Debugger un kernel dans QEMU avec Kdbg

v0.1

Compilation

Pour debugger un fichier ELF, celui-ci doit impérativement être compilé **et** lié avec l'option `-g` :

- Les fichiers sources doivent être compilés par `gcc` avec l'option `-g`
- De même, l'option `-g` doit être spécifiée au linker au moment de l'édition des liens pour obtenir un fichier ELF final contenant les symboles de débogage.

QEMU

QEMU possède un mode « monitor » avec lequel il est possible de stopper l'émulation, inspecter les registres, etc. Pour activer le monitor passer l'option `-monitor stdio` à QEMU.

A tout moment lors de l'émulation, il est possible de taper des commandes dans le monitor. Par exemple :

- `stop` permet de stopper l'émulation
- `cont` permet de reprendre l'émulation
- `info registers` permet d'afficher les registres du CPU

En plus du monitor, QEMU possède un serveur GDB permettant de deboggeur le code s'exécutant dans l'émulateur. Pour ce faire, il est nécessaire de passer les options `-s -S` à QEMU. Lors du démarrage, celui-ci sera en attente d'une connexion avec le debogueur GDB.

Debugger avec KDbg

KDbg

Il s'agit d'une interface graphique pour gdb. Elle prend. en charge le deboguage distant, ce qui en fait un outil de choix pour débogger dans QEMU.

Configuration

En terme de configuration, il faut simplement afficher les panneaux utiles via le menu, à savoir :

- pile
- locales
- registres
- points d'arrêt

et éventuellement

- sortie
- mémoire

Les panneaux étant détachables, on peut les disposer selon son goût.

Debugger avec KDbg

Pour déboguer avec KDbg, il faut d'abord lancer QEMU avec les options `-s -S`, le plus simple étant de créer une cible `debug` dans le makefile, du type :

```
debug: myos.iso
     qemu-system-i386 -monitor stdio -cdrom os.iso -s -S
```

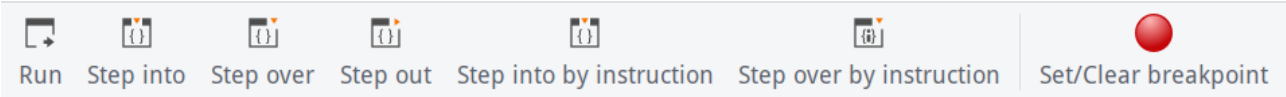
Cela effectué, il faut lancer KDbg selon le schéma suivant :

```
kdbg -r machine:port /local/path/to/binary
```

par exemple :

```
kdbg -r localhost:1234 kernel.elf
```

Il vous reste alors à ouvrir le code source du binaire distant, par exemple `kernel.c`, y placer un ou plusieurs breakpoints et lancer l'exécution via le bouton dans la barre de menu :



Les fonctions offertes permettent, respectivement de :

- lancer l'exécution (Run)
- rentrer dans une fonction et suivre les sous-appels (Step into)
- exécuter l'instruction (Step over)
- sortir des sous-appels (Step out)
- rentrer dans les fonctions et suivre les sous-appels au niveau instruction (Step into by instruction)
- sortir des sous-appels au niveau instruction (Step over by instruction)