

0-1) When you boot the Linux system, the first program that runs is BIOS. Where is this program (the memory location)

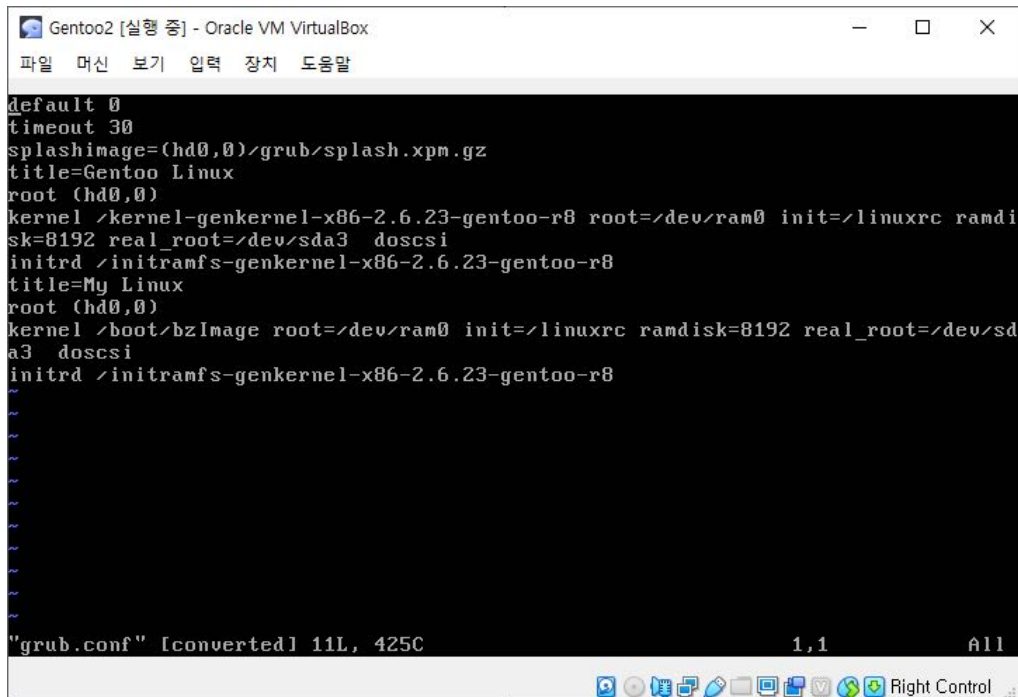
답변 : FFFFFFFF0

0-2) BIOS loads and runs the boot loader program (GRUB in Linux). Where is this GRUB program?

답변 : 00007000

0-3) GRUB loads and runs the Linux executable file. Where is Linux executable file? How GRUB knows the location of Linux executable file?

답변 : Linux executable file의 위치는 '/boot/bzImage'이고, 'grub.conf' 내 grub이 불러드릴 항목들 중 '/boot/bzImage'이 명시되어 있다.

A screenshot of a terminal window titled 'Gentoo2 [실행 중] - Oracle VM VirtualBox'. The terminal displays the contents of the GRUB configuration file, 'grub.conf'. The configuration includes settings for the default boot entry, timeout, splash image, and two boot entries. The first entry is for 'Gentoo Linux' and the second is for 'My Linux'. Both entries specify the kernel path, root device, init file, ramdisk, and real root. The terminal output shows the configuration file being converted to a binary format. The status bar at the bottom indicates the file is 'grub.conf' [converted] 11L, 425C, with a cursor at line 1, column 1. The window has a menu bar with '파일', '메신', '보기', '입력', '장치', and '도움말'. The bottom of the window shows a taskbar with various icons and a 'Right Control' button.

