Compiling an ARM1176 kernel for QEMU

- This page will show you how to compile a QEMU-ready Linux kernel for the ARM1176JZF-S CPU used by the Raspberry Pi. If you want to skip all of this and just have a kernel you can use, you can download one here.

Assumptions

- You're in a Linux environment.
- You are able to install software for your particular Linux distro.
- You have a basic knowledge of the Linux command line.
- You can follow instructions carefully.

Preparing the environment

You will need to install a few packages. I am running Linux Mint 14, modify for your distro.

1. sudo apt-get install git libncurses5-dev gcc-arm-linux-gnueabihf

If you're running a 64-bit system you will also need the 32-bit libraries.

1. sudo apt-get install ia32-libs

Create and enter the directory from which you will work and the directory to install our kernel's modules:

1. mkdir ~/raspidev
2. cd ~/raspidev/
3. mkdir modules

All of the instructions will assume you are working from this directory.

Preparing the kernel

Download the Linux kernel (this will take a while):

- git clone https://github.com/raspberrypi/linux.git

The kernel does not have support for ARM11(ARMv6) for the Versatile board, so we will need to download and apply a patch.

1. wget http://xecdesign.com/downloads/linux-qemu/linux-arm.patch
2. `patch -p1 -d linux/ < linux-arm.patch`

You should not see any errors, do not proceed if you do.

Configuring the kernel

First, we need to know what hardware QEMU emulates:

> The ARM Versatile baseboard is emulated with the following devices:
>
> - ~~ARM926E, ARM1136 or Cortex-A8 CPU~~
> - ~~PL190 Vectored Interrupt Controller~~
> - ~~Four PL011 UARTs~~
> - ~~SMC 91c111 Ethernet adapter~~
> - ~~PL110 LCD controller~~
> - ~~PL050 KMI with PS/2 keyboard and mouse.~~
> - *PCI host bridge*. Note the emulated PCI bridge only provides access to PCI memory space. It does not provide access to PCI IO space. This means some devices (eg. ne2k_pci NIC) are not usable, and others (eg. rtl8139 NIC) are only usable when the guest drivers use the memory mapped control registers.
> - ~~PCI OHCI USB controller.~~
> - *LSI53C895A* PCI SCSI Host Bus Adapter with *hard disk* and *CD-ROM devices*.
> - ~~PL181 MultiMedia Card Interface with SD card.~~

Recent Posts

QEMU – Emulating Raspberry Pi the easy way (Linux or Windows!)
Working with QEMU
QEMU – Compiling for ARM (1176) emulation
Compiling an ARM1176 kernel for QEMU

Blogroll

Ben O'Steen
Hexxeh
Russel Davis

Archives

April 2012
February 2012

Categories

Kernel
QEMU
Raspberry Pi
Tutorials

The devices which are automatically selected when we run the commands below are crossed out and the devices we need to add manually are underlined.

Then we need to configure the kernel accordingly:

1. cd linux
2. make ARCH=arm versatile_defconfig
3. make ARCH=arm menuconfig

Specify the cross-compiler:

> General Setup --->
>
> Cross-compiler tool prefix
>
> (arm-linux-gnueabihf-)

Note: Do not forget the '-' at the end of this string.

Note: When you select an option make sure you don't see 'M', but '*' (you may need to press space twice).

Select the right CPU options:

> System Type --->
>
> [*] Support ARM V6 processor
>
> [*] ARM errata: Invalidation of the Instruction Cache operation can fail
>
> [*] ARM errata: Possible cache data corruption with hit-under-miss enabled

Enable support for hard-float binaries:

> Floating point emulation --->
>
> [*] VFP-format floating point maths

Enable ARM EABI:

> Kernel Features --->
>
> [*] Use ARM EABI to compile the kernel
>
> [*] Allow old ABI binaries to run with this kernel

Enable QEMU's disk support:

> Bus Support --->
>
> [*] PCI Support
>
> Device Drivers --->
>
> SCSI Device Support --->
>
> [*] SCSI Device Support
>
> [*] SCSI Disk Support
>
> [*] SCSI CDROM support
>
> [*] SCSI low-lever drivers --->
>
> [*] SYM53C8XX Version 2 SCSI support

Enable devtmpfs:

> Device Drivers --->

> Generic Driver Options--->
>
> [*] Maintain a devtmpfs filesystem to mount at /dev
>
> [*] Automount devtmpfs at /dev, after the kernel mounted the root

Enable the important file systems:

<*> Ext3 journalling file system support

<*> The Extended 4 (ext4) filesystem

Enable tmpfs:

> File systems --->
>
> Pseudo filesystems--->
>
> [*] Virtual memory file system support (former shm fs)

Enable the event interface:

> Device Drivers --->
>
> Input device support--->
>
> [*] Event interface

Exit, saving the configuration.

