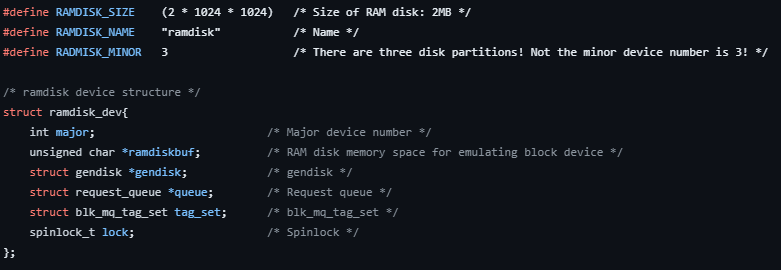
1. 專案26\_ramdisk\_withrequest:
2. 本專案會用RAM來模擬塊設備，進而熟悉Linux Kernel中塊設備API函式。
3. 塊設備驅動實作(Withrequest):
4. 首先字定義設備結構體，用以儲存設備相關訊息。其中

* **gendisk** 是一個指向 gendisk 結構體的指標。gendisk 結構體是 Linux 核心中表示塊設備（block device）的一個重要數據結構。它包含了與塊設備相關的所有資訊，例如設備名稱、容量、分區資訊等。(如下表一及圖一)
* **queue** 是一個指向 request\_queue 結構體的指標。request\_queue 結構體用於管理塊設備的請求佇列。這是核心處理 I/O 請求的核心數據結構之一，它維護了所有待處理的 I/O 請求，並負責將這些請求傳遞給設備驅動程式。(如下表一及圖一)
* **tag\_set** 是一個 blk\_mq\_tag\_set 結構體的實例。blk\_mq\_tag\_set 是用於管理多佇列塊設備的標籤集合（tag set）。在多佇列（multi-queue）塊層（block layer）中，tag set 負責分配和管理 I/O 請求的標籤，用於優化並行 I/O 操作，提高性能。(如下表一及圖一)

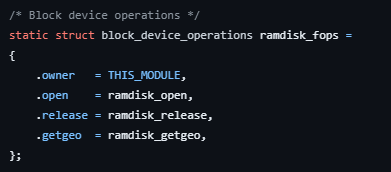
(表一:結構體成員變數及相關功能)

|  |  |
| --- | --- |
| **變數名稱** | **功能描述** |
| major | 主要設備號碼 |
| ramdiskbuf | 用於模擬塊設備的 RAM 磁碟內存空間 |
| gendisk | 指向 gendisk 結構體的指標，表示塊設備 |
| queue | 指向 request\_queue 結構體的指標，管理 I/O 請求佇列 |
| tag\_set | blk\_mq\_tag\_set 結構體的實例，用於管理多佇列塊設備的標籤集合 |
| lock | 自旋鎖，用於同步 |

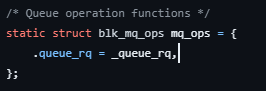


(圖一:設備結構體成員變數)

1. 接著定義，block\_device\_operations結構體操作函數及blk\_mq\_ops操作函數(如下圖二，圖三)



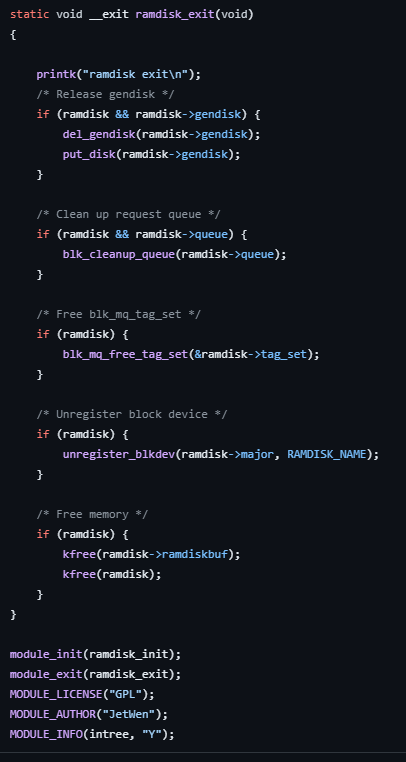
(圖二: block\_device\_operations結構體成員函式)



