**从零使用qemu模拟器搭建arm运行环境**

2015年07月04日 15:26:17 [海枫](https://me.csdn.net/linyt) 阅读数：49297 标签： [kernel](http://so.csdn.net/so/search/s.do?q=kernel&t=blog) [qemu](http://so.csdn.net/so/search/s.do?q=qemu&t=blog) [arm](http://so.csdn.net/so/search/s.do?q=arm&t=blog) [busybox](http://so.csdn.net/so/search/s.do?q=busybox&t=blog) 更多

个人分类： [kernel](https://blog.csdn.net/linyt/article/category/851992)

版权声明：本文为博主原创文章，承蒙转载请注明作者和出处 https://blog.csdn.net/linyt/article/details/42504975

**为什么会有这篇文章**

早在2011年的时候，跟当时同事一起讨论，做Linux系统开发正处于整个Linux开发中间层，没有上层的C/C++业务和数据库的开发经验，也没有底层的内核和驱动开发经验，到底路该如何走……基于对Linux系统开发的理解和内核的兴趣，认为选择Linux内核开发更适合自己。于是到淘宝上买了块三星s3c2440(arm 9)开发板，学起内核开发。没有过多久，机缘巧合，正式加入了公司的内核开发部。就这样跟内核和arm打上交道了。

没有想到这一做就是3年了，arm芯片只有在公司才能使用，回到家里就不能访问了。去年（2014）开始觉得做内核久了，应该向内核社区提交patch，提升知名度和影响力。但在公司提交patch不方便，于是在家里通过qemu方式搭建于ARM A9的运行环境，进行开发和测试验证，一口气提交了好几个patch（[link1](https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=9c986661638c69772f5479c4715061239ec61b29), [link2](https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=c2459d35f5b8de077fde859b8c96a23f309b9ab2), [link3](https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=20cb6abfe0062693183eb2a85f7c73792e96407f), [link4](https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=207a6cb06990c298d0eac982e053d370e216d93d), [link5](https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=279f487e0b471577e2b3c134e2ff9af939129d0f)）并被社区接纳了。

最近在梳理Linux内存机制，无论是[《深入理解Linux内核》](https://www.baidu.com/s?wd=%E3%80%8A%E6%B7%B1%E5%85%A5%E7%90%86%E8%A7%A3Linux%E5%86%85%E6%A0%B8%E3%80%8B&tn=24004469_oem_dg" \t "_blank)还是[《深入理解内核架构》](https://www.baidu.com/s?wd=%E3%80%8A%E6%B7%B1%E5%85%A5%E7%90%86%E8%A7%A3%E5%86%85%E6%A0%B8%E6%9E%B6%E6%9E%84%E3%80%8B&tn=24004469_oem_dg)这两本红宝书都无法告知你每个细节的时候，就需要查看代码细节，修改代码，甚致做行为分析。此时需要修改代码输出调试信息，以帮助更深入体会代码的逻辑。这该是qemu派上场的时候。

其实我[前前后后](https://www.baidu.com/s?wd=%E5%89%8D%E5%89%8D%E5%90%8E%E5%90%8E&tn=24004469_oem_dg)搭建qemu+arm的运行环境已超过5次了，每次都要花上很多时间。碰巧昨天有同事看到我再次搭建，他如搭珍宝，告诉我一定要将搭建qemu的方法告诉他。

所以，如果你想买个开发板来做arm + linux嵌入式开发，完全可以使用qemu进行开发。或者你像我一样，对内核机制关心，而不关心于具体的外设器件，最多是关心arm架构相关的功能，也可以使用qemu进行开发。

一句话：搭建qemu+arm环境，用于做内核开发和功能分析调试。

**搭建环境**

我整个搭建过程都是在笔记本上进行的，Ubuntu 12.04系统；如果是Fodera环境，搭建过程可能略有不同，但关键步骤是不变的。

qemu模拟得最好的arm芯片，要数ARM公司的vexpress A9开发板了，本文的搭建过程都是围绕这个开发板进行的。当然，如果你想搭其它开发板，也不难，只要qemu和内核对它有成熟的支持就够了。

