Rapport de projet – PROG5

ANNÉE 2024-2025

Responsable : Guillaume HUARD

Réalisation d'un éditeur de liens

Phase de réimplantation

Elie HATOUM

Anis DJEBBAR

Houssein MAJED

Daniel ATWI

Abdellah BARROUG

Youssef RIANI

Elouan DURANTON - MAMADOU

3ème année de Licence Informatique

TABLES DES MATIÈRES

I. Introduction :

A. Contexte et motivation du Projet :

Dans le cadre de ce projet, nous avons travaillé sur la réimplantation d'une sous-partie d'un éditeur de liens pour des fichiers objets au format ***ELF***, à partir d'un programme en langage machine ***ARM***. Ce travail avait pour objectif de permettre la lecture et l'analyse de la structure d'un fichier objet, ainsi que la production d'un fichier exécutable unique. Cette démarche visait à développer un module d'éditeur de liens capable de générer un exécutable.

B. Objectifs du projet :

Maîtriser des aspects techniques de la programmation

Ce projet nous amène à concevoir un logiciel de taille intermédiaire, en mobilisant les connaissances acquises au cours du semestre, notamment en programmation de bas niveau et en gestion de mémoire.

Respecter le cahier des charges

Il a été nécessaire de s'approprier des documentations techniques complexes et de sélectionner les informations pertinentes afin de respecter les spécifications fournies.

Appliquer des principes de génie logiciel

Le travail en équipe a nécessité une organisation rigoureuse, comprenant la répartition des tâches et le suivi de leur avancement.

Répondre aux exigences fonctionnelles

L'enjeu consistait à développer un programme modulaire et performant, capable de lire et d'exploiter efficacement les données des fichiers ELF.

II. Mode d'emploi :

A. Prérequis :

Pour compiler et exécuter notre programme, il est indispensable de disposer d'un système d'exploitation compatible, tel que Linux ou tout autre système basé sur Unix, prenant en charge les commandes de compilation standard telles que ***gcc*** et ***make***, ainsi que les options de compilation spécifiées via ***CFLAGS***. De plus, pour assurer le bon déroulement des tests automatisés, le système Linux utilisé doit être configuré en langue anglaise (voir la section dédiée aux tests).

B. Compilation :

Pour compiler les différents modules du projet, il suffit d'exécuter les commandes suivantes:

./configure

make

Le programme sera alors prêt à être utilisé.

C. Utilisation :

Pour la phase 1, utilisez le fichier ***elf_readelf***, qui fonctionne de manière similaire à la commande ***readelf***.

La commande est la suivante :

./elf_readelf <option> <fichier>

Afin d'afficher les options disponibles, il suffit d'exécuter la commande suivante:

./elf_readelf -o

→ *Le programme propose cinq options:*

-h : Affichage de l'en-tête.
-S : Affichage de la table des sections.
-s : Affichage de la table des symboles.
-r : Affichage des tables de réimplantation.
-x : Affichage du contenu d'une section. Cette option nécessite un argument supplémentaire (le numéro ou le nom de la section) en plus du nom du fichier en ligne de commande.

Exemple: ***./elf_readelf -x fichier_elf .text***

Pour la *Phase 2*, il sera nécessaire de passer en argument le fichier relogeable à l'exécutable **process_rel**, qui générera en sortie le fichier **.o** modifié.

La commande est la suivante :

./process_rel <input_elf_fichier>

Enfin, on lance le simulateur en sélectionnant un port libre à l'aide de l'outil **arm_simulator**, puis on exécute le script **load_sim** en lui fournissant comment arguments le fichier **.o** modifié, **localhost** et le **port** déterminé précédemment.

Exemple:

```
./process_rel  Examples_loader/example3.o

./arm_simulator --gdb-port 3842 --trace-registers --trace-memory --trace-state SVC &

./load_sim Examples_loader/example3.o_modified localhost 3842
```

III. Structure du code :

A. Description générale de la structure :

Notre code est structuré autour de deux étapes principales : la lecture et la réimplantation des fichiers ELF.

