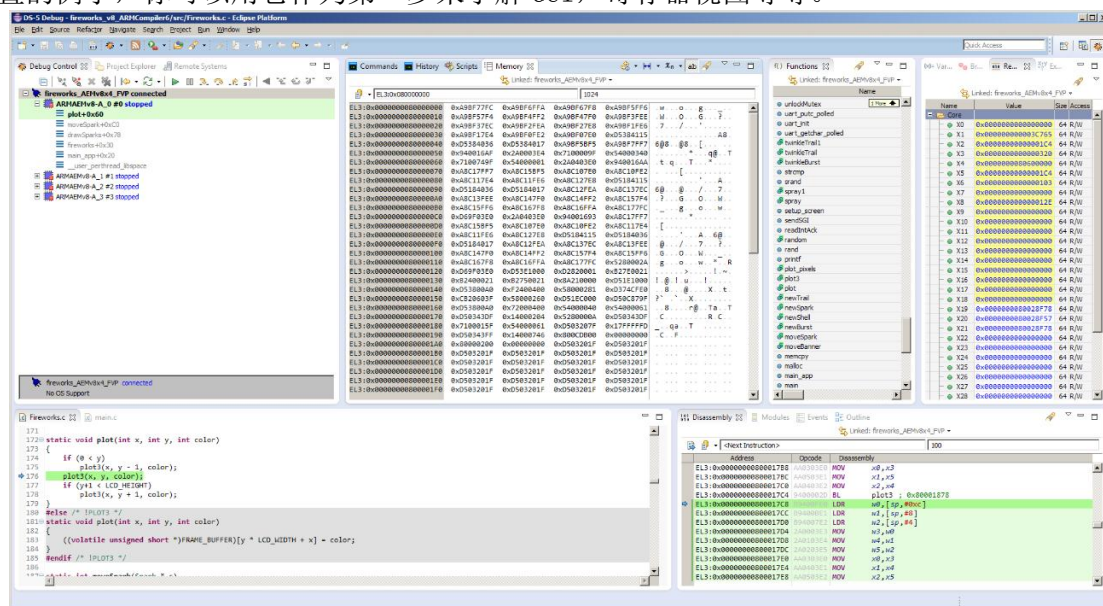


作者原文的名字叫利用 DS-5 旗舰版中的 ARMv8-A 模型启动 Linux, 我觉得名字起的太长, 蛋疼, 就顺手改了, 原文地址[在这](#), 下面是中文翻译。

ARM 最近推出的 DS-5 旗舰版添加了对最新 ARMv8-A 架构和相关 ARM 处理器的支持。安装包中包含一个 ARMv8 (FVP) 模型, 让你可以在没有 ARMv8 硬件之前就可以进行软件开发。我将在本文中向你说明让 Linux 在 FVP 上运行的步骤。

首先, 如果你是一个 DS-5 的新手, DS-5 中提供了一些裸机例子, 你可以使用这些样例来确认一切配置正确。你应该能从 DS-5 的 File 菜单中用 Import... → General → Existing Projects into Workspace 将 Fireworks 的例子从 `\\DS-5\\examples\\Bare-metal_examples_ARMv8.zip` 中导入。这个例子中包含了 Debug 配置, 你可以通过 Debug 控制面板在 FVP 上运行这个例子。这是一个预配置的例子, 你可以用它作为第一步来了解 GUI, 寄存器视图等等。



如果一切正常, 我们可以开始将 Linaro 的 Linux 安装到 FVP 上。最新的 ARMv8-A 的 Linux 分支可以在[这里](#)找到, 基本每个月都会有更新。如果你只是想在 FVP 上启动 Linux, 你可以使用预编译的文件。你需要获得内核文件 `img.axf` 和对应的文件系统文件。下面是 2 个对应的文件系统文件, 需要在主机上解压才能使用。

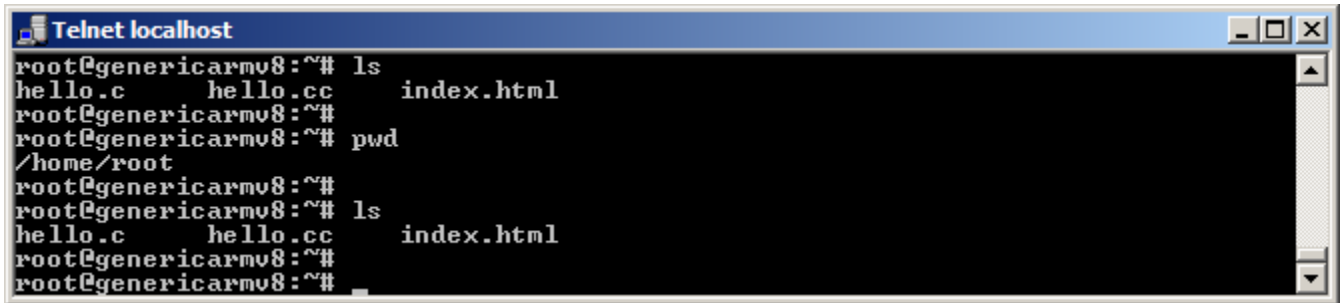
`vexpress64-openembedded_minimal-armv8-GCC-<version_and_date>.img.gz` 是一个最小的文件系统 (~80MB)

