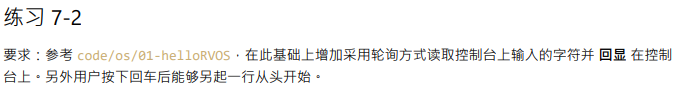
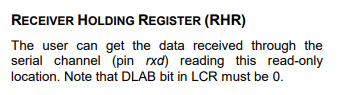
作業系統工程

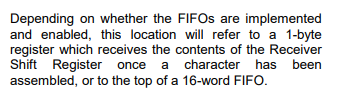
作業(二)

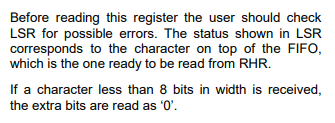
李奕承 611121212

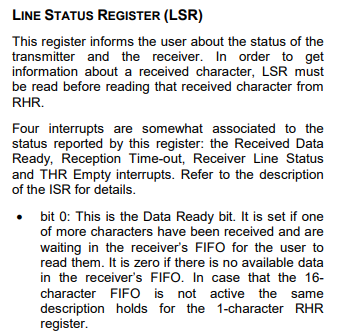


為了能夠讓作業系統從UART收到我們輸入的字，需要先知道要控制哪些UART控制暫存器。於是我開始查閱規格書。











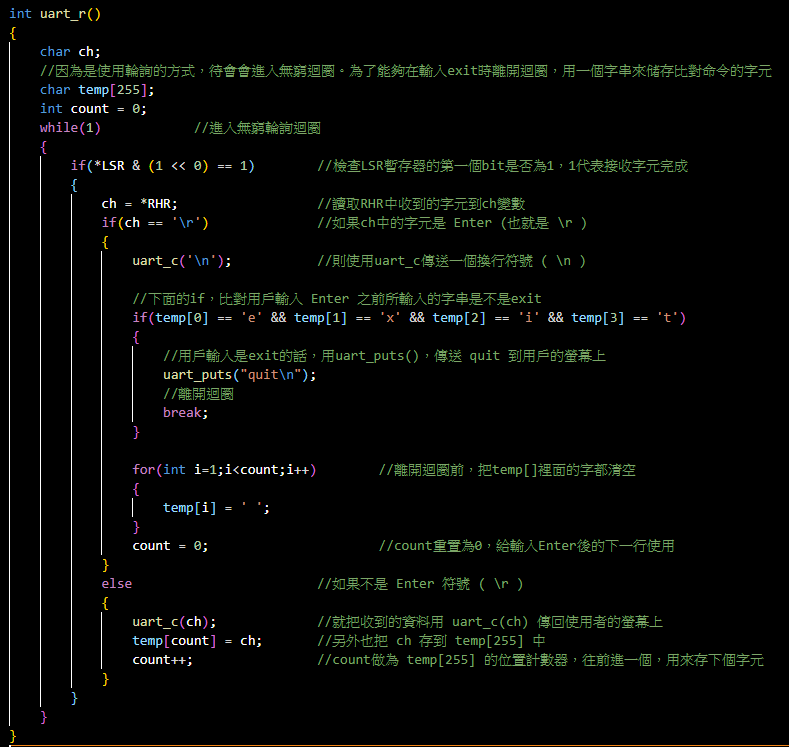
從上面這些敘述中瞭解到，如果要存取RHR中收到的訊號，要先檢查LSR的第0個bit是否為1

於是我開始寫用來收訊息的函數

首先來定義RHR的地址替代巨集。RHR和THR在規格上共用同一個地址

#define RHR  ((volatile unsigned char \*)(0x10000000L + 0))

接著造一個無窮迴圈來不斷檢查是否收到資料了，並判斷用戶輸入的是哪些字



接著將新寫的 uart\_r( ) 加到OS中

extern void uart\_r(void);

在uart初始化後使用它

void main(void)

{

    uart\_init();                        //執行uart\_init函數初始化UART

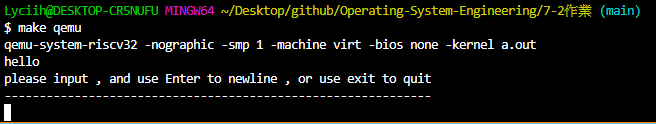
    uart\_puts("hello\n");               //使用uart\_puts函數傳輸 hello 字串

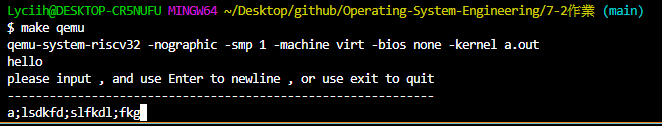
    uart\_puts("please input , and use Enter to newline , or use exit to quit\n");

    uart\_puts("-------------------------------------------------------------\n");