<'grub.conf'파일 내 적혀져있는 사항들>

1) Simple modification of the kernel

```

Gentoo2 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
Linux version 2.6.25.10 (root@localhost) (gcc version 4.1.2 (Gentoo 4.1.2 p1.0.2
)) #4 SMP Fri Sep 13 10:23:29 KST 2019
hello from me
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 0000000000009fc00 (usable)
 BIOS-e820: 0000000000009fc00 - 000000000000a0000 (reserved)
 BIOS-e820: 000000000000f0000 - 00000000000100000 (reserved)
 BIOS-e820: 00000000000100000 - 00000000000100000 (usable)
 BIOS-e820: 00000000000100000 - 00000000000100000 (ACPI data)
 BIOS-e820: 00000000fec00000 - 00000000fec01000 (reserved)
 BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
 BIOS-e820: 00000000fffc0000 - 00000000100000000 (reserved)
WARNING: strange, CPU MTRRs all blank?
-----[ cut here ]-----
WARNING: at arch/x86/kernel/cpu/mtrr/main.c:696 mtrr_trim_uncached_memory+0x16f/
0x17a()
Modules linked in:
Pid: 0, comm: swapper Not tainted 2.6.25.10 #4
[<c011d61f>] warn_on_slowpath+0x40/0x4f
[<c011ddb0>] release_console_sem+0x180/0x199
[<c011ddb0>] release_console_sem+0x180/0x199
[<c011ddb0>] release_console_sem+0x180/0x199
[<c012e205>] printk+0x14/0x10
[<c05c6ec3>] mtrr_trim_uncached_memory+0x16f/0x17a
"x" [converted] 290L, 13583C
21,7 Top

```

<main.c를 수정하여 부팅 때 "hello from me"출력하기>

- 2) start_kernel() calls trap_init(), and there are many trap_init() functions defined in the kernel code. Make an intelligent guess about which trap_init() would be called and insert some printk() in the beginning of this trap_init() to see if it is really called by the kernel. Use grep in the top directory of the linux source tree to find out the locations of trap_init():

```

Gentoo2 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
crash_dump_64.c      ioport.c             ptrace.o             topology.o
doublefault_32.c     ioport.o             quirks.c             trampoline_32.S
doublefault_32.o     irq_32.c             quirks.o             trampoline_32.o
ds.c                 irq_32.o             reboot.c             trampoline_64.S
ds.o                 irq_64.c             reboot.o             traps_32.c
e820_32.c            k8.c                 reboot_fixups_32.c   traps_32.o
e820_32.o            k8.o                 relocate_kernel_32.S traps_64.c
e820_64.c            kdebugfs.c           relocate_kernel_64.S tsc_32.c
early-quirks.c       kdebugfs.o           rtc.c                tsc_32.o
early-quirks.o       kprobes.c            rtc.o                tsc_64.c
early_printk.c       kprobes.o            scx200_32.c          tsc_sync.c
early_printk.o       ldt.c                setup64.c            tsc_sync.o
efi.c                ldt.o                setup_32.c           verify_cpu_64.S
efi_32.c             machine_kexec_32.c    setup_32.o           vm86_32.c
efi_64.c             machine_kexec_64.c    setup_64.c           vm86_32.o
efi_stub_32.S        mca_32.c             sigframe_32.h        vmi_32.c
efi_stub_64.S        mfgpt_32.c           signal_32.c          vmiclock_32.c
entry_32.S           microcode.c           signal_32.o          vmlinux.lds
entry_32.o           module_32.c           signal_64.c          vmlinux.lds.S
entry_64.S           module_32.o           smp_32.c            vmlinux_32.lds.S
genapic_64.c         module_64.c           smp_32.o            vmlinux_64.lds.S
genapic_flat_64.c    module_64.c           smp_64.c            vsmp_64.c
geode_32.c           mpparse_32.c          smptboot_32.c        vsyscall_64.c
head64.c             mpparse_32.o          smptboot_32.o        x8664_ksyms_64.c
localhost kernel #

```

<'trap_init()'는 '/linux-2.6.25.10/arch/x86/kernel/traps_32.c'에 정의되어 있음>

```
Gentoo2 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

printk(KERN_EMERG "math-emulation not enabled and no coprocessor found.\n");

printk(KERN_EMERG "killing %s.\n",current->comm);
force_sig(SIGFPE,current);
schedule();
}

#endif /* CONFIG_MATH_EMULATION */

void __init trap_init(void)
{
    printk("This trap is called from choi\n");
    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
    "traps_32.c" [converted] 1227L, 31321C
    1142,2-9 93%
    Right Control
```

