Bootloader開發

* 摘要

**啟動程式**（boot loader）位於[電腦](https://zh.wikipedia.org/wiki/%E9%9B%BB%E8%85%A6" \o "電腦)或其他計算機應用上，是指啟動[作業系統](https://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)的程式，在嵌入式系統中，功用為初始化硬體、最終目標啟動kernel(在此專案為linux kernel)。之前在開發版上使用過u-boot當作嵌入式linux的啟動程式，並且花了一點時間去閱讀源碼，理解其架構，這次就利用對其的理解，試著做一個小型的bootloader，功能大致有初始化硬體ex: 看門狗、SDRAM、Nand flash等，在最後透linux 開發源碼中setup.h文件裡面的tag結構傳遞內核參數啟動內核。

* 目錄

1. 嵌入式linux 結構與分布

2. 規劃bootloader memory

3. 建構鏈接腳本

4. Bootloadr 第一階段

5. Bootloader 第二階段

1. 嵌入式linux 結構與分布

一般情況下嵌入式Linux系統中的軟件主要分為以下幾部分：

1. **引導程式**：此部分有兩個階段，第一個階段為固化ROM是晶片廠家在出產的時候燒入到ROM裡面的啟動程式，例如有些開發版可以選擇nand flash啟動或者是 nor flash啟動。

2. **Linux內核和驅動程式**。

3. **文件系統**:

包括根文件系統和建立在Flash設備上的文件系統（EXT4，

UBI，CRAMFS等等）。提供管理系統的各種配置文件以及系統執行用戶應

用程序的良好運行環境及載體。

4. **應用程序**。用戶自定義的應用程序，存放於文件系統之中。

2. 規劃bootloader memory

Bootloader裝載的開發版為S32440 Soc，具有nand flash 及 nor flash可以存放程式碼，容量大小分別為64M(nand)、2M(Nor)。我選擇使用較大容量的nandflash放置我們的bootloader程式，並且驅動nandflash 將存放在nand底下的內核(uImage)讀到SDRAM執行

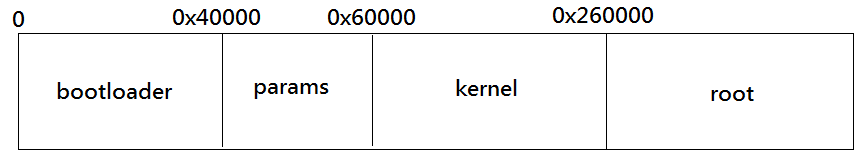


圖1 Nand flash memory 大致規劃

我所使用的開發版在nand flash 啟動時 廠家固化ROM裡面的bootloader會將nand flash前面4K大小的程式載入到內部RAM運行啟動，如果說程式碼大於4K就有必要驅動SDRAM(外部的RAM)將程式碼從nand flash讀到SDRAM運行。

由上面的結果可以得知，我們的內核大小為2M是無法在內部RAM上自動運行的需要我們驅動nand flash將nand flash 0x60000地址存放的內核讀到SDRAM上運行我們在此專案內核鏈接在SDRAM的地址為0x30007FC0。詳細規劃如圖2所示

3. 建構鏈接腳本

有了初步的記憶體空間規畫，便可開始撰寫鏈接腳本，以下為CSDN對鏈接腳本的敘述。

GNU-ld鏈接腳本淺析

每一個鏈接過程都由鏈接腳本(linker script, 一般以lds作為文件的後綴名)控制. 鏈接腳本主要用於規定如何把輸入文件內的section放入輸出文件內, 並控制輸出文件內各部分在程序地址空間內的佈局，連接器有個默認的內置連接腳本，-T選項用以指定自己的鏈接腳本, 它將代替默認的連接腳本。你也可以使用<暗含的連接腳本>以增加自定義的鏈接命令。

由以上的敘述加上之前規畫的記憶體地址我們bootloader決定放在SDRAM叫頂部的位置，避免與內核有衝突

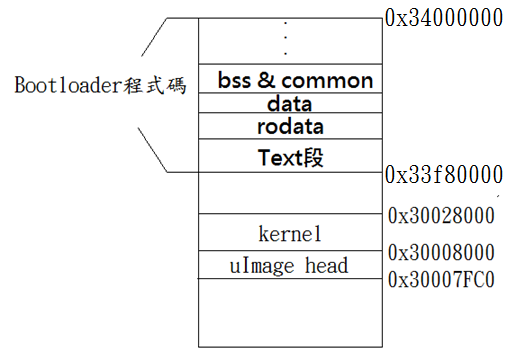


圖2 SDRAM memory 分布圖

4. Bootloadr 第一階段

在多數的單晶片程式中，一開始的程式碼片段多半都是驅動一些必要的硬體，且是用組合語言完成，我們撰寫的bootloader也不意外，在啟動時必須先使用組合語言驅動需要使用的硬體設備，在本階段中分別是關閉看門狗、初始化時鐘頻率(因為需要用到UART做調適)、初始化SDRAM、最後要設置好SP也就是堆疊指針才可以跳轉到C語言程式碼完成更複雜的硬體初始化。

在C語言程式碼中，我實現的nand flash的操作並且將程式碼從nand flash中將bootloader程式碼讀到bootloader所在的鏈接腳本(0x33f80000)，使程式可以在SDRAM用更大的RAM順利執行。

5. Bootloader 第二階段

在第一階段的bootloadr初始化一些必要的硬體設備，接下來就只剩下啟動linux kernel了，在這個階段，我將參考u-boot跳轉linux kernel的程式碼結構撰寫。以下為撰寫的程式碼片段



