

Howto setup Raspberry Pi Emulation with Qemu on Linux or Windows

by



<http://blog.pcfreak.de> [<http://blog.pcfreak.de>] Der PCFreak - Hardware - Software - Repair

Get Qemu

Download and/or install Qemu on your system (Linux or Windows). We need to emulate an ARM1176 CPU with Qemu, but some older versions of Qemu can not emulate this CPU and will not work. To check (on Linux) if your Qemu supports the ARM1176 or at lease the ARM1136-R2 execute the following command:

```
qemu-arm -cpu ?
```

```
pcfreak@dokuwiki $> qemu-arm -cpu ? | grep "arm"
arm1026
arm1136
arm1136-r2
arm1176
arm11mpcore
arm926
arm946
pcfreak@dokuwiki $>
```

The Qemu provided by your Linux distribution should just work (tested on latest Linux Mint - `apt-get install qemu-system-arm`)

For Windows, just use the latest Qemu which you can get here [<http://lassauge.free.fr/qemu/>] or here [<http://www.omledom.com/>].

It is possible, that you need this additional files in the Qemu folder when using Windows.

```
intl.dll
libglib-2.0-0.dll
libgthread-2.0-0.dll
libpng14-14.dll
libssp-0.dll
SDL.dll
zlib1.dll
```

This DLLs can be downloaded here [<http://qemu.weilnetz.de/w32/dll/>].

Get a Raspberry image

Download your favourite Raspberry Pi image (linux based) and save it to your harddrive. This documentation used Raspbian

```
2013-07-26-wheezy-raspbian.img
```

based on Debian Wheezy downloaded from here [<http://www.raspberrypi.org/downloads>].

Get a kernel

To be able to boot the Raspberry image, you need a kernel. You can download a working kernel from here [<http://xecdesign.com/downloads/linux-qemu/kernel-qemu>].

You should now have the file

```
kernel-qemu
```

on your harddrive.

Verify prerequisites

You should now have the following files on your system. Best practice would be to keep them in the same folder.

```
2013-07-26-wheezy-raspbian.img
kernel-qemu
```

On my Linux machine the folder looks like this:

```
pcfreak@dokuwiki $> ls -l
total 1897796
-rw-rw-r-- 1 pcfreak pcfreak 4087349248 Sep 11 16:49 2013-07-26-wheezy-raspbian.img
-rw-rw-r-- 1 pcfreak pcfreak 3470912 Dec 2 2012 kernel-qemu
pcfreak@dokuwiki $>
```

On my Windows machine I additionally have a subfolder where my Qemu lives:

Name	Date modified	Type	Size
 qemu	12.09.2013 09:48	File folder	
 2013-07-26-wheezy-raspbian.img	26.07.2013 14:45	IMG File	1.894.400 KB
 kernel-qemu	11.09.2013 16:11	File	3.390 KB

Resize the image to have more space

A **real** Raspberry Pi uses an SD-Card and you have to copy the image to this card. We do an emulation of this, so we need to resize the image to the size we want to use later. To add 2Gb more space to the Raspbian image we have to execute the following Qemu command.

```
qemu-img resize 2013-07-26-wheezy-raspbian.img +2G
```

This should add additional 2Gb to 2013-07-26-wheezy-raspbian.img.

On Linux it looks like this:

```
pcfreak@dokuwiki $> qemu-img resize 2013-07-26-wheezy-raspbian.img +2G
Image resized.
pcfreak@dokuwiki $> █
```

And on Windows it should look similar:

```
>qemu\qemu-img.exe resize 2013-07-26-wheezy-raspbian.img +2G
Image resized.
```