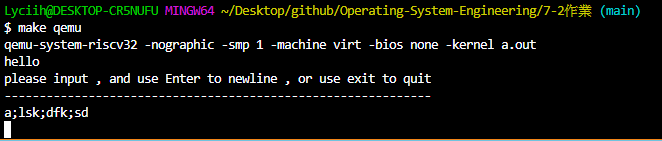
    uart\_r();

編譯執行看看，游標停在最下方等待我們輸入

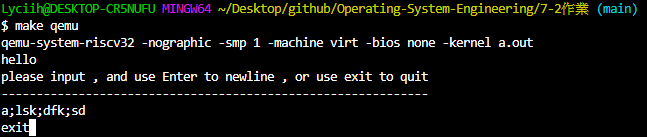


隨便亂打字，都會出現在螢幕上，代表UART成功收到了字元並回傳給我們

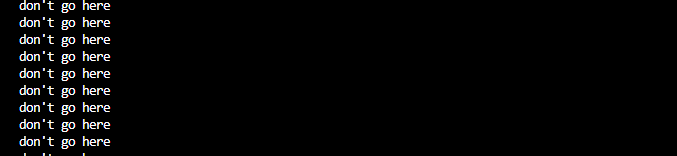
按下Enter看看，成功換行

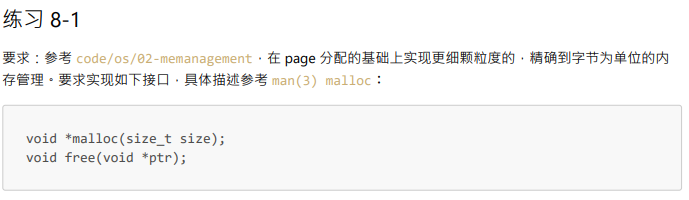


輸入exit指令看看

