Université Grenoble Alpes
621 avenue Centrale
38400 Saint-Martin-d'Hères
Département Informatique

Projet PROG 5

Rapport de projet

Thibault CHASSAGNOL
Bruno DA CRUZ SANTOS
Paul FAURE-JEUNOT
Alban FAURIE
Adrien GLEMBA
Mathis LAMBERT

3e année de Licence Informatique

Responsable :
Guillaume Huard

2021/2022

TABLE DES MATIÈRES

Mode d’emploi

Pour compiler et exécuter chacun des programmes développés dans le cadre de ce projet, il suffit de se placer dans le dossier courant du projet contenant le « Makefile » et de saisir dans un terminal « Bash », par exemple, la commande « make » afin de compiler le projet. Une fois le projet compilé, il suffit de saisir l’une des commandes suivantes en fonction du résultat attendu.

Afficher tout (-h -S -s -r) :	*./main **-a** chemin_du_fichier_ELF*
Afficher l'en-tête du fichier :	*./main **-h** chemin_du_fichier_ELF*
Afficher l'en-tête des sections :	*./main **-S** chemin_du_fichier_ELF*
Afficher la table des symboles :	*./main **-s** chemin_du_fichier_ELF*
Afficher les ré adressages :	*./main **-r** chemin_du_fichier_ELF*
Afficher la section sous forme d'octets:	*./main **-x** chemin_du_fichier_ELF*
Afficher les parties :	*./main **-p** chemin_du_fichier_ELF*
Afficher l'aide-mémoire :	*./main **-H** chemin_du_fichier_ELF*

Description de la structure du code

1.1 Fichier elf_header.c

Le fichier « elf_header.c » contient les fonctions « lire_elf_header() » qui renvoie une structure correspondant à l'entête du fichier ELF passé en paramètre et « afficher_elf_header() » qui affiche une structure correspondant à l'entête d'un fichier ELF.

1.2 Fichier section_header.c

Le fichier « section_header.c » contient les fonctions « lire_section_header() » qui renvoie un tableau de structures correspondant aux entêtes de sections du fichier ELF passé en paramètre et « afficher_section_header() » qui affiche les entêtes des sections d'un fichier ELF.

1.3 Fichier section.c

Le fichier « section.c » contient les fonctions « lire_section() » qui renvoie un tableau d'octets correspondant au contenu de la section passée en paramètre et « afficher_section() » qui affiche le contenu de la section passée en paramètre.

1.4 Fichier symbol.c

Le fichier « symbol.c » contient les fonctions « lire_symbole() » qui crée une structure contenant la table des symboles et « afficher_symboles() » qui affiche la table des symboles correspondant à un fichier ELF.

1.5 Fichier relocation.c

Le fichier « relocation.c » contient les fonctions « lire_relocations() » qui renvoie un tableau de relocations correspondant aux ré-adressages à effectuer,

« liberer_relocations() » qui libère la mémoire occupée par un tableau de relocations, « afficher_relocations() » qui affiche les ré-adressages à effectuer et « afficher_type_relocation() » qui affiche le type de relocations qui correspond.

1.6 Fichier section_sans_rel.c

Le fichier « section_sans_rel.c » contient les fonctions « nb_rel() » qui donne le nombre de sections de relocation qui permet de calculer le nouveau nombre de sections, « maj_section() » qui supprime des sections qui sont des relocations et « maj_ndx() » qui met à jour les symboles, le NDX et la valeur.

1.7 Fichier string_table.c

Le fichier « string_table.c » contient les fonctions « lire_string_table() » qui lit la string table contenant des strings, « lire_strtab() » qui lit la string table contenant les noms des symboles, « lire_shstrtab() » qui lit la string table contenant les noms des sections, « index_string() » qui indique quelle est la section des string et la place dans l'en tête et « afficher_chaine() » qui affiche une chaine de caractère contenue à l'index dans la string table strtab jusqu'à une longueur maximale.

1.8 Fichier fonctions_utilitaires.c

Le fichier « fonctions_utilitaires.c » contient les fonctions « hex2dec() » qui convertit une chaîne de caractère représentant une valeur hexadécimale en décimal, « read_uint16() » qui lit un fichier binaire 16 bits par 16 bits suivant le type d'endian sélectionné, « read_uint32() » qui lit un fichier binaire 32 bits par 32 bits suivant le type d'endian sélectionné, « read_int32() » qui lit un fichier binaire 32 bits par 32 bits suivant le type d'endian sélectionné « compter_bits() » qui renvoie le nombre de bits à 1 dans un mot et « extension_fichier() » qui lit l'extension du fichier et renvoie si c'est un fichier exécutable ou non.

1.9 Fichier main.c

Le fichier « main.c » contient les fonctions « usage() » qui permet d'afficher à l'utilisateur toutes les options qu'il peut utiliser pour utiliser le programme et « main() » qui exécute le programme correspondant à l'option sélectionnée précédemment dans « usage() ».

