

[ARM Assembly](#) [Online Assembler](#) [Exploitation](#) [Lab Environment](#) [TrustZone Research](#) [Self-Improvement](#) [About](#)[TEEs and Arm TrustZone](#)[Trustonic's Kinibi TEE](#)

RASPBERRY PI ON QEMU

Let's start setting up a Lab VM. We will use Ubuntu and emulate our desired ARM versions inside of it.

First, get the latest Ubuntu version and run it in a VM:

- <https://www.ubuntu.com/download/desktop>

For the QEMU emulation you will need the following:

1. A Raspbian Image: <http://downloads.raspberrypi.org/raspbian/images/raspbian-2017-04-10/> (other versions might work, but Jessie is recommended)
2. Latest qemu kernel: <https://github.com/dhruvvyas90/qemu-rpi-kernel>

Inside your Ubuntu VM, create a new folder:

```
$ mkdir ~/qemu_vms/
```

Download and place the Raspbian Jessie image to ~/qemu_vms/.

Download and place the qemu-kernel to ~/qemu_vms/.

```
$ sudo apt-get install qemu-system
$ unzip <image-file>.zip
$ fdisk -l <image-file>
```

You should see something like this:

```
Disk 2017-03-02-raspbian-jessie.img: 4.1 GiB, 4393533440 bytes, 8581120 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x432b3940

Device                Boot  Start      End  Sectors  Size Id Type
2017-03-02-raspbian-jessie.img1      8192  137215  129024   63M  c W95 FAT32 (LBA)
2017-03-02-raspbian-jessie.img2    137216 8581119 8443904   4G  83 Linux
```

You see that the filesystem (.img2) starts at sector 137216. Now take that value and multiply it by 512, in this case it's $512 * 137216 = 70254592$ bytes. Use this value as an offset in the following command:

```
$ sudo mkdir /mnt/raspbian
$ sudo mount -v -o offset=70254592 -t ext4 ~/qemu_vms/<your-img-file.img> /mnt/raspbian
$ sudo nano /mnt/raspbian/etc/ld.so.preload
```

Comment out every entry in that file with '#', save and exit with Ctrl-x » Y.

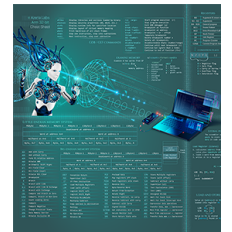
Lab Fundamentals

[ARM Lab VM 1.0](#)[ARM Lab VM 2.0](#)[Running Arm Binaries on : User](#)[Debugging with GDB Intrc](#)

Emulate Raspberry Pi with

[Emulating Arm Firmware](#)[Twitter: @Fox0x01 and @i](#)

New ARM Assembly C

[POSTER](#)[DIGITA](#)

```
$ sudo nano /mnt/raspbian/etc/fstab
```

IF you see anything with mmcblk0 in fstab, then:

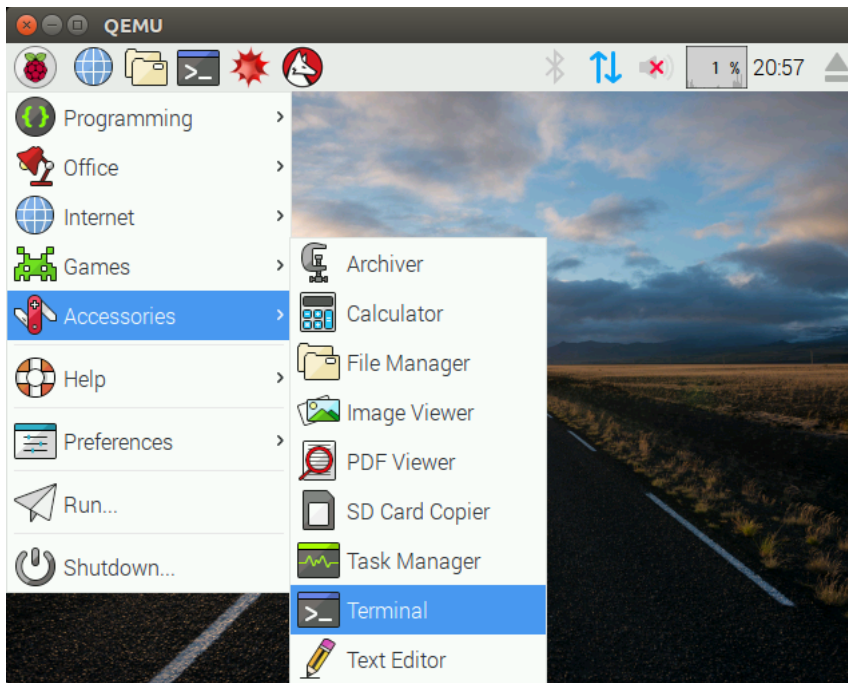
1. Replace the first entry containing /dev/mmcblk0p1 with /dev/sda1
2. Replace the second entry containing /dev/mmcblk0p2 with /dev/sda2, save and exit.

```
$ cd ~  
$ sudo umount /mnt/raspbian
```

