Setting up QEMU

This section will contain a guide to setting up QEMU simulator environment on your Linux machine, the commands may wary slightly from distro to distro. The following procedure is tested and working on Ubuntu 20.04 and 22.04.

Firstly, you should start the install of QEMU, this is done simply by apt-get install.

$ sudo apt-get install qemu

Since we in this project are going to work with an arm platform, we also need some additional packages, quemu-utils and qemu-system-arm. As above these can be fetched by the apt-get install command.

$ sudo apt install qemu-utils qemu-system-arm

Now that we have installed the simulator its time to make a folder to put the device we are simulating in, we called our folder qemu\_vms and put it below the right project folder and enter it.

$ mkdir Project

$ mkdir Project/qemu\_vms

$ cd Project/qemu\_vms

Then let’s get some premade kernels from dhruvvyas90’s github, here is the link to the git:

https://github.com/dhruvvyas90/qemu-rpi-kernel

You should get the entire repository and place it inside te qemu\_vms folder.

Now we need to get a Linux image, we chose to work on a Raspberry Pi using a lite image (Only terminal), but here you can choose somewhat freely the version you want to emulate. The reason I say somewhat is that you need to have kernel support for the version you are choosing. If you are wondering which kernels are supported, you can just take a look inside the qemu-rpi-kernel and you should find the versions supported. Get your image and place it inside the qemu-vms folder.

The image used for the rest of the guide can be found at

https://downloads.raspberrypi.org/raspios\_lite\_armhf/images/raspios\_lite\_armhf-2023-05-03/

If you got the same version of the image as we did it’s in the xz format, hence you need this command to unpack it.

$ xz -v -d 2023-05-03-raspios-bullseye-armhf-lite.img.xz

What does the v and d flags mean?

Now we are getting to the more technical part, so keep your tongue straight or you might have problems booting the device. I strongly encourage you to try to find out what the following commands do if you don’t already before you run them.

First command to run:

$ sudo losetup --show --find --partscan 2023-05-03-raspios-bullseye-armhf-lite.img

This should return something like:

/dev/loop10

Your number is almost certainty not the same is the number I got, but that is no problem. Finding the loop number is the reason we run this command. Now, change the loop number in the following commands to the loop number you got above before running them.

$ ls /dev/loop10\*

This will return all the different subfixes for the loop, you should see the original but also two more p1 and p2 like this:

/dev/loop10 /dev/loop10p1 /dev/loop10p2

Then its time for mounting our raspberry so we can change some stuff before we try to initialize it.

Run the following commands: DON’T FORGET TO CHANGE THE NUMBER

$ sudo mkdir /mnt/{raspbian-boot,raspbian-root}

$ sudo mount /dev/loop10p1 /mnt/raspbian-boot

$ sudo mount /dev/loop10p2 /mnt/raspbian-root

Now we can change the files inside the raspberry, for newer versions there is no standard user and password so let’s make it!

The following command opens the userconf.txt file, here you can add the users you want. You can choose a editor of your choice but I used vim

$ sudo vim /mnt/raspbian-boot/userconf.txt

For this example, we just add the standard “pi” user with password “raspberry”, to do that we enter the following text:

pi:$6$c70VpvPsVNCG0YR5$l5vWWLsLko9Kj65gcQ8qvMkuOoRkEagI90qi3F/Y7rm8eNYZHW8CY6BOIKwMH7a3YYzZYL90zf304cAHLFaZE0

As you might can see the username pi is in cleartext, but the password is written using an encrypted version of the password. Can you figure out how to make you own encrypted password that will work on the pi?

Then we exit the userconf file and enter ld.so.preload, this one is easy. We don’t want to preload anything so just comment out every line.

$ sudo vim /mnt/raspbian-root/etc/ld.so.preload

Now we are done changing the files in the raspberry, so we simply unmount the loops and release the directories we created. Again, remember to change the numbers.

$ sudo umount /dev/loop10p1

$ sudo umount /dev/loop10p2

$ sudo rmdir /mnt/{raspbian-boot,raspbian-root}

$ sudo losetup --detach /dev/loop10

Now, the image file of the raspberry is in raw format, this may be useful but can impact performance and in our case its not needed. Hence we convert it to the qcow2 format.

$ qemu-img convert -f raw -O qcow2 2023-05-03-raspios-bullseye-armhf-lite.img raspbian-bullseye-lite.qcow2

Now we have everything we need to try to run the virtual machine, when trying to start the machine we run the command qemu-system-arm. There are loads of different flags when trying to run the machine, for our case we found that this command runs the device quite stable.

$ qemu-system-arm \

-no-reboot \

-machine versatilepb -cpu arm1176 -m 256 \

-kernel qemu-rpi-kernel/kernel-qemu-5.10.63-bullseye \

-dtb qemu-rpi-kernel/versatile-pb-bullseye-5.10.63.dtb \

-drive format=qcow2,file=raspbian-bullseye-lite.qcow2 \

-append "root=/dev/sda2 panic=1 rootfstype=ext4 rw" \

-nographic \

-nic user,hostfwd=tcp::5022-:22

Here you can play around with arguments, adding serial ports and everything else you might want. But you should really try to understand the basics of the command also, for example do you think this command will connect the device to the internet?

After you run the command, the raspberry should start to boot. If not either try to fix the error messages or try to repeat this guide, as you might have made a mistake.

If everything is correct, the pi should ask you for login information. Here you use the login you defined in the userconf file, which in our case will be “pi” and “raspberry”.

Tips, if you can’t log in there might be something wrong with the userconf file. So, try repeating this process making Shure the changes stick. After doing this you also have to remember to update your image.

Now if you get logged in, you will see that the last line of the launch command indeed enables pass through of internet. If you want to enter the device using SSH or even cloning the code, you want to test from a git you need to make SSH keys. The following commands can be used:

$ ssh-keygen -A

$ sudo service ssh start

After making the keys you need to restart the machine, this is done most nicely by using the

$ sudo halt

Now everything should be set for you to begin development on you brand new ARM raspberry pi simulator.