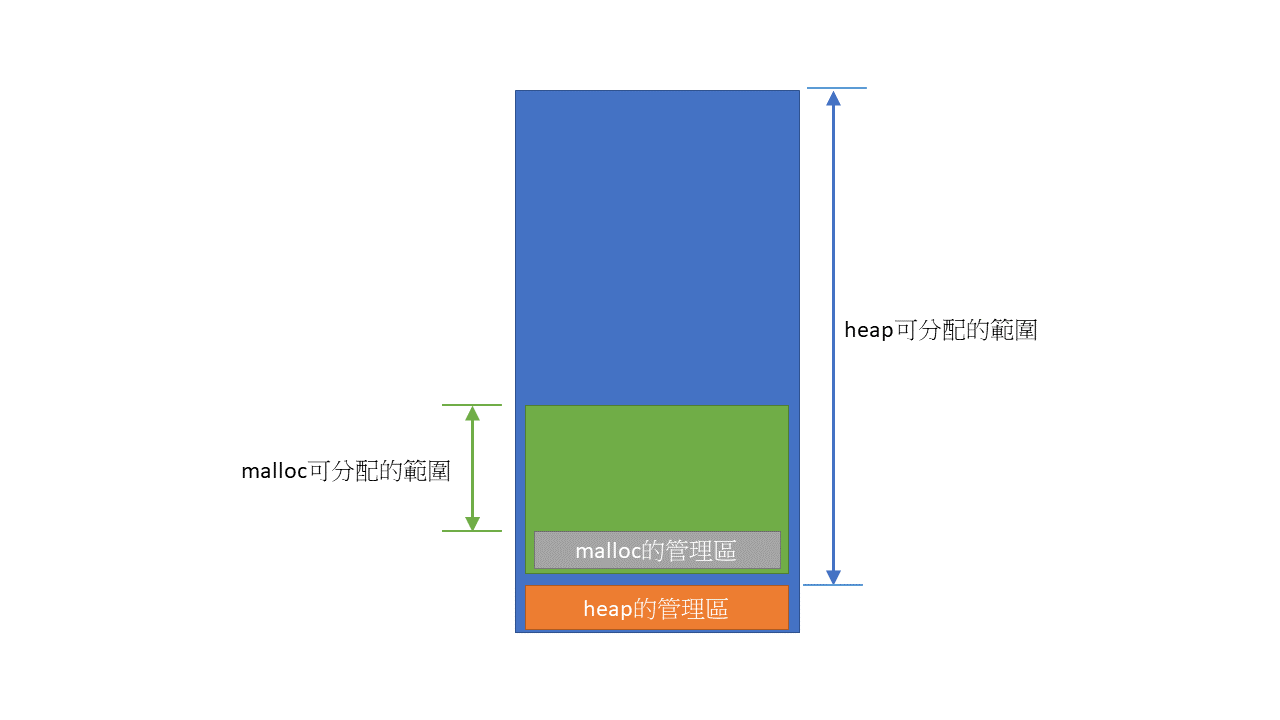


按下Enter，成功跳出迴圈，進入到OS的盡頭

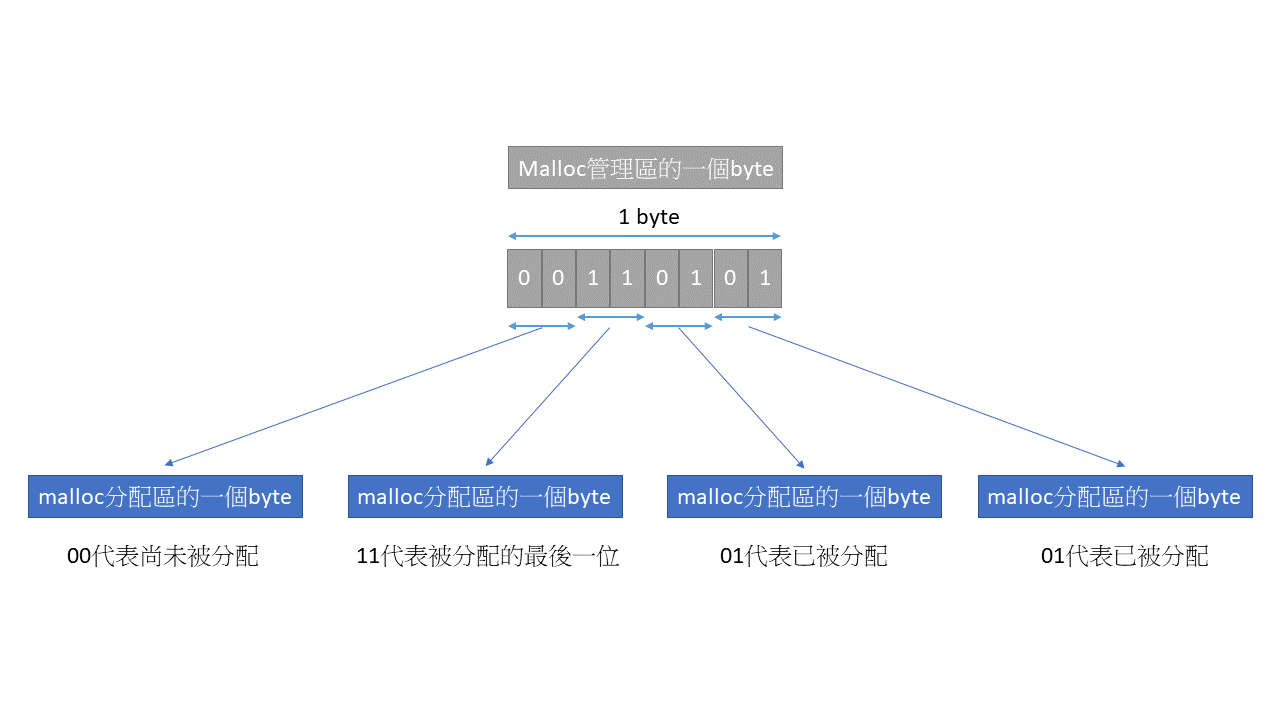




我實作的malloc在初始化時會先用page申請幾個分頁做為malloc可分配的空間。另外我將page的管理模式也用在malloc的實踐上，所以初始化時所申請來的空間有一部份會用來管理malloc分配的空間。



原本的page管理是用一個byte來對映一個page，只用掉2個bit來代表三個不同的狀態，但是這樣太浪費了，所以我在管理malloc時使用一個byte來管理4個byte



