Debugging Linux systems using GDB and QEMU

Khem Raj

Agenda

- Introduction
- Compiling and debugging Linux kernel using QEMU
- Creating image file for root file system
- Compiling and debugging uclibc dynamic linker using QEMU
- Compiling and debugging u-boot 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/
  - Your favourite distribution might have already built it for you

Enable GDB stub

- Add -s and -S options while invoking QEMU
  - -s enables the gdb stub
  - -S instructs QEMU to stop after system restart
    - Waits for gdb to connect

Compiling Kernel for QEMU

- Mainline kernel supports QEMU
- ARM versatilePB is supported

```
qemu-system-arm -M ?
```

- Use versatile configuration

```
make ARCH=arm versatile_defconfig
make ARCH=arm CROSS_COMPILE=arm-oe-linux-uclibceabi- all
```

- Use compressed image 'zImage' in arch/arm/boot

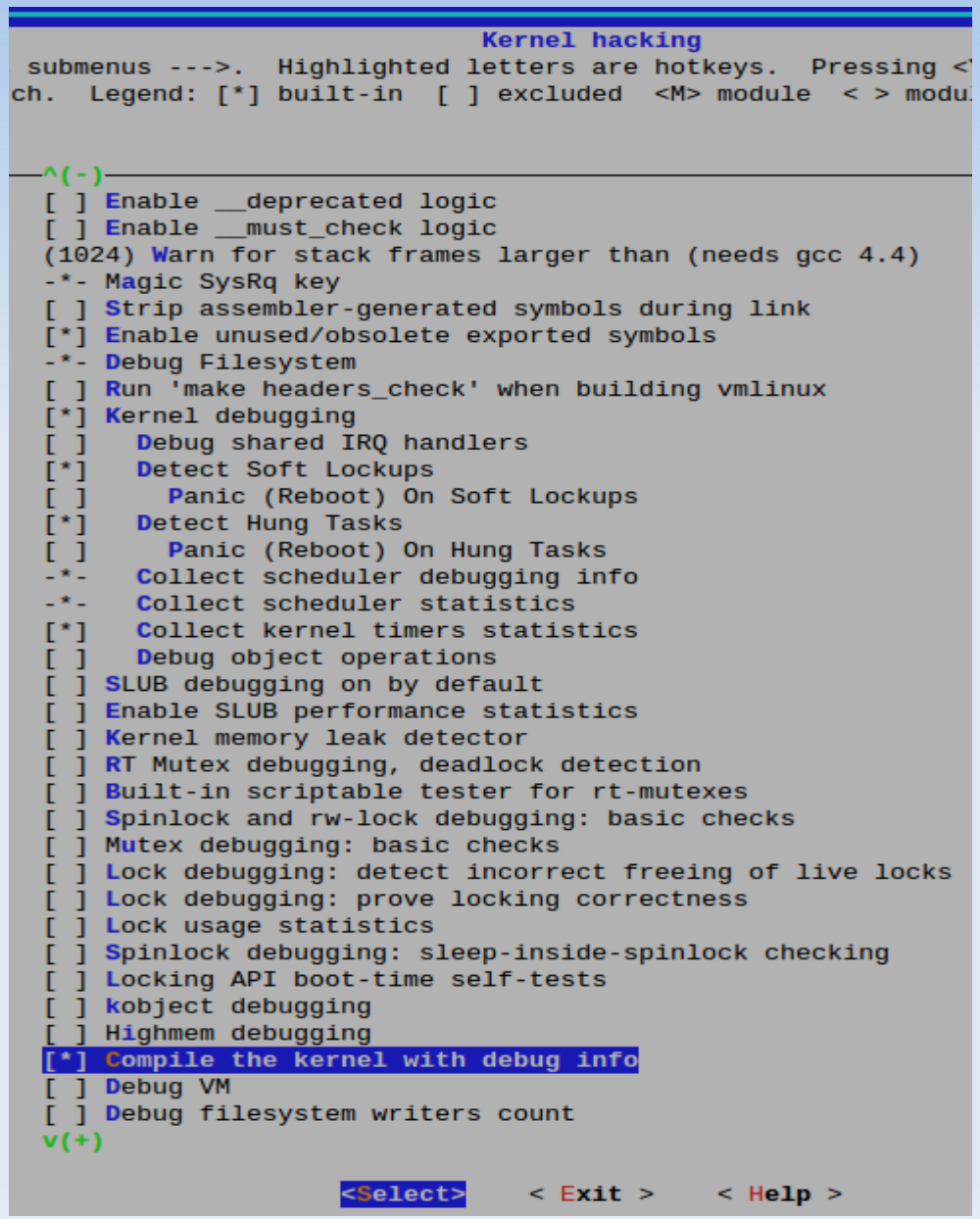
Compiling Kernel for QEMU

- Do not forget to turn debugging on for better experience

```
make ARCH=arm menuconfig
```

```
Kernel Hacking -->Compile the kernel
with debug info
```