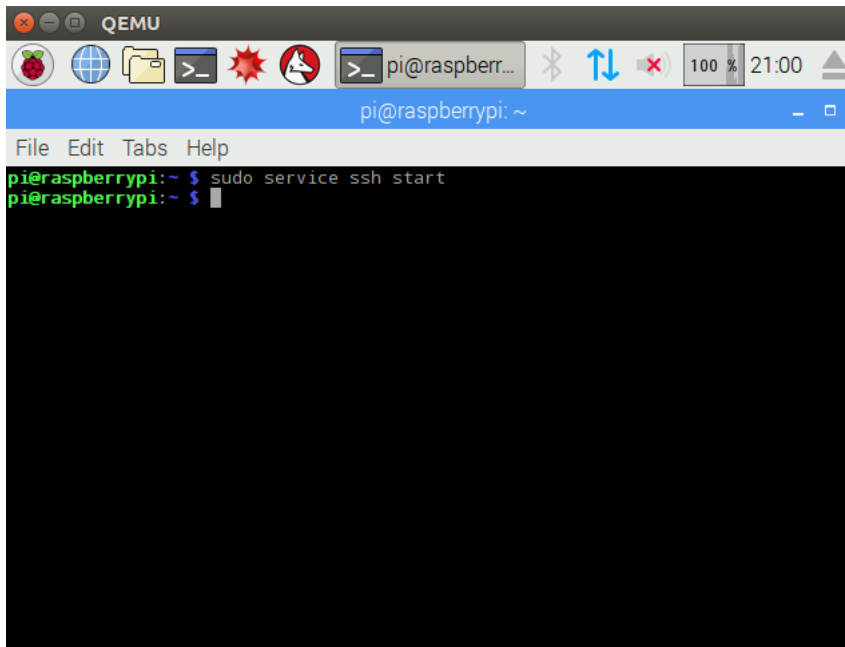
Now you can emulate it on Qemu by using the following command:

```
$ qemu-system-arm -kernel ~/qemu_vms/<your-kernel-qemu> -cpu arm1176 -m 256 -M versatilepb -serial stdio -append
```

If you see GUI of the Raspbian OS, you need to get into the terminal. Use Win key to get the menu, then navigate with arrow keys until you find Terminal application as shown below.



From the terminal, you need to start the SSH service so that you can access it from your host system (the one from which you launched the qemu).



Now you can SSH into it from your host system with (default password – raspberry):

```
$ ssh pi@127.0.0.1 -p 5022
```

For a more advanced network setup see the “Advanced Networking” paragraph below.

Troubleshooting

If SSH doesn't start in your emulator at startup by default, you can change that inside your Pi terminal with:

```
$ sudo update-rc.d ssh enable
```

If your emulated Pi starts the GUI and you want to make it start in console mode at startup, use the following command inside your Pi terminal:

```
$ sudo raspi-config  
>Select 3 - Boot Options  
>Select B1 - Desktop / CLI  
>Select B2 - Console Autologin
```

If your mouse doesn't move in the emulated Pi, click <Windows>, arrow down to Accessories, arrow right, arrow down to Terminal, enter.

Resizing the Raspbian image

Once you are done with the setup, you are left with a total of 3,9GB on your image, which is full. To enlarge your Raspbian image, follow these steps on your Ubuntu machine:

Create a copy of your existing image:

```
$ cp <your-raspbian-jessie>.img raspbian.img
```

Run this command to resize your copy:

```
$ qemu-img resize raspbian.img +6G
```

Now start the original raspbian with enlarged image as second hard drive:

```
$ sudo qemu-system-arm -kernel ~/qemu_vms/<kernel-qemu> -cpu arm1176 -m 256 -M versatilepb -serial stdio -append
```

Login and run:

```
$ sudo cfdisk /dev/sdb
```

Delete the second partition (sdb2) and create a **New** partition with all available space. Once new partition is creates, use **Write** to commit the changes. Then **Quit** the cfdisk.

