PROG 5 Projet 2021-2022

Documentation des Testes Effectués

Kimberly Beauvais, Xuan Li, Nathaniel Tobing
Hugo Roger, Emilien Maillard-Simon, Théo Lanneau

13 janvier 2022

Table des Matières

1 Phase 1

Le but des tests est d'exécuter avec différentes options la fonction créée Readelf sur un jeu significatif de programmes ARM compilés, puis de comparer les résultats à ceux obtenus par l'exécution de la commande Readelf existante.

1.1 Affichage de l'en-tête

On teste d'abord l'affichage de l'entête des fichiers ELF avec un script qui nettoie les sorties des deux fonctions puis les compare. Si les 2 sorties sont identiques, le test est validé.

1.2 Affichage de la table des sections

On teste ensuite l'affichage de l'entête des sections des fichiers ELF.

1.3 Affichage du contenu d'une section

Ici on veut tester l'affichage des hex-dumps des différentes sections des fichiers ELF. Pour chaque fichier ELF on va chercher l'ensemble des noms de ses différentes sections via un nettoyage de la sortie de la fonction Readelf -h. Puis pour chaque section on compare les hex-dumps des deux fonctions. Si une section est vide elle est filtrée et donc pas testée.

1.4 Affichage de la table des symboles

On compare ensuite les deux tables des symboles obtenues à partir des deux fonctions.

1.5 Affichage des tables de réimplantation

Finalement on compare les deux tables de réadressage obtenues à partir des deux fonctions.

2 Phase 2

Dans cette partie, les tests sont faits manuellement. Il s'agit de comparer la sortie faite par la commande

```
./relocation .text=<nombre> .data=<nombre>
  Examples_loader/<fichier.o>
```

et celle avec l'utilisation de la commande

```
readelf -hSs Examples_loader/<fichier.o>
```

2.1 Rénumerotation des sections

Il faut vérifier que les sections du type REL ont bien été supprimées et les sections sont bien renumérotées et aussi que l'adresse des sections .text et .data sont changées.

2.2 Correction des symboles

Il faut verifier que les valeurs et les ndx des symboles dans la table des symboles ont bien été changés et que les ndx pointe vers la même section qu'avant

2.3 Réimplantation de type ARM

Il faut lire le contenu des sections affectées par la réimplantation (readelf -x) et vérifier que les valeurs sont bien changées.(pour vérifier l'écriture du nouveau fichier binaire) Il faut faire

```
readelf -hSs out.bin
```

et vérifier que l'affichage est le même que la sortie de la commande ./relocation

3 Différent Testes

3.1 Tests Automatique

Ces tests sont effectués automatiquement en lançant le scripte

```
./tests\_part1.sh
```

example5.s : Ce test cherche à vérifier si le programme fonctionne en chargeant différentes données dans le même registre.

example6.s : Ce test cherche à vérifier si le programme fonctionne en ignorant une liste d'instructions

example7.s : Ce test cherche à vérifier si le programme fonctionne en n'ayant aucune instruction

example8.s : Ce test cherche à vérifier si le programme fonctionne en ayant un .text vide et un .data rempli

example9.s : Ce test cherche à vérifier si le programme fonctionne en ayant plusieurs balises inutilisées

example10.s : Ce test cherche à vérifier si le programme fonctionne en utilisant une pile

example11.s : Ce test cherche à vérifier si le programme fonctionne en ayant une boucle avec des add et mov

example12.s : Ce test cherche à vérifier si le programme fonctionne en ayant plusieurs données de différents types

example13.s : Ce test cherche à vérifier si le programme fonctionne en utilisant des opérations logiques

example14.s : Ce test cherche à vérifier si le programme fonctionne en utilisant les différents flags pour des branchements

example15.s : Ce test cherche à vérifier si le programme fonctionne en ajoutant plusieurs directives différentes

3.2 Tests Manuels

Tests manuels pour la partie 2
Test simple

```
./relocation .text=0x20 .data=0x2800 Examples\_loader/example1.
o
```

Test avec adresses nulles (donc aucun changement) :

```
./relocation .text=0 .data=0 Examples\_loader/example1.o
```

Test avec un programme sans section .data dans le fichier .s d'origine (aucune différence, car la section .data est quand même créée dans le .o lors de la compilation)

```
./relocation .text=0x20 .data=0x2800 Examples\_loader/example2.o
```

Tests avec adresses données en argument étant des nombres négatifs : Avec ce programme, on interdit les adresses négatives. Au niveau du code, normalement cela fonctionne mais les numéros d'adresses deviennent très grands car on a implémenté avec un type unsigned.

- test avec section .text a une nouvelle adresse étant un nombre négatif

```
./relocation .text=-0x34 .data=0x2800 Examples\_loader/example1.o
```

- test avec section .data a une nouvelle adresse étant un nombre négatif

```
./relocation .text=0 .data=-0x11 Examples\_loader/example1.o
```

- test avec fichier exécutable (donne un message d'erreur, seuls les fichiers avec sections de relocation sont permis) :

```
./relocation .text=0 .data=-0x11 Examples\_loader/example10
```