`vexpress64-openembedded_lamp-armv8-GCC-<version_and_date>.img.gz` 是一个全功能的文件系统 (~430MB)

我们可以独立启动, 也可以使用 debugger 启动 FVP, 步骤几乎一致。对于独立启动, 我建议创建一个批处理文件, 运行下面的命令行。需要注意到把下面的 Kernel 和文件系统文件位置改成你对应的位置:

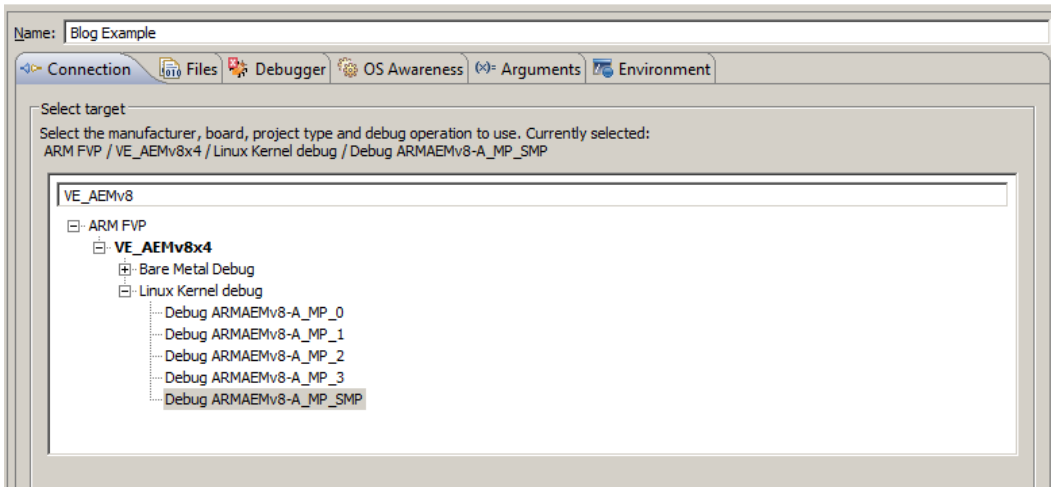
```
<path_to> \ DS-5 \ SW \ models \ BIN \ FVP_VE_ARMv8A.exe \
-a <path_to> img.axf \
- " <path_to> \ <unzipped_filesystem_image> img" \
--parameter motherboard.mmc.p_mmc_file = \
--parameter motherboard.mmc.card_type =的 eMMC \
--parameter motherboard.smc_91c111.enabled=true\
--parameter motherboard.hostbridge.userNetworking =true\
```

--parameter motherboard.hostbridge.userNetPorts = "5555 = 5555,8080 = 8080,22 = 22"
有关这些选项的说明，请参阅最新 FVP 文档。从各种机器测试来看，大约需要 60-120 秒开机的最小文件系统映像，约 4-7 分钟启动了完整的 LAMP 文件系统。

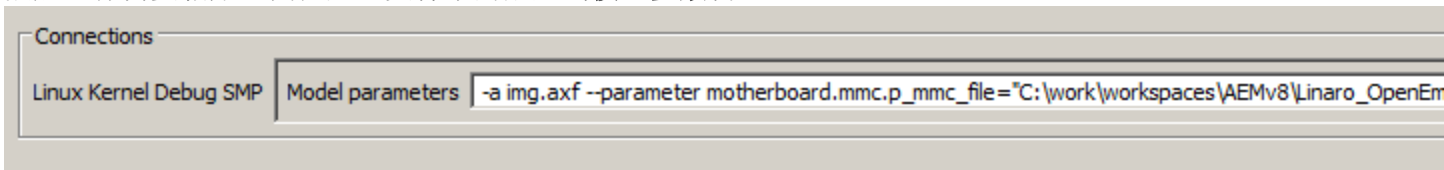


```
Telnet localhost
root@genericarmv8:~# ls
hello.c      hello.cc      index.html
root@genericarmv8:~#
root@genericarmv8:~# pwd
/home/root
root@genericarmv8:~#
root@genericarmv8:~# ls
hello.c      hello.cc      index.html
root@genericarmv8:~#
root@genericarmv8:~#
```