La phase de lecture est répartie sur cinq fichiers distincts, chacun correspondant à une étape spécifique du processus d'analyse des fichiers ELF. Ces fichiers contiennent les fonctions nécessaires pour extraire et analyser les données des fichiers objets. Auxquels s'ajoute un fichier principal, incluant la fonction **main**, qui orchestre l'ensemble du processus en appelant les fonctions des cinq fichiers pour effectuer toutes les opérations de lecture. Cette organisation permet de simuler un comportement similaire à celui de l'outil **readelf**, offrant ainsi une interface pour l'utilisateur.

La phase de réimplantation **:**
Toute cette phase est implémentée dans deux fichiers. Un permettant d'effectuer toute la partie réimplantation et l'adaptation du fichier en conséquence: La suppression des sections de réimplantation et la renumérotation ...etc , le second permet de lancer le simulateur et de charger les sections, qui permettre de consulter les valeurs des registres du processeur ARM simulé et comparer avec le code en conséquence.

B. Spécification des fichiers :

elf_header: Lecture et Affichage de l'en-tête

elf_section: Lecture et Affichage de la table des sections

elf_section_contenu: Lecture et Affichage du contenu d'une section

elf_table_symbole: Lecture et Affichage de la table des symboles

elf_relocation: Lecture et Affichage des tables de réimplantation

process_rel: Un fichier contenant les fonctions principales permettant d'effectuer la rénumérotation des sections,la suppresions des sections REL, la mise à jour de la table des symboles, l'attribution des adresses absolues, et l'interface avec le simulateur ARM .

load_sim: Charge le fichier ELF modifié dans le simulateur ARM.

IV. Fonctionnalités implémentées :

A. Description détaillée des fonctionnalités :

Fonctionnalitées implémentées :

- *Lecture et Affichage du contenu d'un fichier au format ELF*

  - Affichage de l'en-tête
  - Affichage de la table des sections
  - Affichage du contenu d'une section
  - Affichage de la table des symboles
  - Affichage des tables de réimplantation

- *Réimplantation d'un fichiers ELF*

  - Numérotation des sections
  - Correction des symboles
  - Réimplantations de type R_ARM_ABS*
  - Réimplantations de type R_ARM_JUMP24 et R_ARM_CALL
  - Interfaçage avec le simulateur ARM

Fonctionnalitées manquantes :

