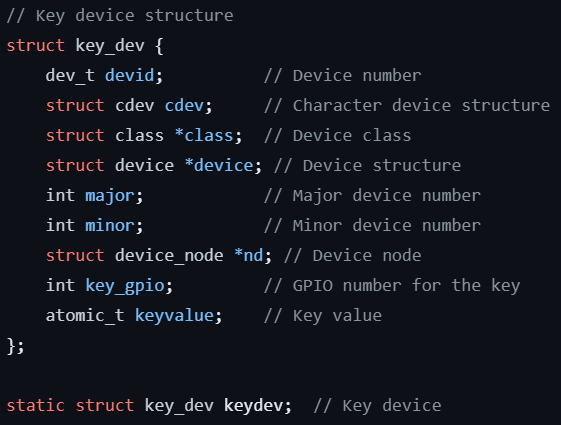
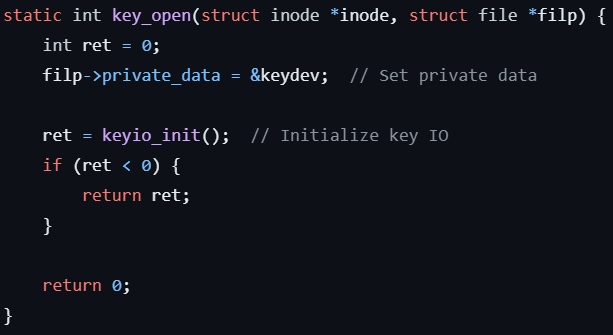
1. 專案11\_key:
2. 這個專案我們使用這些按鍵來完成GPIO輸入驅動程序，同時利用前幾個專案講的原子操作來對按鍵值進行保護。
3. 我們實現按鍵輸入，在驅動程序中使用一個整數變量來表示按鍵值，應用程式通過read()函數來讀取按鍵值，判斷按鍵有沒有按下。在這裡，這個保存按鍵值的變數就是個共享資源，驅動程序要向其寫入按鍵值，應用程序要讀取按鍵值。所以，我們要對其進行保護，對於整數變量而言我們首選的就是原子操作(atomic)，使用原子操作對變數進行賦值以及讀取。Linux下的按鍵驅動原理很簡單，接下來開始編寫驅動。
4. 補充一點，本專案只是為了演示Linux下GPIO輸入驅動的編寫，實際中的按鍵驅動並不會採用本專案中所講解的方法，Linux下的input子系統專門用於輸入設備！
5. 驅動編寫:
6. 設備結構體，特別加入key\_gpio成員變數紀錄GPIO number，和keyvalue成員變數來記錄按鍵key的值。(如下圖一)



(圖一:設備結構體)

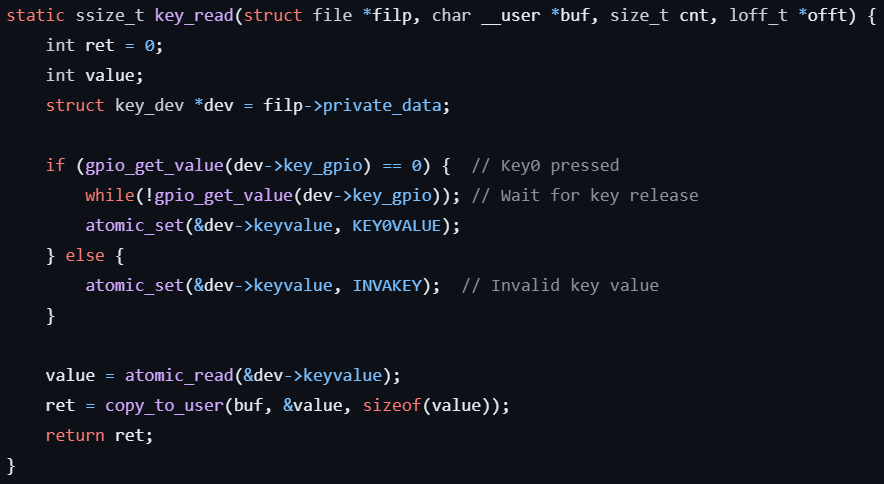
1. 在open()函式中把所有設備結構體資料，儲存到private\_data中以供應用程式使用。以及keyio\_init()函式來初始化按鍵key。

