【网摘】FIT image介绍

2017年7月5日
16:49

1. 前言

Linux kernel在ARM架构中引入device tree（全称是flattened device tree，后续将会以FDT代称）的时候[1]，其实怀揣了一个Unify Kernel的梦想----同一个Image，可以支持多个不同的平台。随着新的ARM64架构将FDT列为必选项，并将和体系结构有关的代码剥离之后，这个梦想已经接近实现：

在编译linux kernel的时候，不必特意的指定具体的架构和SOC，只需要告诉kernel本次编译需要支持哪些板级的platform即可，最终将会生成一个Kernel image，以及多个和具体的板子（哪个架构、哪个SOC、哪个版型）有关的FDT image（dtb文件）。

bootloader在启动的时候，根据硬件环境，加载不同的dtb文件，即可使linux kernel运行在不同的硬件平台上，从而达到unify kernel的目标。

本文将基于嵌入式产品中普遍使用的u-boot，以其新的uImage格式（FIT image，Flattened uImage Tree）为例，介绍达到此目标的步骤，以及背后的思考和意义。

2. Legacy uImage

从u-boot的角度看，它要boot一个二进制文件（例如kernel Image），需要了解该文件的一些信息，例如：

该文件的类型，如kernel image、dtb文件、ramdisk image等等？

该文件需要放在memory的哪个位置（加载地址）？

该文件需要从memory哪个位置开始执行（执行地址）？

该文件是否有压缩？

该文件是否有一些完整性校验的信息（如CRC）？

等等

结合“X-010-UBOOT-使用booti命令启动kernel(Bubblegum-96平台)”中有关booti的例子，上面信息被隐含在我们的命令行中了，例如：

通过DFU工具，将指定的image文件下载到指定的memory地址，间接的指定了二进制文件加载地址；

booti命令本身，说明加载的文件类型是ARM64平台的Image文件；

通过booti的参数，可以指定Kernel Image、ramdisk、DTB文件的执行位置；

等等。

不过，这种做法缺点很明显（总结来说，就是太啰嗦了）：

需要往memory中搬不同的二进制文件（Kernel、DTB、ramdisk等）；

boot指令（booti等）有比较复杂的参数；

无法灵活地处理二进制文件的校验、解压缩等操作；

等等。

为了解决上述缺点，u-boot自定义了一种Image格式----uImage。最初的时候，uImage的格式比较简单，就是为二进制文件加上一个header（具体可参考“include/image.h”中的定义），标示该文件的特性。然后在boot该类型的Image时，从header中读取所需的信息，按照指示，进行相应的动作即可。这种原始的Image格式，称作Legacy uImage，其特征可总结为：

1）使用mkimage工具（位于u-boot source code的tools/mkimage中）生成。

2）支持OS Kernel Images、RAMDisk Images等多种类型的Image。

3）支持gzip、bzip2等压缩算法。

4）支持CRC32 checksums。

5）等等。

最后，之所以称作Legacy，说明又有新花样了，这种旧的方式，我们就不再过多关注了，拥抱新事物去吧。

3. FIT uImage

3.1 简介

device tree在ARM架构中普及之后，u-boot也马上跟进、大力支持，毕竟，美好的Unify kernel的理想，需要bootloader的成全。为了支持基于device tree的unify kernel，u-boot需要一种新的Image格式，这种格

式需要具备如下能力：

1）Image中需要包含多个dtb文件。

2）可以方便的选择使用哪个dtb文件boot kernel。

综合上面的需求，u-boot推出了全新的image格式----FIT uImage，其中FIT是flattened image tree的简称。是不是觉得FIT和FDT（flattened device tree）有点像？没错，它利用了Device Tree Source files（DTS）的语法，生成的image文件也和dtb文件类似（称作itb），下面我们会详细描述。

3.2 思路

为了简单，我们可以直接把FIT uImage类比为device tree的dtb文件，其生成和使用过程为[2]：

```
image source file          mkimage + dtc                              transfer to target
                  +                  ----------------------------> image file ----------------------------------> bootm
image data file(s)
```

其中image source file(.its)和device tree source file(.dts)类似，负责描述要生成的image file的信息（上面第2章描述的信息）。mkimage和dtc工具，可以将.its文件以及对应的image data file，打包成一个image file。我们将这个文件下载到么memory中，使用bootm命令就可以执行了。