- Production d'un fichier exécutable non relogeable

B. Illustration des résultats obtenus :

```
ELF header:
Magic:   7f 45 4c 46 01 02 01 00 00 00 00 00 00 00 00 00
Class : ELF32
Data : 2's complement, big endian
Version : 1 (current)
OS/ABI: UNIX - System V
ABI Version: 0
Type: REL (Relocatable file)
Machine: ARM
Version: 0x1
Entry point address : 0x0
Start of program headers: 0 (bytes into file)
Start of section headers: 776 (bytes into file)
Flags: 0x5000000
Size of this header: 52 (bytes)
Size of program headers: 0 (bytes)
Number of program headers: 0
Size of section headers: 40 (bytes)
Number of section headers: 16
Section header string table index: 15
```

```
There are 16 section headers, starting at offset 0x308:

  [Nr] Name              Type            Addr     Off    Size   ES   Flg      Lk   Inf   Al
  [ 0]                   NULL            00000000 000000 000000 00              0    0    0
  [ 1] .text             PROGBITS        00000000 000034 000010 00   AX         0    0    4
  [ 2] .data             PROGBITS        00000000 000044 000000 00   WA         0    0    1
  [ 3] .bss              NOBITS          00000000 000044 000000 00   WA         0    0    1
  [ 4] .debug_line       PROGBITS        00000000 000044 00003b 00              0    0    1
  [ 5] .rel.debug_line   REL             00000000 000230 000008 08   I         13    4    4
  [ 6] .debug_info       PROGBITS        00000000 00007f 000026 00              0    0    1
  [ 7] .rel.debug_info   REL             00000000 000238 000038 08   I         13    6    4
  [ 8] .debug_abbrev     PROGBITS        00000000 0000a5 000014 00              0    0    1
  [ 9] .debug_aranges    PROGBITS        00000000 0000c0 000020 00              0    0    8
  [10] .rel.debug_aranges REL            00000000 000270 000010 08   I         13    9    4
  [11] .debug_str        PROGBITS        00000000 0000e0 000049 01   MS         0    0    1
  [12] .ARM.attributes   UNKNOWN         00000000 000129 000012 00              0    0    1
  [13] .symtab           SYMTAB          00000000 00013c 0000e0 10             14   13    4
  [14] .strtab           STRTAB          00000000 00021c 000012 00              0    0    1
  [15] .shstrtab         STRTAB          00000000 000280 000088 00              0    0    1
Key to Flags:
W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
L (link order), O (extra OS processing required), G (group), T (TLS),
C (compressed), x (unknown), o (OS specific), E (exclude),
D (mbind), y (purecode), p (processor specific)
```

```
Vidange hexadécimale de la section '.text':
  0x00000000 e3a00005 e2500001 1affffd ef123456 .....P........4V
```

```
Table des symboles dans la section '.symtab':
  Num:    Valeur   Taille Type    Lien   Vis      Ndx   Nom
  ----   --------  ------ ------- ------ -------- ---   ---
  0      00000000  0      NOTYPE  LOCAL  DEFAULT  0
  1      00000000  0      SECTION LOCAL  DEFAULT  1      .text
  2      00000000  0      SECTION LOCAL  DEFAULT  2      .data
  3      00000000  0      SECTION LOCAL  DEFAULT  3      .bss
  4      00000000  0      NOTYPE  LOCAL  DEFAULT  1      $a
  5      00000004  0      NOTYPE  LOCAL  DEFAULT  1      loop
  6      0000000c  0      NOTYPE  LOCAL  DEFAULT  1      end
  7      00000000  0      SECTION LOCAL  DEFAULT  6      .debug_info
  8      00000000  0      SECTION LOCAL  DEFAULT  8      .debug_abbrev
  9      00000000  0      SECTION LOCAL  DEFAULT  4      .debug_line
  10     00000000  0      SECTION LOCAL  DEFAULT  11     .debug_str
  11     00000000  0      SECTION LOCAL  DEFAULT  9      .debug_aranges
  12     00000000  0      SECTION LOCAL  DEFAULT  12     .ARM.attributes
  13     00000000  0      NOTYPE  GLOBAL DEFAULT  1      main
```

```
Relocation section '.rel.debug_line' at offset 0x230 contains 1 entries:
 Offset     Info    Type            Index
0000002e  00000102  R_ARM_ABS32     00000001
Relocation section '.rel.debug_info' at offset 0x238 contains 7 entries:
 Offset     Info    Type            Index
00000006  00000802  R_ARM_ABS32     00000008
0000000c  00000902  R_ARM_ABS32     00000009
00000010  00000102  R_ARM_ABS32     00000001
00000014  00000102  R_ARM_ABS32     00000001
00000018  00000a02  R_ARM_ABS32     0000000a
0000001c  00000a02  R_ARM_ABS32     0000000a
00000020  00000a02  R_ARM_ABS32     0000000a
Relocation section '.rel.debug_aranges' at offset 0x270 contains 2 entries:
 Offset     Info    Type            Index
00000006  00000702  R_ARM_ABS32     00000007
00000010  00000102  R_ARM_ABS32     00000001
```