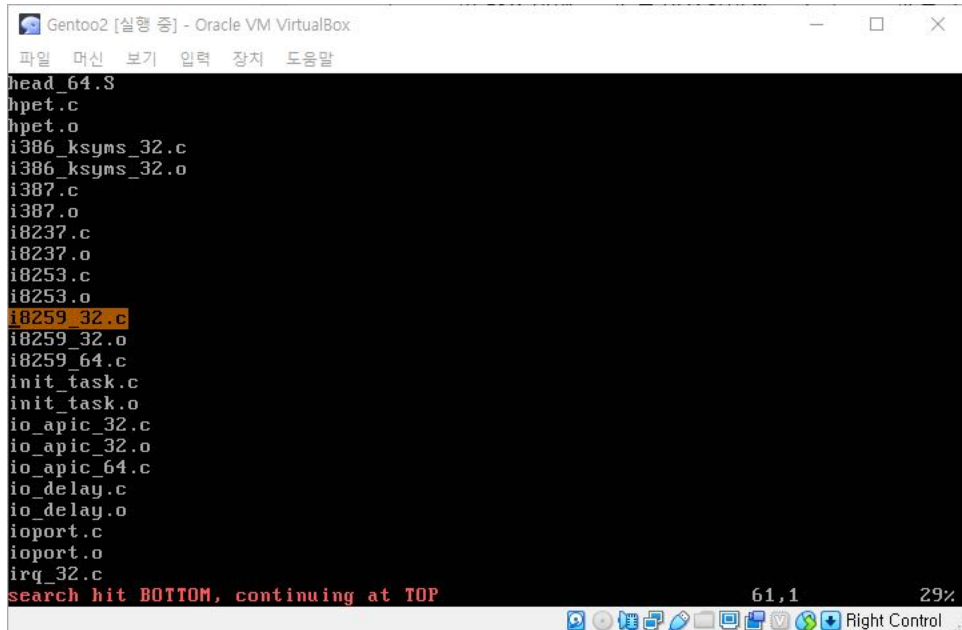
<'trap_init()'에 'printk("This trap is called from choi") 코드 삽입>

```
Gentoo2 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

ACPI: RSDP 000E0000, 0024 (r2 VBOX )
ACPI: XSDT 0FFF0030, 0034 (r1 VBOX VBOXXSDT 1 ASL 61)
ACPI: FACP 0FFF00F0, 00F4 (r4 VBOX VBOXFACP 1 ASL 61)
ACPI: DSDT 0FFF0410, 22EA (r2 VBOX VBOXBIOS 2 INTL 20100528)
ACPI: FACS 0FFF0200, 0040
ACPI: SSDT 0FFF0240, 01CC (r1 VBOX VBOXCPU 2 INTL 20100528)
ACPI: PM-Timer IO Port: 0x4008
Allocating PCI resources starting at 20000000 (gap: 10000000:eec00000)
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 65009
Kernel command line: root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root=/dev/sd
a3 doscsi
This trap is called from choi
No local APIC present or hardware disabled
mapped APIC to ffff0000 (0120b000)
Enabling fast FPU save and restore... done.
Enabling unmasked SIMD FPU exception support... done.
Initializing CPU#0
PID hash table entries: 1024 (order: 10, 4096 bytes)
Detected 3699.765 MHz processor.
Console: colour VGA+ 80x25
console [tty0] enabled
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
search hit BOTTOM, continuing at TOP
63,1 19%
Right Control
```

<부팅 때 위에서 추가한 printk구문이 출력된 것을 확인함>

3) Find also the exact locations of `init_IRQ()` and insert some `printk()` in the beginning of `init_IRQ()` to confirm (actually you insert it in `native_init_IRQ`). Do the same thing for `init_timers()` and `time_init()`

