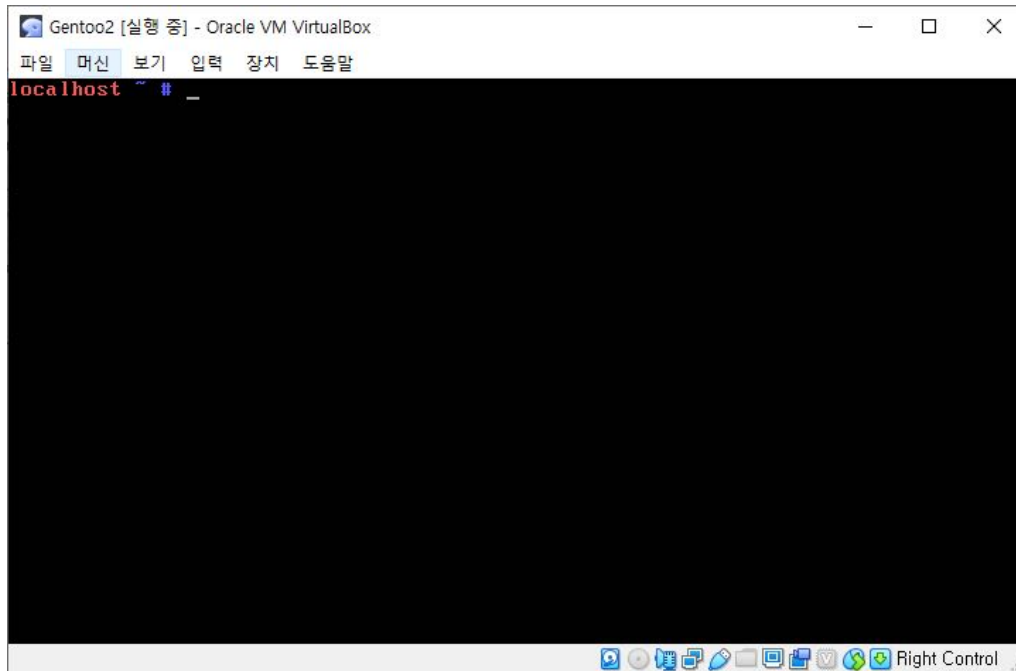
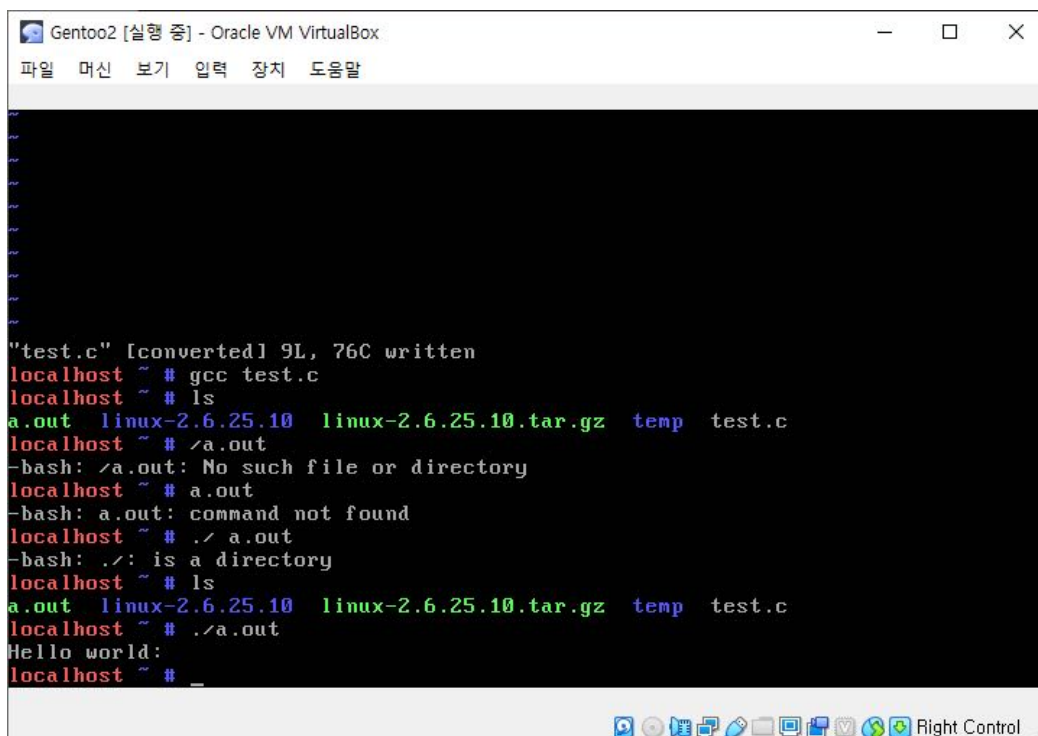