V. Tests et validations :

A. Description des tests effectués :

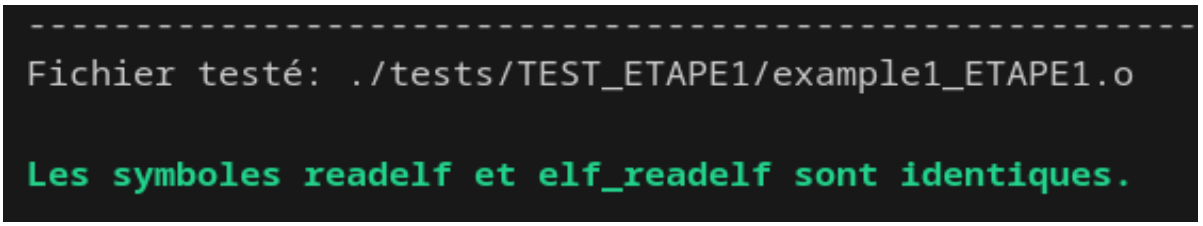
Les tests réalisés pour la lecture et l'affichage consistent à utiliser des scripts shell qui servent à comparer les sorties de notre programme elf_readelf avec celles de l'outil de référence readelf. Si les sorties sont identiques, elles sont affichées en vert, indiquant un succès ou affichant un message indiquant que le test marche avec succès ; en cas de différences, celles-ci sont marquées en rouge pour signaler un écart. Ces tests ont confirmé que notre implémentation produit des résultats conformes à readelf pour ces étapes.

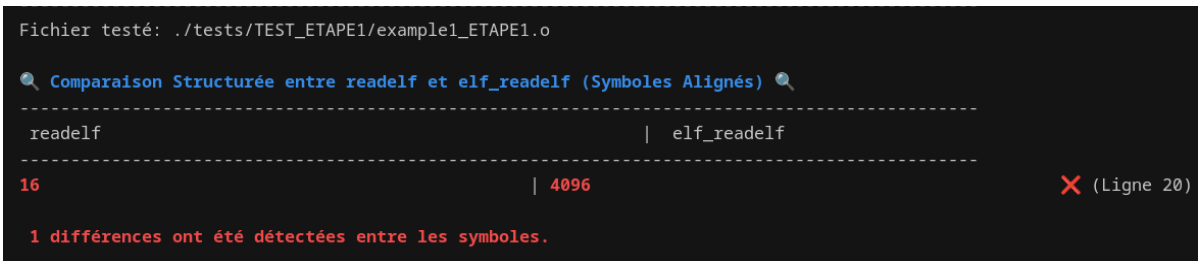
Pour les étapes 8 à 10 , nous testons manuellement les résultats , en comparant le code assembleur et le déroulement du code qu'on a lancé dans le simulateur, dont on se convainc par les valeurs que prennent les différents registres qui sont censés correspondre avec le code source assembleur ARM.

B. Résultats des tests :

Première capture : modèle de test qui marche.
Deuxième capture : modèle de test qui échoue.

ETAPE 1 :





ETAPE 2 :

```
Comparaison Structurée entre readelf et elf_readelf (Différences en rouge avec Éléments en Vert)
--------------------------------------------------------------------------------------------
readelf                                                 | elf_readelf
--------------------------------------------------------------------------------------------
[11] .rel.debug_a[...] REL 00000000 000318 000010 08 I 14 10 4  | [11] .rel.debug_aranges REL 00000000 000318 000010 08 I 14 10 4
[13] .ARM.attributes ARM_ATTRIBUTES 00000000 0001c4 00001f 00 0 0 1  | [13] .ARM.attributes UNKNOWN 00000000 0001c4 00001f 00 0 0 1

2 différences ont été détectées entre readelf et elf_readelf dans ./tests/TEST_ETAPE2/example1_ETAPE2.o.
```

```
--------------------------------------------------------------------------------------------
readelf                                                 | elf_readelf
--------------------------------------------------------------------------------------------
[ 1] .text PROGBITS 00000000 000034 000000 00 AX 0 0 1  | [ 1] .text UNKNOWN 00000000 000034 000000 00 AX 0 0 1
[ 2] .data PROGBITS 00000000 000034 000000 00 WA 0 0 1  | [ 2] .data UNKNOWN 00000000 000034 000000 00 WA 0 0 1
[ 3] .bss NOBITS 00000000 000034 000000 00 WA 0 0 1  | [ 3] .bss UNKNOWN 00000000 000034 000000 00 WA 0 0 1
[ 4] .empty PROGBITS 00000000 000034 000000 00 0 0 1  | [ 4] .empty UNKNOWN 00000000 000034 000000 00 0 0 1
[ 5] .ARM.attributes ARM_ATTRIBUTES 00000000 000034 00001f 00 0 0 1  | [ 5] .ARM.attributes UNKNOWN 00000000 000034 00001f 00 0 0 1
[ 6] .symtab SYMTAB 00000000 000054 000070 10 7 6 4  | [ 6] .symtab UNKNOWN 00000000 000054 000070 10 7 6 4
[ 7] .strtab STRTAB 00000000 0000c4 00000f 00 0 0 1  | [ 7] .strtab UNKNOWN 00000000 0000c4 00000f 00 0 0 1
[ 8] .shstrtab STRTAB 00000000 0000d3 000043 00 0 0 1  | [ 8] .shstrtab UNKNOWN 00000000 0000d3 000043 00 0 0 1

8 différences ont été détectées entre readelf et elf_readelf dans ./tests/TEST_ETAPE2/example7_ETAPE2.o.
```