(圖三: blk\_mq\_ops結構體成員函式)

1. 定義驅動出入口函式及該驅動相關訊息。(如下圖四，圖五，圖六)

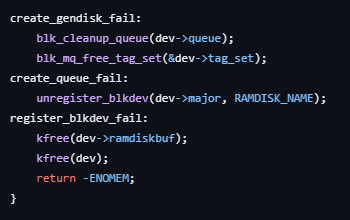
* 驅動入口函數執行了分配內存，向Kernel註冊塊設備透過register\_blkdev函數，又透過create\_req\_queue創建多個queue，最後透過create\_req\_gendisk函式創建block device。(如下圖五，圖六)
* 驅動出口函數就是卸載和註冊驅動入口函數所有動作。(如圖四)



(圖四:驅動出入口函數)



(圖五:驅動入口函式)



(圖六:驅動入口函式)

1. 接著，完善圖二block\_device\_operations結構體成員函式(如圖七)

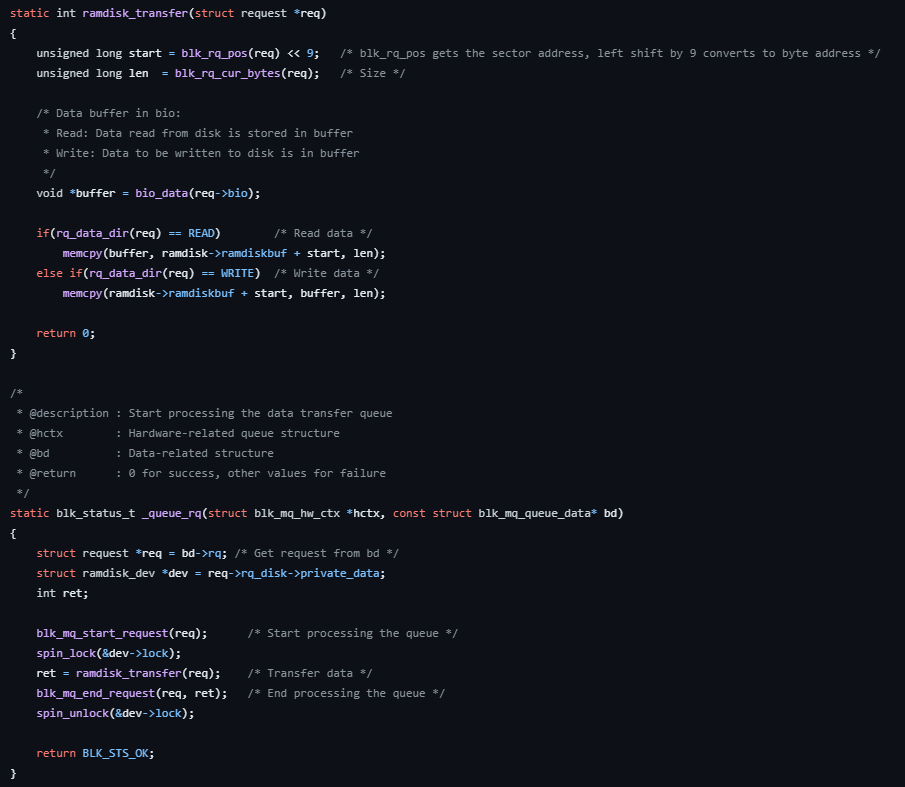
* ramdisk\_open，打開塊設備。
* ramdisk\_release，關閉塊設備。
* ramdisk\_gendisk，用來描述機械硬碟的結構。



(圖七: block\_device\_operations結構體成員函式實現)

