**2021，作業系統，上機考（參考答案）**

說明一：艷紅色的字代表「要回答的問題」。藍色字代表參考答案。黃色底的代表要特別留意的地方

說明二：上機考的時候可以將參考答案和解說影片帶進考場。

題目：

* 呼叫write的system call的程式碼如下（考試的時候會給）：

|  |
| --- |
| **void** call\_sys() {  **char**\* hello\_tc = "全世界，你好\n";  **long** len\_tc = strlen(hello\_tc); //注意我宣告為long，因為long是64位元  **long** ret;  **\_\_asm\_\_** **volatile** (  "mov $1, %%rax\n" //system call number  "mov $2, %%rdi\n" //stderr  "mov %1, %%rsi\n" //  "mov %2, %%rdx\n"  "syscall\n"  "mov %%rax, %0"  : "=m"(ret)  : "g" (hello\_tc), "g" (len\_tc)  : "rax", "rbx", "rcx", "rdx");  }  **int** main() {  printf("address of call\_sys is %p\n", call\_sys);  getchar();  call\_sys();  } |

* 將上述程式碼放到用gcc編譯，編譯出來的程式必須能在『Linux in QEMU』中正確執行（例如：將編譯出來的程式碼複製到在/home/shiwulo/osdi/sharedFolder中）
* 在『dbg-Linux5.0-in-QEMU.sh』中執行你的程式碼，於Eclipse中進行追蹤
  + 首先terminal中執行dbg-Linux5.0-in-QEMU.sh
  + 再執行eclipse，並run ↦ debug history ↦ RunningLinuxKernel 5.0
  + 於eclipse的debugger console中按下「c」
  + 等在terminal的Linux開完機以後，在terminal中執行「./sharedFolder/ test\_syscall」（此步驟相當於在『Linux in QEMU』中執行）
* （問題一，25pt）設定中斷點在test\_syscall發出system call之前（例如：將中斷點設定在call\_sys），請在這個地方截圖
  + 在eclipse的Debugger Console按下ctr-c
  + 在Debugger Console輸入「file /home/shiwulo/osdi/sharedFolder/ test\_syscall」
  + 在Debugger Console輸入「b \*(addr of call\_sys)」或「b call\_sys」
  + 在Debugger Console輸入「c」
  + 當Eclipse停下來時將「terminal」與「Eclipse」合照

一張含有 文字 的圖片

自動產生的描述

（圖一：附上從user space要跳入kernel space前的組合語言碼）

* （問題二，25pt）使用單步追蹤（si），直到syscall後（注意：syscall之後就是跳入到Linux kernel），先在Debugger Console輸入「file /home/shiwulo/osdi/vmlinux」，然後對Eclipse拍照。這個時機點剛好就是從user space進入kernel space時。

一張含有 文字 的圖片

自動產生的描述

（圖二）附上組合語言與原始碼對照的「system call進入點」

* （問題三，25pt）請說明Linux kernel如何用RAX暫存器判斷要呼叫哪個Linux內部的函數，請說明該函數的名稱
  + 答案：



**nr = array\_index\_nospec(nr, NR\_syscalls);**

**regs->ax = sys\_call\_table[nr](regs);**

1. **（注意期中考時，這一項應該會不一樣，應該不是ksys\_write）**

**SYSCALL\_DEFINE3(write, unsigned int, fd, const char \_\_user \*, buf, size\_t, count) {**

**return ksys\_write(fd, buf, count);**

**}**

* （問題四，25pt）請用50~200個「有意義的文字」大致說明作業系統如何處理該system call（例如：可以將註解英翻中，或者列出呼叫流程。請盡可能表達。這一題會依照所寫的內容的完整性給分）。
  + **\_\_do\_sys\_write呼叫ksys\_write後者為Ｃ函數，他的參數剛好和wrtie(2)一模一樣**
  + **在ksys\_write中首先呼叫fdget\_pos()，這個函數似乎是「鎖定」這個檔案，同時他會回傳一個資料結構用以代表「檔案」，這個資料結構包含fields及function pointers。**
  + **在ksys\_write接下來呼叫file\_pos\_read()獲得要寫入的這一個檔案的偏移值**
  + **呼叫vfs\_write(f.file, buf, count, &pos)做真正的寫入。因為每個檔案可能對應到不同的裝置，因此vfs\_write的行為，取決於f.file**
    - **這個地方應該繼續往下追，vfs\_write應該是呼叫f裡面的某個函數**
  + **如果上述的write成功，那麼會將pos更新到f.file**
  + **因為已經更新完畢，呼叫fdput\_pos，解開「鎖定」**

繳交：

* Pdf檔案即可