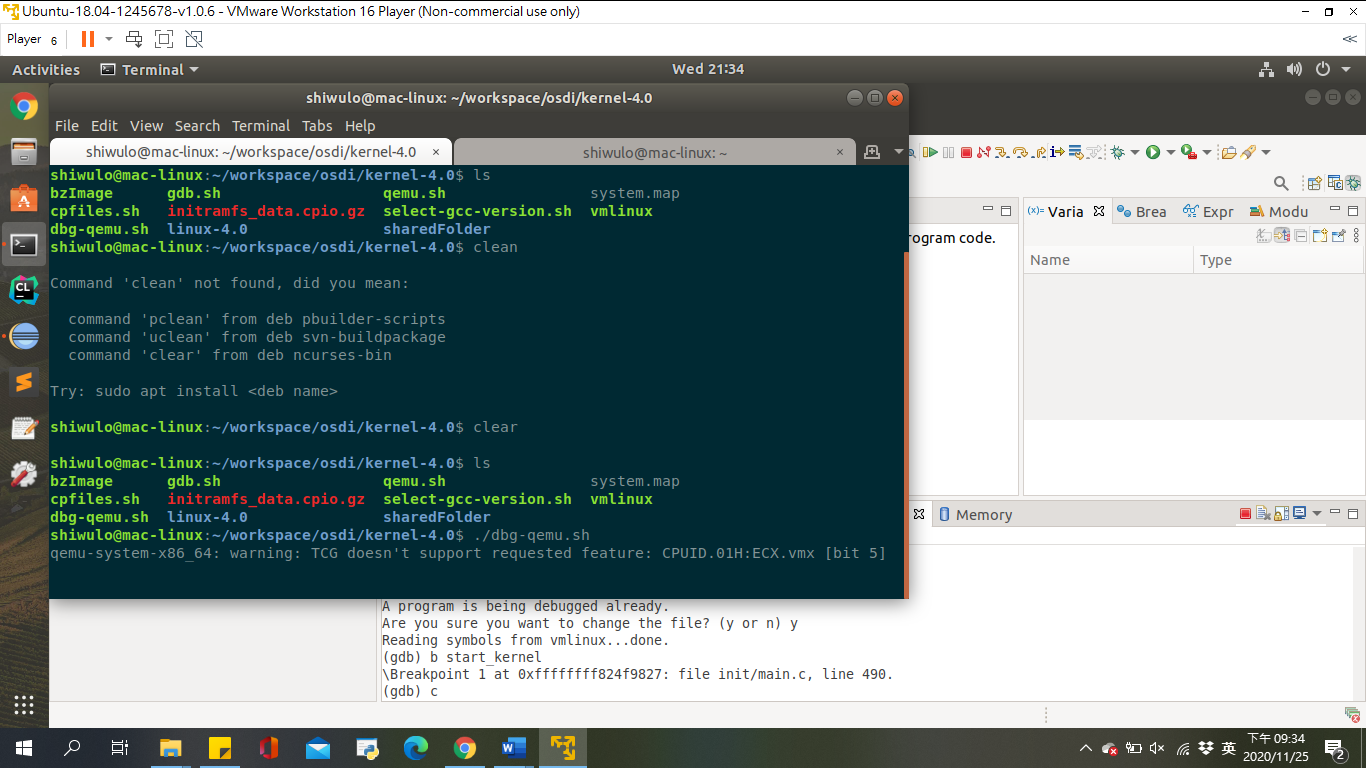
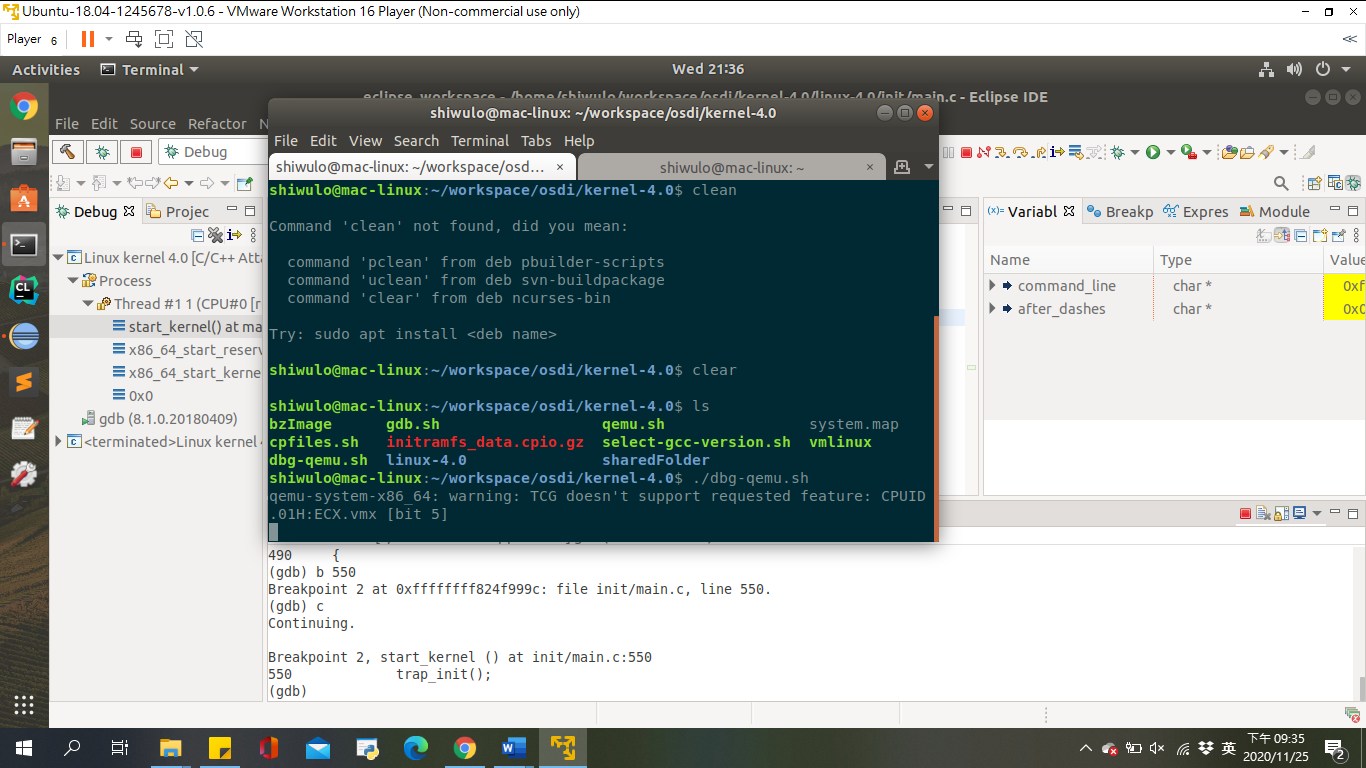
一、

