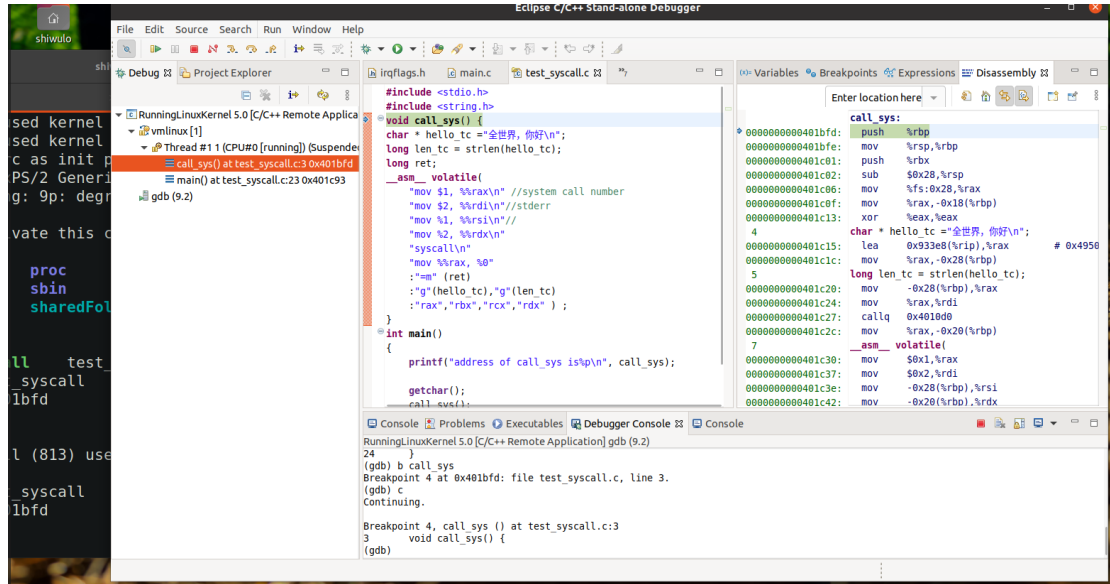


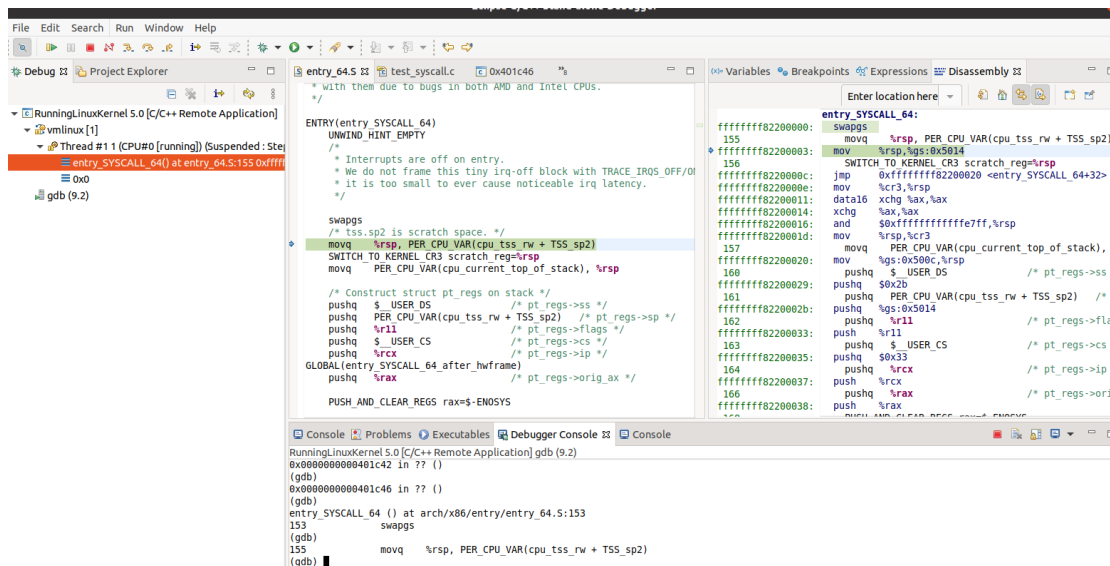
## [ 作業系統概論 HW8 ]

409410025 邱 x 恩

[ 1 ] 設定中斷點在 test\_syscall 發出 system call 之前，請在這個地方截圖



[ 2 ] 使用單步追蹤 (si)，直到 Linux kernel，請在進入 Linuxkernel 時截圖



[ 3 ] 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

利用以下兩個函式

1. `sys_call_table` 是一個函式指標。

是暫存器的名稱。

```
nr = array_index_nospec(nr, NR_syscalls);  
regs->ax = sys_call_table[nr](regs);
```

2.