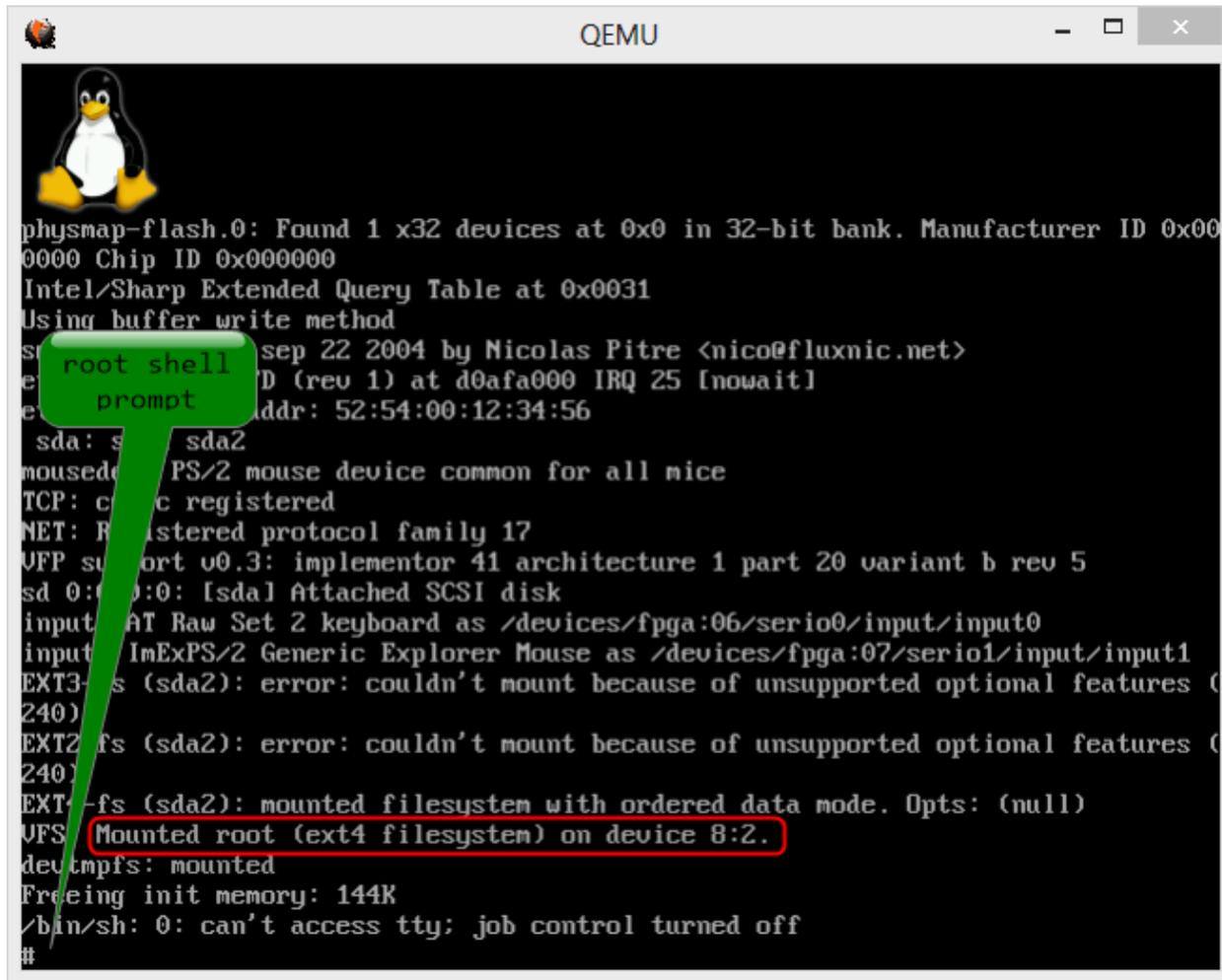
Boot to root shell and fix some stuff

Before we can do a "real" boot with the Raspbian image, **we need to modify some files to make it work**. Some other documentations you can find on the net tell you to mount the image and then do the changes - which is easy under Linux - but when using Windows you cannot easily mount the image for modifications. An easier way is to boot Raspberrian into a root shell and do the changes there

Therefore fire-up Qemu with the following command (**Don't forget the 'init=/bin/sh rw' part**)

```
qemu-system-arm -kernel kernel-qemu -cpu arm1176 -m 256 -M versatilepb -append "root=/dev/sda2 panic=1 init=/bin/sh rw" -hda 2013-07-26-wheezy-raspbian.img
```

When the system has bootet into a /bin/sh root shell in read/write mode, it should look like this:



A screenshot of a QEMU terminal window. The window title is "QEMU". The terminal output shows the boot process of a Linux system. It starts with hardware initialization messages, including "physmap-flash.0: Found 1 x32 devices at 0x0 in 32-bit bank. Manufacturer ID 0x000000 Chip ID 0x000000". It then shows the kernel version "Linux version 2.6.32-041-generic (Sep 22 2004 by Nicolas Pitre <nico@fluxnic.net>)" and the root device "root=hd (rev 1) at d0afa000 IRQ 25 [nowait]". The root filesystem is mounted as "rootfs (sda2): mounted filesystem with ordered data mode. Opts: (null)". A red box highlights the message "Mounted root (ext4 filesystem) on device 8:2.". The terminal ends with a root shell prompt "#". A green speech bubble with the text "root shell prompt" points to the prompt.

```
physmap-flash.0: Found 1 x32 devices at 0x0 in 32-bit bank. Manufacturer ID 0x000000 Chip ID 0x000000
Intel/Sharp Extended Query Table at 0x0031
Using buffer write method
Linux version 2.6.32-041-generic (Sep 22 2004 by Nicolas Pitre <nico@fluxnic.net>)
root=hd (rev 1) at d0afa000 IRQ 25 [nowait]
rootfs (sda2): error: couldn't mount because of unsupported optional features (
240)
EXT2-fs (sda2): error: couldn't mount because of unsupported optional features (
240)
EXT1-fs (sda2): mounted filesystem with ordered data mode. Opts: (null)
VFS: Mounted root (ext4 filesystem) on device 8:2.
devtmpfs: mounted
Freeing init memory: 144K
/bin/sh: 0: can't access tty; job control turned off
#
```