1. 先執行「./dbd-qemu.sh」
2. 再使用eclipse :「file vmlinux -> b start\_kernel -> c」



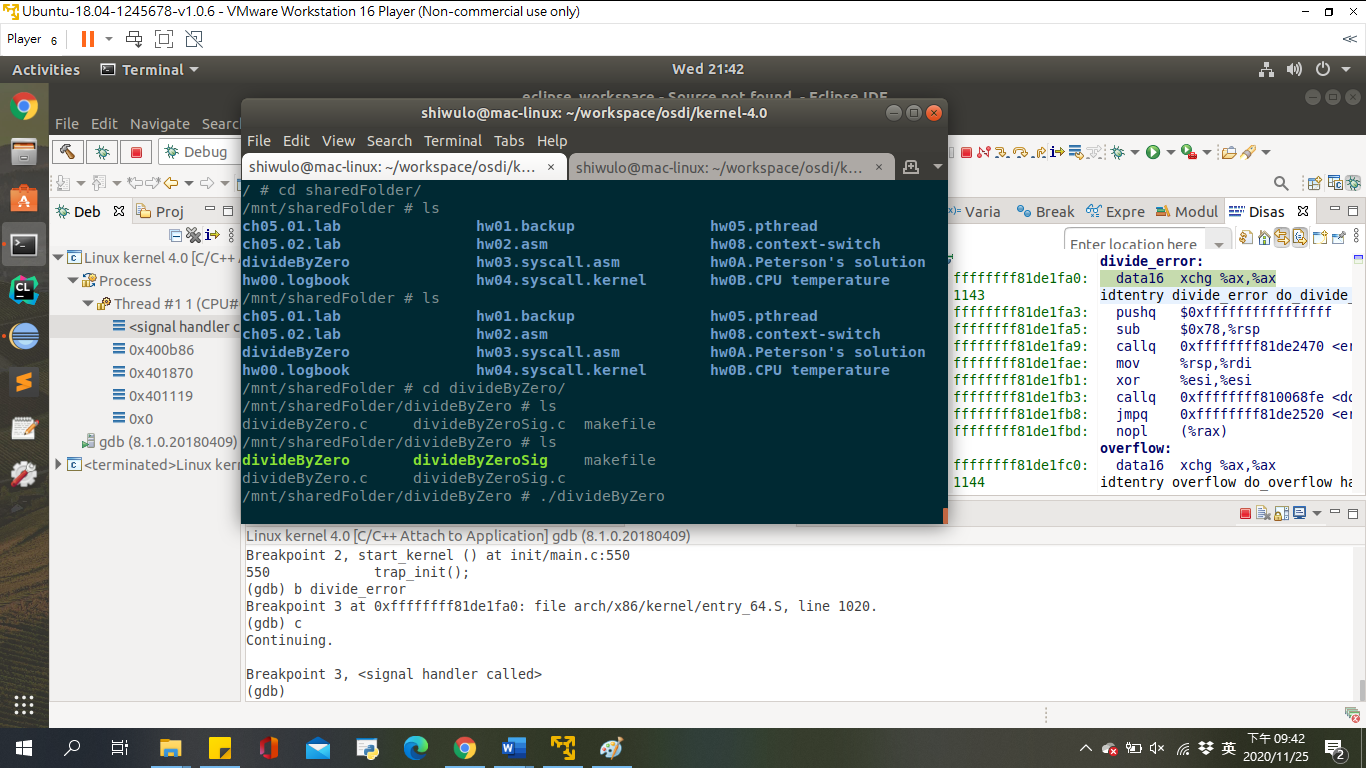
1. 接著:「b 550 -> c」，將中斷點設立在trap\_init



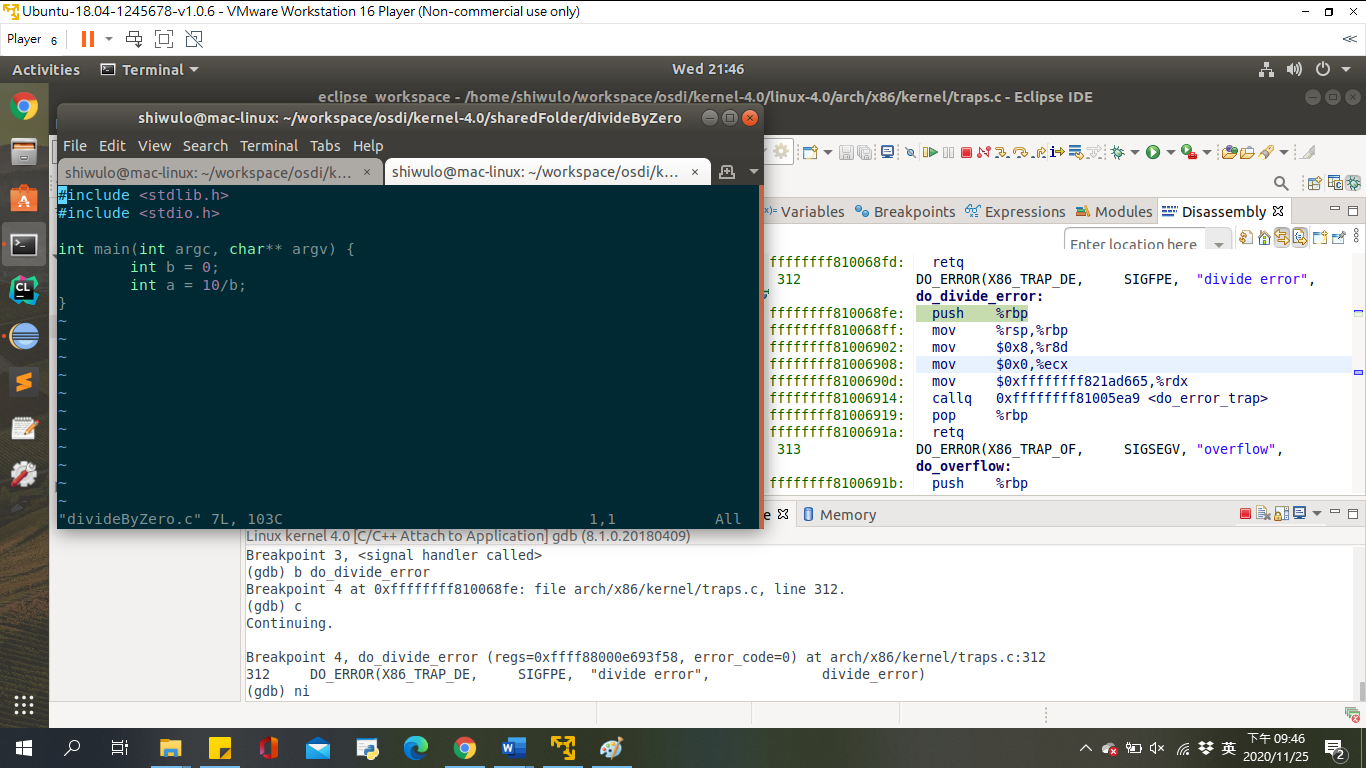
1. 最後 : 「b divide\_error -> c」，設立新中斷點讓它重新開機