ETAPE 3 :

```
🔍 Comparaison de la section 9 entre readelf et elf_readelf
--------------------------------------------------------------------------------------------
 elf_readelf                                            |  readelf
--------------------------------------------------------------------------------------------
                                                        |                                                                  ✅
0x00000000 00000064 000000c8 0000012c 00000190 ...d.......,.... | 0x00000000 00000064 000000c8 0000012c 00000190 ...d.......,.... ✅
                                                        |                                                                  ✅

Les données de la section 9 sont parfaitement identiques.
```

```
🔍 Comparaison de la section .strtab entre readelf et elf_readelf
--------------------------------------------------------------------------------------------
 readelf                                                |  elf_readelf
--------------------------------------------------------------------------------------------
                                                        |                                                                  ✅
0x00000000 00246100 5f737461 727400 .$a._start.         |                                                                  ❌
                                                        |                                                                  ✅

1 différences détectées dans la section .strtab.
```

ETAPE 4 :

```
🔍 Test sur le fichier : ./tests/TEST_ETAPE4/example7_ETAPE4.o 🔍
0 00000000 0 NOTYPE LOCAL DEFAULT UND  | 0 00000000 0 NOTYPE LOCAL DEFAULT 0  ✔
1 00000000 0 SECTION LOCAL DEFAULT 1 .text  | 1 00000000 0 SECTION LOCAL DEFAULT 1 .text  ✔
2 00000000 0 SECTION LOCAL DEFAULT 2 .data  | 2 00000000 0 SECTION LOCAL DEFAULT 2 .data  ✔
3 00000000 0 SECTION LOCAL DEFAULT 3 .bss  | 3 00000000 0 SECTION LOCAL DEFAULT 3 .bss  ✔
4 00000000 0 SECTION LOCAL DEFAULT 4 .custom  | 4 00000000 0 SECTION LOCAL DEFAULT 4 .custom  ✔
5 00000000 0 NOTYPE LOCAL DEFAULT 4 $d  | 5 00000000 0 NOTYPE LOCAL DEFAULT 4 $d  ✔
6 00000000 0 SECTION LOCAL DEFAULT 5 .ARM.attributes  | 6 00000000 0 SECTION LOCAL DEFAULT 5 .ARM.attributes  ✔
7 00000000 0 NOTYPE GLOBAL DEFAULT 4 custom_section  | 7 00000000 0 NOTYPE GLOBAL DEFAULT 4 custom_section  ✔
Test réussi : Les symboles readelf et elf_readelf sont parfaitement identiques.
```

```
🔍 Test sur le fichier : ./tests/TEST_ETAPE4/example7_ETAPE4.o 🔍
0 00000000 0 NOTYPE LOCAL DEFAULT UND  | 0 00000000 0 NOTYPE LOCAL DEFAULT 0  ✔
1 00000000 0 SECTION LOCAL DEFAULT 1 .text  | 1 00000000 0 SECTION LOCAL DEFAULT 1 .text  ✔
2 00000000 0 SECTION LOCAL DEFAULT 2 .data  | 2 00000000 0 SECTION LOCAL DEFAULT 2 .data  ✔
3 00000000 0 SECTION LOCAL DEFAULT 3 .bss  | 3 00000000 0 SECTION LOCAL DEFAULT 3 .bss  ✔
4 00000000 0 SECTION LOCAL DEFAULT 4 .custom  | 4 00000000 0 SECTION LOCAL DEFAULT 4 .custom  ✔
5 00000000 0 NOTYPE LOCAL DEFAULT 4 $d  | 5 00000000 0 NOTYPE LOCAL DEFAULT 4 <invalid>  ❌
6 00000000 0 SECTION LOCAL DEFAULT 5 .ARM.attributes  | 6 00000000 0 SECTION LOCAL DEFAULT 5 .ARM.attributes  ✔
7 00000000 0 NOTYPE GLOBAL DEFAULT 4 custom_section  | 7 00000000 0 NOTYPE GLOBAL DEFAULT 4 <invalid>  ❌
2 différences numériques détectées dans ./tests/TEST_ETAPE4/example7_ETAPE4.o.
```