Compile with debug information



Debugging kernel

- qemu-system-arm -M versatilepb --snapshot -gdb tcp::1234 -s -S -m 256 -kernel /scratch/oe/qemuarm/deploy/uclibc/images/qemuarm/zImage-qemuarm.bin -hda /scratch/oe/qemuarm/deploy/uclibc/images/qemuarm/console-image-qemuarm.ext2 -usb -usbdevice wacom-tablet -nographic --no-reboot -localtime -append 'console=ttyAMA0 console=ttyS0 root=/dev/sda rw debug user_debug=-1' -net none

Debugging Kernel

- Start GDB in another window

  /scratch/oe/qemuarm/cross/armv5te/bin/arm-oe-linux-uclibceabi-gdb
  /scratch/oe/qemuarm/work/qemuarm-oe-linux-uclibceabi/linux-rp-2.6.25+2.6.26-rc4-r6/linux-2.6.25/vmlinux

- Connect to the waiting QEMU remotely

  (gdb) target remote :1234

  Remote debugging using :1234

  0xc001eb30 in calibrate_delay ()

- Set breakpoints in the kernel start-up code somewhere in start_kernel ()

  (gdb) b pidmap_init

Useful breakpoints

- System-wide breakpoints to halt current thread of execution

```
(gdb) b panic
Breakpoint 4 at 0xc030be54
(gdb) b sys_sync
Breakpoint 3 at 0xc00b3bd4
```

Debugging early startup userspace with QEMU

- Dynamic linker is one of first userspace program started after kernel boots.
- Dynamic linkers rely on printf debugging
- Its helpful in porting to new architectures
- Adding more functionality or fixing bugs

Generating an ext2 root filesystem image

- Create a sysroot
  - Install the packages into the sysroot
- Create device nodes
  - `makedevs -r <sysroot> -D <device table>`
- Create an ext2 image
  - `genext2fs -b <size> -d <sysroot> -i 8192 <image>.ext2`

Device table

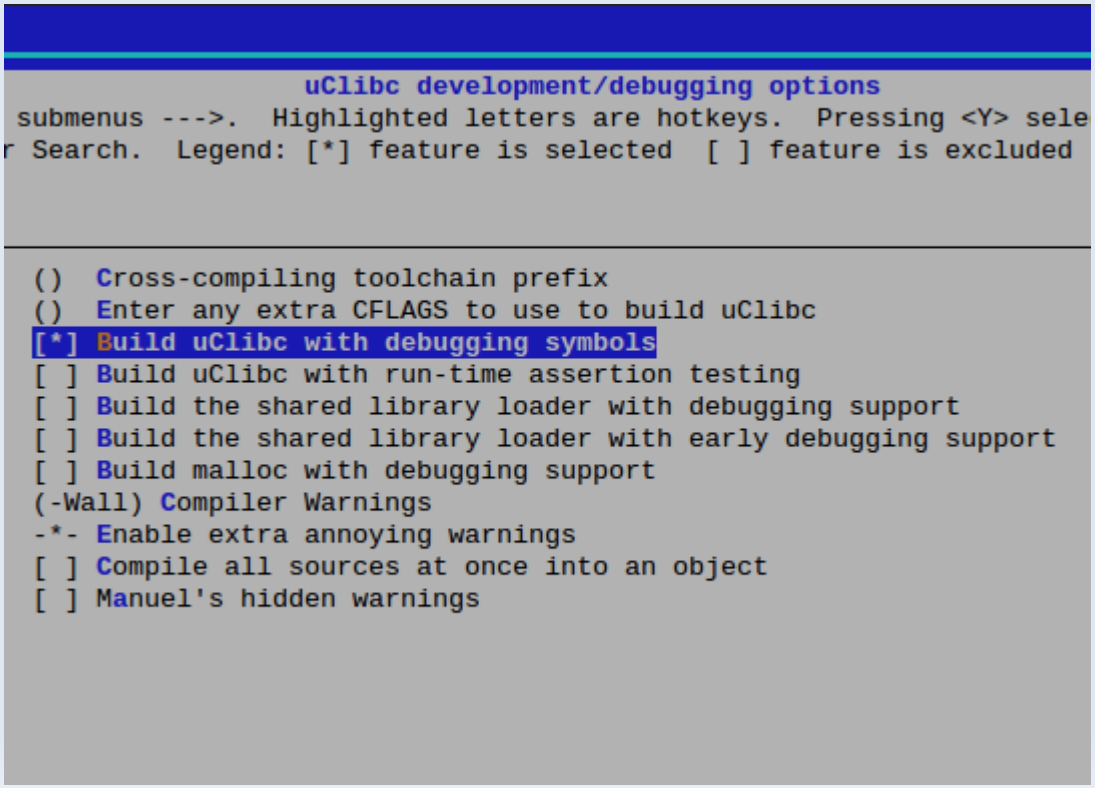
```
$ cat /home/kraj/work/oe/openembedded/files/device_table-minimal.txt
#<path> <type> <mode>   <uid>   <gid>   <major> <minor> <start> <inc>   <count>
#/dev/mem    c      640      0       0       1       1       0         0         -
#
#type can be one of:
#    f  A regular file
#    d  Directory
#    c  Character special device file
#    b  Block special device file
#    p  Fifo (named pipe)

/dev            d       755     0       0       -       -       -       -       -
/dev/initctl    p       600     0       0       -       -       -       -       -
/dev/apm_bios   c       660     0       46      10      134     -       -       -
/dev/fb0        c       600     0       0       29      0       -       -       -
/dev/hda        b       660     0       6       3       0       -       -       -
/dev/hda        b       660     0       6       3       1       1       1       19
/dev/kmem       c       640     0       15      1       2       -       -       -
/dev/mem        c       640     0       15      1       1       -       -       -
/dev/null       c       666     0       0       1       3       -       -       -
```