1번 답변



<'gentoo'를 실행시켜, 'root'로 login 완료함>



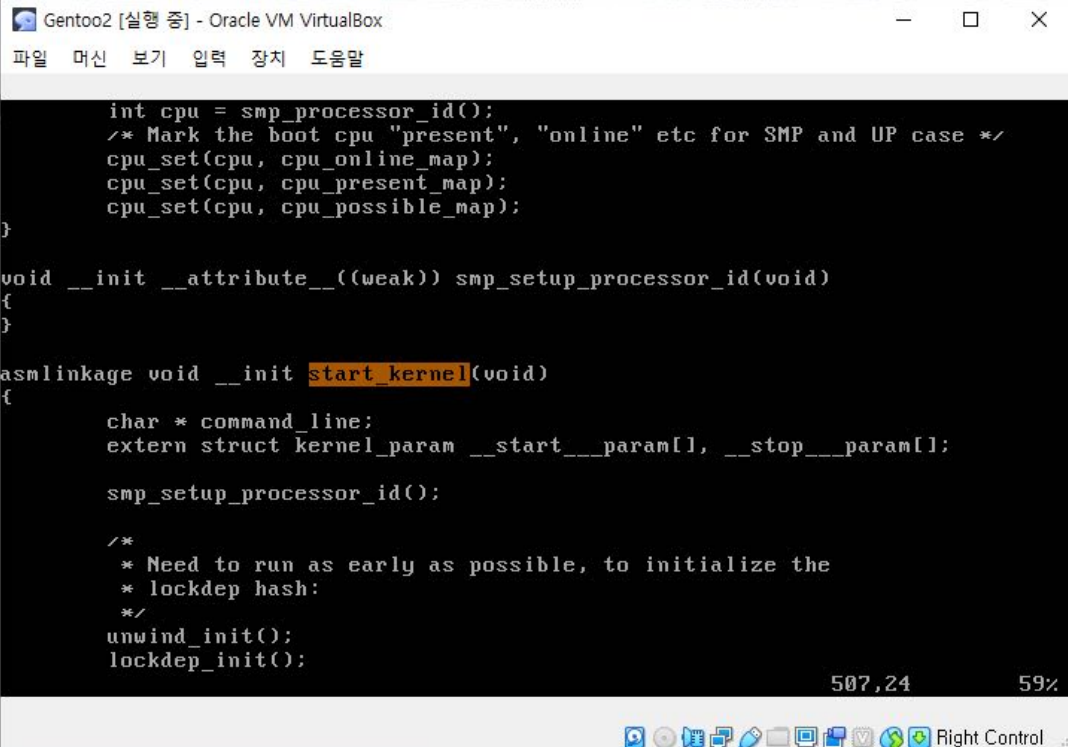
<'test.c'파일을 생성하고 컴파일하여, 'Hello world'를 화면에 출력함>

2번 답변

1. main.c의 위치 : linux-2.6.25.10/init/main.c
2. fork.c의 위치 : linux-2.6.25.10/kernel/fork.c
3. exit.c의 위치 : linux-2.6.25.10/kernel/exit.c
4. sched.c의 위치 : linux-2.6.25.10/kernel/sched.c
5. entry_32S의 위치 : linux-2.6.25.10/arch/x86/kernel/entry_32.S
6. head_32.S의 위치 : linux-2.6.25.10/arch/x86/kernel/head_32.S

※ irq_32.c, process_32.c, time_32.c 등의 코드들도 'linux-2.6.25.10/arch/x86/kernel'에 존재하는 것을 확인함