(keyio\_init()函數內容，如圖五，圖六，圖七所示)



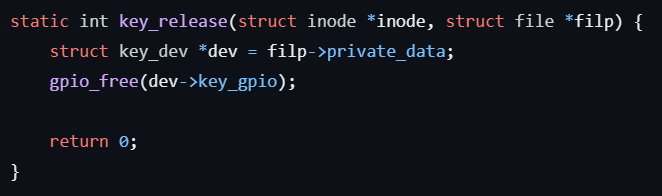
(圖二:open()函式內容)

1. 在read()函式中以gpio\_get\_value()函式獲取gpio的voltage，透過atomic\_set()函式來把按鍵值儲存到設備結構體的keyvalue成員變數中。再透過atomic\_read()函式把按鍵值讀取出來。最後以copy\_to\_user()函式把讀取到的值傳到User Space。



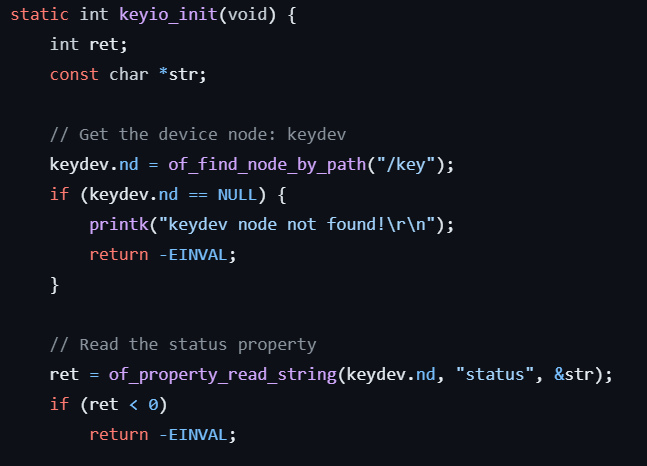
(圖三:read()函式內容)

1. release()函式，只有gpio\_free()函數去把釋放key\_gpio成員變數。

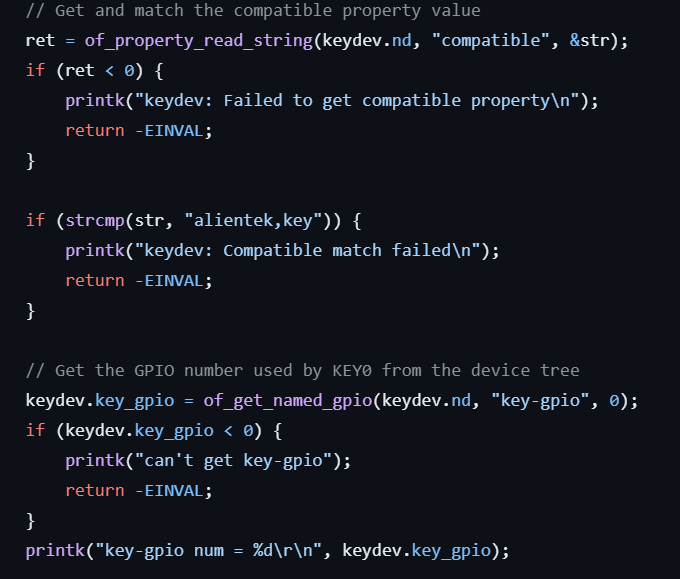


(圖四:release()函式內容)

