Analyse de processus en cours d'exécution

Description

L'objectif de ce projet est de concevoir un outil permettant de comprendre pourquoi un processus ne réalise pas la tâche pour laquelle il a été programmé. L'idée est de se rapprocher au maximum des fonctionnalités offertes par un débuggeur traditionnel, au moyen des connaissances vues en classe. Vous avez *presque* toute liberté pour réaliser cette implémentation. Si vous manquez d'inspiration, imaginez que vous ne disposez pas des outils qui existent aujourd'hui et que vous avez un programme erroné (crash, deadlock, lancement impossible…).

De quoi avez-vous besoin pour trouver la raison du problème ?

Pistes de réflexion

Votre outil pourrait d'abord un analyseur de programme en fournissant **par exemple** les informations suivantes:

- informations de bases du processus (PID, PPID, GID, localisation du programme, etc.)
- l'état de sa mémoire
- Les bibliothèques chargées
- les variables globales disponibles
- la liste complète des fonctions disponibles
- …*Toute information qui vous permet finalement de "caractériser" un processus*

Ensuite, il faudra réfléchir aux fonctionnalités qui permettent d'aider un processus erroné, **par exemple**:

- La pile d'appel de fonctions lors du crash
- Le signal ayant mené au crash
- l'adresse / la raison du crash
- …

Il est permis (et même encouragé), d'ajouter autant de fonctionnalités qu'il vous semblerait nécessaire à vous pour débugger un programme. Les fonctionnalités non listées ici seront grandement appréciées et jugées au cas par cas:

- Liste des arguments (et leur valeur) des fonctions de la pile d'appel au moment de l'erreur
- Liste des variables automatiques de la fonction courante
- Affichage du code source (voir format "DWARF")
- Pistes de correction d'erreurs
- …

Rendu & Evaluation

Il existe différentes approches pour réaliser cette tâche, il est entendu que vous devrez développer vous-même l'outil réalisant cette analyse. Cependant, vous êtes autorisés à reposer sur des tierces dépendances (bibliothèques) pour vous **aider**. Le projet sera écrit en C/C++ (car beaucoup d'intéractions avec la libc), tout autre choix devra être fortement justifié. De plus il vous appartient de vous assurer que le programme se compile selon les règles "classiques" (Makefile, CMake, Autotools ou équivalent). La qualité de la gestion logicielle (structure, propreté, documentation, compilation…) sera aussi évaluée.

Il existe globalement deux méthodes que vous êtes libres d'explorer à votre convenance:

- Greffon interne: on vient injecter un bout de code qui vient s'éxécuter dans le programme (hint: utilisation de *signaux*)
- Processus externe: un processus parent lance le processus erroné et réalise son analyse de l'extérieur avec les outils adéquats (hint: fonction *ptrace*)

Rendu avant le Dimanche 20 mars 2021 avant 23:59:59

Ce projet est à faire seul ou en binôme.

- le code source **intégral** documenté.
- un document technique expliquant votre approche et les fonctionnalités que vous avez implémentées. Ceci n'est pas un rapport détaillé de tout votre travail, il n'a pas besoin d'être long.

Tout effort de rendu sera pris en compte, il est clairement plus simple de rendre un code versionné, en ligne (dépôt Github), qu'une archive par mail qui contient encore toute les fichiers objets & binaires. Merci de faire un effort !

> Notre principale attente est liée à votre code et sa démo, c'est pour cela que nous insistons sur la taille du rapport qui ne devra être qu'un résumé succint de votre travail.

Soutenance le Lundi 28 mars 2021

Votre programme sera lancé sur différents programmes précompilés (des échantillons vous seront donnés pour vos travaux) afin d'observer les différentes informations retournées et leur niveau de cohérence par rapport au problème effectif. Vous aurez pour mission de faire:

- une démo du programme sur des codes fournis en dernière minute le jour de la soutenance;
- une présentation des fonctionnalités qui n'auraient pas été couvertes (pas de slides, que la démo + questions/réponses).