PROG5 - Projet logiciel 2024-2025

Nouvelles et annonces importantes

Au cours du projet, cette section de la page web sera mise à jour avec les informations et les précisions arrivant après le démarrage. Pensez à la consulter régulièrement (au moins une fois par jour).

Description du projet

De manière générale, le projet a plusieurs objectifs :

- mettre en pratique les compétences de programmation acquises pendant le semestre ;
- réaliser un logiciel de taille moyenne et associé à une problématique réaliste ;
- apprendre à appréhender des documentations techniques volumineuses ;
- apprendre à gérer le travail au sein d'un groupe de taille moyenne.

Cette année, le projet porte sur le développement d'un éditeur de liens. Les détails se trouvent dans l'énoncé et le code fourni :

- présentation du projet 2024-2025
- sujet du projet 2024-2025
- code source fourni
- simulateur ARM big endian compilé pour Linux/x86

Documentation technique

- Spécification du format ELF
- Compléments de spécification ELF relatifs à l'architecture ARM
- Manuel de référence ARMv7

Organisation

- Présence des enseignants : Au moins un passage par matinée. Un enseignant viendra à l'UFR durant le créneau 9h45-11h15 et fera le tour des salles réservées au projet. En l'absence de questions, l'enseignant repartira.
- Amphi d'explications et discussions sur le projet, lundi 16 décembre 2024 en F022.
- Audits de code, jeudi 19 décembre 2024, pousser votre code la veille sur le dépôt fourni, planning à venir...
- Rendu final, jeudi 09 janvier 2024 avant 12h, pousser sur le dépôt fourni :
  - la version du projet que vous souhaitez rendre dans le commit le plus récent de la branche `master`
  - l'intégralité du code doit compiler
  - les documents demandés :
    - Bref mode d'emploi expliquant comment compiler et lancer chacun des programmes utilitaires développés dans le cadre du projet
    - Descriptif de la structure du code développé : principales fonctions et fichiers correspondants (inutile de décrire le code fourni par les enseignants, sauf en cas de

modifications importantes)
    - Liste des fonctionnalités implémentées et manquantes
    - Liste des éventuels bogues connus mais non résolus
    - Liste et description des tests effectués. Pour chacun des tests, la description correspondante doit permettre de connaître :
        - son objectif (ce qu'il cherche à vérifier)
        - la marche à suivre pour le lancer
        - la façon de conclure sur le résultat observé (réussite ou échec du test), si le test n'est pas automatisé
    - Journal décrivant la progression du travail et la répartition des tâches au sein du groupe (à remplir quotidiennement).
- Soutenances : vendredi 10 janvier 2024, planning à venir...

Informations pratiques

- Salles réservées pour le projet : consulter ADE
- Serveurs utilisables pour le projet : le serveur de référence est mandelbrot, tous les projets doivent fonctionner sur ce serveur. Le code fourni devrait fonctionner correctement sur les autres machines de l'UFR.

Foire aux questions

- **Je suis redoublant et je souhaite conserver ma note de l'an passé.**
  Cela n'est plus possible. Depuis 2020-2021, la note de projet est comme toutes les autres notes, elle n'est pas conservée l'année suivante si l'UE n'est pas validée.
- **Problèmes liés à l'infrastructure de compilation automake/autotools**
  Vous rencontrerez peut-être des problèmes liés à l'infrastructure de compilation fournie, en raison de différences de versions entre les outils installés sur différentes machines. La séquence de commande suivantes devrait vous permettre de régler ces problèmes :
  `make distclean`
  `autoreconf`
  et éventuellement
  `autoreconf Exemples_loader`
- **Comment éditer/visualiser le contenu "brut" d'un fichier ?**
  Pour la visualisation, il existe des utilitaires tels que `hexdump` ou `od`. Pour l'édition, il existe des modes spéciaux dans `vim` (via la commande `:%!xxd`) et `emacs` (mode `hexl-mode`); il existe également des éditeurs graphiques (par exemple Ghex sous Gnome).
- **Le projet sur ma ubuntu/debian**
  La principale chose pouvant manquer est le compilateur croisé. Le paquet `gcc-arm-none-eabi` et les paquets dont il dépend devrait faire l'affaire. Attention, évitez le paquet `gcc-arm-linux-gnueabi` qui ne compile pas avec le code source fourni. En effet, il impose l'existence d'une table de segments correcte pour le chargement alors que le code fourni n'en crée aucune. Cette table de segments fait l'objet de l'étape 11 et vous pourrez passer au paquet `gcc-arm-linux-gnueabi` si vous arrivez jusque là.
- **flags de section non documentés**
  Les standards pouvant évoluer, il est possible que certains flags de section ou autre ne soient pas documentés dans les documentations techniques fournies. De manière générale, vous pouvez chercher une version plus à jour de ces documentations - qui n'existe peut être pas encore - mais vous n'êtes pas tenus de le faire. En particulier, l'exercice demandé dans ce projet concerne une partie ancienne et très stable de la spécification qui n'est pas amenée à changer.
- **R_ARM_V4BX**
  Ce type de réimplantation est uniquement lié à la compatibilité du code : il indique les emplacements contenant des instructions `bx` (instruction recommandée pour le retour de fonction) non supportées en ARMv4. Cela permet à l'éditeur de liens classique de les remplacer par des `mov` (qui fonctionnent

également mais ne sont pas recommandée pour des questions de style). Depuis les versions les plus récentes, `gcc` refuse de générer du code sans `bx` et laisse à l'éditeur de liens le soin de gérer la compatibilité ascendante. En ce qui nous concerne, il faudra donc tenir compte de ces réimplantations qui ne demandent quasiment aucun travail : seul leur offset devra être corrigé.

Enseignants :

Pour contacter l'équipe pédagogique par courriel, utilisez systématiquement l'adresse de la hotline: im2ag-l3-info-hotline@univ-grenoble-alpes.fr

- Guillaume Huard (responsable)
- Vincent Danjean
- Philippe Waille
- Maxime de Sousa
- Fateh Boulmaiz
- Cyril Labbé