3번 답변



```
int cpu = smp_processor_id();
/* Mark the boot cpu "present", "online" etc for SMP and UP case */
cpu_set(cpu, cpu_online_map);
cpu_set(cpu, cpu_present_map);
cpu_set(cpu, cpu_possible_map);
}

void __init __attribute__((weak)) smp_setup_processor_id(void)
{
}

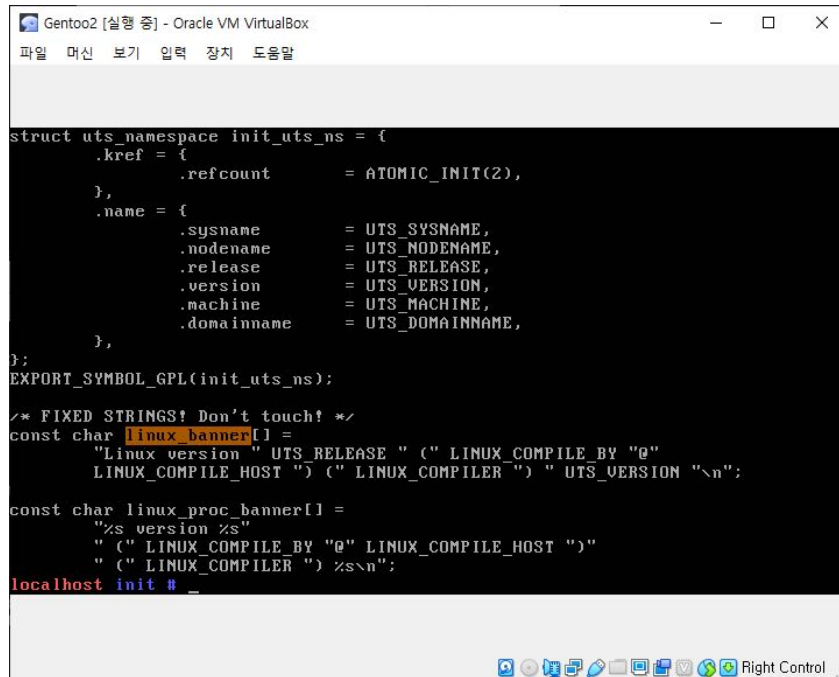
asmlinkage void __init start_kernel(void)
{
    char * command_line;
    extern struct kernel_param __start___param[], __stop___param[];

    smp_setup_processor_id();

    /*
     * Need to run as early as possible, to initialize the
     * lockdep hash:
     */
    unwind_init();
    lockdep_init();
}
```

<'init'디렉토리에 존재하는 'main.c'파일 내에 'start_kernel()'이 존재하는 것을 확인함>

4번 답변

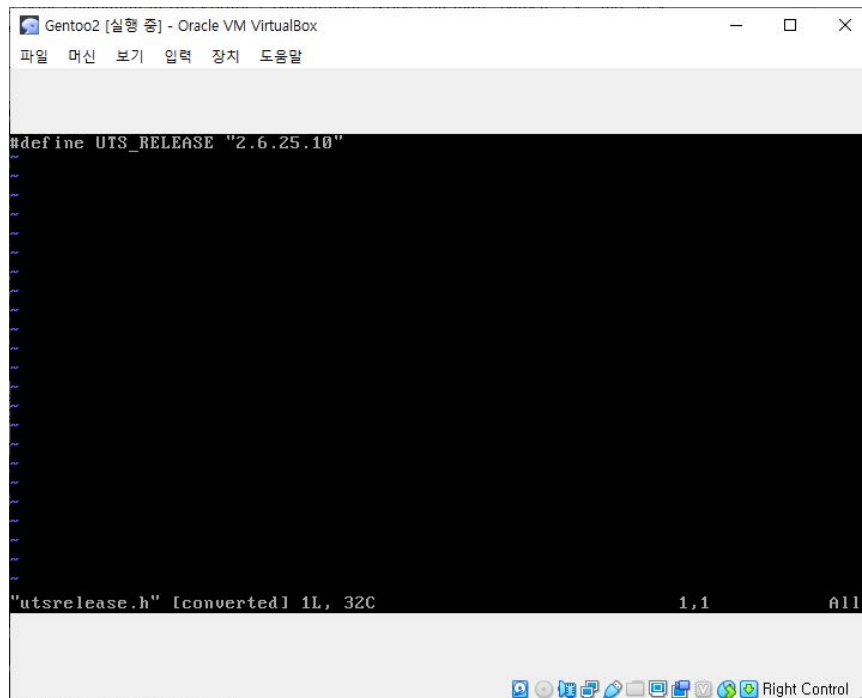


```
struct uts_namespace init_uts_ns = {
    .kref = {
        .refcount = ATOMIC_INIT(2),
    },
    .name = {
        .sysname = UTS_SYSNAME,
        .nodename = UTS_NODENAME,
        .release = UTS_RELEASE,
        .version = UTS_VERSION,
        .machine = UTS_MACHINE,
        .domainname = UTS_DOMAINNAME,
    },
};
EXPORT_SYMBOL_GPL(init_uts_ns);

/* FIXED STRINGS! Don't touch! */
const char linux_banner[] =
    "Linux version " UTS_RELEASE " (" LINUX_COMPILE_BY "@"
    LINUX_COMPILE_HOST ") (" LINUX_COMPILER ") " UTS_VERSION "\n";

const char linux_proc_banner[] =
    "%s version %s"
    " (" LINUX_COMPILE_BY "@" LINUX_COMPILE_HOST ")"
    " (" LINUX_COMPILER ") %s\n";
localhost init # _
```

