

**PROJET PROG 5, EQUIPE 4**

**Réalisation d'un éditeur de liens | Phase de réimplantation**

# Table des matières

[Table des matières 1](#_Toc92958577)

[Mode d’emploi pour la compilation et l’exécution du code 2](#_Toc92958578)

[Descriptif de la structure du code développée 2](#_Toc92958579)

[Fonctionnalités 2](#_Toc92958580)

[Fonctionnalités implémentées 2](#_Toc92958581)

[Phase 1 3](#_Toc92958582)

[Phase 2 3](#_Toc92958583)

[Fonctionnalités manquantes 4](#_Toc92958584)

[Bugs et difficultés 4](#_Toc92958585)

[Bugs et difficultés résolus 4](#_Toc92958586)

[Bugs et difficultés non résolus 4](#_Toc92958587)

[Tests 4](#_Toc92958588)

[Journal de bord 5](#_Toc92958589)

# Mode d’emploi pour la compilation et l’exécution du code

Pour l’exécution du code, exécutez les commandes suivantes :

**./configure**

**Make**

Pour la phase 1, exécutez :

**./readelf2 -[options] [fichier]**

S’il n’y a pas d’options pour la commande, on affiche la liste des options possibles

Pour la phase 2, exécutez :

**./writeElf -[options] [fichier] [adressestext][adresses data]**

Pour l’exécution des scripts de tests, exécutez :

**Sh test\_phase1\_part1.sh**

**Sh test\_phase1\_part2.sh**

**Sh test\_phase1\_part4.sh**

# Descriptif de la structure du code développée

Nous avons créé un fichier **.c** et **.h** pour chaque étape, et chaque fichier d’une étape dépend de celui de la précédente.

Voici la liste des fichiers que nous avons créés :

|  |  |
| --- | --- |
| **L’étape du projet** | **Les fichiers correspondants** |
| Etape 1 | read\_elfHead.c  read\_elfHead.h |
| Etape 2 | read\_elfSection.c  read\_elfSection.h |
| Etape 3 | read\_section\_data.c  read\_section\_data.h |
| Etape 4 | read\_elfSymbol.c  read\_elfSection.h |
| Etape 5 | read\_relocation\_table.c  read\_relocation\_table.h |
| Etape 6 et 7 | renum\_correction.c  renum\_correction.h |
| Etape 8 et 9 | reimplantation\_ARM.c  reimplantation\_ARM.h |

# Fonctionnalités

## Fonctionnalités implémentées

### Phase 1

Read\_elfHead :

* Fonction de test du bigendian
* Fonction d'inversion des bits de poids fort et faible
* Fonction de récupération du header
* Fonction de vérification du type du fichier
* Fonction de récupération de la classe du fichier
* Fonction de récupération type de données du fichier
* Fonction de récupération de la version du fichier
* Fonction de récupération de l'OS du fichier
* Fonction de récupération du type du fichier
* Fonction d'affichage du header du fichier

Read\_elfSection :

* Fonction de stockage du tableau des sections
* Fonction de lecture d’une section
* Fonction d'affichage des sections
* Fonction de récupération du nom de la section
* Fonction de récupération du type de la section
* Fonction de récupération du flag de la section

Read\_sectiondata :

* Fonction de lecture des données de la section
* Fonction de verification de la présence d'une section
* Fonction de vérification de la taille de la section
* Fonction d'affichage des données de la section

Read\_elfSymbol :

* Fonction de lecture de la table des symboles
* Fonction de lecture d’un symbole
* Fonction récupération du nom d'un symbole
* Fonction récupération de la taille de la table des symboles
* Fonction d'affichage de la table des symboles

Read\_relocation\_table :

* Fonction de récupération du nombre de section de réadressage
* Fonction de récupération des données des fonctions de réadressage
* Fonction d'affichage de la table de réadressage

### Phase 2

Renum\_correction : (correspondante aux parties 6 et 7 du projet)

* Fonction de création de la table de renumérotation
* Fonction de comptage de section vide
* Fonction de renumérotation des section
* Fonction de renumérotation et correction de la table des symbole
* Fonction de trie de la table des symboles

Reimplantation\_ARM : (correspondante aux parties 8 et 9 du projet)

Nous avons fait les fonctions de renumérotation de la table des symboles suivantes :

* Fonction d’application de réadressage de type R\_ARM\_JUMP ou R\_ARM\_CALL
* Fonction d’application de réadressage de type R\_ARM\_ABS\*
* Fonction de modification d’un tableau de donnée de section
* Fonction de réadressage du fichier
* Fonction d’affichage des données des sections modifiées
* Fonction de libération de la mémoire du tableau de section

## Fonctionnalités manquantes

Pour les étapes 6, 7, 8, et 9 : Nous avons fait uniquement l’affichage, et pas la réécriture dans un autre fichier.

Nous n’avons pas eu le temps de faire les étapes 10 et 11.