1. ramdisk\_transfer函式的功能是處理從請求中讀取或寫入數據的操作。具體步驟如下(參考圖八)：

* **計算數據傳輸的起始位址和長度：**start 是請求開始的扇區地址，通過 blk\_rq\_pos(req) 獲取，並左移 9 位轉換為位元組地址。len 是請求的大小，通過 blk\_rq\_cur\_bytes(req) 獲取。
* **獲取數據緩衝區：**buffer 是數據緩衝區，通過 bio\_data(req->bio) 獲取。
* **根據請求的方向（讀取或寫入）進行數據傳輸：**如果請求是讀取（rq\_data\_dir(req) == READ），則將數據從 RAM 磁碟（ramdisk->ramdiskbuf + start）拷貝到 buffer。如果請求是寫入（rq\_data\_dir(req) == WRITE），則將數據從 buffer 拷貝到 RAM 磁碟（ramdisk->ramdiskbuf + start）。



(圖八: ramdisk\_transfer和\_queue\_rq內容實現)

1. \_queue\_rq這個函式的功能是開始處理數據傳輸隊列中的請求。具體步驟如下(參考圖八)：

* **獲取請求：**從 bd 中獲取請求 req。獲取 ramdisk\_dev 結構體的指標 dev，這是請求中 rq\_disk 的私有數據。
* **開始處理請求：**調用 blk\_mq\_start\_request(req) 開始處理請求。
* **加鎖並傳輸數據：**使用 spin\_lock(&dev->lock) 獲取自旋鎖，以確保數據傳輸過程中的同步。調用 ramdisk\_transfer(req) 進行數據傳輸。
* **結束請求處理並解鎖：**調用 blk\_mq\_end\_request(req, ret) 結束請求處理，ret 是數據傳輸的返回值。使用 spin\_unlock(&dev->lock) 釋放自旋鎖。
* **返回狀態：**返回 BLK\_STS\_OK 表示請求成功處理。

1. create\_req\_queue作用是初始化與隊列相關的操作，具體包括配置和創建請求隊列。以下是詳細的解釋(如下圖九)：

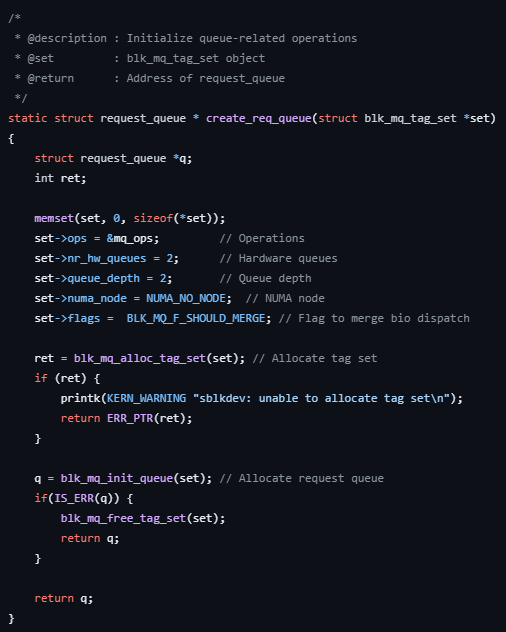
* **清零 set 變數：**使用 memset 將 set 變數清零，確保所有欄位初始化為 0。
* **配置 set 結構：**set->ops = &mq\_ops; 設置操作函式指標，這些操作函式定義了如何處理佇列中的請求。set->nr\_hw\_queues = 2; 設置硬體佇列的數量。set->queue\_depth = 2; 設置佇列的深度，即每個佇列中能夠同時處理的請求數量。set->numa\_node = NUMA\_NO\_NODE; 設置 NUMA 節點，這裡設置為不特定的 NUMA 節點。set->flags = BLK\_MQ\_F\_SHOULD\_MERGE; 設置標誌，允許合併 bio 分派。
* **分配標籤集合：**調用 blk\_mq\_alloc\_tag\_set(set) 分配標籤集合，標籤集合是多佇列塊層（block layer）中用於管理 I/O 請求標籤的結構。如果分配失敗，打印警告訊息並返回錯誤指標 ERR\_PTR(ret)。
* **初始化請求佇列：**調用 blk\_mq\_init\_queue(set) 初始化請求佇列，這將根據配置的 set 創建並初始化請求佇列。如果初始化失敗，釋放之前分配的標籤集合並返回錯誤指標。
* **返回請求佇列的地址：**如果一切順利，返回請求佇列的地址。