Fix endless boot loop

At the root prompt open `/etc/ld.so.preload` with nano (**remember you have an english keyboard**)

```
nano /etc/ld.so.preload
```

```
UFS: Mounted root (ext4 filesystem) on device 8:2.  
devtmpfs: mounted  
Freeing init memory: 144K  
/bin/sh: 0: can't access tty; job control turned off  
# atkbd serio0: Unknown key pressed (raw set 2, code 0x1f on fpga:06).  
atkbd serio0: Use 'setkeycodes 1f <keycode>' to make it known.  
# nano /etc/ld.so.preload
```

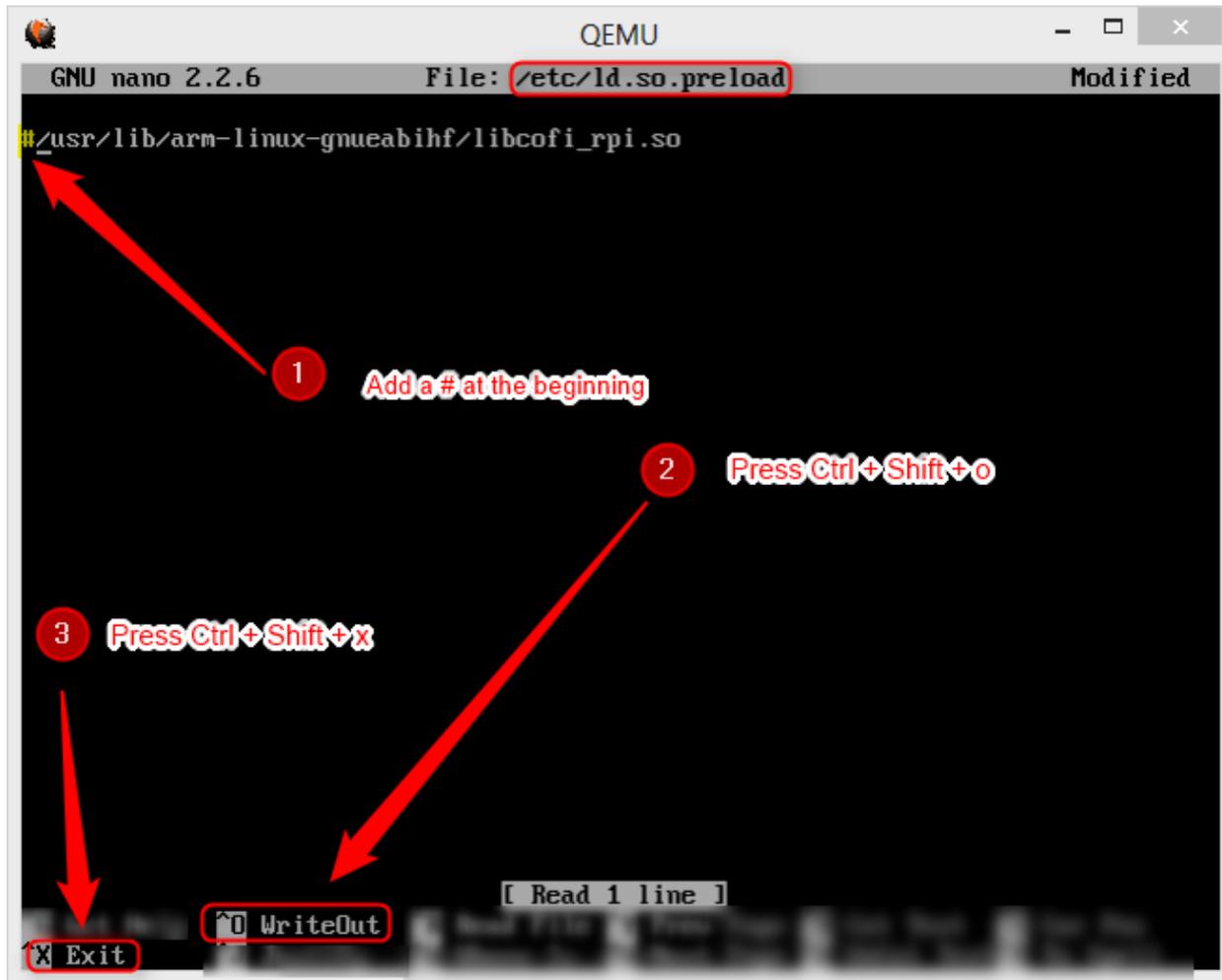
A file will open. It contains exactly one line.

```
/usr/lib/arm-linux-gnueabi/libcofi_rpi.so
```

Add a # (SHIFT+3 on german keyboard) at the beginning of the line, so it looks now

```
#/usr/lib/arm-linux-gnueabi/libcofi_rpi.so
```

Now press **Ctrl+O <ENTER>** to write the file and then **Ctrl+X** to exit the editor



This change will make sure, that libcofi_rpi.so will not be loaded. This change was necessary to avoid an endless boot loop.