1. 在裡面執行:「./divideByZero」，可以看到右邊組語divide\_error



1. 設一個新的中斷點到:「b do\_divide\_error」，可以發現它為的c函式，會將當初初發生的divide\_error去做處理

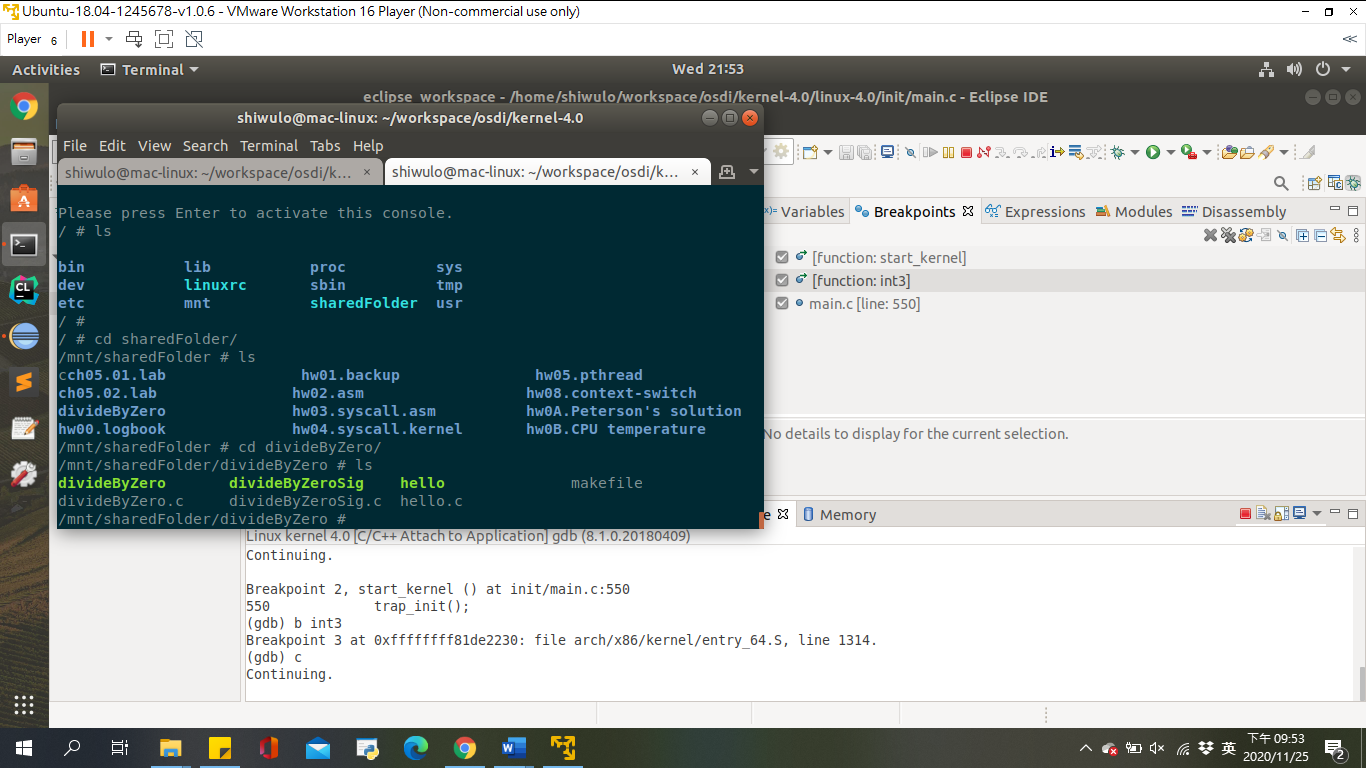


二、