3.3 image source file的语法

image source file的语法和device tree source file完全一样（可参考[3][4][5]中的例子），只不过自定义了一些特有的节点，包括images、configurations等。说明如下：

1）images节点

指定所要包含的二进制文件，可以指定多种类型的多个文件，例如multi.its[5]中的包含了3个kernel image、2个ramdisk image、2个fdt image。每个文件都是images下的一个子node，例如：

```
kernel@2 {
    description = "2.6.23-denx";
    data = /incbin/("./2.6.23-denx.bin.gz");
    type = "kernel";
    arch = "ppc";
    os = "linux";
    compression = "gzip";
    load = <00000000>;
    entry = <00000000>;
    hash@1 {
        algo = "sha1";
    };
};
```

可以包括如下的关键字：

description，描述，可以随便写；

data，二进制文件的路径，格式为----/incbin/("path/to/data/file.bin")；

type，二进制文件的类型，"kernel", "ramdisk", "flat_dt"等，具体可参考中[6]的介绍；

arch，平台类型，“arm”，“i386”等，具体可参考中[6]的介绍；

os，操作系统类型，linux、vxworks等，具体可参考中[6]的介绍；

compression，二进制文件的压缩格式，u-boot会按照执行的格式解压；

load，二进制文件的加载位置，u-boot会把它copy对应的地址上；

entry，二进制文件入口地址，一般kernel Image需要提供，u-boot会跳转到该地址上执行；

hash，使用的数据校验算法。

2）configurations

可以将不同类型的二进制文件，根据不同的场景，组合起来，形成一个个的配置项，u-boot在boot的时候，以配置项为单位加载、执行，这样就可以根据不同的场景，方便的选择不同的配置，实现unify kernel目标。还以multi.its[5]为例，

```
configurations {
    default = "config@1";

     config@1 {
```

```
            description = "tqm5200 vanilla-2.6.23 configuration";
            kernel = "kernel@1";
            ramdisk = "ramdisk@1";
            fdt = "fdt@1";
        };

        config@2 {
            description = "tqm5200s denx-2.6.23 configuration";
            kernel = "kernel@2";
            ramdisk = "ramdisk@1";
            fdt = "fdt@2";
        };

        config@3 {
            description = "tqm5200s denx-2.4.25 configuration";
            kernel = "kernel@3";
            ramdisk = "ramdisk@2";
        };
};
```

它包含了3种配置，每种配置使用了不同的kernel、ramdisk和fdt，默认配置项由“default”指定，当然也可以在运行时指定。

3.4 Image的编译和使用

FIT uImage的编译过程很简单，根据实际情况，编写image source file之后（假设名称为kernel_fdt.its），在命令行使用mkimage工具编译即可：

```
$ mkimage -f kernel_fdt.its kernel_fdt.itb
```

其中-f指定需要编译的source文件，并在后面指定需要生成的image文件（一般以.itb为后缀，例如kernel_fdt.itb）。

Image文件生成后，也可以使用mkimage命令查看它的信息：

```
$ mkimage -l kernel.itb
```

最后，我们可以使用dfu工具将生成的.idb文件，下载的memory的某个地址（没有特殊要求，例如0x100000），然后使用bootm命令即可启动，步骤包括：

1）使用iminfo命令，查看memory中存在的images和configurations。

2）使用bootm命令，执行默认配置，或者指定配置。

使用默认配置启动的话，可以直接使用bootm：

```
bootm 0x100000
```

选择其它配置的话，可以指定配置名:

```
bootm 0x100000#config@2
```

以上可参考“doc/uImage.FIT/howto.txt[2]”，具体细节我们会在后续的文章中结合实例说明。

4. 总结

本文简单的介绍了u-boot为了实现Unify kernel所做的努力，但有一个问题，大家可以思考一下：bootloader的unify怎么保证呢？

SOC厂家提供（固化、提供二进制文件等）？

格式统一，UEFI？

后面有时间的话，可以追着这个疑问研究研究。

5. 参考文档

[1] Device Tree（一）：背景介绍

[2] doc/uImage.FIT/howto.txt

[3] doc/uImage.FIT/kernel.its

[4] doc/uImage.FIT/kernel_fdt.its

[5] doc/uImage.FIT/multi.its

[6] doc/uImage.FIT/source_file_format.txt

源文档 <http://www.wowotech.net/u-boot/fit_image_overview.html>