Liste des fonctionnalités

2.1 Liste des fonctionnalités implémentées

Étape 1 : Affichage de l'en-tête

Étape 2 : Affichage de la table des sections

Étape 3 : Affichage du contenu d'une section

Étape 4 : Affichage de la table des symboles

Étape 5 : Affichage des tables de réimplantation

Étape 6 : Renumérotation des sections

Étape 7 : Correction des symboles

2.2 Liste des fonctionnalités manquantes

Étape 8 : Réimplantations de type R ARM ABS*

Étape 9 : Réimplantations de type R ARM JUMP24 et R ARM CALL

Étape 10 : Interfaçage avec le simulateur ARM

Étape 11 : Production d'un fichier exécutable non relogeable

Liste des bogues connus non résolus

3.1 Aucun bogue connu

Aucun bogue connu n'est à déclarer.

Liste et description des tests effectués

4.1 Script tests.sh

L'objectif de ce script « tests.sh » est d'automatiser la réalisation des tests liés à l'affichage des fonctions que nous avons réalisés afin de nous assurer qu'elles fonctionnent correctement. Pour exécuter ce script il suffit de se placer dans le dossier courant du projet et de saisir dans un terminal Bash, par exemple, l'une des commandes ci-dessous en fonction du résultat souhaité.

Activer tous les affichages :	*sh tests/tests.sh* ***all***
Afficher uniquement les tests qui échouent :	*sh tests/tests.sh* ***failed***
Afficher uniquement si les tests réussissent ou non :	*sh tests/tests.sh* ***minimal***
Afficher le message d'aide :	*sh tests/tests.sh* ***help***

Une fois les tests exécutés, un récapitulatif du nombre de tests réussit et échoué apparaît dans le terminal avec un pourcentage global du nombre de test qui ont réussi et qui ont échoué. Dans le cas d'un test avec l'option « failed », une phrase indiquant le but du test qui échoue apparaît dans le terminal et est suivi du nom du fichier test correspondant. Un total de 862 tests ont été réalisés pour ce projet.

Journal de bord du projet

Jour 0 - Avant les vacances

Lors de la formation des groupes, nous étions quatre à nous connaitre à savoir Paul, Mathis, Adrien et Alban nous nous sommes liés au binôme que forme Thibault et Bruno afin de créer un groupe de six personnes. Nous avons fait connaissance les-uns avec les autres puis nous avons commencé à lire le sujet pour pouvoir démarrer le projet. C'est avec le sujet en tête que nous sommes partis en vacances.

Jour 1 – Lundi 3 Janvier

Après une relecture du sujet au retour des vacances, nous avons remarqué que les cinq premières étapes pouvaient être réalisées en parallèle pour gagner en efficacité. Nous décidons alors de réaliser la première étape tous ensemble pour se mettre dans le bain, puis une fois cette dernière terminée, nous nous sommes réparti les étapes suivantes en binômes. Nous avons eu quelques difficultés pour lire le fichier ELF, mais une fois que ce problème a été réglé nous avons pu continuer. À la fin de la journée, nous avons réalisé une fonction qui déchiffre les fichiers ELF en 64 bits et en Big Endian, malheureusement ce n'était pas ce qui était attendu. Avant de passer à l'étape 2, nous devrons donc refaire cette fonction afin qu'elle respecte les consignes du sujet.

Jour 2 – Mardi 4 Janvier

Nous avons pour but de finir la première étape d'ici la fin de la journée. Notre gros problème a été de transposer les données lues de Big Endian vers Little Endian. En effet, le sens de lecture étant différent, nous n'obtenions pas les résultats attendus. Après quelques heures passées à chercher une méthode pour passer du Big Endian au Little Endian, nous sommes parvenus à obtenir une méthode fonctionnelle. En se penchant sur la seconde étape, nous remarquons qu'il est nécessaire de lire des données à partir d'une certaine adresse. Nous récupérons facilement cette adresse grâce à une fonction réalisée à l'étape précédente, mais le problème est d'arriver à lire

un fichier binaire à partir d'une certaine adresse, ici ce qui nous intéresse c'est l'adresse correspondant à la table des sections.

Jour 3 – Mercredi 5 Janvier

On décide de se répartir autrement le travail à réaliser de façon à avancer de manière homogène. Un trinôme réalise la quatrième partie, un binôme se consacre à la deuxième et troisième partie et une personne se penche sur la cinquième partie. Une bonne partie du travail consiste à étudier la forme des données à récupérer, afin de savoir comment les organiser et comment les lires. Cela nous aura pris la matinée. Petit problème en fin de matinée, Internet n'était plus disponible dans les bâtiments de la fac à partir de 11 h sur les ordinateurs de l'IM2AG. Nous sommes donc revenus à 12 h 10, et Internet n'était toujours pas revenu, nous avons donc décidé de continuer le projet depuis chez nous. À la fin de la journée, les parties quatre et cinq ont bien avancé et seront bientôt terminées.

Jour 4 – Jeudi 6 Janvier

L'audit d'oral s'est bien passé, nous sommes relativement dans les temps et notre code à l'air correct, même s'il existe toujours quelques petites améliorations à apporter. Il nous a été conseillé de factoriser les fonctions d'ouverture de fichier afin d'éviter les répétitions de code et d'automatiser nos jeux de test pour faciliter les étapes suivantes. Nous avons donc continué de travailler en groupe afin d'avancer de façon homogène. À la fin de la journée, la deuxième, quatrième et cinquième partie sont presque complètes puisqu'il ne leur manque que quelques éléments de finition. Nous avons comme objectif de bien avancer la troisième étape demain afin de pouvoir terminer la Phase 1 durant le week-end.