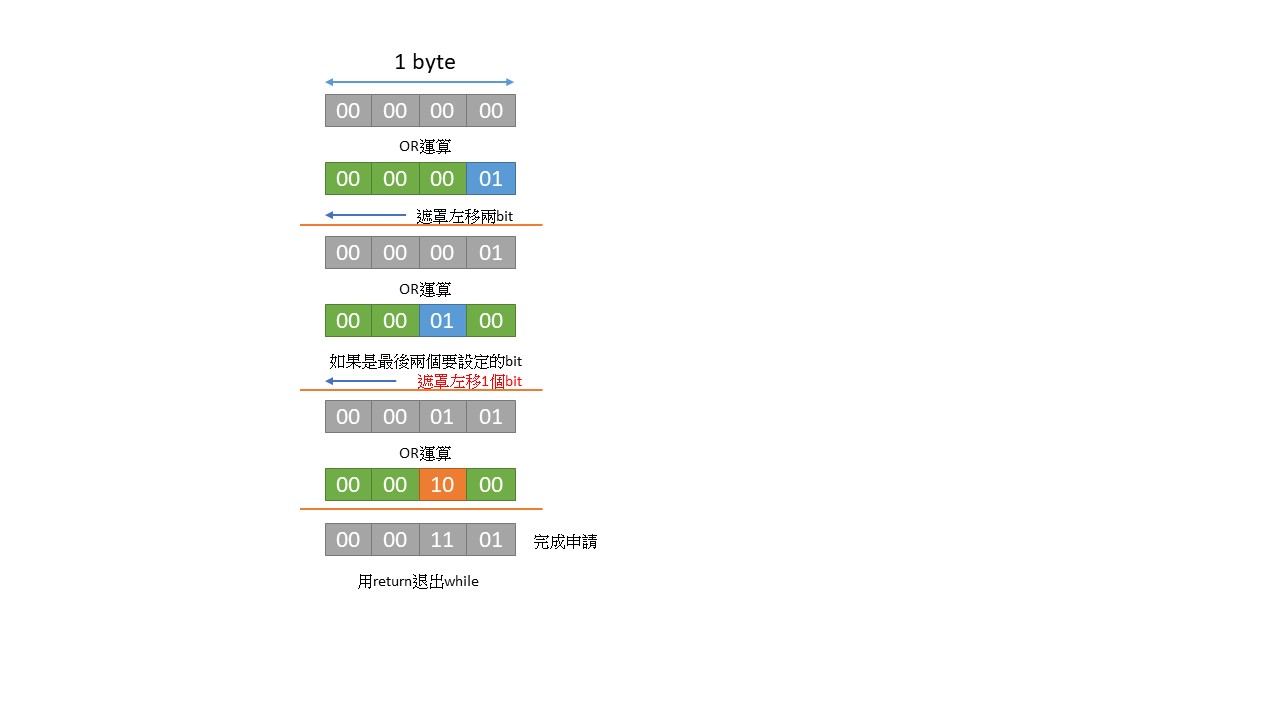
使用1個byte管理4個byte的原理

宣告一個名為offset的uint8\_t變數當遮罩，並將offset的值用來做為while是否繼續運行的依據

while(offset)



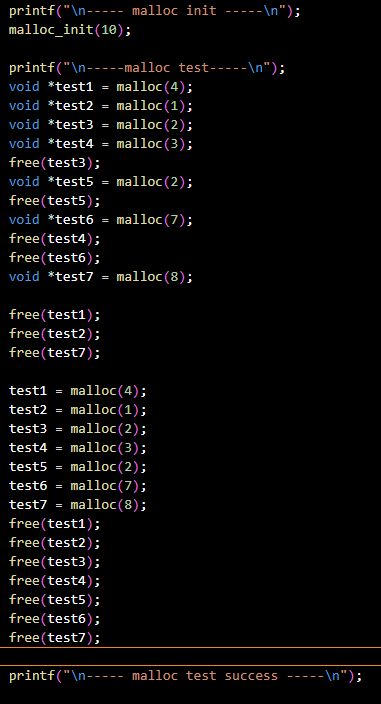
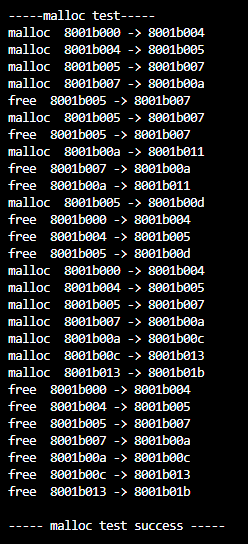
設定最後一個byte的方法



而Free也是利用一樣的原理

實際的程式碼太長不好排版，我將每一行都做了注釋，請老師直接看程式碼

在OS中使用malloc\_init初始化並加上一段malloc跟free的測試，編譯並執行看看成果

測試成功，可以看到申請的位置是一路連續下去，並且在釋放後產生空位時，新的申請會優先放在第一個找到的可以放得下的區段。接著測試申請不是5的倍數的page會不會被阻擋，這裡申請11個page



成功，並且初始化失敗後，malloc跟free都被禁止

