EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

Programmation Système Unix Avancée

ftrace

Benjamin Gaillard benjamin.gaillard@epitech.eu

Abstract: ftrace est un traceur d'appels de fonction pour Unix.

Table des matières

Chapitre I

Introduction

I.1 Description

Ftrace permet de générer un graphe contenant le *call-graph* d'un programme existant ou lancé par *!*ftrace*!*. Ce *call-graph* devra contenir ces différentes informations :

- appels système ;
- appels de fonctions internes au programme avec leur nom et leur adresse ;
- récupération des signaux envoyés aux autres programmes ;
- appels de fonctions contenues dans les bibliothèques partagées (.so) ;
- toute autre information intéressante.

Toutes ces informations devront être distinguables les unes des autres dans le graphe généré. Le processus à inspecter pourra être précisé via son PID (option `-p`) dans le cas d'un processus déjà lancé ou via une commande et ses arguments ; dans ce dernier cas, c'est à vous de créer le processus.

Suivant les éléments dont vous disposez, l'affichage peut se restreindre, par exemple si l'exécutable que vous invoquez ne dispose pas de table des symboles. Vous devez tout de même suivre les appels de fonctions et en afficher une description. Par exemple : `func_0x8765FDE0@a.out`.

I.2 Usage

```
./ftrace <commande>
./ftrace -p <pid>
```

Chapitre II

Consignes

II.1 Rendu

- Les sources doivent être rendues dans le répertoire `AUSP_ftrace`.
- L'intégralité de votre projet devra compiler avec un unique *Makefile* comportant (au moins) les règles `all`, `clean`, `fclean` et `re`.
- L'exécutable sera nommé `ftrace`.
- Ce projet est à réaliser en groupe de 2 à 4 personnes.
- Votre rendu devra contenir un fichier `auteur` avec les logins de chaque membre du groupe, séparés par un « ; ».

II.2 Autres

- Vous devrez faire ce projet au moins sur *x86-64/Linux*.
- Bibliothèques autorisées : libc, libelf, libm.
- Langage autorisé : C.

Chapitre III

Idées de bonus

- Afficher le code des fonctions appelées quand les symboles de débogage le permettent (niveaux de verbosité).
- Décoloration (*unmangling*) des noms de fonction.
- Décodage explicite des adresses de `CALL`.
- Fonctionner également en 32-bit (*x86*).
- Être portable sur différentes architectures matérielles (ARM, PowerPC, SPARC, etc.).
- Être portable sur différents systèmes.
- Un *call-graph* sexy en 3D.
- Lorsque le programme envoie un signal à un autre programme, vous pouvez commencer à tracer l'autre programme.