# Bugs et difficultés

## Bugs et difficultés résolus

* Difficultés à comprendre le fonctionnement des réadressages de type r\_arm\_jump et r\_arm\_call
* Difficultés pour accéder à la table des strings afin de récupérer le nom des sections
* Erreurs mémoires liées à l’utilisation excessive de reverse\_endianness dans les fonctions (corrigées dans la structure)

## Bugs et difficultés non résolus

* Difficultés de réécriture dans un nouveau fichier pour les parties 6, 7, 8 et 9.
* On ne sait pas comment ajouter les nouveaux symboles à la table des symboles modifiée

# Tests

* **Example5.s** : un programme ARM qui calcule le max entre 3 nombres
* **Example6.s** : un programme ARM qui calcule un nombre d’une position donnée de la suite de fibonacci
* **Example7.s** : un programme qui calcule le PGCD de deux nombres

Nous avons aussi réalisé des scripts Shell, pour chaque partie qui, pour chaque fichier de la forme Examples\_loader/example\*.o, exécute la commande readelf et compare les résultats avec les résultats de nos programmes (voir et exécuter les fichiers .sh du programme)

# Journal de bord

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Taches** | **Membre chargé de la tache** | **Remarques** |
| 16/12/2021 | * Début projet | Toute l’équipe |  |
| * Lecture sujet et DOC fichiers ELF. |
| * Rédactions individuelles de résumés de la DOC ELF. |
| 17/12/2021 | * Documentation sur le projet | Toute l’équipe |  |
| * Codage de l’étape 1 (globalement fonctionnelle) | Eliot & Tristan |
| * Réalisation du script bash permettant la comparaison des résultats avec ceux de la commande readelf pour la partie 1 | Lucas & Thierry |
| 03/01/2022 | * Résolution des cas manquants pour la première étape (e\_machine, e\_type) | Eliot & Tristan | Fin de la partie 1 |
| * Début de la partie 2 | Lucas & Thierry |
| * Début de la partie 3 | Eliot & Tristan |
| * Début de la partie 4 | Yahia & Karine |
| * Codage de la partie 3 | Eliot & Tristan |
| 04/01/2022 | * Réalisation des scripts des tests partie 2 | Lucas & Thierry | Tests étape 1 : plusieurs types de fichiers, plusieurs formats ELF… de manière générale on teste sur plusieurs fichiers différents et on compare à la fonction readelf.  Tests étape 3 : tester sections inexistante, sections vide, sections remplies.  Étapes 2 et 3 terminées, mais problème dans l’étape 1, car le programme ne fonctionne pas avec les fichiers en littleendian.  Nous avons essayé de modifier la fonction reverse\_endianess en utilisant le header, mais le problème persiste.  Étapes 4 et 5 en fin de production, l’étape 5 n’étant pas testable à cause de ce problème. |
| * Réalisation des scripts des tests partie 4 | Yahia & Karine |
| 05/01/2022 | * Organisation du programme et tests | Toute l’équipe | Pas possible de tester le main final tant que les 5 parties ne sont pas complètement finis |
| * Modification des parties pour inclure la nouvelle version de reverse endian | Toute l’équipe |
| * Continuation du travail sur le codage des partie 2 3 4 | Toute l’équipe |
| * Codage de la partie 5 | Eliot & Tristan |
| * Début du codage du main final de la phase 1 | Eliot & Tristan |
| 06/01/2022 | * Tests et finalisation de la partie 5 | Eliot & Tristan | Parties 1 2 3 4 fonctionnelles  Rassemblement des parties du programme  Réunion avec le prof |
| * Finalisation des parties 2 3 4 | Toute l’équipe |
| 07/01/2022 | * Séparer les fonctions d’affichage et de lecture/stockage en mémoire pour la partie 5 et les autres parties | Toute l’équipe |  |
| * Ecrire plus d’exemples de programmes ARM | Yahia & Karine |
| * Création d’un nouveau main avec les variables booléennes | Eliot & Tristan |
| * Documentation sur la phase 2 | Toute l’équipe |
| 10/01/2022 | * Début/documentation sur les parties 6 et 7 | Toute l’équipe | Partie 6 et 7 :  problèmes au niveau des adresses car nous n’avons pas encore compris comment les changer |
| 11/01/2022 | * Documentation et travail/essaies sur les parties 6 et 7 | Toute l’équipe |  |
| * Documentation sur les parties 8 et 9 |
| 12/01/2022 | * Finalisation des parties 6 et 7 | Toute l’équipe | Parties 6 7 8 9 fonctionnelles |
| * Codage et tests sur les parties 8 et 9 |
| * Finalisation de la partie 8 |
| 13/01/2022 | * Création du main de la phase 2 * Rédaction de rapport * Vérifications du bon fonctionnement de ce qui a été fait * Push de tous les fichiers dans la branche main (branche principale) | Toute l’équipe |  |