Optional configuration

Enable /proc/config.gz:

> General Setup --->
>
> [*] Kernel .config support
>
> [*] Enable access to .config through /proc/config.gz

The default font is fairly small, which is good if you want to see more text on the screen.

You can optionally disable the default compiled-in font for a more standard linux look or modify the kernel further to your own taste:

> Device Drivers --->
>
> Graphics Support --->
>
> Console display driver support --->
>
> [ ] Select compiled-in fonts
>
> [*] Bootup logo (optional)

There are many more options which you may find helpful or even require to get your distro or certain applications working, so consult their documentation as I cannot possibly cover everything.

Exit, saving the configuration.

Compiling the kernel

If you're not there already, cd into your linux source folder then compile the kernel:

1. cd linux
2. make ARCH=arm
3. make ARCH=arm INSTALL_MOD_PATH=../modules modules_install

Copy the compiled kernel to your working directory:

1. cp arch/arm/boot/zImage ../

Done! You should name have a QEMU-bootable linux kernel and modules in your work directory.

Filed under: Kernel, Raspberry Pi, Tutorials Leave a comment

Comments (15) Trackbacks (2) (subscribe to comments on this post)

gunflame
July 17th, 2012 - 05:54

I'm very grateful for the tutorial, but I have been unable to compile my own working kernel.
I'm doing the compiling on my Phenom X6, Ubuntu 12.04 32 bits. It compiles just fine but during the boot process it stops halfway.

Could you help me a bit? I know the actual qemu works just fine because the kernel you posted works like a charm:
http://xecdesign.com/qemu-emulating-raspberry-pi-the-easy-way/
http://xecdesign.com/downloads/linux-qemu/kernel-qemu

My goal? very simple…. I want to change the raspberry logo into something much bigger (600×400 or so). For this I need to compile the kernel while changing the logo. The funny part is that the modified kernel works displays the new logo, but it stops 1/2 way through the booting sequence.

(REPLY)

Shift
November 24th, 2012 - 12:53

Sorry, don't have much experience in that area.

(REPLY)

Andrew
July 24th, 2012 - 18:11

I found I needed to set CONFIG_MMU otherwise the kernel would hang after compression but before the first kernel boot up message under QEMU.

(REPLY)

Shift
December 2nd, 2012 - 21:08

That's odd, I found it automatically enabled before I ran menuconfig.

(REPLY)

Mephisto
July 26th, 2012 - 08:50

Thank you very much! Great Tutorial

(REPLY)

Angel Genchev
November 5th, 2012 - 05:45

ThanX! That`s a lot of information. What do the striked-through lines from the emulated devices mean ?

(REPLY)

Shift
November 24th, 2012 - 12:55

Just that those options don't need to be manually enabled.

(REPLY)

madtrapper
November 15th, 2012 - 10:26

Why I can not use Qemu boot the kernel copy from Pi's SD card?
Thx

(REPLY)

Shift
November 24th, 2012 - 12:55

It has been a while since I tried, but I think that it is missing support for the hardware emulated by qemu.

(REPLY)

Stefan
December 11th, 2012 - 18:17

What a great tutorial. Couldn't find the QEMU emulated hardware specs anywhere before. Saved me Thx dude!

(REPLY)

Shift
December 11th, 2012 - 19:20

Thanks, let me know if you have any problems with it.

(REPLY)

John Lane
January 21st, 2013 - 09:31

I had to tinker with the kernel options to get the Raspberry Pi Arch Linux image (archlinux-hf-2012-09-18.img) to boot. I had to add "CONFIG_CGROUPS=y" becuase this is a requirement for systemd, which Arch Linux now uses. I also had to enable stuff so /boot could be mounted, otherwise the boot always drops to maintenance mode:

CONFIG_VFAT_FS=y
CONFIG_NLS_CODEPAGE_437=y
CONFIG_NLS_ISO8859_1=y

I have a script in my library that will build this kernel automatically. See "kernel/build-kernel-qemu" in my repository at https://github.com/johnlane/rpi-utils.git.

(REPLY)

Daniel
February 5th, 2013 - 06:29

Thank you so much for your build-kernel-qemu script!

(REPLY)

siaak

February 22nd, 2013 - 21:39

Is there any way to configure the kernel to allow for 512MB RAM, or is it a limitation of the versatilepb board?

(REPLY)

Shift

February 22nd, 2013 - 22:23

As far as I understand it it's the limitation of the board, however Torlus is working on a proper raspberry pi target for qemu and can already boot the official kernel and it quite usable. Google 'torlus qemu raspberry pi' if you're interested.

(REPLY)

Leave a comment

Name (required)

Email (required)

Website

Submit

QEMU – Compiling for ARM (1176) emulation »

Go to top ↑