下面是step by step的搭建过程，建议没有特别诉求的朋友，按照下在面的步骤操作。或者先根据下面的步骤成功搭建vexpress运行环境之后，再根据自己的需求进行更改。

**下载Linux内核**

下载内核有两种方法，一种是用git直接下载内核代码树，方便后面的内核开发。另一种是直接到内核社区下载稳定版本（详见：https://www.kernel.org/pub/linux/kernel/v4.x/）。下面演示从Linux kernel主线下载代码进行编译。

git clonegit://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git

整个下载过程会持续半个小时左右，有足够时间来泡杯茶。

**安装arm的交叉编译工具链**

想必做嵌入式开发的朋友，对交叉编译工具链不陌生。如果你订制一个交叉编译工具链，建议你使用[crosstool-ng](http://crosstool-ng.org/)开源软件来构建。但在这里建议直接安装arm的交叉编译工具链：

sudo apt-get install gcc-arm-linux-gnueabi

**编译Linux内核**

生成vexpress开发板子的config文件：

make CROSS\_COMPILE=arm-linux-gnueabi- ARCH=arm vexpress\_defconfig

编译：

make CROSS\_COMPILE=arm-linux-gnueabi- ARCH=arm

生成的内核镱像位于arch/arm/boot/zImage， 后续qemu启动时需要使用该镜像。

**下载和安装qemu模拟器**

其实Ubuntu 12.04有qemu的安装包，但由于版本较低，对vexpress开发板支持不友好，建议下载高版本的qemu:

wget http://wiki.qemu-project.org/download/qemu-2.0.2.tar.bz2

配置qemu前，需要安装几个软件包：

sudo apt-get install zlib1g-dev

sudo apt-get install libglib2.0-0

sudo apt-get install libglib2.0-dev

配置qemu，支持模拟arm架构下的所有单板：

./configure --target-list=arm-softmmu --audio-drv-list=

编译和安装：

make

make install

**测试qemu和内核能否运行成功**

qemu已经安装好了，内核也编译成功了，到这里最好是测试一下，编译出来的内核是否OK，或者qemu对vexpress单板支持是否够友好。

运行命令很简单：

qemu-system-arm -M vexpress-a9 -m 512M -kernel **/path/to/kernel/dir/**arch/arm/boot/zImage -dtb  **/path/to/kernel/dir/**arch/arm/boot/dts/vexpress-v2p-ca9.dtb -nographic -append "console=ttyAMA0"

这里的 **/path/to/kernel/dir/** 是内核的下载目录，或者编译目录。

如果看到内核启动过程中的打印，说明前面的搭建是成功的。

这里简单介绍下qemu命令的参数：

-M vexpress-a9 模拟vexpress-a9单板，你可以使用-M ?参数来获取该qemu版本支持的所有单板

-m 512M 单板运行物理内存512M

-kernel **/path/to/kernel/dir/**arch/arm/boot/zImage  告诉qemu单板运行内核镜像路径

-nographic 不使用图形化界面，只使用串口

-append "console=ttyAMA0" 内核启动参数，这里告诉内核vexpress单板运行，串口设备是那个tty。

注意：

我每次搭建，都忘了内核启动参数中的console=参数应该填上哪个tty，因为不同单板串口驱动类型不尽相同，创建的tty设备名当然也是不相同的。那vexpress单板的tty设备名是哪个呢？ 其实这个值可以从生成的.config文件CONFIG\_CONSOLE宏找到。

如果搭建其它单板，需要注意内核启动参数的console=参数值，同样地，可从生成的.config文件中找到。

**制作根文件系统**

到这里是否[大功告成](https://www.baidu.com/s?wd=%E5%A4%A7%E5%8A%9F%E5%91%8A%E6%88%90&tn=24004469_oem_dg" \t "_blank)了呢？ 其实在上面的测试中，你会发现内核报panic，因为内核找不到根文件系统，无法启init进程。