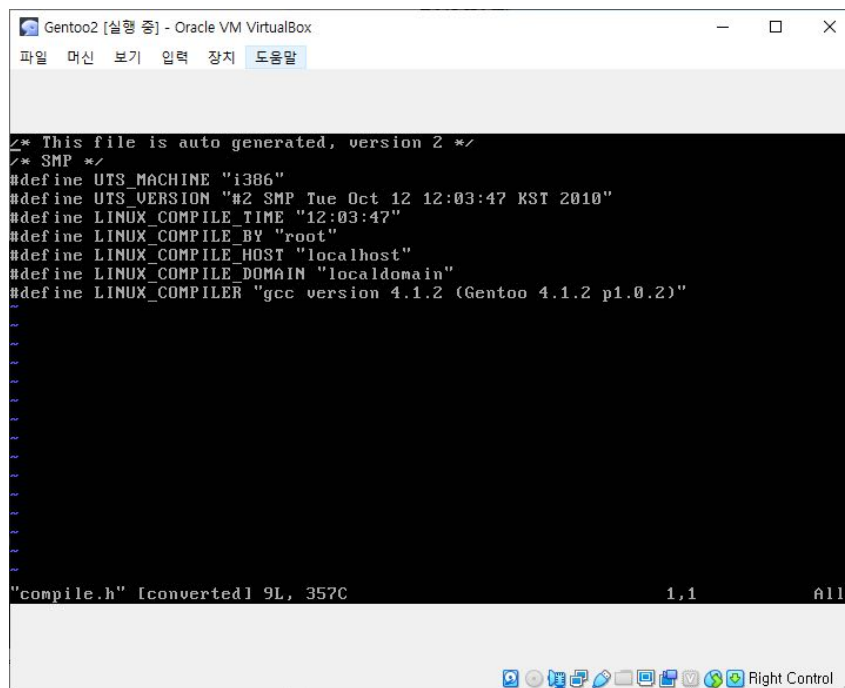
<'start_kernel()' 함수 선언 내 존재하던 코드 'printk(linux_banner)'의
매개변수 'linux_banner' 선언 부분 확인>



```
#define UTS_RELEASE "2.6.25.10"
```

"utsrelease.h" [converted] 1L, 32C 1,1 All

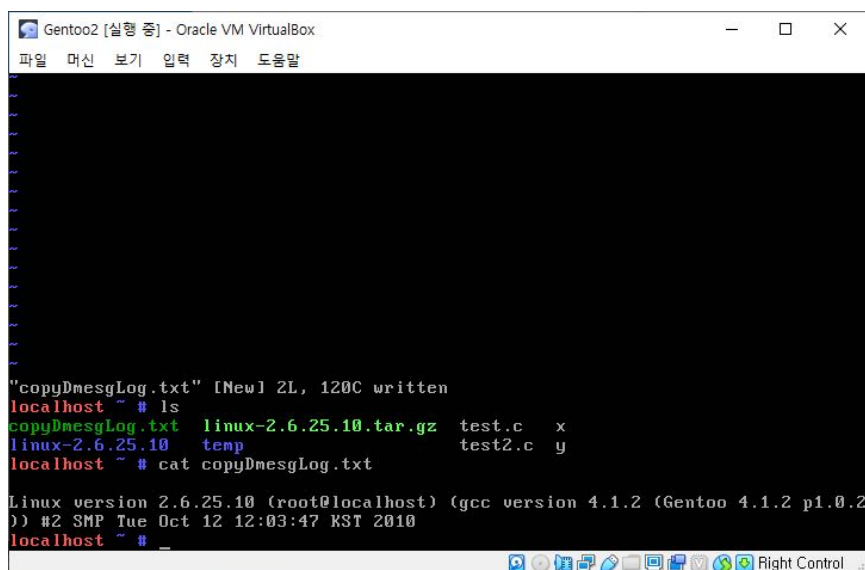
<'linux_banner' 배열 내 존재하는 'UTS_RELEASE' 선언 확인>



```
/* This file is auto generated, version 2 */
/* SMP */
#define UTS_MACHINE "i386"
#define UTS_VERSION "#2 SMP Tue Oct 12 12:03:47 KST 2010"
#define LINUX_COMPILE_TIME "12:03:47"
#define LINUX_COMPILE_BY "root"
#define LINUX_COMPILE_HOST "localhost"
#define LINUX_COMPILE_DOMAIN "localdomain"
#define LINUX_COMPILER "gcc version 4.1.2 (Gentoo 4.1.2 p1.0.2)"

"compile.h" [converted] 9L, 357C 1,1 All
```

