

# PARTIE 1

## ARCHITECTURE DE BASE

### 1.1 L'objectif du projet

Notre objectif est de réaliser un outils qui permet de démarrer un processus et le déboguer, ou s'attacher à un processus existant. Et parcourir le code en une seule étape, définir des points d'arrêt et les exécuter, examiner les valeurs des variables et empiler les traces.

### 1.2 l'architecture de base du programme

#### 1.2.1 Ptrace

Ce projet est principalement basé sur l'appel système appelé ptrace. Il s'agit d'un appel système unique avec une multitude de types de requêtes différents. Celui-ci permet au traceur (*debugger ou tracer*) de suivre un observé (*programme à déboguer ou tracee*). Ainsi, le traceur peut changer la mémoire de l'observé, changer ses registres, tracer ses appels système ou mettre des points d'arrêt.

les requêtes de **ptrace** utilisés :

- **PTRACE\_CONT** : Requête qui demande que l'inférieur continue l'exécution.
- **PTRACE\_TRACEME** : Processus enfant demande au noyau du système d'exploitation de laisser son parent le tracer.
- **PTRACE\_GETSIGINFO** : Récupérer des renseignements sur le signal qui a provoqué l'arrêt.
- **PTRACE\_PEEKTEXT** : Lire un mot à l'adresse addr dans la mémoire du tracee, renvoyant le mot comme résultat de l'appel à ptrace ().
- **PTRACE\_GETREGS** : Copier les registres généraux ou du processeur en virgule flottante du tracee, vers l'adresse data du tracer.

## 1.3 L'attente du tracee

### 1.3.1 waitpid

Nous avons utilisé l'appel système `waitpid` pour notifié le tracer sur l'état du *tracee*, Nous passons l'identifiant du processus que nous voulons attendre à travers le `pid`. *status* est un paramètre de sortie qui encode la raison pour laquelle le trace s'est arrêté. Pour décoder la raison nous utilisons des macros pour surveiller nos tracee.

## 1.4 Backtrace

Un backtrace est un suivi des fonctions appelées avant la fonction courante à travers la pile d'appels. Ainsi, pour obtenir un backtrace, il faut parcourir la pile jusqu'à ce que `ptrace` renvoie -1.

L'appel *backtrace()* remplit un tableau avec le compteur de programme de chaque fonction appelante, et l'appel séparé *backtrace\_symbols()* peut rechercher les noms symboliques pour chaque adresse. La sortie affiche la trace avec l'adresse de chaque site d'appel de fonction, et nous avons ajouté l'option `-rdynamic` pour obtenir des noms symboliques utiles.

## BIBLIOGRAPHIE

- [1] "Ptrace manuel Linux", <https://man7.org/linux/man-pages/man2/ptrace.2.html>
- [2] "Playing with ptrace", <https://www.linuxjournal.com/article/6100?page=0,11>
- [3] "gcc backtrace support", <https://codingrelic.geekhold.com/2009/05/pre-mortem-backtracing.html>