12/12/2019:

Familiarisation et compréhension du le projet. Planification de la répartition du projet.

13/12/2019:

Nous recherchons comment utiliser Gitlab à la place de Github. Après plusieurs heures on décide d'utiliser Github pour des raisons de facilités. Un TP expliquant comme faire fonctionner Github est disponible sur Moodle.

Premier problème rencontré, la compilation est plutôt étrange et n'a pas l'air de fonctionner. Pour régler la compilation nous avons du changer 'am__api_version=1.14' pour 'am__api_version=1.15' dans configure du dossier principal et 'am__api_version=1.16' pour 'am__api_version=1.15' dans configure du dossier Examples_loader

Pour compiler il est décidé d'utiliser `clang -c test.c && clang -o test test.o` avec test étant le fichier à compiler.

La étape 1 est partiellement finie.

14/12/2019

Jour de documentation

15/12/2019

Recherche de lien pouvant être utile :
https://stackoverflow.com/questions/29052125/reading-the-contents-of-an-elf-sectionprogrammatically

16/12/2019

Alors, le 1.3 est normalement finie. Le 1.2 ne devrait pas tarder à être fini. Il est important de noter que nous avons tous stuck sur la recherche du nom des sections et que Martin, a résolu le problème.

1.5 : pas compris de comment on passe du type de relocation qui est donné comme un entier à genre RX86_64_32. "La spéc du type dépend de l'architecture matérielle" mais je trouve pas les spécifications en question.

17/12/2019

1.2 et 1.5 fonctionnelles et fusionnées sur Master. 1.3 semi-fonctionnelle, nous avons un problème dessus : nous regardons toute la section et un peu plus loin. Cette dernière ne sera pas fusionnée avec Master aujourd'hui.

18/12/2019

1.3 est réparée, nous ne regardions pas si nous avons dépassés la taille lors de l'affichage d'une ligne. Elle est fusionnée à Master.

Nouveau problème trouvé sur la 1.2, l'affichage de section était limité aux 20 premières pour ne pas surcharger le terminal. Nous réglons ce problème et nous mettons ça sur le git.

Merge de la 1.4 ; Anthony a commencé un regroupement des fonctions de byteshift ou du genre

19/12/2019

Nous faisons un point, histoire d'être tous sur la même longueur d'onde. Après lecture et compréhension de la seconde phase, nous décidons de nous répartir le travail. Afin de gagner du temps nous choisissons de faire trois équipes de deux. Une sur l'étape 6, une sur l'étape 7 et une dernière sur l'étape 8.

20/12/2019

Début de la seconde phase. On est tous un peu perdu pour le coup...

Passage à l'oral

Il a été retenu de mettre au propre le code en le commentant, en supprimant toutes nos fonctions byteshift et changer l'ordre des octets au début de la lecture, de regrouper les fonctions utilisées à droite à gauche pour les mettres dans un seul fichier. De plus nous devons automatiser les tests afin de gagner du temps de travail et de comparaisons. Pour faciliter l'utilisation, nous devons avoir un seul main et importer chaques fichiers dedans.

21/12/2019

Anthony a fait la séparation des mains, le makefile qui va avec et un script permettant d'automatiser les tests.

Reprise le 28/12/2019

Durant cette semaine de repos, Anthony a effectué tout ce qui a été décidé le dernier jours. Utilisation de macro, utilisation de byteshift et tout autre.

29/12/2019

Mise au propre de l'étape 1

06/01/2019

Avancée de l'étape 2.6 avec des ajouts de commentaires pour une meilleure compréhension. Correction de la librairie de gestion du big endian. Ajout d'une fonction de lecture pour lire et inverser les headers/sections/rel/etc.

Correction de bug divers, il est impossible de créer un fichier à partire de '../Exemple_loarder/etc'

07/01/2019

Remise à niveau du travail d'Anthony ; Recherche de solutions qui n'ont menées à rien, début de fonctionnement de l'étape 2.6 grâce à Martin

08/01/2019

Essai d'intégrer le code du 2.7 au reste. Correction 1.5, 1.4 et amélioration de la lisibilité du code de étape 2.6. Implementation d'un script de tests