Debug and develop uClibc using QEMU

Khem Raj
Embedded Linux Conference
11-13 April 2011, San Francisco
Agenda

- QEMU Setup
- Compiling uClibc for debugging
- Debugging uclibc dynamic linker using QEMU
- Some more on gdb
- Q & A
What is QEMU?

- Processor emulator
  - Emulates ARM, x86, powerpc, mips, SH ...
  - Has a built-in GDB stub

- Getting QEMU
  - [http://bellard.org/qemu/](http://bellard.org/qemu/)
  - Your favourite distribution might have already built it for you
Enable GDB stub

- **QEMU options**
  - `-s` enables the gdb stub
  - `-S` instructs QEMU to stop after system restart
    - Waits for gdb to connect
  - `-gdb tcp::1234`
    - Enables port 1234 on host
Debugging uClibc

- Use printf debugging
  - Compile with LD_DEBUG_EARLY
- Use JTAG with debugger e.g. BDI
  - Expensive
- Use emulators
  - QEMU
Dynamic linker is one of first userspace program started after kernel boots.

Dynamic linkers rely on printf debugging

Helpful in porting to new architectures

Debugging functionality

fixing bugs
Compiling uClibc for debugging

- Compile uclibc with debug information

```
make CROSS=/scratch/oe/qemuarm/cross/armv5te/bin/arm-oe-linux-uclibceabi- menuconfig
```
Debugging ld.so

- Find out start address offset of ld.so
  
  ```
  objdump -f ld-uClIBC.so.0 | grep start
  ```

  start address 0x00000ed0

- Find the virtual address mapping of ld.so
  
  - gdb's command `info shared`
  
  - Use `SUPPORT_LD_DEBUG_EARLY` which dumps the address
  
  - Gdb command `info proc mapping` or reading `proc/<pid>/maps`

- Add start address with virtual address to get the final address to load symbol information
Setup debugging environment

- **Launch QEMU system emulation**
  
  ```
  qemu-system-arm -M versatilepb -m 256 -gdb tcp::1234 -s -S
  -kernel <kernel> -drive file=<image> -append
  'console=ttyAMA0 console=ttyS0 root=/dev/sda rw debug
  user_debug=-1'
  ```

- **Launch cross gdb in another terminal**
  
  arm-oe-linux-uclibceabi-gdb

- **Connect to waiting QEMU**
  
  ```
  (gdb) target remote :1234
  Remote debugging using :1234
  0xc001eb30 in calibrate_delay()
  ```
Launch QEMU system emulation

```
qemu-system-arm -M versatilepb -m 256 -gdb tcp::1234 -s -S
-kernel <kernel> -drive file=<image> -append
'console=ttyAMA0 console=ttyS0 root=/dev/sda rw debug
user_debug=-1'
```

Use add-symbol-file <address> to load the debug info to right address.

Set breakpoint in _dl_get_ready_to_run ()

```
(gdb) b _dl_get_ready_to_run
Breakpoint 1 at 0x40005f94: file ldso/ldso/ldso.c, line 366.
```
Debugging ld.so

- Connect to remote target
- 'Continue' should hit the breakpoint in ld.so
Convenience
Frontends to gdb

- Data Display Debugger (DDD)
  - Uses gdb to control the target
  - Provided rich GUI experience
- Eclipse CDT
- Kdevelop
- Insight
Happy Debugging