圖3 Bootloader 第二階段程式碼

從u-boot中可以得知，再啟動linux 內核時需要將傳遞的參數存放在某個地址，當內核啟動後程式將會到該地址取出參數並且按照參數的設定順利啟動內核。並且得知內核參數會用一種特定的結構體保存此結構圖如下所示

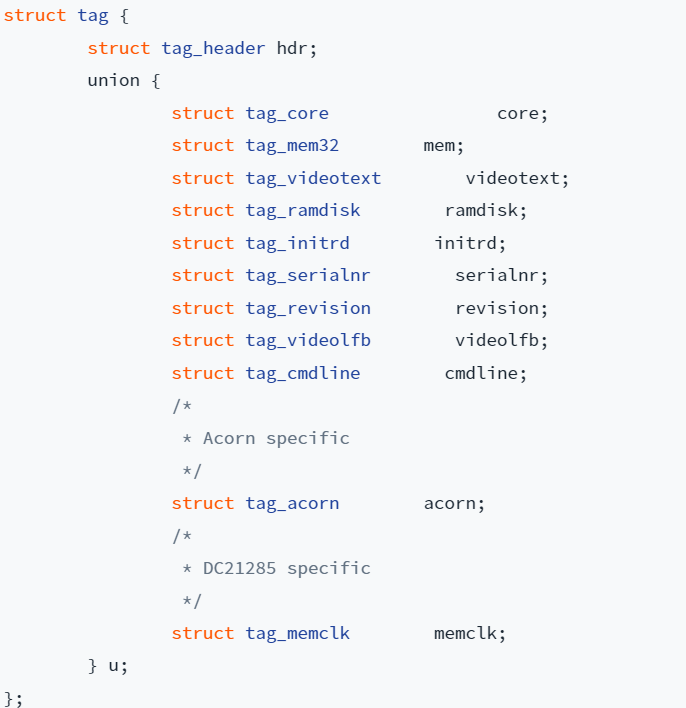


圖4 Tag結構體

透過tag標籤結構體，我們可以存放參數在特定的地址，內核可以從這個地址讀取參數啟動，我傳遞的內核參數分別為起始標籤、記憶體標籤、命令標籤、結束標籤，分別對應setup\_start\_tag、setup\_memory\_tags、setup\_commandline\_tag、setup\_end\_tag函式。

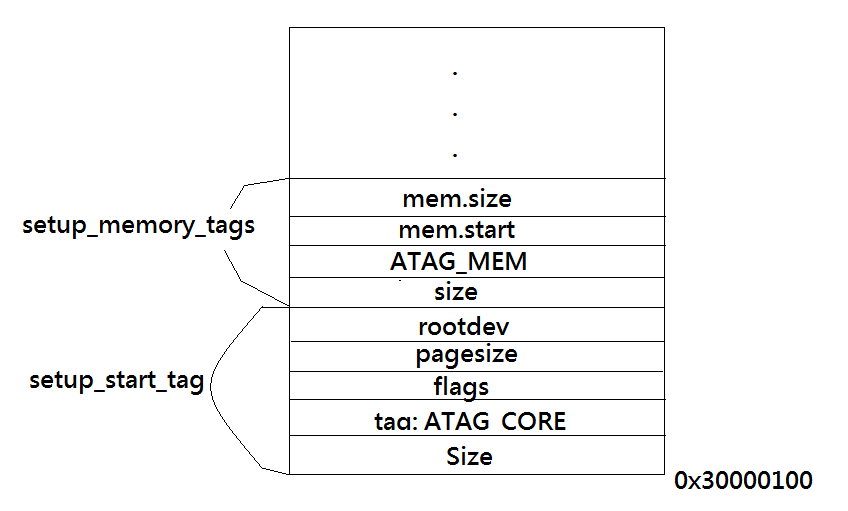


圖 5 SDRAM tag memory 分佈

最後透過以下函式啟動內核

void (\*theKernel)(int, int, unsigned int);

theKernel = (void (\*)(int, int, unsigned int))(kernel\_addr + 64);

theKernel(0,MACH\_TYPE\_S3C2440,params\_addr);

至於為什麼要跳轉到地址0x30007FC0偏移64bytes的地址運行內核呢?

原來是內核編譯出來的映象檔案室uImage格式

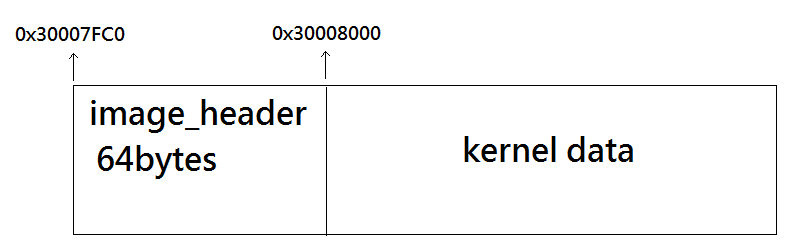


圖6 uImage格式

在uImage格式中會在內核資料前面64bytes空間填入一些資訊，例如內核運行的地址等等，但是因為我完成的bootloader為一個簡單的版本，並沒有去分析開頭64bytes uImage header的資訊，因此跳轉到主要的內核地址運行即可。

參考文獻

1. [GNU-ld鏈接腳本淺析](http://blog.csdn.net/yili_xie/article/details/5692007)

2. [u-boot 1.1.6](https://www.denx.de/wiki/U-Boot)

3. [s3c2440 datasheet](https://www.rockbox.org/wiki/pub/Main/DataSheets/um_s3c2440a_rev10.pdf)