根文件系统要考虑两个方面：

1. 根文件系统的内容

    如果你看过《[Linux From Scratch](http://www.linuxfromscratch.org/lfs/view/stable/" \t "_blank)》，相信你会对这一步产生恐惧感，但如果一直从事嵌入式开发，就可以放下心来。根文件系统就是简单得不能再简单的几个命令集和态动态而已。为什么Linux From Scratch会有那么复杂，是因为它要制作出一个Linux发生版。但在嵌入式领域，几乎所有的东西，都是mini版本，根文件系统也不例外。

   本文制本的根文件系统 = busybox(包含基础的Linux命令)  + 运行库 + 几个字符设备

2. 根文件系统放在哪里

    其实依赖于每个开发板支持的存储设备，可以放到Nor Flash上，也可以放到SD卡，甚至外部磁盘上。最关键的一点是你要清楚知道开发板有什么存储设备。

    本文直接使用SD卡做为存储空间，文件格式为ext3格式

**下载、编译和安装busybox**

 wget http://www.busybox.net/downloads/busybox-1.20.2.tar.bz2

make defconfig

make CROSS\_COMPILE=arm-linux-gnueabi-

make install CROSS\_COMPILE=arm-linux-gnueabi-

安装完成后，会在busybox目录下生成\_install目录，该目录下的程序就是单板运行所需要的命令。

**形成根目录结构**

先在Ubuntu主机环境下，形成目录结构，里面存放的文件和目录与单板上运行所需要的目录结构完全一样，然后再打包成镜像（在开发板看来就是SD卡），这个临时的目录结构称为根目录

1.  创建rootfs目录（根目录），根文件系统内的文件全部放到这里

mkdir -p rootfs/{dev,etc/init.d,lib}

2. 拷贝busybox命令到根目录下

sudo cp busybox-1.20.2/\_install/\* -r rootfs/

3. 从工具链中拷贝运行库到lib目录下

sudo cp -P /usr/arm-linux-gnueabi/lib/\* rootfs/lib/

4. 创建4个tty端终设备

sudo mknod rootfs/dev/tty1 c 4 1

sudo mknod rootfs/dev/tty2 c 4 2

sudo mknod rootfs/dev/tty3 c 4 3

sudo mknod rootfs/dev/tty4 c 4 4

**制作根文件系统镜像**

1. 生成32M大小的镜像

 dd if=/dev/zero of=a9rootfs.ext3 bs=1M count=32

2. 格式化成ext3文件系统

mkfs.ext3 a9rootfs.ext3

3.  将文件拷贝到镜像中

sudo mkdir tmpfs

sudo mount -t ext3 a9rootfs.ext3 tmpfs/ -o loop

sudo cp -r rootfs/\*  tmpfs/

sudo umount tmpfs

**系统启动运行**

完成上述所有步骤之后，就可以启动qemu来模拟vexpress开发板了，命令参数如下：

qemu-system-arm -M vexpress-a9 -m 512M -kernel **/path/to/kernel/dir/**arch/arm/boot/zImage -dtb**/path/to/kernel/dir/**arch/arm/boot/dts/vexpress-v2p-ca9.dtb -nographic -append "**root=/dev/mmcblk0** console=ttyAMA0" **-sd a9rootfs.ext3**

**从内核启动打印，到命令行提示符出现，激动人心的时刻出现了……**

**写在后面的话**

通过上面的步骤，搭建出来一个最小的qemu+arm开发环境，你可以上面的基础上修改内核，或者增加一些测试程序在单板上运行，甚至使用单板的flash设备。

在此，你可以做纯arm架构的内核开发，或者与架构无关的内核开发，也可以做单板相关的驱动开发。  
  
**最后的最后**：本文已经过多次修改，如果各位读者按照本文方法搭建遇到问题，还请各位反馈给我(lyt2008bj at 163 dot com)，我会修改直至完全成功。