ETAPE 5 :

```
Comparaison pour le fichier ELF : ./tests/TEST_ETAPE5/example7_ETAPE5.o

Comparaison Structurée entre readelf et elf_readelf (Avec Mots-Clés Sym.Value et Index en Vert)
-----------------------------------------------------------------------------------------------
readelf                                                    | elf_readelf
-----------------------------------------------------------------------------------------------
Offset Info Type Sym.Value  | Offset Info Type Index
00000022 00000902 R_ARM_ABS32 00000000  | 00000022 00000902 R_ARM_ABS32 00000009
00000026 00000902 R_ARM_ABS32 00000000  | 00000026 00000902 R_ARM_ABS32 00000009
00000030 00000902 R_ARM_ABS32 00000000  | 00000030 00000902 R_ARM_ABS32 00000009
00000035 00000902 R_ARM_ABS32 00000000  | 00000035 00000902 R_ARM_ABS32 00000009
0000003d 00000102 R_ARM_ABS32 00000000  | 0000003d 00000102 R_ARM_ABS32 00000001
Offset Info Type Sym.Value  | Offset Info Type Index
00000008 00000702 R_ARM_ABS32 00000000  | 00000008 00000702 R_ARM_ABS32 00000007
0000000d 00000802 R_ARM_ABS32 00000000  | 0000000d 00000802 R_ARM_ABS32 00000008
00000011 00000102 R_ARM_ABS32 00000000  | 00000011 00000102 R_ARM_ABS32 00000001
00000016 00000a02 R_ARM_ABS32 00000000  | 00000016 00000a02 R_ARM_ABS32 0000000a
0000001a 00000a02 R_ARM_ABS32 00000000  | 0000001a 00000a02 R_ARM_ABS32 0000000a
0000001e 00000a02 R_ARM_ABS32 00000000  | 0000001e 00000a02 R_ARM_ABS32 0000000a
Offset Info Type Sym.Value  | Offset Info Type Index
00000006 00000602 R_ARM_ABS32 00000000  | 00000006 00000602 R_ARM_ABS32 00000006
00000010 00000102 R_ARM_ABS32 00000000  | 00000010 00000102 R_ARM_ABS32 00000001
```

```
Comparaison Structurée entre readelf et elf_readelf (Avec Mots-Clés Sym.Value et Index en Vert)
-----------------------------------------------------------------------------------------------
readelf                                                    | elf_readelf
-----------------------------------------------------------------------------------------------
Offset Info Type Sym.Value  | Offset Info Type Index
00000022 00000902 R_ARM_ABS32 00000000  | 22000000 02090000 Unknown 00020900
00000026 00000902 R_ARM_ABS32 00000000  | 26000000 02090000 Unknown 00020900
00000030 00000902 R_ARM_ABS32 00000000  | 30000000 02090000 Unknown 00020900
00000035 00000902 R_ARM_ABS32 00000000  | 35000000 02090000 Unknown 00020900
0000003d 00000102 R_ARM_ABS32 00000000  | 3d000000 02010000 Unknown 00020100
Offset Info Type Sym.Value  | Offset Info Type Index
00000008 00000702 R_ARM_ABS32 00000000  | 08000000 02070000 Unknown 00020700
0000000d 00000802 R_ARM_ABS32 00000000  | 0d000000 02080000 Unknown 00020800
00000011 00000102 R_ARM_ABS32 00000000  | 11000000 02010000 Unknown 00020100
00000016 00000a02 R_ARM_ABS32 00000000  | 16000000 020a0000 Unknown 00020a00
0000001a 00000a02 R_ARM_ABS32 00000000  | 1a000000 020a0000 Unknown 00020a00
0000001e 00000a02 R_ARM_ABS32 00000000  | 1e000000 020a0000 Unknown 00020a00
Offset Info Type Sym.Value  | Offset Info Type Index
00000006 00000602 R_ARM_ABS32 00000000  | 06000000 02060000 Unknown 00020600
00000010 00000102 R_ARM_ABS32 00000000  | 10000000 02010000 Unknown 00020100

97 différences numériques ont été détectées entre les tableaux.
```

