Linux World

Using qemu to debug kernel

The qemu when used in the system mode can be used to run a complete operating system in another. For e.g. if we want to test a kernel image compiled for ARM on the x86 we can do it using the qemu in system mode

Another use of qemu is to debug or single step through the kernel code to debug it or to just understand the code.

To be able to debug the kernel running inside the qemu we will first need to have a kernel that is compiled with debug info, i.e. the option *CONFIG_DEBUG_INFO* should be set which can.be done as follows:

Download the source code from kernel.org
Create default config using

```
Make defconfig
```

Launch the configuration menu using

```
Make menuconfig
```

Now go to option *kernel hacking* and select the option *compile kernel with debug info*

Note: You will require package ncurses-devel to be installed .

If menuconfig does not launch then you cna open the .config file using any editor and search for *CONFIG_DEBUG_INFO* and set it to *y*
Now run :

```
Make
```

If there are no errors the bzimage would have got generated in arch/x86/boot under the source tree which we compiled.

Now let us boot this kernel using qemu and see how we can debug it. To just run the newly compiled kernel we can use the command

```
qemu-system-i386 -kernel bzImage
```

But to be able to debug the we will have to add the following options :

S: to ask the qemu to stop as soon as it starts,so that we can single execute when we are ready with the gdb connection s: to inform qemu to wait for HSBC connection on port 1234 kernel: To point to the bzimage that we wish to boot. The command with the options will be:

```
qemu-system-i386 -s -S -kernel "path to bzImage"
```

After executing the command,a qemu window is launched,which stops booting because of the *S* option.We can see the message *stopped* written in the top title bar of qemu.

Now open a new terminal and launch gdb passing to it the vmlinux that got created as a result of the kernel compilation

requi...

Download from gnuarm site keeps getting disrupted

gnuarm: can not execute binary file

▶ March (7)

▶ February (10)

▶ January (10)

▶ 2011 (69)

▶ 2010 (85)

▶ 2009 (1)

Followers

About Me

Tux Think

View my complete profile

```
gdb "path to vmlinux"
```

The gdb will read the debugging symbols that have got compiled with the kernel from vmlinux thus help us in debugging the kernel.

Now we need to connect this gdb session with the qemu session launched previously, which can be done by using the following command in gdb prompt

```
target remote localhost:1234
```

This connects the gdb to the qemu.
To set a breakpoint for a kernel function use the command *break* in the gdb prompt. For eg

```
break kernel_init
```

This will set a break point for the function *kernel_init*
To start execution of qemu use the command *c* which stands for continue.

The other gdb commands and their working can be found in the post "Introduction to gdb" .

To stop debugging use the command *quit* in gdb, which will also kill the qemu session.

Posted by Tux Think

5 comments

Add a comment as Mohammad Alshamlan

Top comments

Tux Think shared this via Google+ 1 year ago - Public

· Reply

Anonymous 2 months ago - Public

Thank you for your response. So make, and make bzImage are the same?

Tux Think 2 months ago

Yes make by default creates a bzImage, if you want to create any other format like a uImage etc then you need to specify it.

Anonymous 2 months ago - Public

Thank you for the post. I have a question though. In this statement: qemu-system-i386 -kernel bzImage

I have about 7 linux-* folders under
/usr/src/

They all contain the folders: arch/x86/boot
but none of these folders contain any bzimage.

I would like to ask what am I missing? Thanks.

Tux Think 2 months ago

If I am right you are looking at the kernel headers/source that come with the installation. They will not contain bzImage as you have not compiled the code yourself. The safest thing is download a fresh copy of source code from kernel.org for which ever kernel version you are interested in and then compile it yourself using the commands given above. Only after the successful compilation you will get the bzImage.

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Follow by Email

Email address... Submit