Fix disks for later use

Still at the root prompt create the file

```
/etc/udev/rules.d/90-qemu.rules
```

by executing the nano editor again:

```
nano /etc/udev/rules.d/90-qemu.rules
```

A terminal window with a black background and white text. The prompt is '# nano /etc/udev/rules.d/90-gemu.rules'. The text is highlighted with a red box. The terminal has a jagged top edge.

```
# nano /etc/udev/rules.d/90-gemu.rules
```

Enter the following lines into the file

```
-----  
KERNEL=="sda", SYMLINK+="mmcblk0"  
KERNEL=="sda?", SYMLINK+="mmcblk0p%n",  
-----
```

Then press **Ctrl+O** <ENTER> to write the file and then **Ctrl+X** to exit the editor.

```
QEMU
GNU nano 2.2.6 File: /etc/udev/rules.d/90-qemu.rules Modified
KERNEL=="sda", SYMLINK+="mmcblk0"
KERNEL=="sda", SYMLINK+="mmcblk0pzn",
^X Exit
^O WriteOut
```

1 don't forget the Komma

2 Ctrl+Shift+o

3 Ctrl+Shift+x

Finally enter the command

```
sync
```

to make sure everything is written to disk.

```
#
# sync
```

Now just close Qemu to stop the virtual machine.

Now we have applied all needed fixes to make the emulation boot without problems and being able to resize the root partition with raspi-config.

First run

To fire-up your machine just execute the following command (pointing to the correct files).

```
qemu-system-arm -kernel kernel-qemu -cpu arm1176 -m 256 -M versatilepb -append "root=/dev/sda2 panic=1" -hda 2013-07-26-wheezy-raspbian.img
```

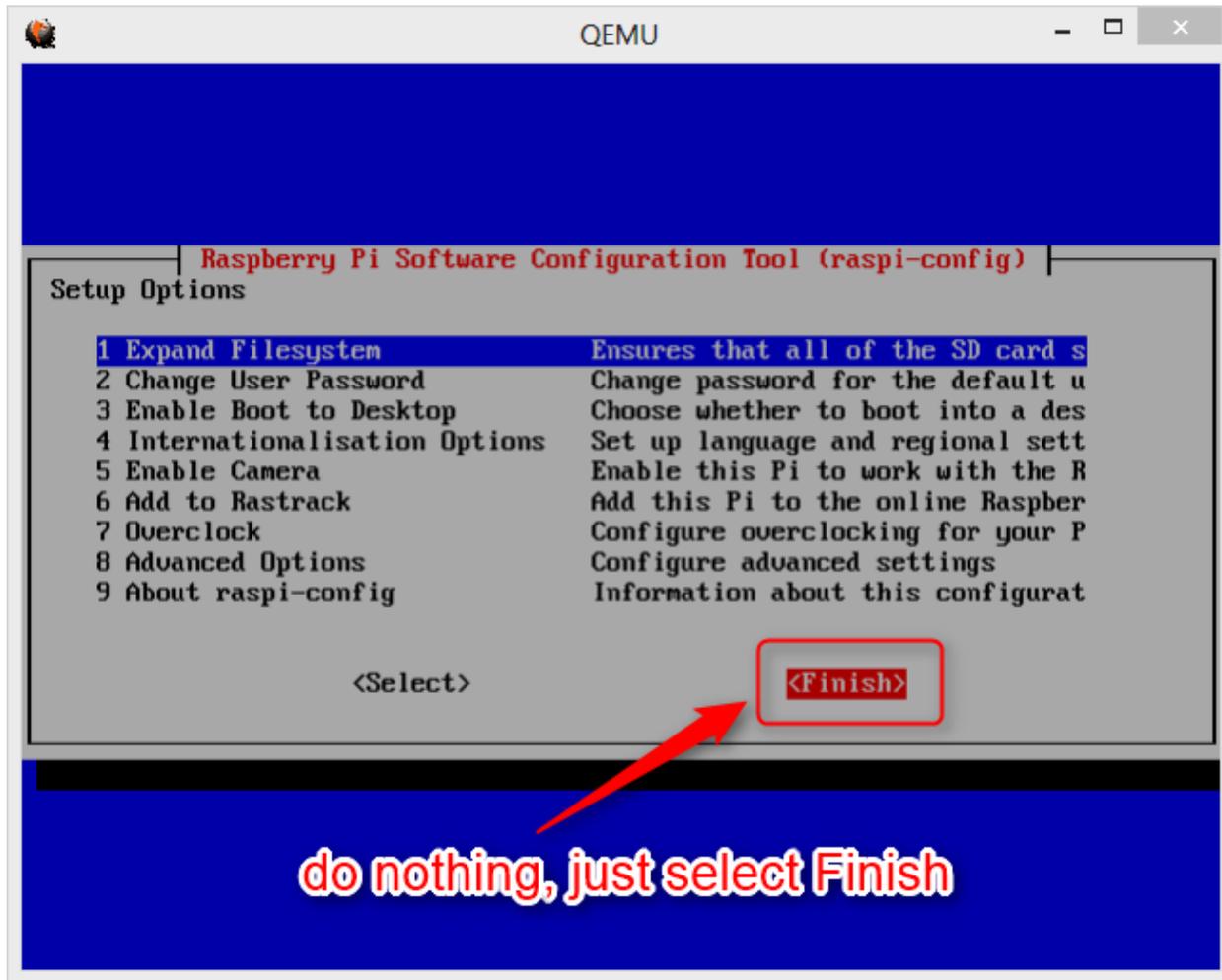
If Qemu automatically prompts you with a root shell asking for a **fsck** to fix the expanded file system execute the following

```
fsck /dev/sda2
```

