資工三 405235035 王博輝

作業系統概論 作業二

問題一：完成_do_fork 的函數呼叫圖

```
Reading symbols from ./vmlinux...done.
(gdb) set serial baud 115200
(gdb) target remote /dev/ttyS1
Remote debugging using /dev/ttyS1

kgdb_breakpoint () at kernel/debug/debug_core.c:1073
1073            wmb(); /* Sync point after breakpoint */
(gdb)
(gdb) b _do_fork
Breakpoint 1 at 0xffffffff810f5390: file kernel/fork.c, line 2098.
(gdb) c
Continuing.
[Switching to Thread 2]

Thread 3 hit Breakpoint 1, _do_fork (clone_flags=8390417,
    stack_start=18446744071580010144, stack_size=18446612139861022400,
    parent_tidptr=0x0 <irq_stack_union>, child_tidptr=0x0 <irq_stack_union>, tls=0)
    at kernel/fork.c:2098
2098    {
(gdb)
```

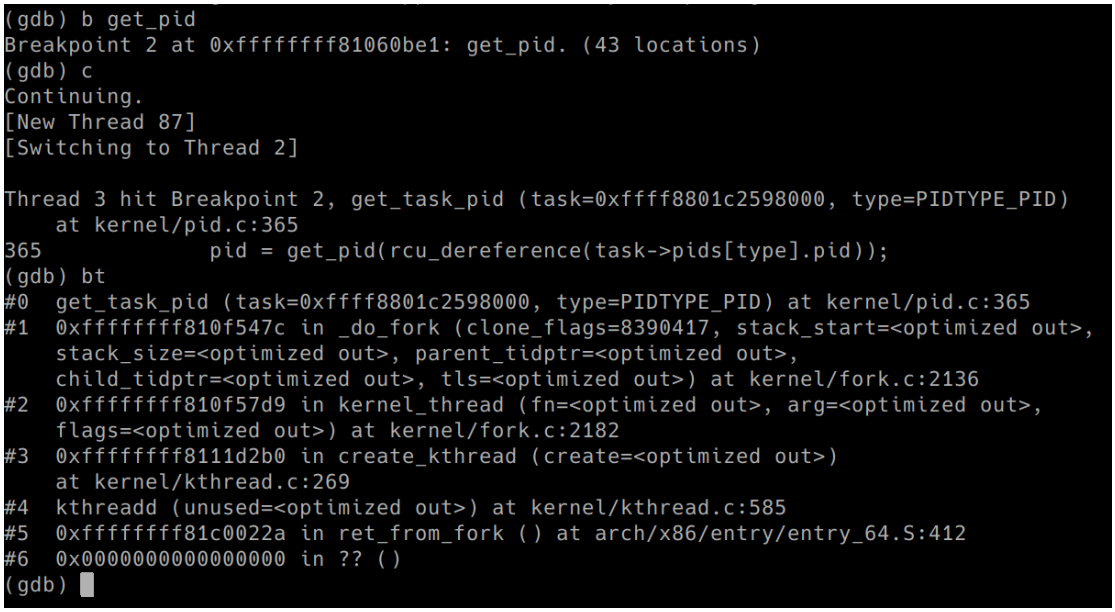
```
(gdb) bt
#0  _do_fork (clone_flags=8390417, stack_start=18446744071580010144,
    stack_size=18446612139861022400, parent_tidptr=0x0 <irq_stack_union>,
    child_tidptr=0x0 <irq_stack_union>, tls=0) at kernel/fork.c:2098
#1  0xffffffff810f57d9 in kernel_thread (fn=<optimized out>, arg=<optimized out>,
    flags=<optimized out>) at kernel/fork.c:2182
#2  0xffffffff8111d2b0 in create_kthread (create=<optimized out>)
    at kernel/kthread.c:269
#3  kthreadd (unused=<optimized out>) at kernel/kthread.c:585
#4  0xffffffff81c0022a in ret_from_fork () at arch/x86/entry/entry_64.S:412
#5  0x0000000000000000 in ?? ()
(gdb)
```

可以看到_do_for 這個函數是從 entry_64.S 開始一層一層呼叫上去的。entry_64.S 中的 ret_from_fork 呼叫 kthreadd -> create_kthreadd -> kernel_thread，最後才會呼叫_do_fork。

```
ENTRY(entry_SYSCALL_64)
        UNWIND_HINT_EMPTY
        /*
         * Interrupts are off on entry.
         * We do not frame this tiny irq-off block with TRACE_IRQS_OFF/ON,
         * it is too small to ever cause noticeable irq latency.
         */

        swapgs
        /*
         * This path is only taken when PAGE_TABLE_ISOLATION is disabled so it
         * is not required to switch CR3.
         */
        movq    %rsp, PER_CPU_VAR(rsp_scratch)
        movq    PER_CPU_VAR(cpu_current_top_of_stack), %rsp

        /* Construct struct pt_regs on stack */
        pushq   $__USER_DS                      /* pt_regs->ss */
        pushq   PER_CPU_VAR(rsp_scratch)        /* pt_regs->sp */
        pushq   %r11                            /* pt_regs->flags */
        pushq   $__USER_CS                      /* pt_regs->cs */
        pushq   %rcx                            /* pt_regs->ip */
GLOBAL(entry_SYSCALL_64_after_hwframe)
        pushq   %rax                            /* pt_regs->orig_ax */

        PUSH_AND_CLEAR_REGS rax=$-ENOSYS

        TRACE_IRQS_OFF

        /* IRQs are off. */
        movq    %rax, %rdi
        movq    %rsp, %rsi
        call    do_syscall_64           /* returns with IRQs disabled */
```

到 LXR 查看 entry_64.S 的原始碼後，發現在這個 entry 中，中斷是關閉的，所以我們無法對它除錯。

問題二：完成 getpid 的函數呼叫圖



發現 getpid 也是從 entry_64.S 中的 ret_from_fork 開始呼叫，再呼叫 kthreadd -> create_kthreadd -> kernel_thread -> _do_fork，最後才由_do_fork 來呼叫 get_task_pid。

問題三：指出_do_fork 和 getpid 的返回路徑是否不同，如果不同，為什麼不同

因為 getpid 是被_do_fork 所呼叫的，所以兩者的返回路徑不同，getpid 要返回的話會比_do_fork 多一道關卡，那就是從 getpid 返回到_do_fork