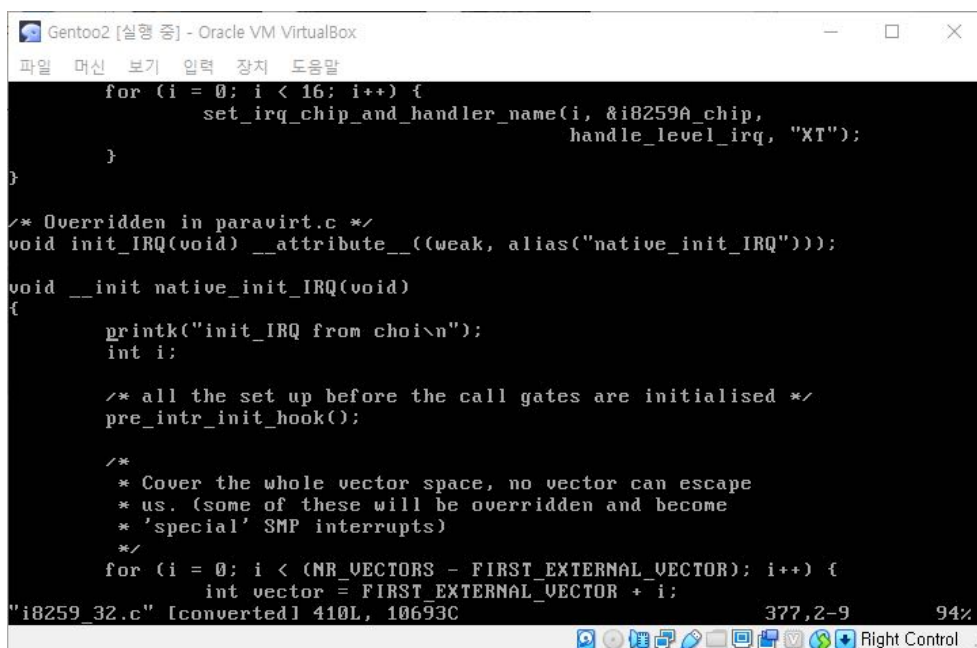


```

head_64.S
hpet.c
hpet.o
i386_ksyms_32.c
i386_ksyms_32.o
i387.c
i387.o
i8237.c
i8237.o
i8253.c
i8253.o
i8259_32.c
i8259_32.o
i8259_64.c
init_task.c
init_task.o
io_apic_32.c
io_apic_32.o
io_apic_64.c
io_delay.c
io_delay.o
ioport.c
ioport.o
irq_32.c
search hit BOTTOM, continuing at TOP
61,1 29%

```

<'native_init_IRQ()'는 '/linux-2.6.25.10/arch/x86/kernel/i8259_32.c'에 정의되어 있음>



```

    for (i = 0; i < 16; i++) {
        set_irq_chip_and_handler_name(i, &i8259A_chip,
                                       handle_level_irq, "XT");
    }
}

/* Overridden in paravirt.c */