Then reboot the emulation with

```
shutdown -r now
```

At the next boot Raspbian should automatically boot into raspi-config. **Immediately quit raspi-config** by selecting <FINISH>.



Then create a link with the following command

```
sudo ln -snf mmcblk0p2 /dev/root
```

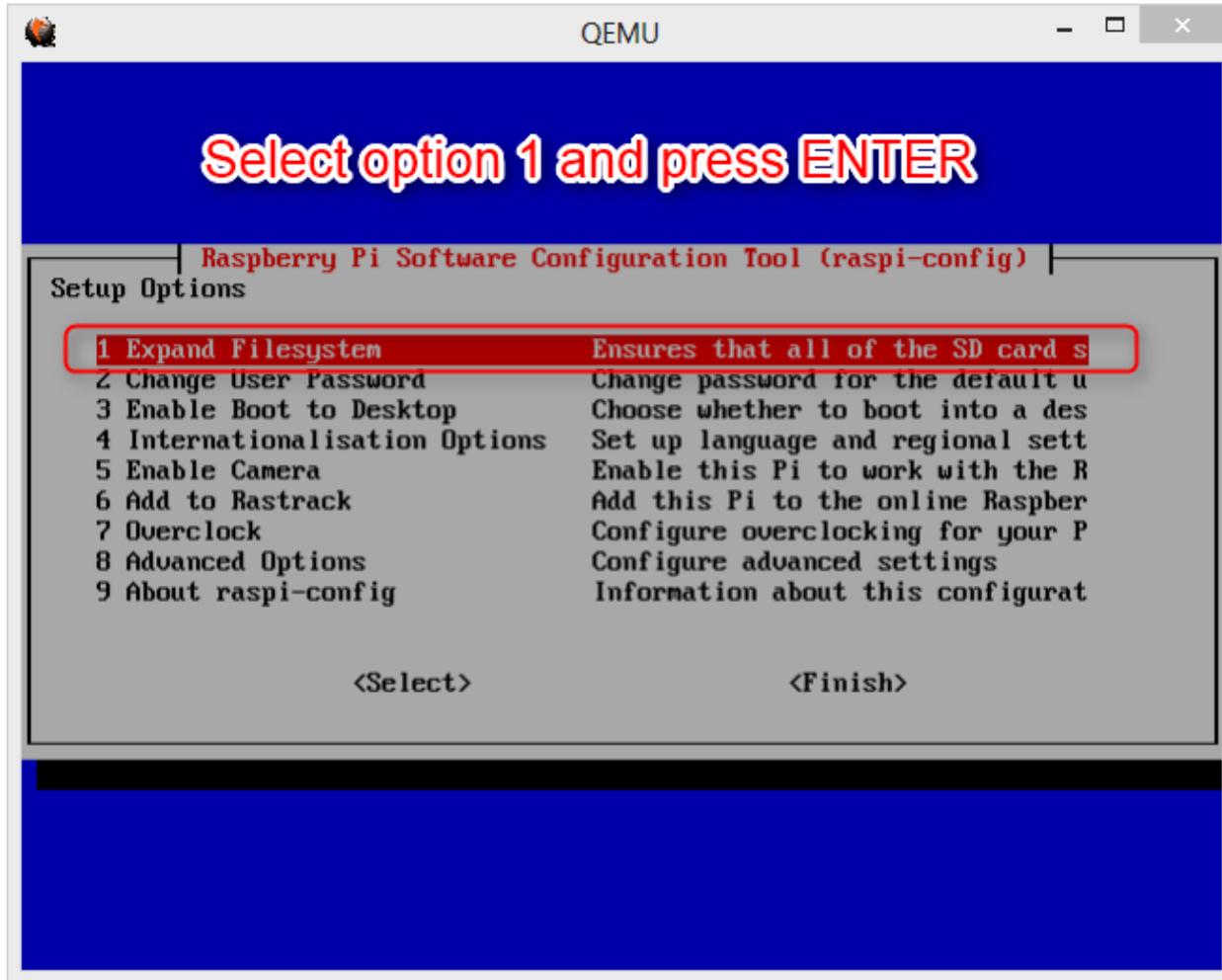
Then manually restart raspi-config with

```
sudo raspi-config
```

```
pi@raspberrypi ~ $ sudo ln -snf mmcblk0p2 /dev/root 1
pi@raspberrypi ~ $ sudo raspi-config 2
```