Jour 5 – Vendredi 7 Janvier

Cette journée a été perturbée par le Covid car Mathis reçoit enfin le résultat de son test effectué hier et ce dernier se révèle être positif. Il ne pourra donc pas être présent lors de la soutenance du projet. Adrien aussi a été malade toute la journée en raison d'un gros rhume. Alban et Paul ayant été en contact avec Mathis mardi dernier

sont allés se faire tester en fin de la matinée. Alban et Paul sont testés négatifs au Covid. Adrien passe tout son après-midi à faire la queue dans une pharmacie pour finalement qu'on lui annonce qu'il n'y a plus assez de test pour lui. Malgré toutes ces péripéties, la troisième partie avance bien et est presque terminée à la fin de la journée.

Weekend – Samedi 8 Janvier / Dimanche 9 Janvier

Le but du week-end a été de finir correctement la Phase 1, comprenant les cinq premières étapes. Tout le monde travaille donc de façon à terminer ces différentes parties. La troisième étape est complètement terminée d'un point de vue pratique, il reste juste à implémenter la lecture de la section en paramètre. À la fin de la semaine, la Phase 1 est globalement complète, il reste encore quelques détails sur la deuxième et quatrième étape, mais ils seront réglés très prochainement.

Jour 6 – Lundi 10 Janvier

Le travail est réparti en deux groupes, un groupe essayant de réaliser la sixième et septième partie et un second groupe qui se concentre sur la huitième partie. Nous avions identifié la partie ARM comme étant la partie critique de ce projet. La sixième partie avance bien et est quasiment terminée à l'issue de cette journée. Nous commençons également à réfléchir sur la réalisation de la septième partie.

Jour 7 – Mardi 11 Janvier

Nous avons terminé la sixième partie. En se penchant sur la septième partie, nous remarquons que les membres du groupe ne comprennent pas l'énoncé de la même manière et cela ne nous aide pas à avancer. Dans le même temps, nous commençons à mettre en place les éléments nécessaires aux différents rendus ainsi qu'à la soutenance de projet. Cependant, la huitième partie n'avancent que très peu aussi par manque de compréhension.

Jour 8 – Mercredi 12 Janvier

Tout le monde essaye d'avancer tant bien que mal sur la septième et huitième partie. Après concertation, nous avons décidé de laisser tomber l'avancement des différentes étapes manquantes pour se concentrer sur la soutenance ainsi que sur les améliorations possibles, comme l'automatisation des tests et la réalisation des différents documents. Le rapport de projet est quasiment terminé et les tests sont automatisés. Le but de la journée de demain sera de préparer au mieux la soutenance, en réalisant un diaporama et en nous répartissant équitablement le temps de parole.

Jour 9 – Jeudi 13 Janvier

Pour cette dernière matinée, toutes les informations sont mises en commun afin de préparer la soutenance à cinq car Mathis ne pourra pas y participer en raison de son test positif au covid. Nous sommes finalement assez contents d'être arrivés jusque-là, il nous aurait fallu tout de même un peu plus de temps pour bien comprendre les étapes suivantes et pouvoir continuer le projet .

Conclusion

Lors de ce projet, nous avons dû réaliser l'implémentation d'une sous-partie d'un éditeur de liens en C.

Ce projet que nous avons réalisé sur deux semaines nous a permis de mettre en pratique des connaissances vues en cours de PROG 5 à travers l'utilisation du langage C, l'utilisation de script Bash et l'utilisation de Git. Le cours d'ALM nous a aussi été utile pour comprendre le fonctionnement et l'organisation d'un fichier ELF. Nous avons appris à utiliser une documentation technique. Cela nous a également permis de travailler dans un groupe de taille moyenne et de réaliser un projet lui aussi de taille moyenne.

En plus d'avoir été enrichissant du point de vue technique, ce projet a également été instructif du point de vue humain. Grâce à ce projet, nous avons amélioré notre capacité à travailler en groupe, à utiliser des outils et différentes méthodes de travail pour avancer le plus possible sur ce projet.

Webographie

Site du projet, https://prog5.gricad-pages.univ-grenoble-alpes.fr/Projet/
Le site du projet nous a permis d'avoir connaissances du sujet, des différentes étapes à réaliser, d'être tenu aux courants vis-à-vis de certaines informations liées au projet et aussi d'avoir accès aux différents contacts des enseignants pouvant nous aider tout au long du projet.

Elf.pdf, https://prog5.gricad-pages.univ-grenoble-alpes.fr/Projet/elf.pdf
La documentation technique ELF nous a permis de comprendre comment sont structurés les fichiers ELF.

Arm7.pdf, https://prog5.gricad-pages.univ-grenoble-alpes.fr/Projet/arm7.pdf
Le manuel de Référence ARM v7 est la documentation technique liée au langage machine ARM.