void init_IRQ(void) __attribute__((weak, alias("native_init_IRQ")));

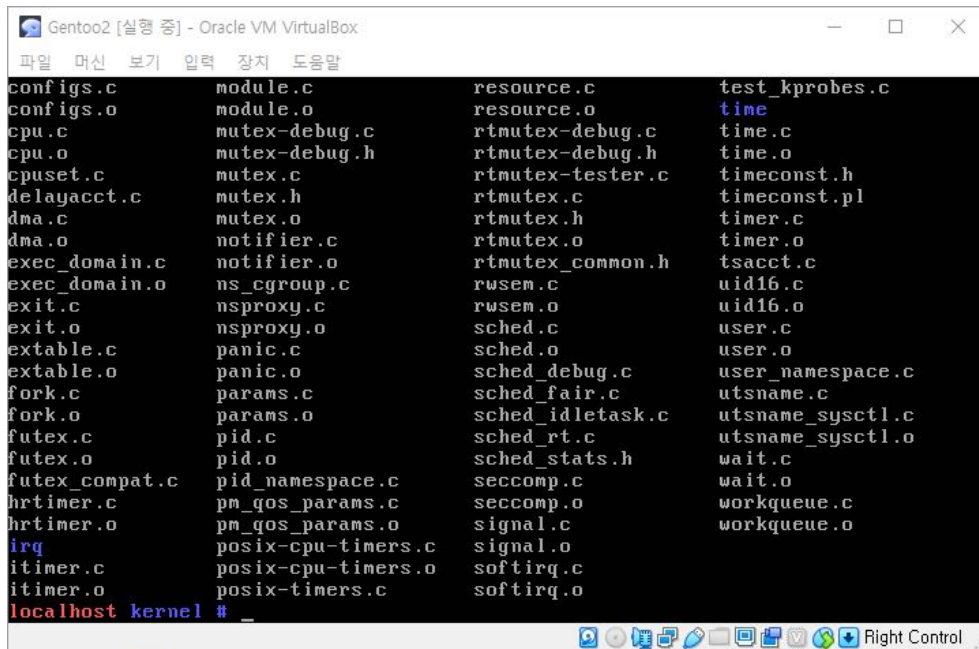
void __init native_init_IRQ(void)
{
    printk("init_IRQ from choi\n");
    int i;

    /* all the set up before the call gates are initialised */
    pre_intr_init_hook();

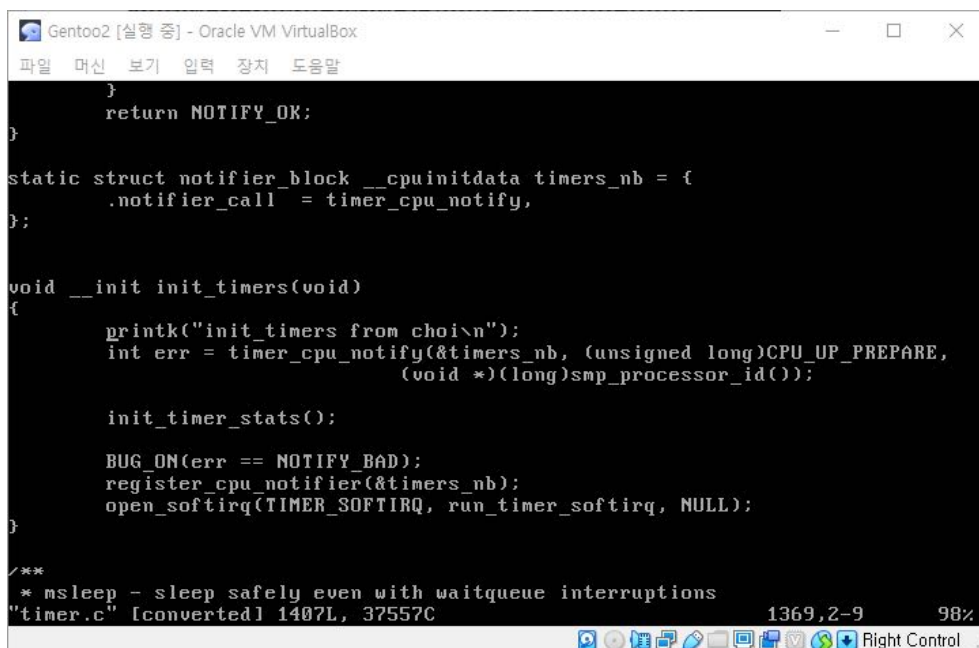
    /*
     * Cover the whole vector space, no vector can escape
     * us. (some of these will be overridden and become
     * 'special' SMP interrupts)
     */
    for (i = 0; i < (NR_VECTORS - FIRST_EXTERNAL_VECTOR); i++) {
        int vector = FIRST_EXTERNAL_VECTOR + i;
        "i8259_32.c" [converted] 410L, 10693C
    }
}
377,2-9 94%