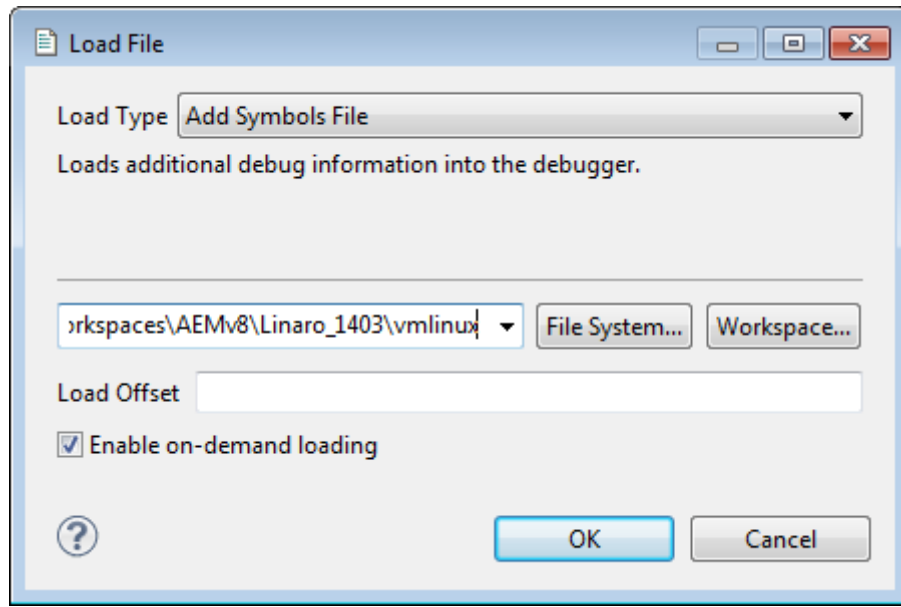
对于从 DS-5 调试器启动，从 Run Debug Configurations...，然后点击左上角的 New launch configuration 按钮。添加配置的名称，然后找到 VE_AEMv8。我发现很容易通过在 Filter 窗口中输入 VE_AEMv8 找到目标，并且这些工具会自动跳转到它。展开树形目录到 Linux Kernel Debug，然后选择 Debug ARMAEMv8-A_MP_SMP，连接到包含四个 CPU 的模型。