and select

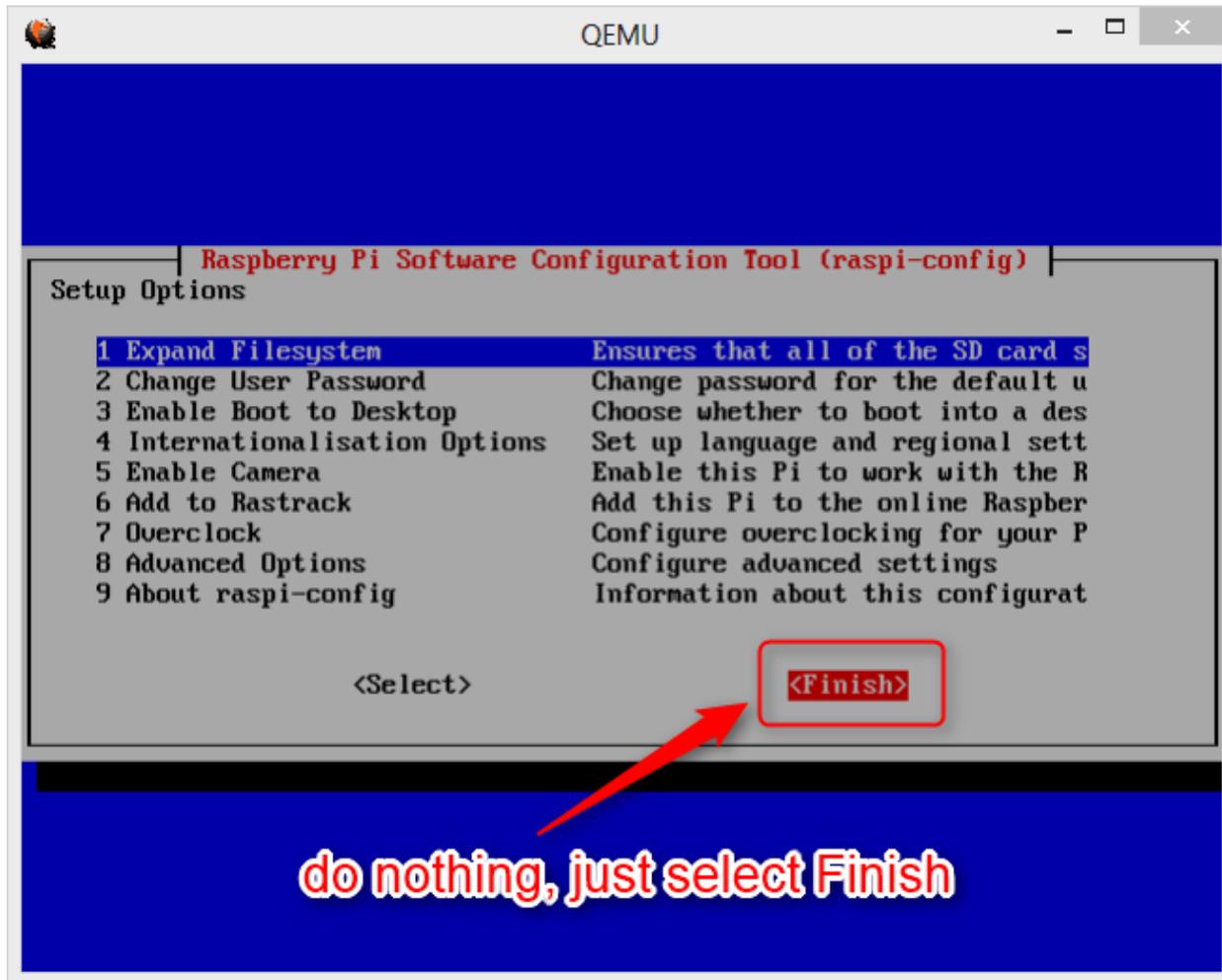
1 Expand Filesystem Ensures that all of the SD card s...



The following screen should tell you that everything went fine.



Accept the message by pressing ENTER and exit raspi-config afterwards by selecting FINISH.



You will get asked if you like to reboot. Select YES to initiate the reboot.