```

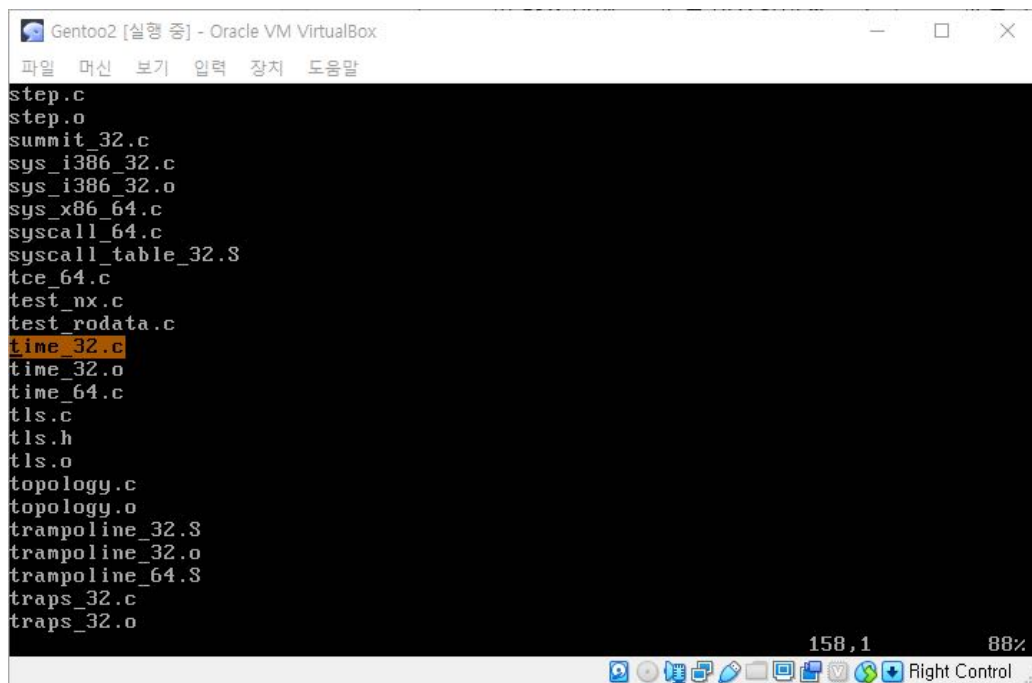
<'native_init_IRQ()'에 `printk()`추가>



<'init_timers()'는 '/linux-2.6.25.10/kernel/timer.c'에 정의되어 있다.>

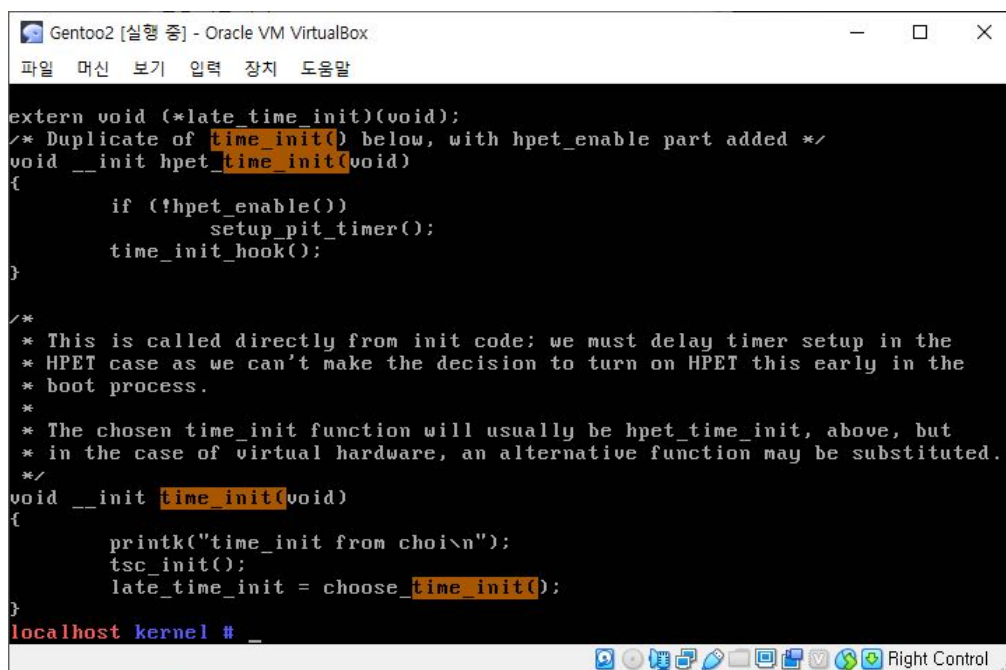


<'init_timers()'에 'printk()'추가>



```
step.c
step.o
summit_32.c
sys_i386_32.c
sys_i386_32.o
sys_x86_64.c
syscall_64.c
syscall_table_32.S
tce_64.c
test_nx.c
test_rodata.c
time_32.c
time_32.o
time_64.c
tls.c
tls.h
tls.o
topology.c
topology.o
trampoline_32.S
trampoline_32.o
trampoline_64.S
traps_32.c
traps_32.o
```