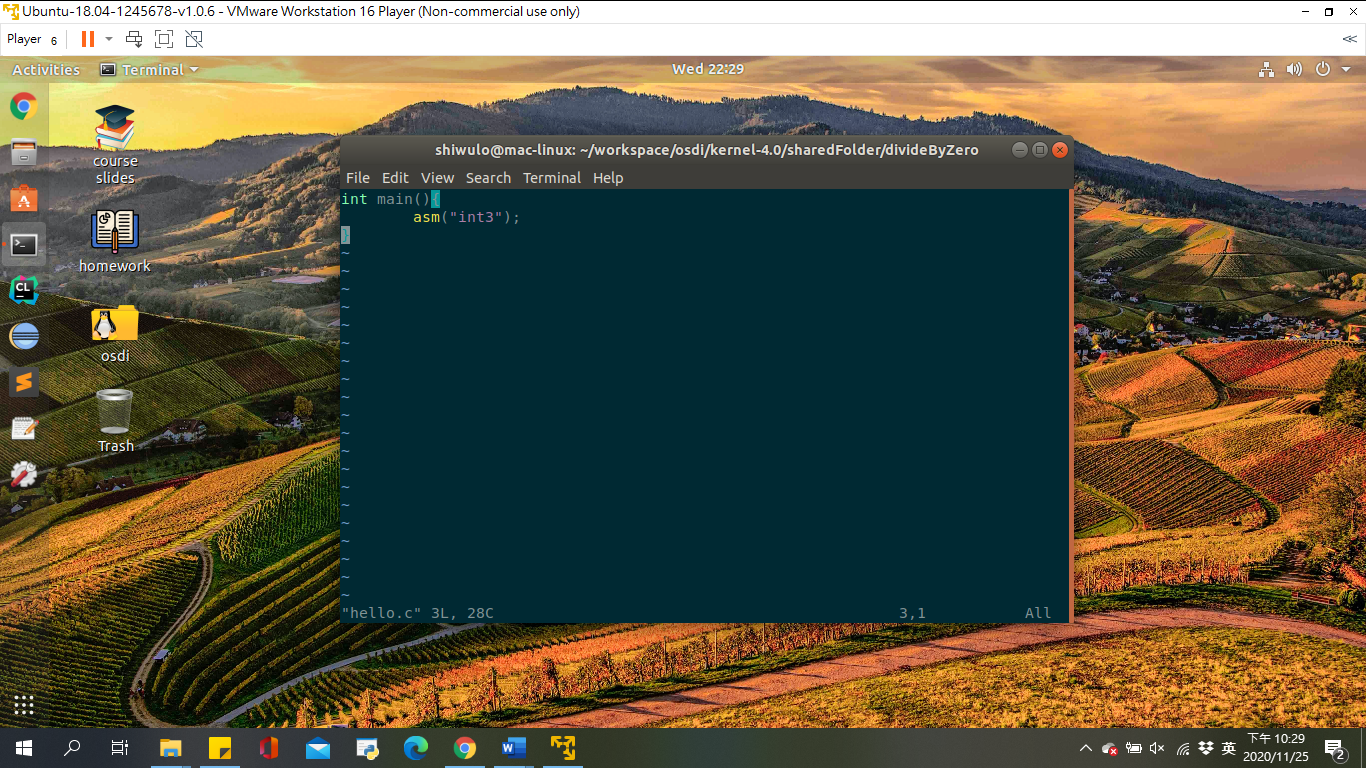
1. 一樣先執行「./dbd-qemu.sh」

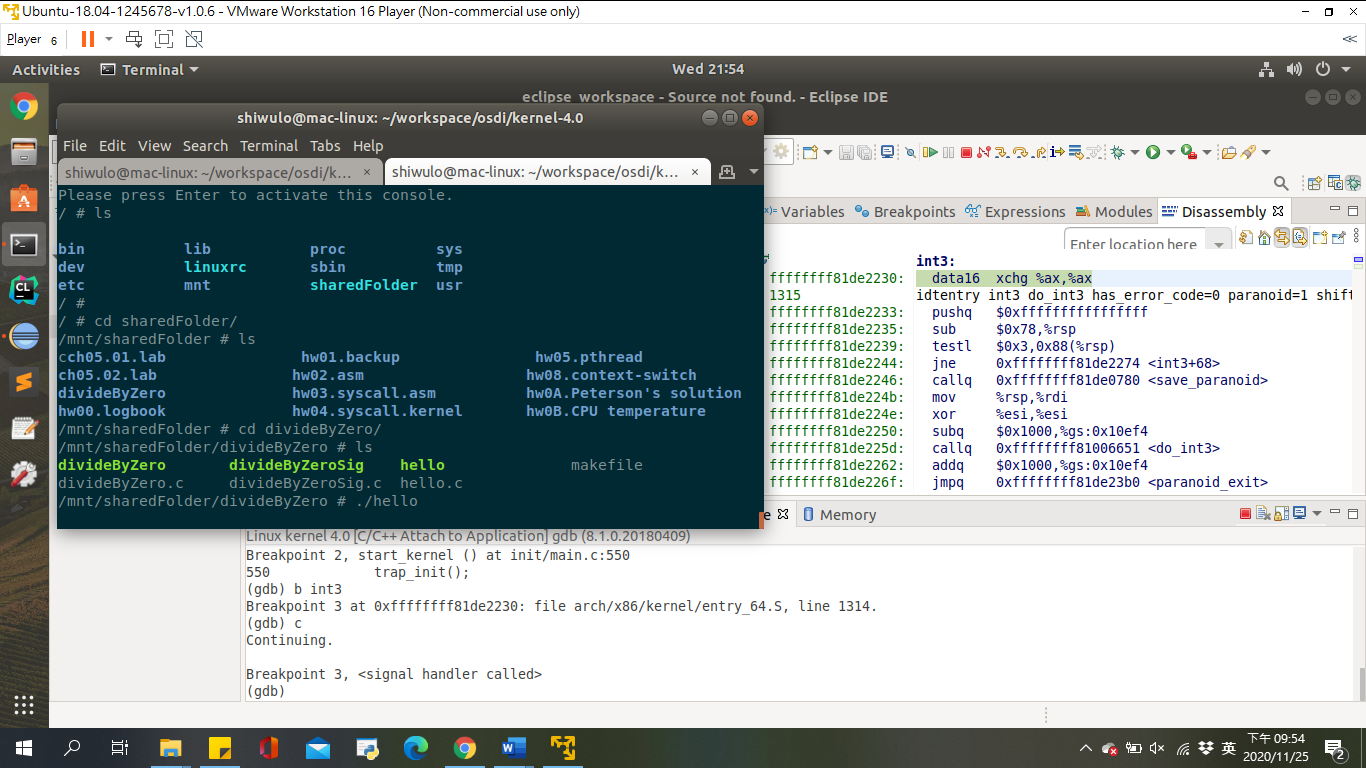
再使用eclipse :「file vmlinux -> b start\_kernel -> c -> b 550 -> c」

