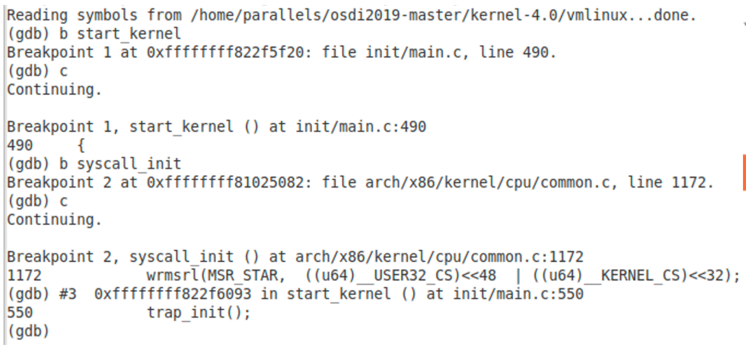
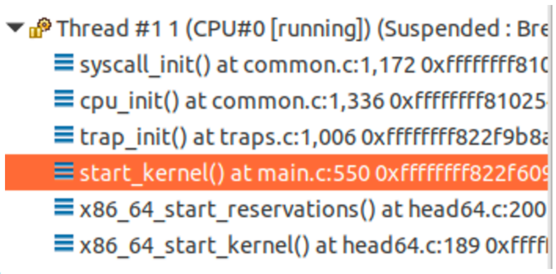
- 將中斷點設定在「syscall_init」，並回答下列問題

    1.當應用程式發出system call時，Linux會從哪裡開始執行Linux kernel

從start_kernel 開始執行trap_init，再從trap_init 執行cpu_init，cpu_ini執行syscall_init。

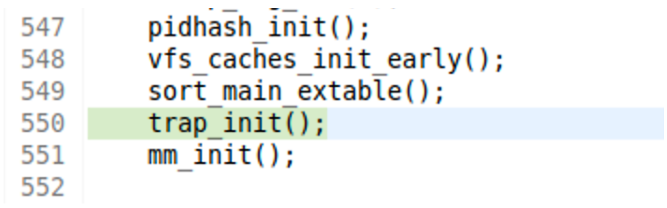


```
Reading symbols from /home/parallels/osdi2019-master/kernel-4.0/vmlinux...done.
(gdb) b start_kernel
Breakpoint 1 at 0xffffffff822f5f20: file init/main.c, line 490.
(gdb) c
Continuing.

Breakpoint 1, start_kernel () at init/main.c:490
490     {
(gdb) b syscall_init
Breakpoint 2 at 0xffffffff81025082: file arch/x86/kernel/cpu/common.c, line 1172.
(gdb) c
Continuing.

Breakpoint 2, syscall_init () at arch/x86/kernel/cpu/common.c:1172
1172            wrmsrl(MSR_STAR,  ((u64)__USER32_CS)<<48  | ((u64)__KERNEL_CS)<<32);
(gdb) #3  0xffffffff822f6093 in start_kernel () at init/main.c:550
550             trap_init();
(gdb)
```



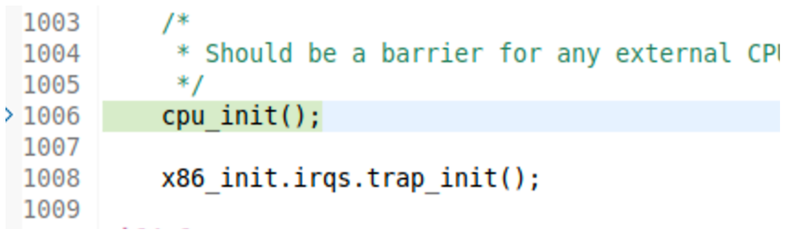
```
▼ Thread #1 1 (CPU#0 [running]) (Suspended : Bre
    syscall_init() at common.c:1,172 0xffffffff810
    cpu_init() at common.c:1,336 0xffffffff81025
    trap_init() at traps.c:1,006 0xffffffff822f9b8
    start_kernel() at main.c:550 0xffffffff822f60
    x86_64_start_reservations() at head64.c:200
    x86_64_start_kernel() at head64.c:189 0xffff
```

start_kernel:



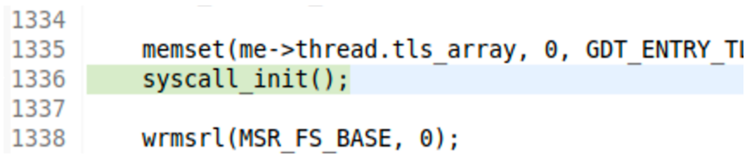
```
547     pidhash_init();
548     vfs_caches_init_early();
549     sort_main_extable();
550     trap_init();
551     mm_init();
552
```

trap_init:



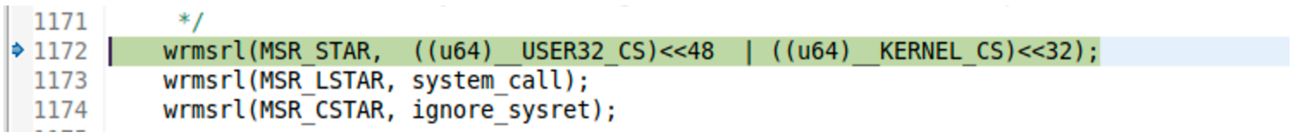
```
1003    /*
1004     * Should be a barrier for any external CP
1005     */
1006    cpu_init();
1007
1008    x86_init.irqs.trap_init();
1009
```

cpu_init:



```
1334
1335    memset(me->thread.tls_array, 0, GDT_ENTRY_TI
1336    syscall_init();
1337
1338    wrmsrl(MSR_FS_BASE, 0);
```

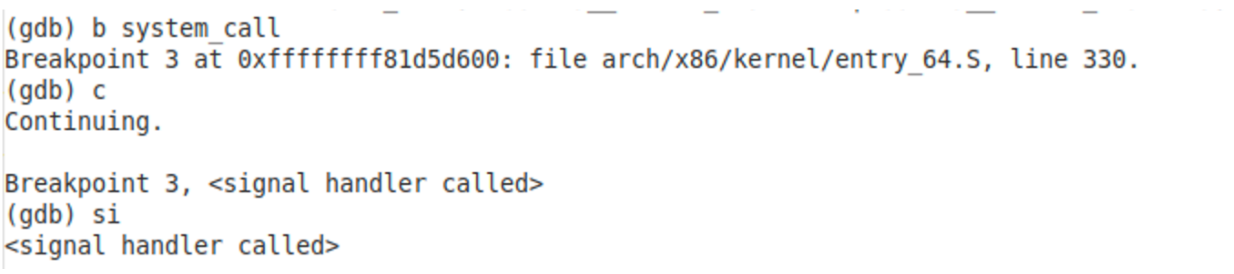
syscall_init:



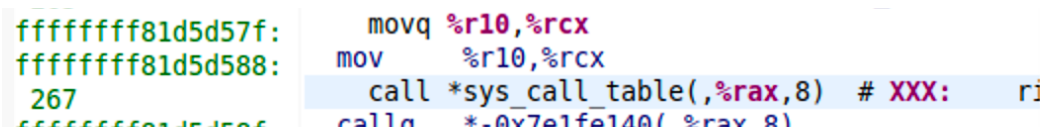
```
1171     */
1172    wrmsrl(MSR_STAR,  ((u64)__USER32_CS)<<48  | ((u64)__KERNEL_CS)<<32);
1173    wrmsrl(MSR_LSTAR, system_call);
1174    wrmsrl(MSR_CSTAR, ignore_sysret);
```

2.如何使用rax暫存器呼叫對應的system call處理函數?

使用sys_call_table可以找到對應的處理函數

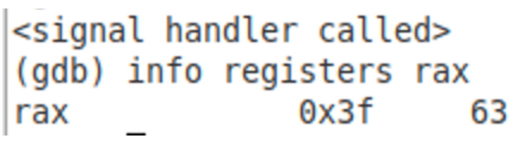


```
(gdb) b system_call
Breakpoint 3 at 0xffffffff81d5d600: file arch/x86/kernel/entry_64.S, line 330.
(gdb) c
Continuing.

Breakpoint 3, <signal handler called>
(gdb) si
<signal handler called>
```



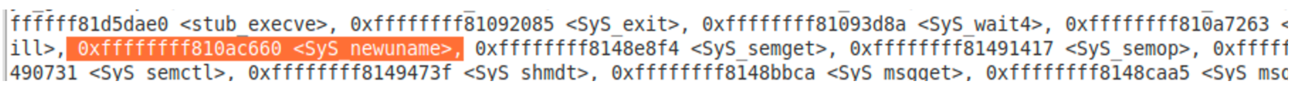
```
ffffffff81d5d57f:    movq %r10,%rcx
ffffffff81d5d588:  mov      %r10,%rcx
 267                 call *sys_call_table(,%rax,8)  # XXX:    ri
                   callq    *-0x7e1fe140(,%rax,8)
```

透過gdb 知道是63號 system call:



```
<signal handler called>
(gdb) info registers rax
rax            0x3f     63
```

第63號為newuname:



```
ffffff81d5dae0 <stub_execve>, 0xffffffff81092085 <SyS_exit>, 0xffffffff81093d8a <SyS_wait4>, 0xffffffff810a7263 <
ill>, 0xffffffff810ac660 <SyS_newuname>, 0xffffffff8148e8f4 <SyS_semget>, 0xffffffff81491417 <SyS_semop>, 0xfffff
490731 <SyS_semctl>, 0xffffffff8149473f <SyS_shmdt>, 0xffffffff8148bbca <SyS_msgget>, 0xffffffff8148caa5 <SyS_msc
```