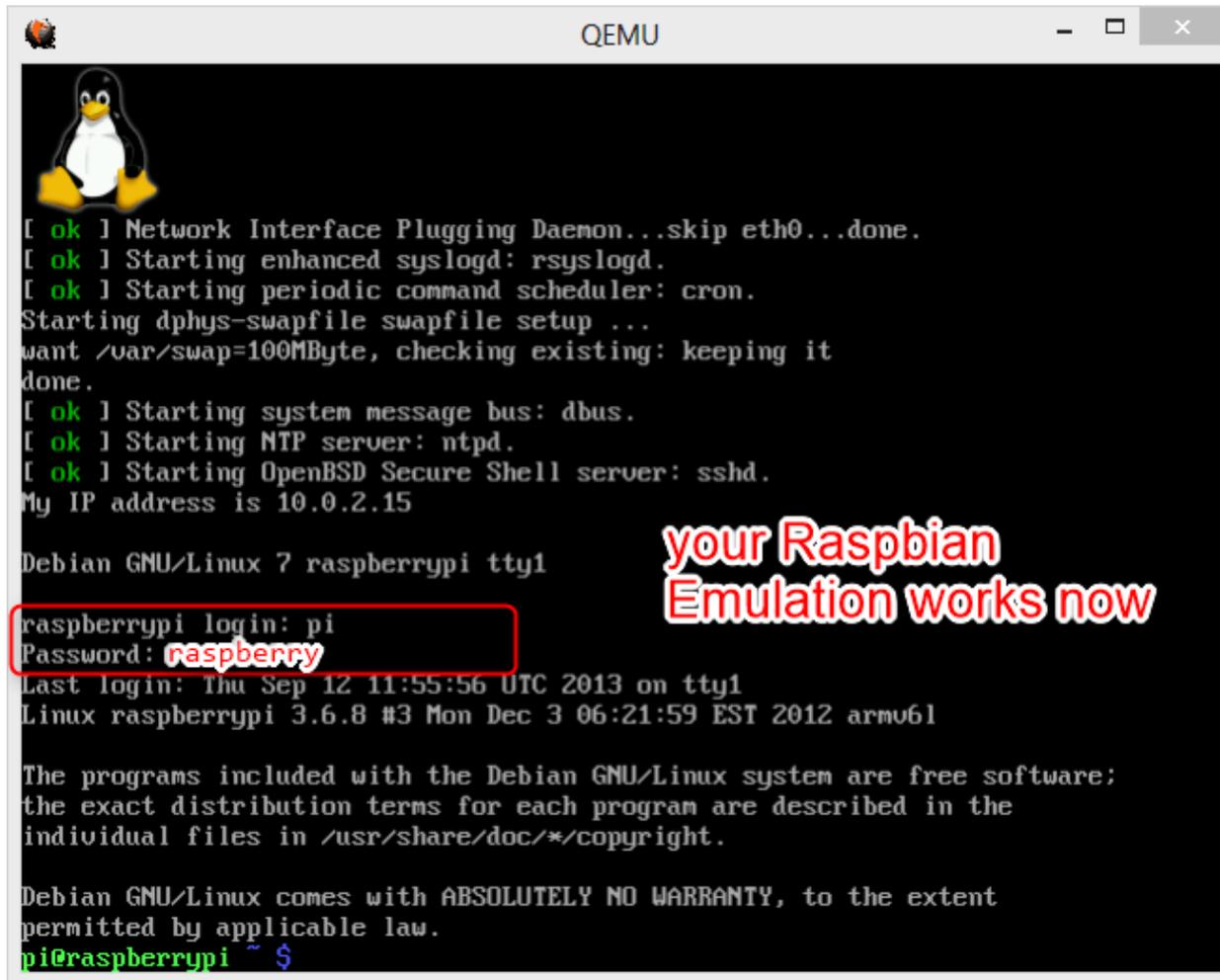


During the reboot the disk will be resized and at the end you should arrive at the login prompt.

You can then login with the following credentials:

```
username: pi  
password: raspberry
```

Please keep in mind, that the keyboard layout is english by default, make sure you type the **y** correctly.



```
QEMU
[ ok ] Network Interface Plugging Daemon...skip eth0...done.
[ ok ] Starting enhanced syslogd: rsyslogd.
[ ok ] Starting periodic command scheduler: cron.
Starting dphys-swapfile swapfile setup ...
want /var/swap=100MByte, checking existing: keeping it
done.
[ ok ] Starting system message bus: dbus.
[ ok ] Starting NTP server: ntpd.
[ ok ] Starting OpenBSD Secure Shell server: sshd.
My IP address is 10.0.2.15

Debian GNU/Linux 7 raspberrypi tty1
raspberrypi login: pi
Password: raspberrypi
Last login: Thu Sep 12 11:55:56 UTC 2013 on tty1
Linux raspberrypi 3.6.8 #3 Mon Dec 3 06:21:59 EST 2012 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi ~ $
```

your Raspbian Emulation works now

Fixes inside the emulation

/etc/X11/xorg.conf

To be able to run X with 800×600 we need to manually create an xorg.conf by executing

```
sudo nano /etc/X11/xorg.conf
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

