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01\_chrdevbase.c

首先需要向Linux系統定義設備入口函數及出口函數，用於加載和卸載驅動模塊。(如下圖一)

chrdevbase\_init(void)函式:

主要功能是向Linux Kernel註冊字元設備

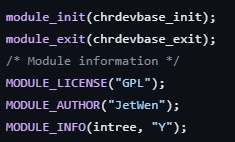
chrdevbase\_exit(void)函式:

主要功能是向Linux Kernel取消註冊字元設備



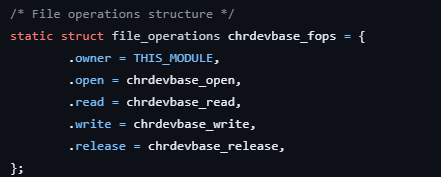
(圖一: 設備入口函數及出口函數)

1. 定義設備入口函數和出口函數完成後需要向Linux系統註冊此兩個函數，透過module\_init()和module\_exit()來實現。(如下圖二)，觀察圖二可以發現MODULE\_LISENCE(“GPL”)，這是用來宣稱模組適用Linux中GPL規範，而MODULE\_AUTHOR()則是作者訊息。



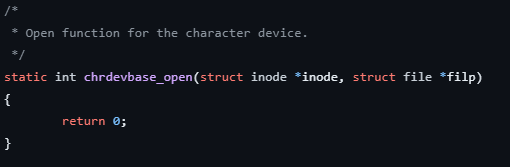
(圖二)

最後是溝通Kernel Space和User Space的函式，他定義在Linux Kernel文件include/linux/fs.h中有的file\_operations結構體，此結構體就是Linux Kernel驅動操作函数集合。而最常使用的就是open(), read(), write(), release()。(如下圖三)

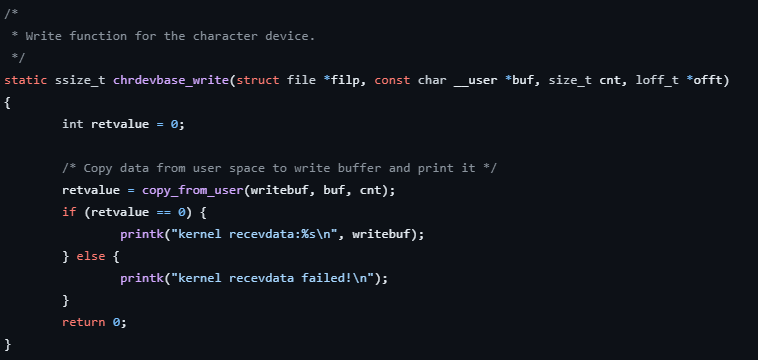


(圖三: file\_operations結構體)

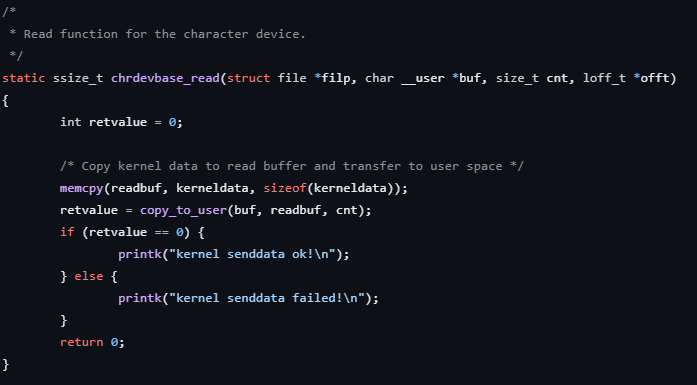
1. 我們還需要個別去定義結構體內的函式(如下圖四\_1 ~ 圖四\_4)。



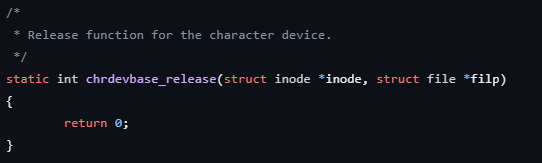
(圖四\_1:自定義open函式內容)



(圖四\_2:自定義write函式內容)



(圖四\_3:自定義read函式內容)



(圖四\_4:自定義release函式內容)

※總結:

如此一來我們就完成最基本的字元設備(Char Device)的驅動框架了。

幾乎所有字元設備(Char Device)驅動都包含以上內容。

主要流程為:

1. 向Linux Kernel註冊加載函式及卸載函式。
2. 並在加載函式中註冊字元設備及卸載函式中取消註冊字元設備。
3. 接著定義open(), read(), write(), release()函式內容。