然后，你需要粘贴上面批处理文件中的配置到模型参数窗口。



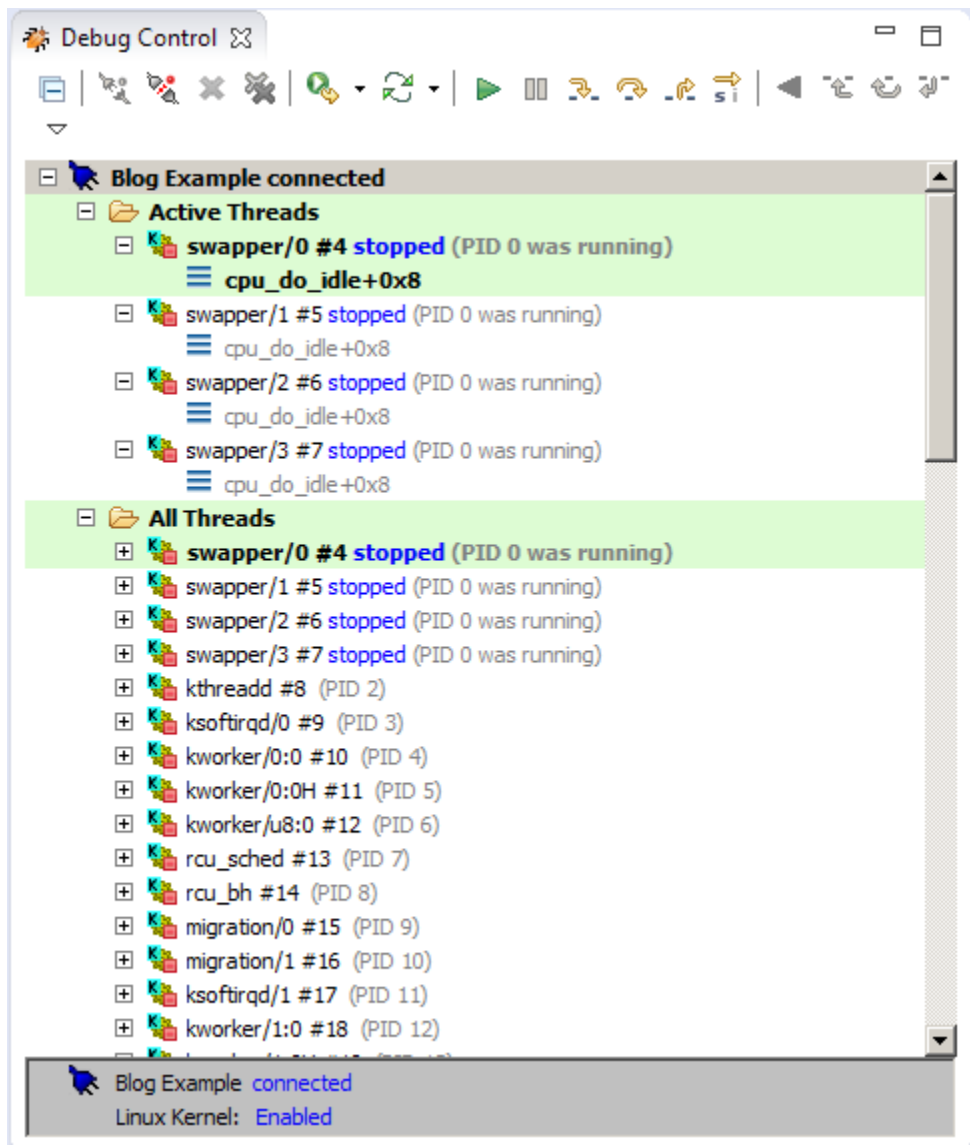
点击调试按钮，然后选择 Connect Only。点击 Debug 连接调试器，点击 go，让系统和前面一样启动。你现在也可以用调试器控制系统执行（启动/停止等）。如果你已经重新编译包含调试信息的内核（预构建映像不包含调试信息），你可以停止目标，现在使用调试控制面板中的加载功能



或从 CLI 输入:

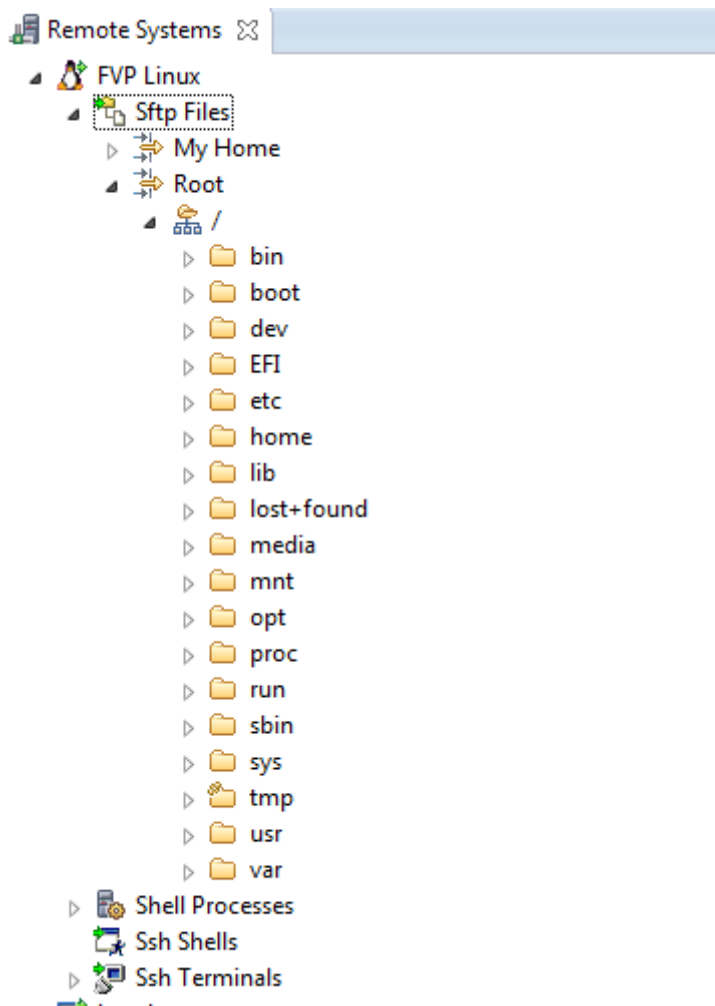
```
add-symbol-file <path_to>\vmlinux
```