1. create\_req\_gendisk函式的目的是創建塊設備，並提供給User Space使用(如下圖十) (參考下表二)。

* **分配和初始化 gendisk:**dev->gendisk = alloc\_disk(RADMISK\_MINOR); 這行代碼分配一個新的 gendisk 結構體，指定的次要設備號數量為 RADMISK\_MINOR。if (dev->gendisk == NULL) return -ENOMEM; 檢查分配是否成功，如果失敗，返回 -ENOMEM 錯誤碼。
* **添加（註冊）磁碟:**dev->gendisk->major = ramdisk->major; 設置主要設備號。dev->gendisk->first\_minor = 0; 設置起始次要設備號。dev->gendisk->fops = &ramdisk\_fops; 設置塊設備操作函式指標。dev->gendisk->private\_data = set; 設置私有數據。dev->gendisk->queue = dev->queue; 設置請求佇列。sprintf(dev->gendisk->disk\_name, RAMDISK\_NAME); 設置磁碟名稱。set\_capacity(dev->gendisk, RAMDISK\_SIZE / 512); 設置設備容量，單位為扇區（每扇區 512 字節）。add\_disk(dev->gendisk); 註冊磁碟，使其在系統中可用。

(表二: create\_req\_gendisk函式實現步驟及功能)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| 分配和初始化 gendisk | 分配一個新的 gendisk 結構體，指定次要設備號數量為 RADMISK\_MINOR。檢查分配是否成功，如果失敗，返回 -ENOMEM 錯誤碼。 |
| 添加（註冊）磁碟 | 設置主要設備號、起始次要設備號、操作函式指標、私有數據、請求佇列、磁碟名稱和設備容量（單位為扇區）。註冊磁碟，使其在系統中可用。 |



(圖九: create\_req\_queue內容實現)



(圖十: create\_req\_gendisk函式內容實現)

※總結:

所有步驟整理成如下表格。

(表三:塊設備應用request開發步驟)

|  |  |
| --- | --- |
| **步驟** | **說明** |
| **包含必要的頭文件** | 包含 Linux 核心模塊開發所需的各種標頭文件。 |
| **定義常數和數據結構** | 定義 RAM 磁碟大小、名稱、次要設備號和 ramdisk\_dev 結構體。 |
| **聲明全域變數** | 聲明全域變數 ramdisk。 |
| **處理數據傳輸請求** | 定義函式 ramdisk\_transfer，負責處理數據的讀寫。 |
| **開始處理數據傳輸隊列** | 定義函式 \_queue\_rq，負責開始處理隊列中的請求。 |
| **定義佇列操作函式** | 定義操作函式 mq\_ops，包括 \_queue\_rq。 |
| **打開塊設備** | 定義 ramdisk\_open 函式，負責打開塊設備。 |
| **釋放塊設備** | 定義 ramdisk\_release 函式，負責釋放塊設備。 |
| **獲取磁碟幾何結構** | 定義 ramdisk\_getgeo 函式，負責獲取磁碟幾何結構。 |
| **定義塊設備操作函式** | 定義操作函式 ramdisk\_fops，包括打開、釋放和獲取磁碟幾何結構的函式。 |
| **初始化佇列相關操作** | 定義 create\_req\_queue 函式，負責初始化佇列相關操作並分配佇列。 |
| **創建塊設備** | 定義 create\_req\_gendisk 函式，負責創建和初始化 gendisk 結構體並註冊磁碟。 |
| **模塊初始化** | 定義 ramdisk\_init 函式，負責分配內存、初始化自旋鎖、註冊塊設備、創建佇列和創建塊設備。 |
| **模塊退出** | 定義 ramdisk\_exit 函式，負責釋放資源，包括 gendisk、請求佇列、標籤集合、內存等。 |
| **模塊入口和出口** | 使用 module\_init 和 module\_exit 宏定義模塊的入口和出口函式。 |
| **模塊信息** | 定義模塊的許可證、作者和其他信息。 |