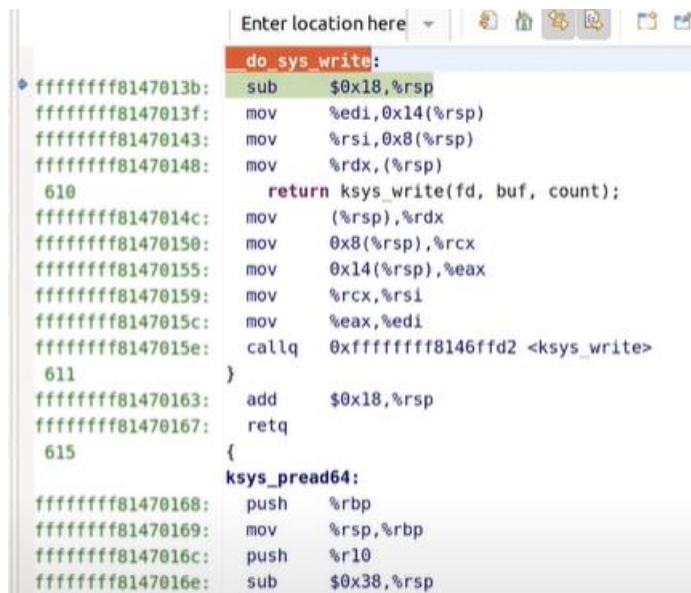
```
SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,  
size_t, count)  
{  
    return ksys_write(fd, buf, count);  
}
```

其中第二個是一個 macro，

對照組合語言來看，

展開之後可得知是由 `do_sys_write` 呼叫

如下圖



```
Enter location here  
do_sys_write:  
ffffffffff8147013b: sub    $0x18,%rsp  
ffffffffff8147013f: mov    %edi,0x14(%rsp)  
ffffffffff81470143: mov    %rsi,0x8(%rsp)  
ffffffffff81470148: mov    %rdx,(%rsp)  
610      return ksys_write(fd, buf, count);  
ffffffffff8147014c: mov    (%rsp),%rdx  
ffffffffff81470150: mov    0x8(%rsp),%rcx  
ffffffffff81470155: mov    0x14(%rsp),%eax  
ffffffffff81470159: mov    %rcx,%rsi  
ffffffffff8147015c: mov    %eax,%edi  
ffffffffff8147015e: callq  0xffffffff8146ffd2 <ksys_write>  
611      }  
ffffffffff81470163: add    $0x18,%rsp  
ffffffffff81470167: retq  
615      {  
ksys_pread64:  
ffffffffff81470168: push   %rbp  
ffffffffff81470169: mov    %rsp,%rbp  
ffffffffff8147016c: push   %r10  
ffffffffff8147016e: sub    $0x38,%rsp
```

[ 4 ] 請大致說明作業系統如何處理 write 。

```
ssize_t ksys_write(unsigned int fd, const char __user *buf, size_t count)
{
    struct fd f = fdget_pos(fd);
    ssize_t ret = -EBADF;

    if (f.file) {
        loff_t pos = file_pos_read(f.file);
        ret = vfs_write(f.file, buf, count, &pos);
        if (ret >= 0)
            file_pos_write(f.file, pos);
        fdput_pos(f);
    }

    return ret;
}
```

仔細看了 ksys\_write 的內容後，  
可以推得 OS 對於 WRITE 的處理。

首先會對檔案進行處理。  
所以會有 fdgetpos，  
以及 file\_pos\_read 對 file 進行讀取的動作。

接著就是寫入檔案  
呼叫了 vfs\_write 這個函式  
並用一個 ssize\_t 的資料結構去接住這些資訊  
而為什麼要這樣做呢？

原因是：**要判斷這個檔案的寫入是否成功與否。**

可以看到這邊判斷 ret >= 0 時，  
會利用 file\_pos\_write 將 pos 紀錄回檔案，  
最後就完成了 write 的 system call。

---