```
pi@raspberrypi ~ $ sudo nano /etc/X11/xorg.conf
```

Write the following into the file

```
Section "Screen"
Identifier "Default Screen"
SubSection "Display"
    Depth 16
    Modes "800x600" "640x480"
EndSubSection
EndSection
```

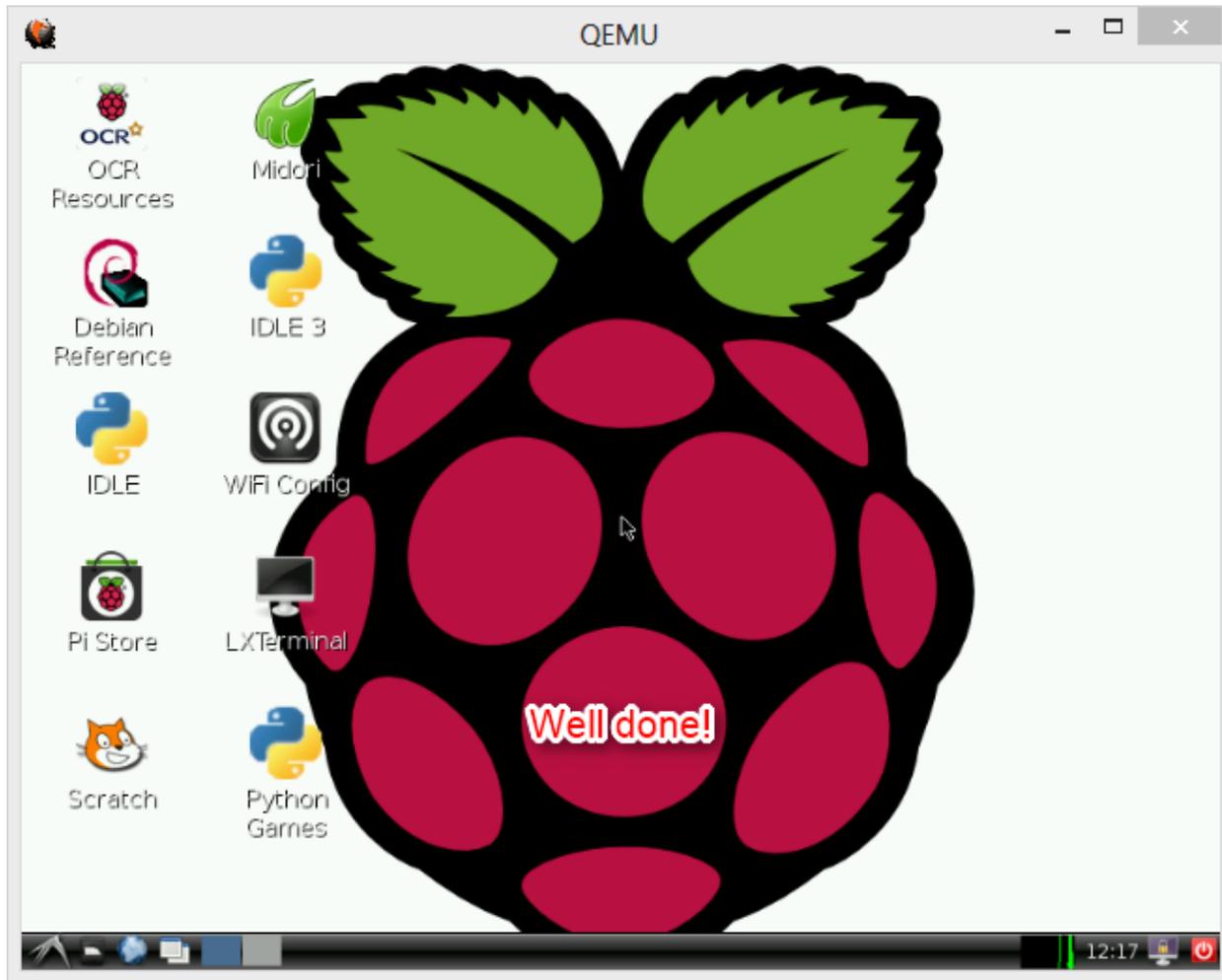
and then WriteOut (Ctrl+Shift+o) and Exit (Ctrl+Shift+X) in nano.

```
QEMU
GNU nano 2.2.6 File: /etc/X11/xorg.conf Modified
Section "Screen"
Identifier "Default Screen"
  SubSection "Display"
    Depth 16
    Modes "800x600" "640x480"
  EndSubSection
EndSection
^X Exit
```

Just for testing you can now start the X server with

```
startx
```

It should present you the desktop with a resolution of 800×600 pixels as shown in this image.



Additional Notes

Memory

Don't try to use more than 256MB memory within the qemu command line. The value is hard-coded in the emulated arm-chip, so you cannot change it!

Graphics

At the moment it seems, that the maximum resolution you can emulate is 800×600 with a depth of 16bit.