但這次設立「b int3 -> c」為新的中斷點，讓它重新開機

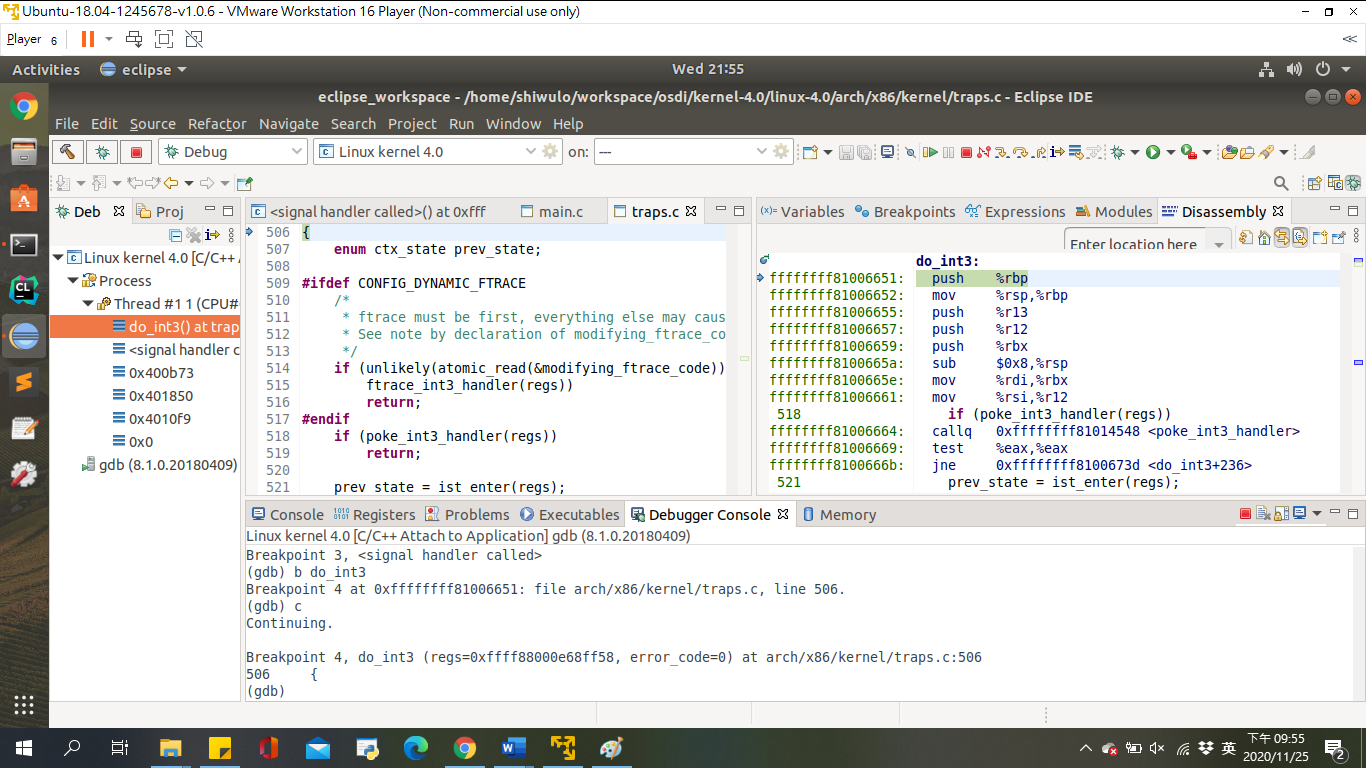


2. 而執行預先編譯完的hello.c檔，從右邊的組語可以看到後面有跳到do\_int3





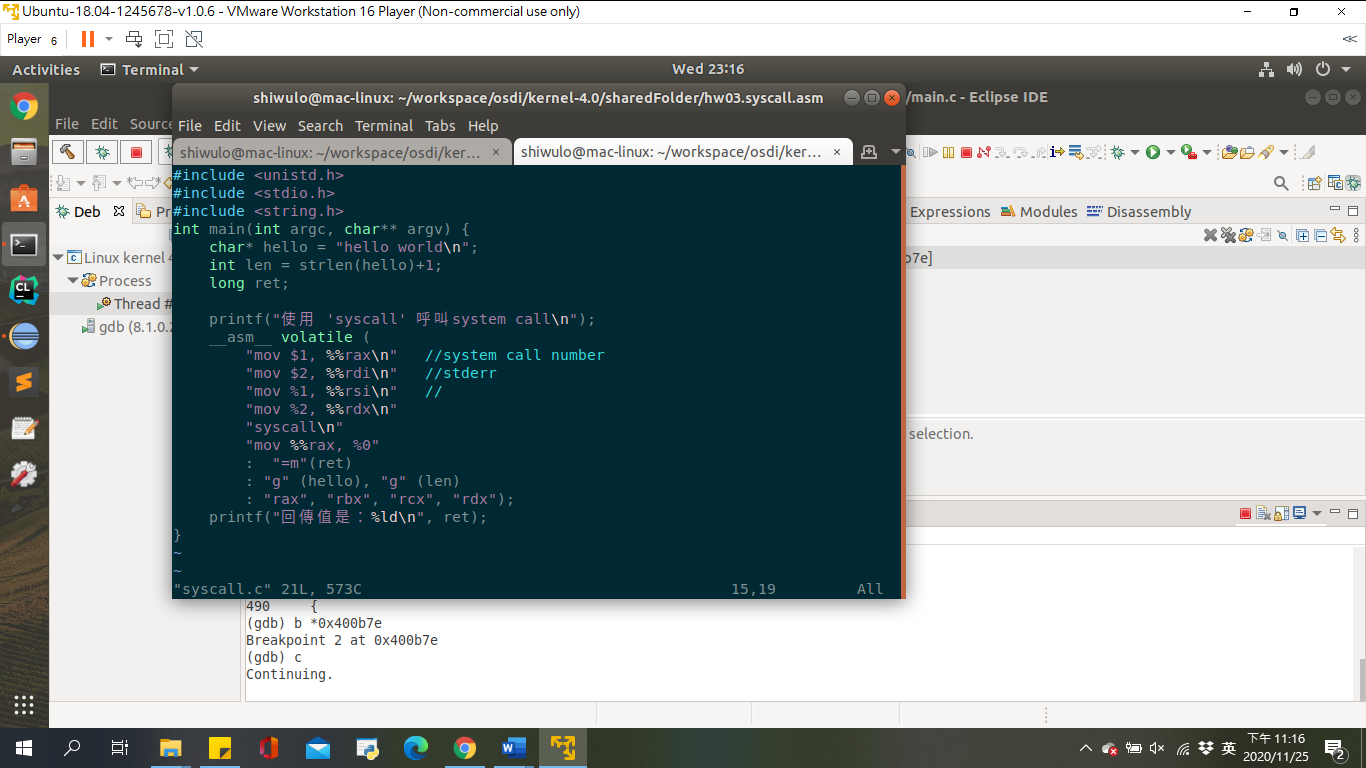
3. 設立新的中斷點「b do\_int3」，可以發現它同為處理int3的c語言函式