ETAPE 6 :

```
Vérification de ./tests/TEST_ETAPE6/example6_ETAPE6.o_modified ...
Le fichier ./tests/TEST_ETAPE6/example6_ETAPE6.o_modified ne contient pas de section .REL. ✅
```

```
Vérification de ./tests/TEST_ETAPE6/example6_ETAPE6.o ...
Le fichier ./tests/TEST_ETAPE6/example6_ETAPE6.o contient une section .REL. ❌
```

ETAPE 7 :

```
=== Vérification (ind. du modifié) : valeur_orig + addr_section_mod = valeur_mod ? ===
[OK] $a => 00000000 + 0000005c = 0000005c
[OK] data_var => 00000000 + 0000006c = 0000006c
[OK] bss_var => 00000000 + 00000070 = 00000070
[OK] $d => 00000000 + 00000070 = 00000070
[OK] func1 => 00000000 + 0000005c = 0000005c
[OK] func2 => 00000008 + 0000005c = 00000064

✅ Aucune incohérence détectée pour example2_ETAPE7.o.
```

```
=== Vérification (ind. du modifié) : valeur_orig + addr_section_mod = valeur_mod ? ===
[OK] $a => 00000000 + 0000005c = 0000005c
[OK] $a => 00000000 + 0000005c = 0000005c
[OK] data_var => 00000000 + 00000064 = 00000064
[OK] bss_var => 00000000 + 00000068 = 00000068
[OK] $d => 00000000 + 00000068 = 00000068
[OK] func1 => 00000000 + 0000005c = 0000005c
[ERREUR] func2 => 00000000 + 00000070 != 00000000

❌ 1 incohérence(s) détectée(s) pour example2_ETAPE7.o.
```

VI. Conclusion :

En conclusion, ce projet a réussi son objectif qui est de développer un logiciel de réimplantation d'un fichier binaire translatable au format ELF, afin de permettre l'obtention d'un fichier exécutable. Nous avons travaillé en pour mener à bien cette tâche, en mettant à profit nos compétences en programmation bas niveau et en analyse de la documentation technique.

Bien que certaines étapes aient présenté des défis, notamment dans la compréhension de leurs énoncés, nous avons néanmoins réussi à accomplir la majorité des fonctionnalités attendues. Des tests automatisés ont été mis en place pour valider nos résultats, et plusieurs pistes d'amélioration ont été identifiées pour renforcer la qualité du produit final.

Globalement, nous sommes satisfaits des résultats obtenus et fiers du travail accompli. Ce projet nous aura permis de développer des compétences techniques précieuses, que nous estimons essentielles pour notre avenir professionnel.