Linux Driver編寫流程:

在Linux底下一切接文件，包含驅動程式。由於Linux Driver跑在Kernel Space，而Linux應用程式跑在User Space，所以我們需要個別編寫。

再者常見的Linux Driver分為三類 : 分別是字元設備(Char Device)，塊設備(Block Device)以及網路設備(Network Device)

* **字符設備驅動**（Character Device Drivers）：這類設備驅動通常用於處理按字節流讀寫的設備，如串口、鍵盤、鼠標等。字符設備驅動提供了一組接口，允許應用程序直接對設備進行字節級別的操作。
* **塊設備驅動**（Block Device Drivers）：這類設備驅動主要用於處理按塊讀寫的設備，如硬盤、SSD等。塊設備驅動通常通過緩存和批處理來提高性能，允許操作系統和應用程序高效地進行大塊數據的讀寫操作。
* **網絡設備驅動**（Network Device Drivers）：這類設備驅動專門用於處理網絡接口設備，如以太網卡、Wi-Fi卡等。網絡設備驅動管理網絡數據包的發送和接收，並提供適當的接口以便操作系統進行網絡通信

1. 專案02\_led:

首先為了方便管理，Linux中每個設備都有一個設備號，設備號由主設備號和次設備號兩部分組成。主設備號表示某一個具體的驅動，次設備號表示使用這個驅動的各個設備。Linux提供了一個名為dev\_t的數據類型表示設備號，dev\_t定義在文件include/linux/types.h裡面。而在此驅動中則是以#define LED\_MAJOR和#define LED\_NAME去實作。(如下圖1)



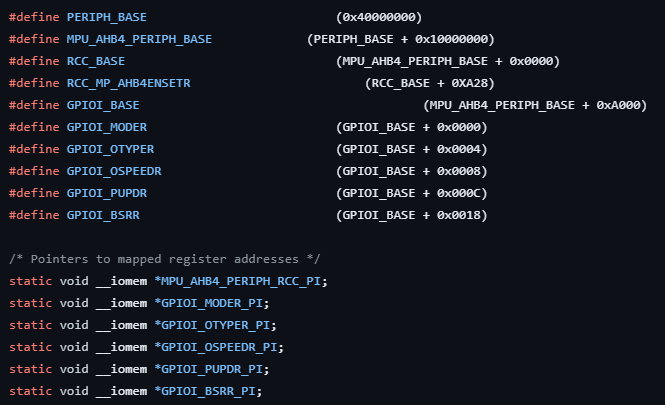
(圖1: 設備號和設備名稱)

1. 在此之前，我們需要先簡單了解一下MMU這個單元，MMU全稱叫做Memory Manage Unit，也就是內存管理單元。在舊版本的Linux中要求處理器必須有MMU，但是現在Linux內核已經支持無MMU的處理器了。MMU主要完成的功能如下：

* 完成虛擬空間到物理空間的映射。
* 內存保護，設置存儲器的訪問權限，設置虛擬存儲空間的緩衝特性。

系統裡面的對應虛擬地址，這裡就涉及到了物理內存和虛擬內存之間的轉換，需要用到兩個函數：ioremap 和 iounmap。

我們透過#define來去定義LED暫存器實際物理地址，接著以static void \_\_iomem去取得定義LED暫存器映射後的首地址(如下圖2\_1)



(圖2\_1:為ioremap後的地址)

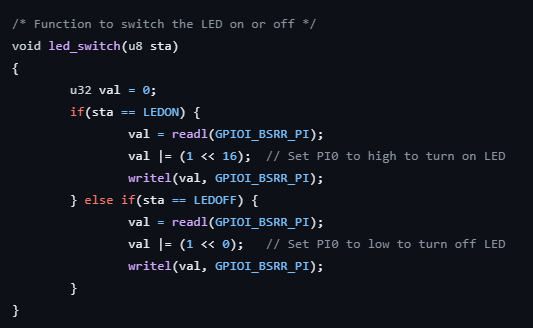
1. ioremap 函數用於獲取指定物理地址空間對應的虛擬地址空間，定義在 arch/arm/include/asm/io.h 文件中，接著我們在led\_init中呼叫ioremap函式

獲取LED物理地址空間對應的虛擬地址空間。此一來就可以進行LED的操作了。(如下圖2\_2)。



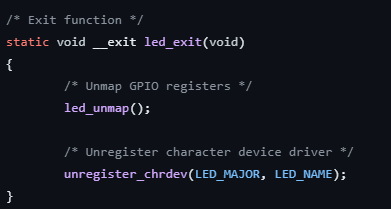
(圖2\_2:led\_init()函式)

1. 最後定義led\_switch函式來實作LED的亮滅操作。(如下圖\_3)

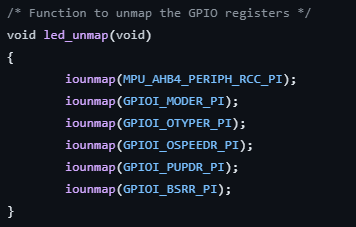


(圖\_3:led\_switch函式)

1. 當要卸載驅動時需要把剛剛的映射地址給取消映射，並且取消註冊字元設備。(如下圖\_4\_1 ~ 圖\_4\_2)



(圖\_4\_1:led\_exit()函式)



(圖\_4\_2:led\_unmap()函式)

※結論:

這個專案整體流程為:

1. 註冊驅動模塊入口(led\_init())和出口(led\_exit())函式。
2. 接著再led\_init()函式中實現地址映射(ioremap)，並且註冊字元設備。
3. 編寫led\_switch()函式來操作LED燈亮滅。
4. 最後led\_exit()時，取消地址映射iounmap，以及取消註冊字元設備。