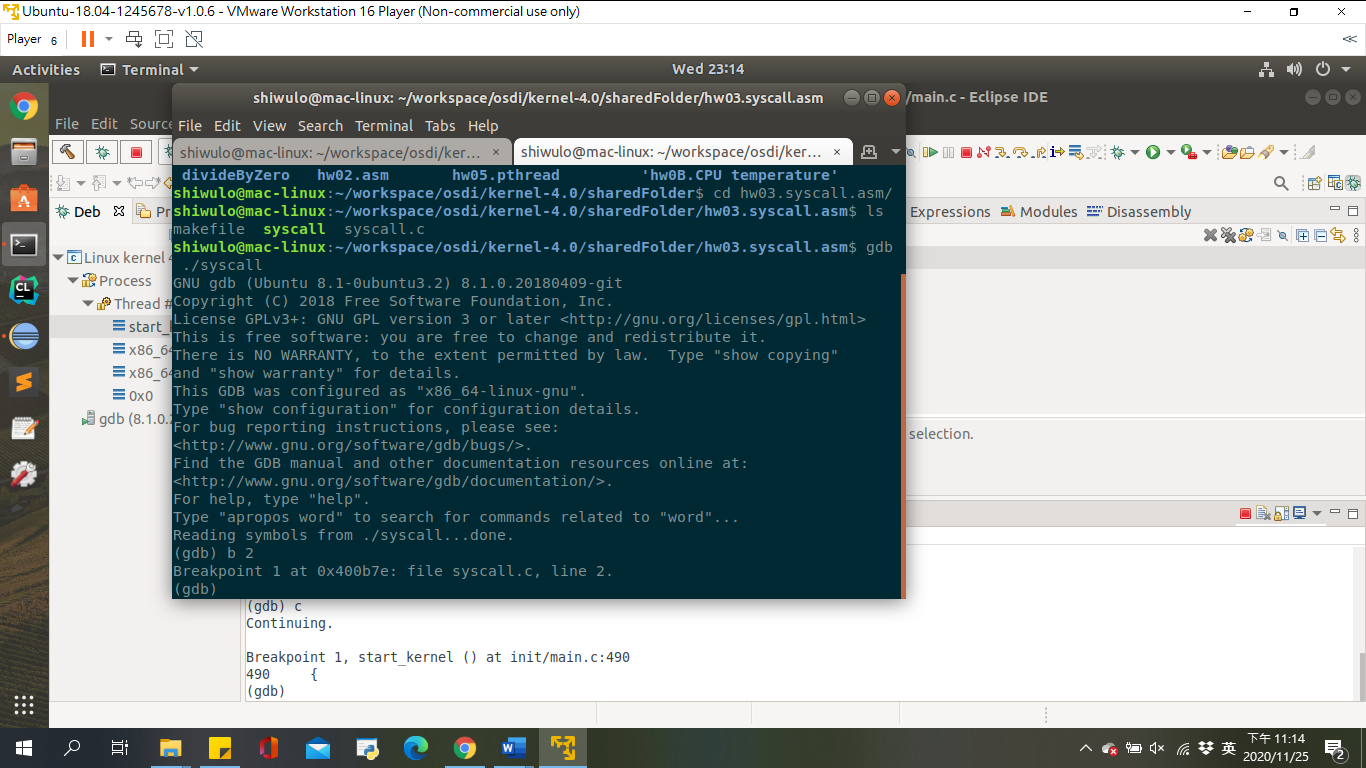
1. 因此在「set\_nmi\_gate(X86\_TRAP\_BP, &int3);」裡，X86\_TRAP\_BP為exception的編號，而int3發生這個exception時，CPU會執行int3這個指標的程式碼。

