kgdb with Virtualbox

Author: Swapnil Pimpale (swapnil@geeksofpune.in)

This document explains in brief the steps required to get kgdb up and running using a virtualbox VM.

Prerequisites:

1. Host should be running Linux – 3.5.0-43-generic (in my case)
2. Oracle Virtualbox should be installed – 4.3.12 (in my case)
3. Socat binary installed on the host. This is used to link the pipe file (FIFO) created by virtualbox with a pseudo terminal on the host. Normally, GDB takes a physical terminal (like ttyS0), but in our case we will provide a pseudo terminal created by socat as the remote target
4. A linux guest VM installed in virtualbox – FC20 (in my case)
5. Vanilla linux source code should be accessible to both the guest and the host – Can be achieved using shared folders in virtualbox

Setting up the guest:

- Configuring the virtual serial port: Right click on your VM instance, go to *Settings* tab. On *Port 1* tab, choose *"Enable Serial Port"*. Select port mode to be *"Host Pipe"*. Enter a pipe file name in the text field - */home/swapnil/sp/geep/kernel_seminar/serial* (in my case)

File sharing between host and guest:

- Install guest additions to enable shared folders for the guest VM. Please refer to virtualbox documentation available online on how to install guest additions for linux.
- Configure *Shared Folders* from the *Settings* tab of the VM

Preparing and installing a kgdb-enabled kernel on guest:

- The linux kernel source (3.10.43) can be compiled either on the guest or on the host. This source should be kept in the shared folder so that it can be accessed from both host and guest. Note that compilation is time consuming on the guest. I compiled the source on host machine.
- Follow the following steps for kernel compilation:
  - *make menuconfig (on host/guest):* Make sure that the following options are set/unset correctly
    - *CONFIG_DEBUG_INFO=y*
    - *CONFIG_FRAME_POINTER=y*
    - *CONFIG_DEBUG_RODATA=n*
    - *CONFIG_KGDB_SERIAL_CONSOLE=y*
    - *CONFIG_HAVE_ARCH_KGDB=y*
    - *CONFIG_KGDB_LOW_LEVEL_TRAPS=y*
    - *CONFIG_MAGIC_SYSRQ=y*
  - *make (on host/guest)*
  - *sudo make modules_install (on guest)*
  - *sudo make install (on guest)*
- Edit the kernel command line for the newly installed kernel to include these kernel parameters
  - *kgdboc=kbd,ttyS0,115200 – kbd* is optional here
  - *kgdbwait* – If you want to set breakpoints in early boot stages

Linking the serial file on the host to the pseudo terminal

- We use *socat* to do this linking – *socat -d -d /home/swapnil/sp/geep/kernel_seminar/serial PTY:*
- When this command is run it returns the pseudo terminal that's allocated (ex: */dev/pts/5*)
- Do not terminate the socat command as long as your debugging session is on. It needs to be running in the background else it breaks the stream

Start GDB

- Enter the linux kernel source directory where it was compiled and start gdb as follows
- *gdb ./vmlinux*
- *(gdb) target remote /dev/pts/5*
- This connects us to the guest linux kernel and now you can use any of the gdb commands to debug the guest kernel.

Further reading:

1. https://www.kernel.org/pub/linux/kernel/people/jwessel/kgdb/
2. https://www.kernel.org/pub/linux/kernel/people/jwessel/kdb/

References:

1. http://www.opensourceforu.com/2011/03/kgdb-with-virtualbox-debug-live-kernel/