<'time_init()'은 '/linux-2.6.25.10/arch/x86/kernel/time_32.c'에 정의되어 있음>



```
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();
}

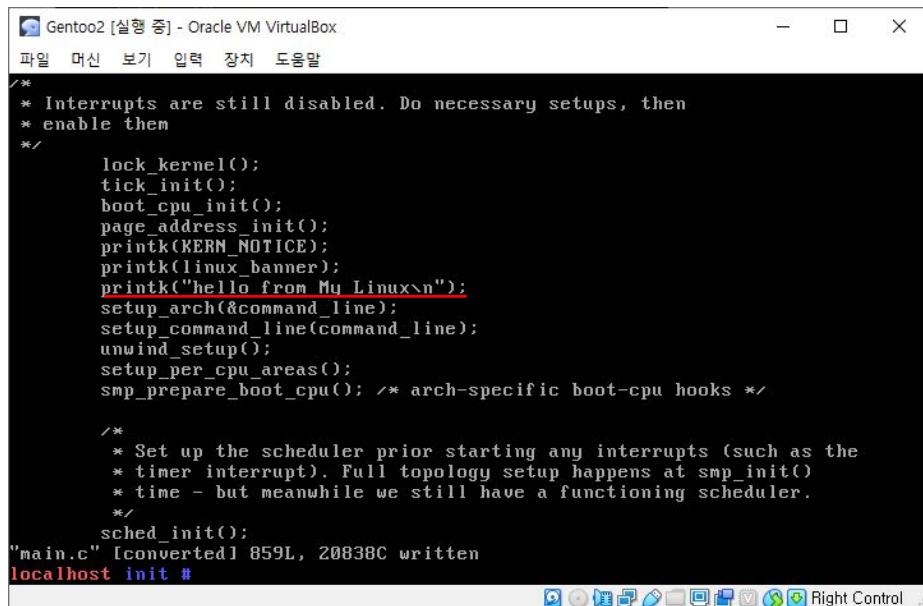
/*
 * This is called directly from init code: we must delay timer setup in the
 * HPET case as we can't make the decision to turn on HPET this early in the
 * boot process.
 *
 * The chosen time_init function will usually be hpet_time_init, above, but
 * in the case of virtual hardware, an alternative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

<'time_init()'에 'printk()'추가>


```
Gentoo2 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
Allocating PCI resources starting at 20000000 (gap: 10000000:eec00000)
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 65009
Kernel command line: root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root=/dev/sd
a3 doscsi
This trap is called from choi
No local APIC present or hardware disabled
mapped APIC to fffffb000 (0120b000)
Enabling fast FPU save and restore... done.
Enabling unmasked SIMD FPU exception support... done.
Initializing CPU#0
init_IRQ from choi
PID hash table entries: 1024 (order: 10, 4096 bytes)
init_timers from choi
time_init from choi
Detected 3699.937 MHz processor.
Console: colour VGA+ 80x25
console [tty0] enabled
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 248536k/262080k available (3163k kernel code, 13020k reserved, 1660k dat
a, 284k init, 0k highmem)
virtual kernel memory layout:
  fixmap : 0xffe14000 - 0xfffff000 (1964 kB)
  pkmap : 0xff800000 - 0xffc00000 (4096 kB)
72,19 21%
```

<부팅 때 추가한 printk문 확인 완료>

4. Modify `/boot/grub/grub.conf` so that GRUB displays another Linux selection, My Linux2. Set the location of the kernel for this linux as `/boot/bzImage2`. Prepare two versions of My Linux such that when you select "My Linux" the kernel will display "hello from My Linux", and when you select "My Linux2", it displays "hello from My Linux2".



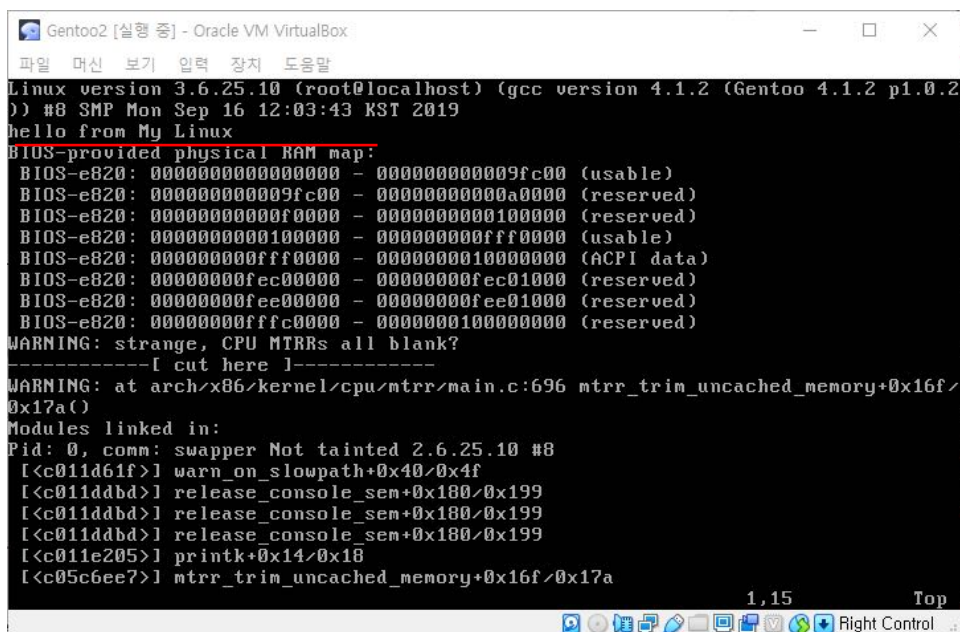
```