三、

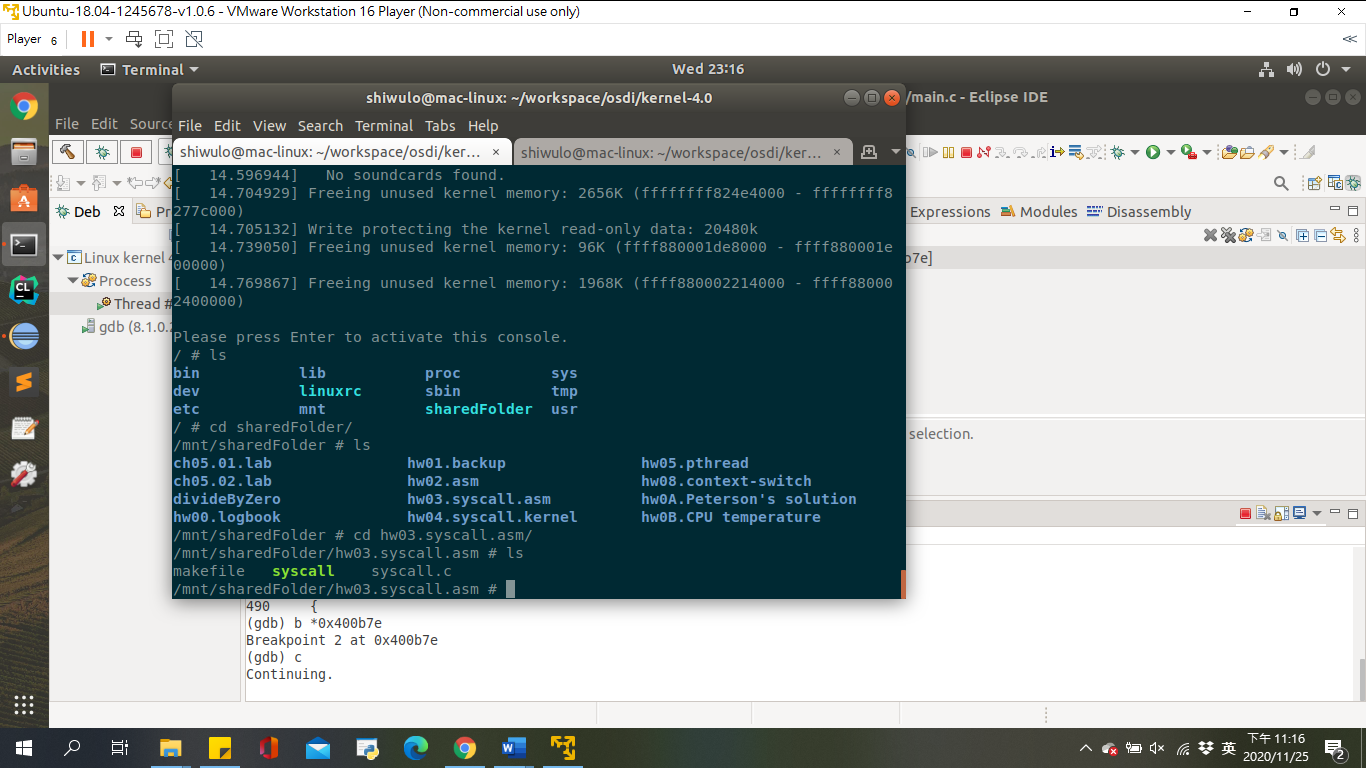
1. 利用作業3，用組語寫system\_call的程式碼



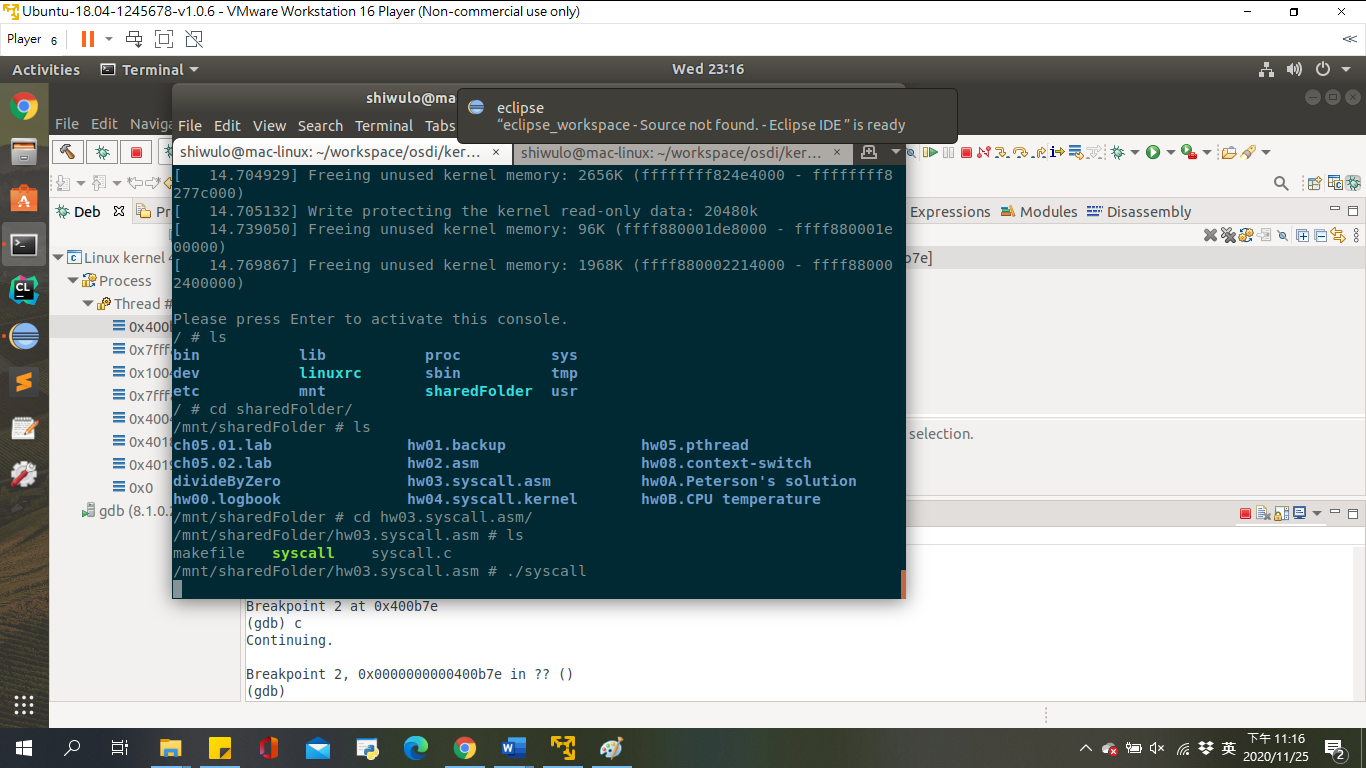
2. 再利用gdb找到該檔案的指標

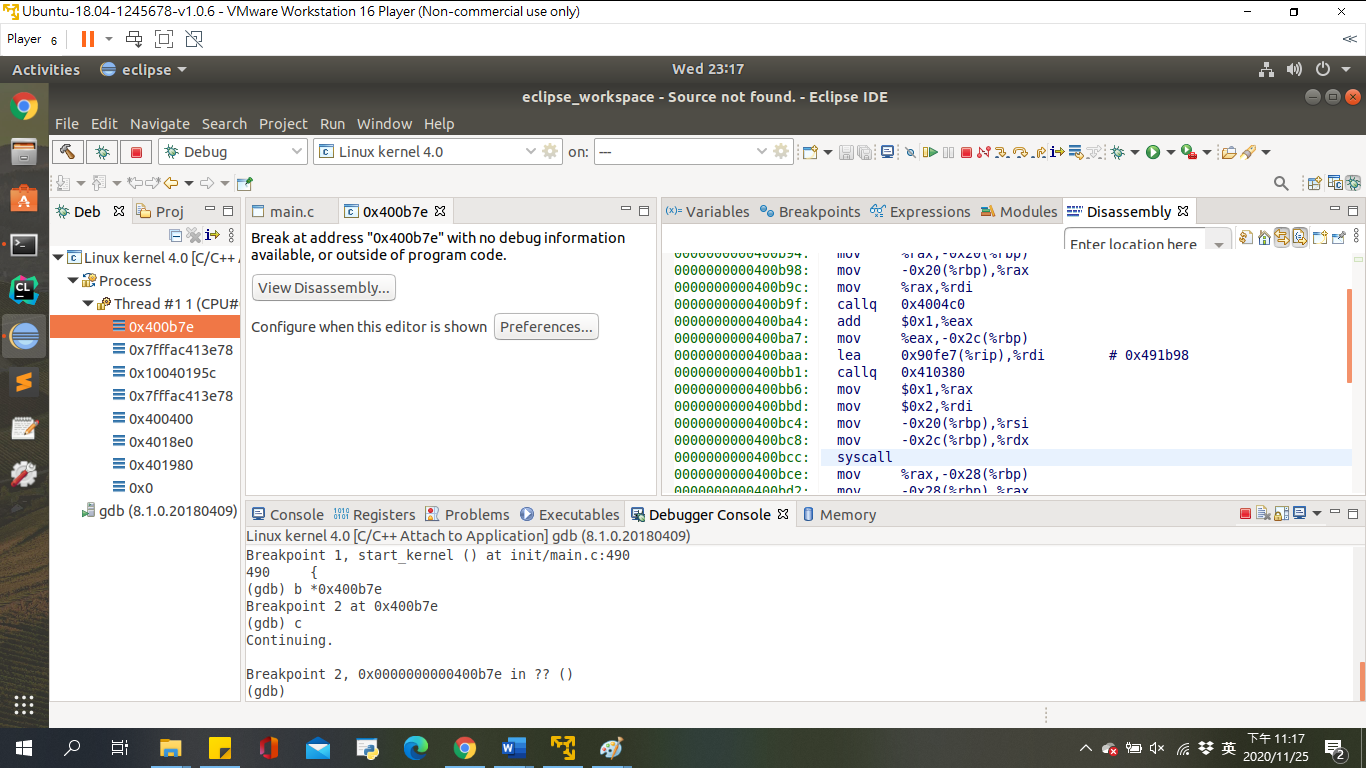