Resize and check the old partition and shutdown.

```
$ sudo resize2fs /dev/sdb2
$ sudo fsck -f /dev/sdb2
$ sudo halt
```

Now you can start QEMU with your enlarged image:

```
$ sudo qemu-system-arm -kernel ~/qemu_vms/<kernel-qemu> -cpu arm1176 -m 256 -M versatilepb -serial stdio -append
```

Advanced Networking

In some cases you might want to access all the ports of the VM you are running in QEMU. For example, you run some binary which opens some network port(s) that you want to access/fuzz from your host (Ubuntu) system. For this purpose, we can create a shared network interface (tap0) which allows us to access all open ports (if those ports are not bound to 127.0.0.1). Thanks to [@0xMitsurugi](#) for suggesting this to include in this tutorial.

This can be done with the following commands on your **HOST** (Ubuntu) system:

```
azeria@labs:~$ sudo apt-get install uml-utilities
azeria@labs:~$ sudo tuncctl -t tap0 -u azeria
azeria@labs:~$ sudo ifconfig tap0 172.16.0.1/24
```

After these commands you should see the tap0 interface in the ifconfig output.

```
azeria@labs:~$ ifconfig tap0
tap0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
inet 172.16.0.1 netmask 255.255.255.0 broadcast 172.16.0.255
ether 22:a8:a9:d3:95:f1 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

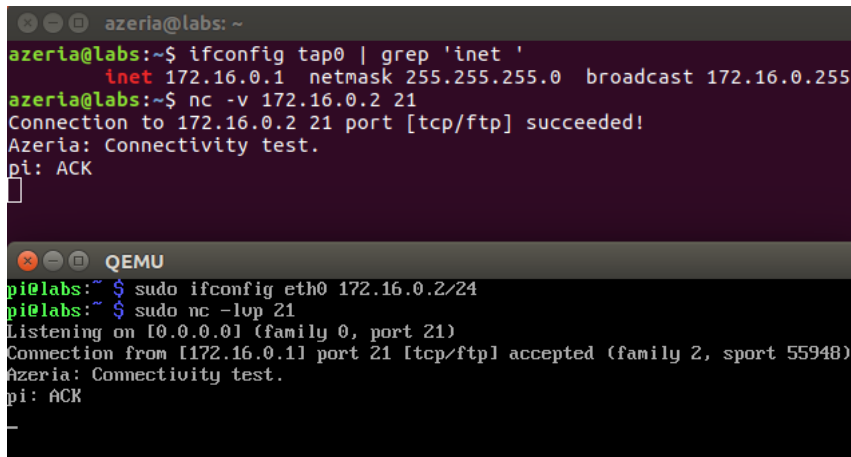
You can now start your QEMU VM with this command:

```
azeria@labs:~$ sudo qemu-system-arm -kernel ~/qemu_vms/<kernel-qemu> -cpu arm1176 -m 256 -M versatilepb -serial
```

When the QEMU VM starts, you need to assign an IP to it's eth0 interface with the following command:

```
pi@labs:~$ sudo ifconfig eth0 172.16.0.2/24
```

If everything went well, you should be able to reach open ports on the GUEST (Raspbian) from your HOST (Ubuntu) system. You can test this with a netcat (nc) tool (see an example below).



```
azeria@labs: ~  
azeria@labs:~$ ifconfig tap0 | grep 'inet '  
inet 172.16.0.1 netmask 255.255.255.0 broadcast 172.16.0.255  
azeria@labs:~$ nc -v 172.16.0.2 21  
Connection to 172.16.0.2 21 port [tcp/ftp] succeeded!  
Azeria: Connectivity test.  
pi: ACK  
[ ]  
  
QEMU  
pi@labs:~$ sudo ifconfig eth0 172.16.0.2/24  
pi@labs:~$ sudo nc -lvp 21  
Listening on [0.0.0.0] (family 0, port 21)  
Connection from [172.16.0.1] port 21 [tcp/ftp] accepted (family 2, sport 55948)  
Azeria: Connectivity test.  
pi: ACK  
-
```