Compiling uClibc ld.so for debugging

- Compile uclibc with debug information

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



Debugging early startup userspace with QEMU

- Find out entry address of ld.so

```
objdump -f ld-uClibc.so.0 |grep start

start address 0x00000ed0
```

- Find the load address of ld.so
  - On a booted target use gdb's command info shared
  - Use SUPPORT_LD_DEBUG_EARLY which dumps the address
- Add entry address and load address to get the final virtual address

Debugging early startup userspace with QEMU

- Launch QEMU system emulation
- 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 early startup userspace with QEMU

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

```
Breakpoint 1, _dl_get_ready_to_run (tpnt=0x400a9730, load_addr=0x40007158, auxvt=0x0, envp=0x4000f030, argv=0x40006b24) at ldso/ldso/ldso.c:366
366                     nextp = unsecure_envvars;
(gdb) c
Continuing.
```

```
Breakpoint 1, _dl_get_ready_to_run (tpnt=0x400a9730, load_addr=0x40007158, auxvt=0x0, envp=0x4000f030, argv=0x40006b24) at ldso/ldso/ldso.c:366
366                     nextp = unsecure_envvars;
(gdb)
Continuing.
```

.gdbinit

- All can be put into a file that gdb reads

File Edit View Scrollback Bookmarks Settings Help

```
set output-radix 16
#add-symbol-file /scratch/oe/qemuarm/work/qemuarm-oe-linux-uclibceabi/uclibc-0.9.31+gitrf26c5f6952ce9bf8edec9c1571c47addb1bcc442-r34.0/git/lib/ld-uClibc.so.0 0x40000000+0xed0
#add-symbol-file /scratch/oe/build/work/qemumips-oe-linux-uclibc/sysvinit-2.86-r57/sysvinit-2.86/src/init 0x401570

define loadlibc
       add-symbol-file /scratch/oe/qemuarm/work/qemuarm-oe-linux-uclibceabi/uclibc-0.9.31+gitrf26c5f6952ce9bf8edec9c1571c47addb1bcc442-r34.0/git/lib/ld-uClibc.so.0 $arg0+0xed0
end

#b *0x2aba24d4
#b *(0x2aafb000 + 0x5aad8)
#b _dl_perform_mips_global_got_relocations
#b _dl_parse_relocation_information
#b elfinterp.c:217
#b ldso.c:1003
#b __dl_runtime_resolve
#b vsyslog
#b __uClibc_main
#b _dl_get_ready_to_run
#b __start
target remote localhost:1234
~
~
~
```

Compile u-boot for running in QEMU

- Download stable u-boot
- Compile for versatilePB

```
make
CROSS_COMPILE=/scratch/oe/qemuarm/cross/armv5te/bin/arm-oe-
linux-uclibceabi- versatilepb_config
```

```
make
CROSS_COMPILE=/scratch/oe/qemuarm/cross/armv5te/bin/arm-oe-
linux-uclibceabi- ARCH=arm
```

Debugging u-boot in QEMU

- Invoke QEMU system emulator
  - `qemu-system-arm -M versatilepb -m 256 -kernel u-boot.bin`
  - Start ARM gdb in another window and load u-boot

```
//scratch/oe/qemuarm/cross/armv5te/bin/arm-oe-linux-uclibceabi-gdb -nx ./u-boot
```

  - Connect to remote target i.e. QEMU

```
(gdb) target remote :1234
```

Debugging u-boot in QEMU

- Set breakpoints (say do_printenv ())

```
(gdb) b do_printenv
Breakpoint 1 at 0x10081b8: file cmd_nvedit.c, line 147.
Continue
```

- Issue printenv command on u-boot CLI

```
VersatilePB # printenv
```

- Execution breaks at do_printenv in gdb

```
Breakpoint 1, do_printenv (cmdtp=0x10151e4, flag=0, argc=1,
argv=0xfddef4) at cmd_nvedit.c:147
```

Frontends to gdb

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

Questions

Happy Debugging