```c
// ...
#include <linux/completion.h>
#include <linux/sched.h>
#include <linux/kthread.h>
#include <asm/signal.h>
// ...

static struct task_struct* thread_id;
static wait_queue_head_t wq;
static DECLARE_COMPLETION(on_exit);

static int thread_code( void *data ) {
    unsigned long timeout;
    // erlaubt signal SIGTERM für aktuellen thread
    allow_signal(SIGTERM);
    for (int i = 0; i < 100; i++) {
        timeout = 2 * HZ;
        /*  versetzt aktuellen thread in zustand warten (interruptible), wenn condition (timeout == 0) false
            wird aufgeweckt wenn timeout abgelaufen ist oder on signal */
        timeout = wait_event_interruptible_timeout(wq, (timeout == 0), timeout);
        printk("thread_code: woke up after 2 secs ...\n");
        // wird von signal geweckt
        if(timeout == -ERESTARTSYS) {
            printk("got signal, break\n");
            break;
        }
    }
    thread_id = 0;
    // complete and exit with returnvalue 0
    complete_and_exit(&on_exit, 0);
}

static int __init ModInit(void) {
    // ...
    init_waitqueue_head(&wq);

    thread_id = kthread_create(thread_code, NULL, "MyKThread");
    if(thread_id == 0) {
        return -EIO;
    }
    wake_up_process(thread_id);
    return 0;
}

static void __exit ModExit(void) {
    if(thread_id){
        kill_pid(task_pid(thread_id), SIGTERM, 1);
    }
    wait_for_completion(&on_exit);
    // ...
}
// ...
```