<'UTS_VERSION', 'LINUX_COMPILE_BY', 'LINUX_COMPILE_HOST',
'LINUX_COMPILER'선언 확인>



```
"copyDmesgLog.txt" [New] 2L, 120C written
localhost ~ # ls
copyDmesgLog.txt  linux-2.6.25.10.tar.gz  test.c  x
linux-2.6.25.10  temp                  test2.c  y
localhost ~ # cat copyDmesgLog.txt

Linux version 2.6.25.10 (root@localhost) (gcc version 4.1.2 (Gentoo 4.1.2 p1.0.2
)) #2 SMP Tue Oct 12 12:03:47 KST 2010
localhost ~ #
```

<dmesg 내 첫 줄 복사 후 'copyDmesgLog.txt'파일에 붙여넣기>

5번 답변

trap_init() : 부팅시에 SystemCall 테이블을 초기화하는 함수이다.

```
void __init trap_init(void)
{
    int i;

#ifdef CONFIG_EISA
    void __iomem *p = early_ioremap(0xFFFFD9, 4);
    if (readl(p) == 'E'+('I'<<8)+'S'<<16)+'A'<<24) {
        EISA_bus = 1;
    }
    early_iounmap(p, 4);
#endif

#ifdef CONFIG_X86_LOCAL_APIC
    init_apic_mappings();
#endif

    set_trap_gate(0,&divide_error);
    set_intr_gate(1,&debug);
    set_intr_gate(2,&nmi);
    set_system_intr_gate(3, &int3); /* int3/4 can be called from all */
    set_system_gate(4,&overflow);
    set_trap_gate(5,&bounds);
}
```

1139,0-1 94%

init_IRQ() : 해당 시스템의 인터럽트 컨트롤러를 초기화하고 각각의 인터럽트 번호에 따른 핸들러들을 준비하는 함수이다.

```
void __init init_IRQ(void)
{
    int irq;

    for (irq = 0; irq < NR_IRQS; irq++)
        irq_desc[irq].status |= IRQ_NOREQUEST | IRQ_NOPROBE;

#ifdef CONFIG_SMP
    bad_irq_desc.affinity = CPU_MASK_ALL;
    bad_irq_desc.cpu = smp_processor_id();
#endif
    init_arch_irq();
}

#ifdef CONFIG_HOTPLUG_CPU
static void route_irq(struct irq_desc *desc, unsigned int irq, unsigned int cpu)
{
    pr_debug("IRQ%u: moving from cpu%u to cpu%u\n", irq, desc->cpu, cpu);

    spin_lock_irq(&desc->lock);
    desc->chip->set_affinity(irq, cpumask_of_cpu(cpu));
    spin_unlock_irq(&desc->lock);
}
```

178,6-13 83%

sched_init() : cpu스케줄을 초기화하는 함수이다.

```
void __init sched_init(void)
{
    int highest_cpu = 0;
    int i, j;

#ifdef CONFIG_SMP
    init_defrootdomain();
#endif

#ifdef CONFIG_GROUP_SCHED
    list_add(&init_task_group.list, &task_groups);
#endif

    for_each_possible_cpu(i) {
        struct rq *rq;

        rq = cpu_rq(i);
        spin_lock_init(&rq->lock);
        lockdep_set_class(&rq->lock, &rq->rq_lock_key);
        rq->nr_running = 0;
        rq->clock = 1;
        init_cfs_rq(&rq->cfs, rq);
        init_rt_rq(&rq->rt, rq);
#ifdef CONFIG_FAIR_GROUP_SCHED

