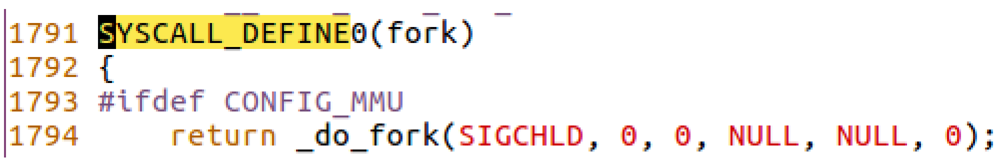
Driving - fork()

노트북:	SW		
만든 날짜:	2018-04-22 오후 7:33	수정한 날짜:	2018-04-29 오후 8:59
작성자:	fstopdg@gmail.com		

How to find sys_fork
1.
In kernel/linux-4.4
-> vi -t task_sturct (enter into 144)
-> :cs find 0 fork (enter into 76 - kernel/fork.c)
2.
On vim,
-> grep -rn "fork" | grep "SYSCALL"
-> vi kernel/fork.c
-> /SYSCALL_DEFINE(fork)



```
SYSCALL_DEFINE0(fork)
{
#ifdef CONFIG_MMU
	return _do_fork(SIGCHLD, 0, 0, NULL, NULL, 0);
```

_do_fork
parameter : SIGCHLD, 0, 0, NULL, NULL, 0

Enter into '_do_fork'

```
/* SIGCHLD, 0, 0, NULL, NULL, 0 */
long _do_fork(unsigned long clone_flags,
          unsigned long stack_start,
          unsigned long stack_size,
          int __user *parent_tidptr,
          int __user *child_tidptr,
          unsigned long tls)
{
    struct task_struct *p;
    int trace = 0;
    long nr;

    /*
     * Determine whether and which event to report to ptracer.  When
     * called from kernel_thread or CLONE_UNTRACED is explicitly
     * requested, no event is reported; otherwise, report if the event
     * for the type of forking is enabled.
     */

    /*CLONE_UNTRACED = 0x00800000 */
    if (!(clone_flags & CLONE_UNTRACED)) {
        /*CLONE_VFORK = 0x00004000*/
        if (clone_flags & CLONE_VFORK)
            trace = PTRACE_EVENT_VFORK;
        /*CSIGNAL = 0x000000ff
          if clone_flags != SIGCHLD, trace = PTRACE_EVENT_CLONE = 3
          if clone_flag = SIGCHLD, trace = PTRACE_EVENT_FORK = 1 */
        else if ((clone_flags & CSIGNAL) != SIGCHLD)
            trace = PTRACE_EVENT_CLONE;
        else
            trace = PTRACE_EVENT_FORK;

        if (likely(!ptrace_event_enabled(current, trace)))
        trace = 0;
    }

    p = copy_process(clone_flags, stack_start, stack_size,
             child_tidptr, NULL, trace, tls);
    /*
     * Do this prior waking up the new thread - the thread pointer
     * might get invalid after that point, if the thread exits quickly.
     */
    if (!IS_ERR(p)) {
        struct completion vfork;
        struct pid *pid;

        trace_sched_process_fork(current, p);

        pid = get_task_pid(p, PIDTYPE_PID);
        nr = pid_vnr(pid);

        if (clone_flags & CLONE_PARENT_SETTID)
            put_user(nr, parent_tidptr);

        if (clone_flags & CLONE_VFORK) {
            p->vfork_done = &vfork;
            init_completion(&vfork);
            get_task_struct(p);
        }

        wake_up_new_task(p);

        /* forking complete and child started to run, tell ptracer */
        if (unlikely(trace))
            ptrace_event_pid(trace, pid);

        if (clone_flags & CLONE_VFORK) {
            if (!wait_for_vfork_done(p, &vfork))
                ptrace_event_pid(PTRACE_EVENT_VFORK_DONE, pid);
        }

        put_pid(pid);
    } else {
        nr = PTR_ERR(p);
    }
    return nr;
}
```

There are 3 parts.
First, Above   p = copy_process(clone_flags, stack_start, stack_size, child_tidptr, NULL, trace, tls);
Second, p = copy_process(clone_flags, stack_start, stack_size, child_tidptr, NULL, trace, tls);
Finally, Under p = copy_process(clone_flags, stack_start, stack_size, child_tidptr, NULL, trace, tls);

-> First section,

```
/* SIGCHLD, 0, 0, NULL, NULL, 0 */
long _do_fork(unsigned long clone_flags,
          unsigned long stack_start,
          unsigned long stack_size,
          int __user *parent_tidptr,
          int __user *child_tidptr,
          unsigned long tls)
{
    struct task_struct *p;
    int trace = 0;
    long nr;

    /*
     * Determine whether and which event to report to ptracer.  When
     * called from kernel_thread or CLONE_UNTRACED is explicitly
     * requested, no event is reported; otherwise, report if the event
     * for the type of forking is enabled.
     */

    /*CLONE_UNTRACED = 0x00800000 */
    if (!(clone_flags & CLONE_UNTRACED)) {
        /*CLONE_VFORK = 0x00004000*/
        if (clone_flags & CLONE_VFORK)
            trace = PTRACE_EVENT_VFORK;
        /*CSIGNAL = 0x000000ff
          if clone_flags != SIGCHLD, trace = PTRACE_EVENT_CLONE = 3
          if clone_flag = SIGCHLD, trace = PTRACE_EVENT_FORK = 1 */
        else if ((clone_flags & CSIGNAL) != SIGCHLD)
            trace = PTRACE_EVENT_CLONE;
        else
            trace = PTRACE_EVENT_FORK;

        if (likely(!ptrace_event_enabled(current, trace)))
            trace = 0;
    }
```
SIGCHID = 17 ~ 20 = 10001 ~ 10100 = 0x11 ~ 0x14
if -> else
trace = PTRACE_EVENT_FORK = 1

if(likely(!ptrace_event_enabled(current, trace)))
likely -> trace = 0 (iot depends on value of ptrace)
Anyway, trace = 0

-> Second section
```
p = copy_process(clone_flags, stack_start, stack_size, child_tidptr, NULL, trace, tls);
```
parameter : SIGCHLD, 0, 0, NULL, NULL, 0, 0

retval = security_task_create(clone_flags)