调试器现在将显示在 FVP 上运行的所有线程的状态:



请注意，以下所有操作需要使用上述全功能的文件系统（或者你已经建立了自己一个相似系统）。

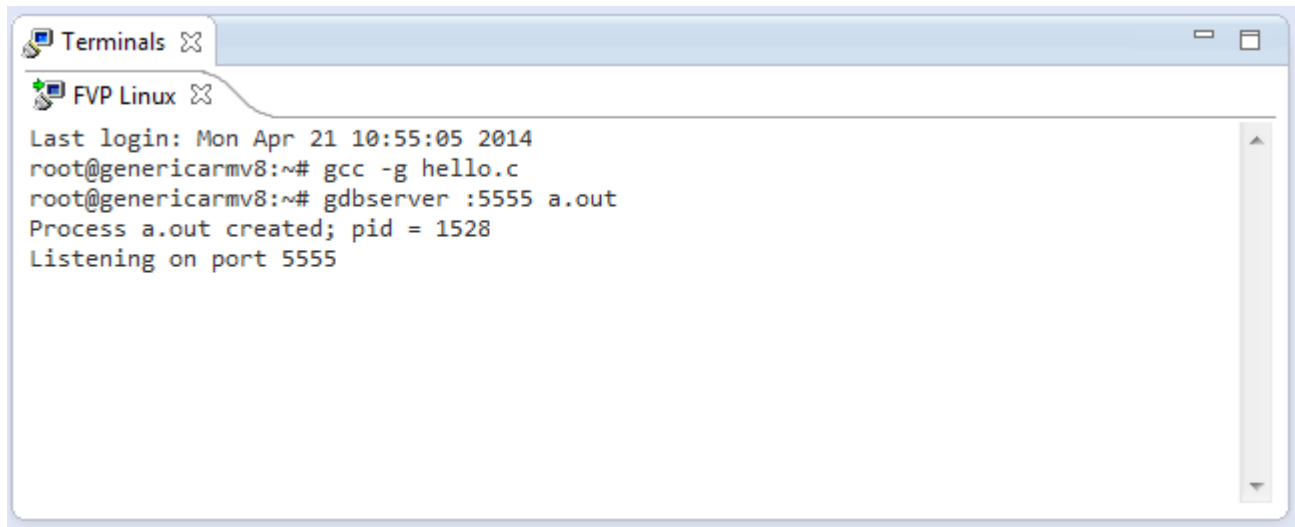
你可以创建一个远程系统查看文件系统，通过打开远程系统窗口中，创建一个（右击）→ New Connection → General → Linux 连接，设置主机名设置为 localhost，并给该连接的任意连接名称（下图中的 FVP Linux）。然后单击下一步，然后在配置部分中选择 ssh.files。单击 Finish（完成）。然后，你将能够扩展文件系统视图（如果有窗口弹出，设置用户名作为 root）。



你

从这个视图中，你也可以在 *ssh* 的终端单击鼠标右键，在 DS-5 的 GUI 中启动一个终端窗口，从而省去了来回跳转到 *telnet* 窗口。

最后，你也可以使用 DS-5 在 FVP 上调试应用程序。我在 FVP 上新建的 `hello.c`，用 `gcc -g hello.c` 进行编译生成 `a.out`。然后，我启动一个 `gdbserver` 的会话 “`gdbserver: 5555 a.out`”（你会发现这个端口是在启动模式时指定的模型参数）。然后我就可以创建一个应用程序调试配置，如下，调试应用。



Name: FVP Application Debug

Connection Files Debugger OS Awareness Arguments Environment

Select target

Select the manufacturer, board, project type and debug operation to use. Currently selected:

Linux Application Debug / Application Debug / Connections via AArch64 gdbserver / Connect to already running application

Filter platforms

- Linux Application Debug
 - Application Debug
 - Connections via AArch64 gdbserver
 - Connect to already running application
 - Download and debug application
 - Start gdbserver and debug target-resident application
 - Connections via gdbserver

DS-5 Debugger will connect to an already running gdbserver on the target system.

Connections

gdbserver (TCP)

Address: localhost

Port: 5555

☒ Use Extended Mode

☒ Terminate gdbserver on disconnect

Apply

Debug Control

FVP Application Debug connected

Active Threads

Thread 1528 #1 stopped on breakpoint #1

main

X:0x0000007FB7EB9D50

All Threads

FVP Application Debug connected
No OS Support

另外，如果应用程序是在 PC 上生成的，您可以使用“下载和调试”选项，用调试器通过远程系统的连接将应用程序下载到 FVP，。

