Debugger un kernel dans QEMU avec GDB

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 le fichier ELF final.

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 connection avec le deboggeur GDB.

Configuration de gdb

La configuration proposées ci-après paramètre *gdb* avec la syntaxe intel, coupe la pagination externe et permet la gestion de l'historique des commandes.

Le fichier de configuration est `~/.gdbinit`, il se présente comme suit :

```
# Intel syntax is more readable
set disassembly-flavor intel

# When inspecting large portions of code the scrollbar works better than
'less'
set pagination off

# Keep a history of all the commands typed. Search is possible using
ctrl-r
set history save on
set history filename ~/.gdb_history
set history size 65536
set history expansion on
```

Pour faire bonne figure, il faut également créer le fichier d'historique des commandes via `touch ~/.gdb_history`.

Debugger avec GDB

Voici les étapes à suivre pour débugger le code du kernel en utilisant GDB :

- Exécuter QEMU avec les options suivantes :
  `qemu-system-i386 -monitor stdio -s -S`
- Exécuter GDB avec `gdb`
- Les commandes qui suivent sont à exécuter au sein de GDB
- Se connecter au serveur GDB de QEMU :
  `target remote :1234`
- Charger le fichier ELF du kernel :
  `file kernel.elf`
- Placer un point d'arrêt à la ligne 7 du source `kernel.c` :
  `br kernel.c:7`
- Continuer l'exécution jusqu'au point d'arrêt :
  `cont`

Exemples de commandes GDB

`enter`	repeats the last command
`next`	next line of code (C)
`nexti`	next instruction
`step`	step into function (C)
`print x`	print the value of variable x
`print *x`	print the value pointed by x
`info reg`	display registers
`ctrl-c`	stop emulation
`cont`	continue emulation until next breakpoint
`c`	continue emulation until next breakpoint
`x 0x1000`	inspect memory at address 0x1000
`x/10b 0xb8000`	display 10 bytes at address 0xb8000
`x/10h 0xb8000`	display 10 half-words (16-bits)
`x/10w 0xb8000`	dipslay 10 words (32-bits)

`x/10sb 0xb8000`	display 10 bytes as string
`x/10xb 0xb8000`	display 10 bytes as hex values
`x/10db 0xb8000`	display 10 bytes as decimal values
`x/10ib 0xb8000`	display 10 bytes as instructions (disassemble)
`br *0x1000`	set breakpoint at 0x1000
`info br`	display breakpoints
`disable 2`	disable breakpoint #2
`delete 2`	delete breakpoint #2
`enable 2`	enable breakpoint #2
`disas 0x10000,0x12000`	disassemble memory from 0x10000 to 0x12000

Examining memory with gdb: http://www.delorie.com/gnu/docs/gdb/gdb_56.html