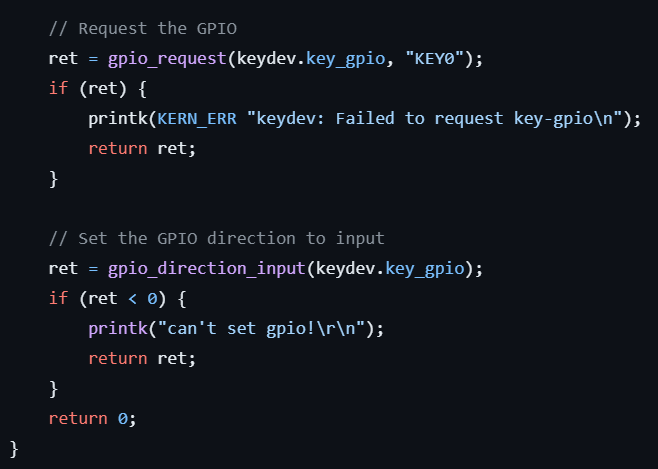
1. keyio\_init()函式內容主要是初始化按鍵，透過大量of函式取得Device Tree中相關的設備訊息。



(圖五:keyio\_init()函式內容)



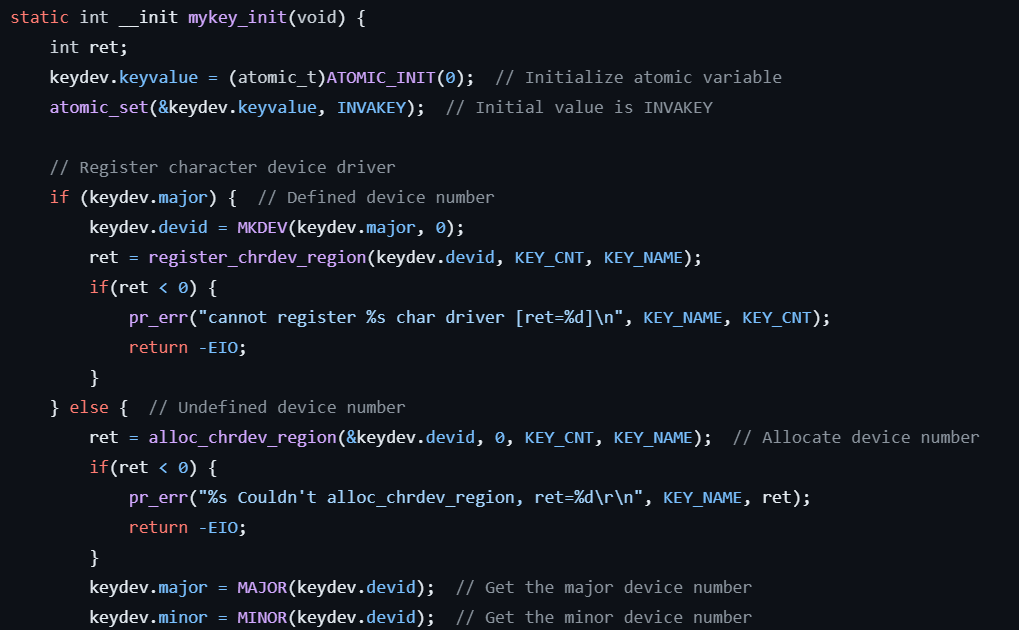
(圖六:keyio\_init()函式內容)



(圖七:keyio\_init()函式內容)

1. 當然要使用atomic原子操作，需要再驅動入口函式初始化atomic的值。

其餘就是一般字元設備驅動編寫。(如下圖八)



(圖八:驅動入口函數內容)

※結論:

整體流程與字元設備驅動開發流程大同小異，只不過把file\_operations中的成員函式更加完善。透過加入Linux系統中API函式對gpio進行操作。

最後在驅動入口函數加入atomic原子操作初始化，進而在file\_operation中的函式使用atomic原子操作。

此處使用原子操作的原因是，避免驅動程式(回傳按鍵值)和應用程式(讀去按鍵值)發生衝突。