Programmation Système Unix

strace

Contents

Description

Strace permet de tracer un programme en temps réel et d'afficher tous les appels système faits par un programme dans leur ordre d'apparition.
Vous devez développer une alternative à strace, supportant les options suivantes :

- -p vous permettra d'utiliser un PID spécifique au lieu d'executer une commande
- -s affichera les arguments détaillés (voir ci-dessous)

Par défaut, votre strace n'affichera les arguments et les valeurs de retour que sous forme hexadécimale.

Si l'appel système a pour type de retour void, vous afficherez un point d'interrogation '?'

Si vous utilisez l'option -s, votre programme devra se rapprocher autant que possible du véritable strace. Ainsi, il affichera :

- les entiers sous forme décimale
- les pointeurs sur chaine de caractères sous forme de chaine de caractères
- les structures de façon détaillées (valeur pour chaque champ de celle-ci)

L'affichage avec l'option -s devra être aussi proche que possible de celui proposé par la commande strace telle que présente sur votre système.

```
1  Usage : ./strace [-s] [-p <pid>|<command>]
```

Voici un exemple de sortie sans option -s pour un appel système

```
1 write(0x1, 0x7ef23a43, 0x4) = 0x4
```

Rendu

- Les sources doivent être rendues dans le répertoire `PSU_2015_strace`.
- 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 `strace`.
- Ce projet est à réaliser en seul ou en binôme.

Autres

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

`Vous n'avez pas le droit d'utiliser PTRACE_SYSCALL.`

Bonus

- Gérer les flags
- Fonctionner également en 32-bit (*x86*).
- Être portable sur différentes architectures matérielles (ARM, PowerPC, SPARC, etc.).
- Être portable sur différents systèmes.