```

7261,13

88%

time_init() : 시간 초기화를 수행하는 함수이다.

```

}

extern void (*late_time_init)(void);
/* Duplicate of time_init() below, with hpet_enable part added */
void __init hpet_time_init(void)
{
    if (!hpet_enable())
        setup_pit_timer();
    time_init_hook();
}

/*
 * This is called directly from init code; we must delay timer setup in the
 * HPET case as we can't make the decision to turn on HPET this early in the
 * boot process.
 *
 * The chosen time_init function will usually be hpet_time_init, above, but
 * in the case of virtual hardware, an alternative function may be substituted.
 */
void __init time_init(void)
{
    tsc_init();
    late_time_init = choose_time_init();
}

```

135,1

Bot

console_init() : console(명령조작에 사용하는 애플리케이션이나 OS자체)을 초기화하는 함수이다.

```
/*
 * Initialize the console device. This is called *early*, so
 * we can't necessarily depend on lots of kernel help here.
 * Just do some early initializations, and do the complex setup
 * later.
 */
void __init console_init(void)
{
    initcall_t *call;

    /* Setup the default TTY line discipline. */
    (void) tty_register_ldisc(N_TTY, &tty_ldisc_N_TTY);

    /*
     * set up the console device so that later boot sequences can
     * inform about problems etc..
     */
    call = __con_initcall_start;
    while (call < __con_initcall_end) {
        (*call)();
        call++;
    }
}
```

4037,2 98%

mem_init() : 메모리의 초기화를 담당하는 함수이다.

```
void __init mem_init(void)
{
    int codesize, reservedpages, datasize, initsize;
    int tmp, bad_ppro;

#ifdef CONFIG_FLATMEM
    BUG_ON(!mem_map);
#endif
    bad_ppro = ppro_with_ram_bug();

#ifdef CONFIG_HIGHMEM
    /* check that fixmap and pkmap do not overlap */
    if (PKMAP_BASE + LAST_PKMAP*PAGE_SIZE >= FIXADDR_START) {
        printk(KERN_ERR
               "fixmap and knap areas overlap - this will crash\n");
        printk(KERN_ERR "pkstart: %lxh pkend: %lxh fixstart %lxh\n",
               PKMAP_BASE, PKMAP_BASE + LAST_PKMAP*PAGE_SIZE,
               FIXADDR_START);
        BUG();
    }
#endif
    /* this will put all low memory onto the freelists */
    totalram_pages += free_all_bootmem();
}
```

569,13 72%

rest_init() : 초기화된 메모리를 사용할 수 있게 하고, init()함수를 실행시켜 부팅 과정을 끝내는 함수이다.

```
/*
 * We need to finalize in a non-__init function or else race conditions
 * between the root thread and the init thread may cause start_kernel to
 * be reaped by free_initmem before the root thread has proceeded to
 * cpu_idle.
 *
 * gcc-3.4 accidentally inlines this function, so use noinline.
 */
static void noinline __init_refok rest_init(void)
{
    __releases(kernel_lock)

    int pid;

    kernel_thread(kernel_init, NULL, CLONE_FS | CLONE_SIGHAND);
    numa_default_policy();
    pid = kernel_thread(kthreadd, NULL, CLONE_FS | CLONE_FILES);
    kthreadd_task = find_task_by_pid(pid);
    unlock_kernel();

    /*
     * The boot idle thread must execute schedule()
     * at least once to get things moving:
     */
}
```

432,1 50%

6번 답변

'printf()'는 'stdio.h'라는 헤더파일에 정의되어있고 해당 헤더파일은 'root/linux-2.6.25.10/include/linux' 경로에 존재하지 않는다. 그렇기 때문에 커널(운영체제 코드)을 내포하고 있는 파일들 내에는 'stdio.h'가 include되어 있지 않다. 그러나 'printk()'는 'kernel.h'라는 헤더파일에 정의되어있고 해당 헤더파일은 'root/linux-2.6.25.10/include/linux'에 존재하며, 커널을 내포하고 있는 파일들 내에는 'kernel.h'가 include 되어있다. 이 때문에, 'printf()'는 사용할 수 없고 'printk()'만 사용할 수 있다.