1. 專案07\_atomic

Linux是一個多任務操作系統，肯定會存在多個任務共同操作同一段記憶體或者設備的情況，多個任務甚至中斷都能訪問的資源叫做共享資源。在驅動開發中要注意對共享資源的保護，也就是要處理對共享資源的並發訪問。

1. 併發與競爭:

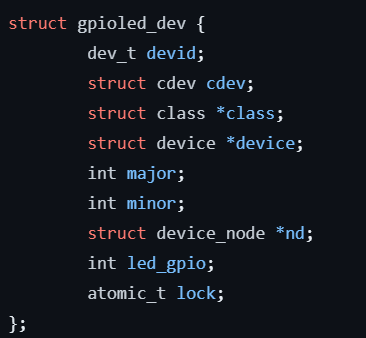
Linux系統是個多任務操作系統，會存在多個任務同時訪問同一片記憶體區域，這些任務可能會相互覆蓋這段記憶體中的數據，造成記憶體數據混亂。大致上原因有以幾項:

1. 多線程並發訪問，Linux是多任務（線程）的系統，所以多線程訪問是最基本的原因。
2. 搶佔式並發訪問，從2.6版本內核開始，Linux內核支持搶佔，也就是說調度程序可以在任意時刻搶佔正在運行的線程，從而運行其他的線程。
3. 中斷程序並發訪問，因為硬體中斷的權利是很大的。
4. SMP（多核）核間並發訪問，現在ARM架構的多核SOC很常見，多核CPU存在核間並發訪問。"
5. 保護的內容與方法:

我們實際要保護的內容就是共享數據，透故過防止併發的訪問來進行保護。

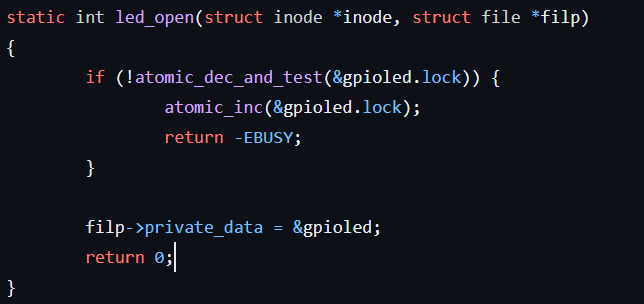
* 保護的方法有:
  + - 原子操作(atomic):指的是不能再進一步分割的操作。而Linux Kernel中定義了atomic\_t的結構體於include/linux/types.h 文件中。
    - 自旋鎖(Spinlock):之後專案08\_spinlock會提到。
    - 訊號量(Semaphore): 之後專案09\_semaphore會提到。
    - 互斥鎖(Mutex): 之後專案10\_mutex會提到。

1. 實際atomic使用:
2. 在設備結構體中定義atomic\_t成員變數(如下圖一)



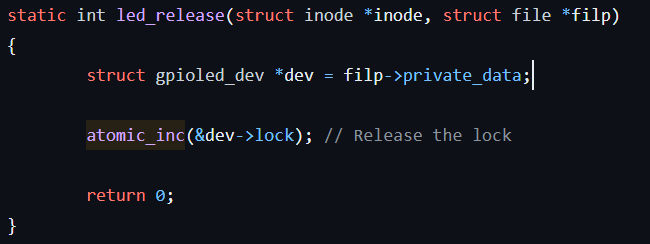
(圖一: 設備結構體)

1. 在open()函式中加入原子操作函數atomic\_dec\_and\_test()，目的是確保同一時間只能有一個任務能打開並是用這個設備。(如下圖二)



(圖二:open()函式內容)

1. 並在release()函式中加入原子操作函數atomic\_inc()，目的是釋放先前(如圖二)在open()函式中的鎖。(如圖三)



(圖三:release()函式內容)

1. 在驅動入口函數init()中加入， ATOMIC\_INIT(0)初始化原子變數鎖。透過原子操作函數atomic\_set()將鎖設置為1，(如下圖四)



(圖四:驅動入口函數init()內容)

※總結:

先透過ATOMIC\_INIT(0)初始化gpioled.lock為0，再透過atomic\_set()將其設置為1，每當有設備訪問時減一。當有其他任務想訪問時透過atomic\_dec\_and\_test來檢查是否有設備正在訪問，若gpioled.lock為0表示此設備被占用；若gpioled.lock為1表示此設備可以被訪問，最後當設備結束訪問時透過atomic\_inc()加1，使其他設備又可以訪問。