTENTION : La manière pour comparer les affichages n'est pas "très propre" (ie, j'ai récupéré les valeurs "utiles" des deux affichages, en supprimant toute la mise en page, et ensuite comparées), cela implique que le texte renvoyé n'est pas très lisible.

Les headers sont comparés simplement, ensuite les tables de section, symbole et réimplantation sont comparés ligne par ligne ainsi que le test de chaque section.

Fusion:

Ce test compare un fichier fusionné avec arm-none-eabi et l'autre fusionné par notre programme. Les headers de ces fichiers sont ensuite comparés.
On teste aussi que certaines fusion échoue, par exemple la fusion d'un fichier avec lui-même, etc…

Journal décrivant la progression du travail et la répartition des tâches au sein du groupe:

Lors de la semaine précédant les vacances, nous avons commencé par comprendre le sujet en lisant tous les documents mis à disposition. Nous avons décidé de nous organiser en binôme dans un premier temps, et puisque les étapes de la phase 1 sont globalement réalisables en parallèle, nous avons pu avancer rapidement au début. Un premier binôme Gaëtan/Alban a touché aux étapes 2, 3 et 5, tandis que le binôme Fabien/Clément a réalisé l'étape 4 et le binôme Thomas/Maxime,a réalisé l'étape 1. A la fin de la semaine, nous avions bien avancé la Phase 1, et chaque binôme a touché a un peu toutes les étapes.
Au retour des vacances, nous voulions finir la Phase 1 pour l'audit de code du mercredi 4 janvier, et ainsi garder assez de temps pour la Phase 2 tout en ayant appliqué les modifications suggérées durant l'audit.
Cependant, nous avons pris un peu de retard à ce moment-là.
A cause d'un problème de fichier ouvert et rouvert dans tout le programme, il nous a été suggéré d'ouvrir une unique fois le fichier et le l'écrire en mémoire. Un binôme s'est donc occupé de réaliser une structure Lecteur, qui permet de garder le pointeur vers le fichier en mémoire, tandis que les autres membres du groupe s'occupaient de finaliser les autres étapes de la Phase 1.
Les binômes ont ensuite évolué selon les difficultés rencontrées, afin de combler nos lacunes.
La Phase 2 fût plus compliquée pour nous, de par le manque de documentation comparé à la Phase 1, et du fait que les étapes ne sont plus vraiment réalisables en parallèle. Pour ce faire, nous avons continué de travailler par petits groupes, de façon à ce que chacun touche un peu à toutes les étapes. Tandis qu'un groupe Thomas/Gaëtan/Fabien s'occupait de l'étape 6, Maxime commençait à se renseigner sur l'étape 7 et le binôme Clément/Alban sur les étapes 8 et 9. A la fin de la semaine du 3 janvier, nous avions l'étape 6 correcte, et nous pouvions commencer la dernière semaine sur l'étape 7.
Les étapes 7 et 8 ont été les plus dures pour notre groupe, à cause d'une multitude de petits problèmes qui s'accumulent, notamment sur l'étape 8, où un problème d'offset nous aura bloqué pendant 2 jours.
Toutefois, nous finissons ce projet avec seulement l'étape 8 qui fonctionne partiellement, tandis que les autres sont correctes.