3. 設立中斷點於這個檔案「b \*0x400b7e ->c」

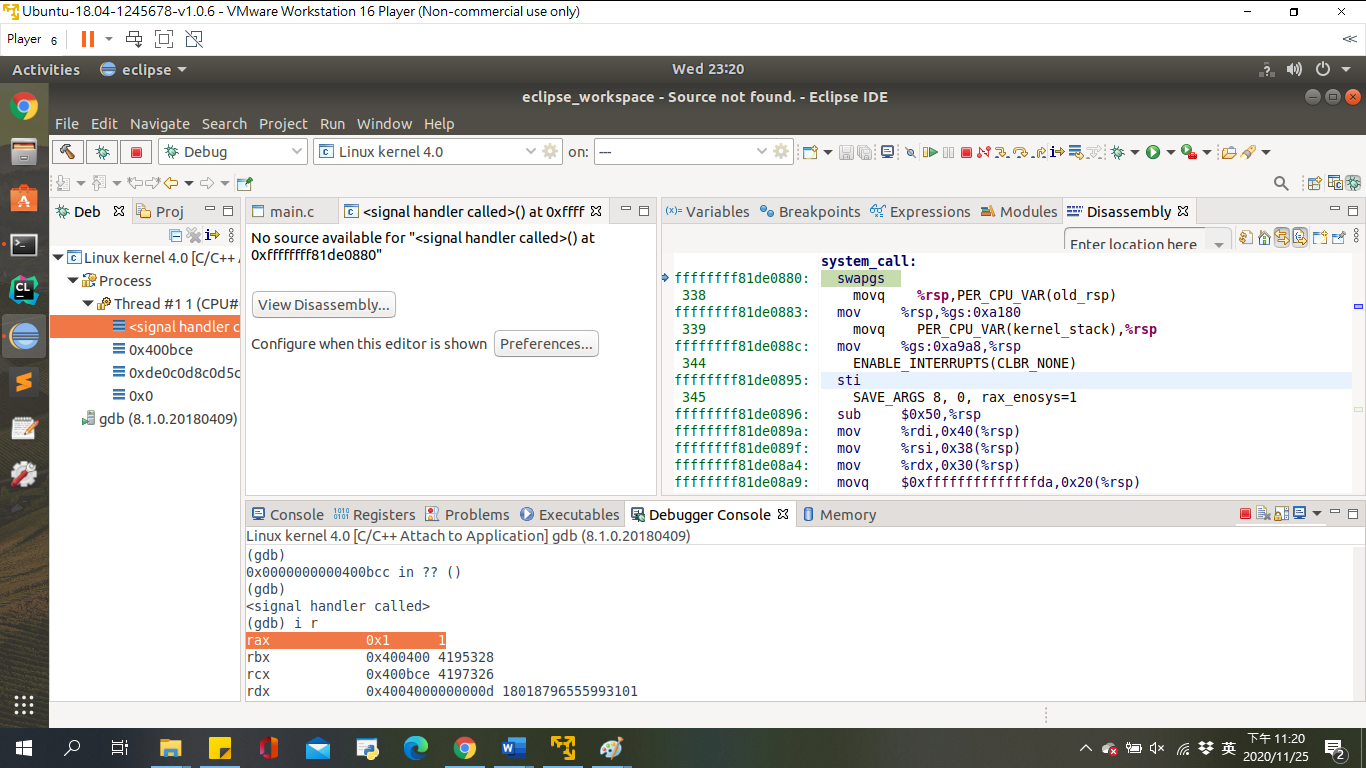


4. 執行該檔案，我們可以用eclipse找到syscall該組語的位置





1. 設立中斷點進入，並用「i r」觀看rax的值



1. 最後透過si指令發現會將rax存入的值1找到對應到的就是sys\_write這個syscall