/*
 * Interrupts are still disabled. Do necessary setups, then
 * enable them
 */
lock_kernel();
tick_init();
boot_cpu_init();
page_address_init();
printk(KERN_NOTICE);
printk(linux_banner);
printk("hello from My Linux\n");
setup_arch(&command_line);
setup_command_line(Command_line);
unwind_setup();
setup_per_cpu_areas();
smp_prepare_boot_cpu(); /* arch-specific boot-cpu hooks */

/*
 * Set up the scheduler prior starting any interrupts (such as the
 * timer interrupt). Full topology setup happens at smp_init()
 * time - but meanwhile we still have a functioning scheduler.
 */
sched_init();
"main.c" [converted] 859L, 20838C written
localhost init #

```

<'main.c'에 'printk("hello from My Linux")'를 추가함>

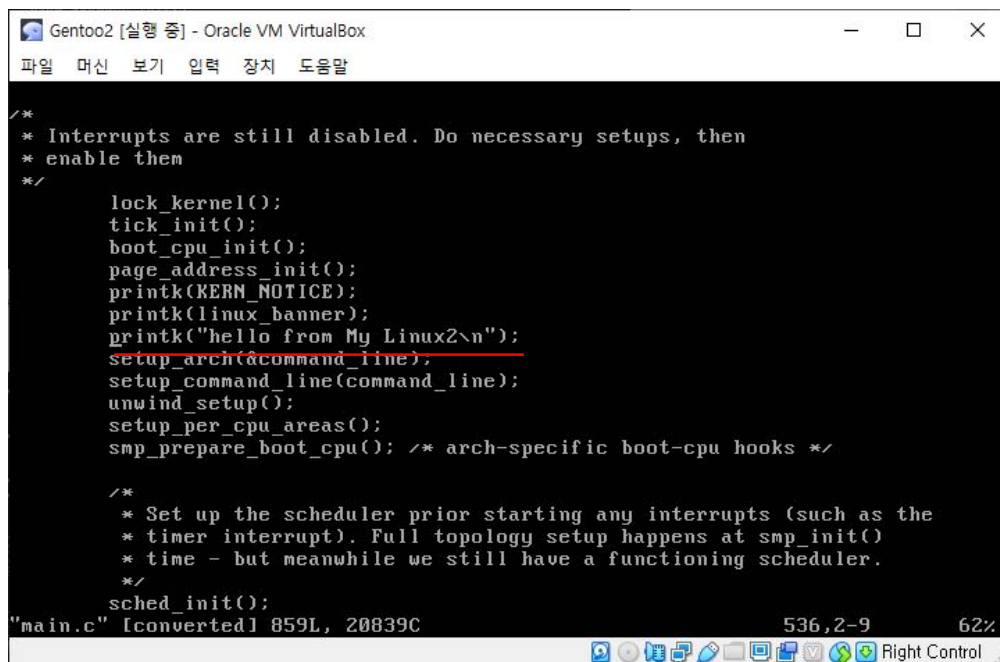


```

Linux version 3.6.25.10 (root@localhost) (gcc version 4.1.2 (Gentoo 4.1.2 p1.0.2
)) #8 SMP Mon Sep 16 12:03:43 KST 2019
hello from My Linux
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
 BIOS-e820: 000000000009fc00 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000f0000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000000fff000 (usable)
 BIOS-e820: 0000000000fff000 - 0000000010000000 (ACPI data)
 BIOS-e820: 00000000fec00000 - 00000000fec01000 (reserved)
 BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
 BIOS-e820: 00000000fffc0000 - 0000000010000000 (reserved)
WARNING: strange, CPU MTRRs all blank?
-----[ cut here ]-----
WARNING: at arch/x86/kernel/cpu/mtrr/main.c:696 mtrr_trim_uncached_memory+0x16f/
0x17a()
Modules linked in:
Pid: 0, comm: swapper Not tainted 2.6.25.10 #8
[<c011d61f>] warn_on_slowpath+0x40/0x4f
[<c011ddb0>] release_console_sem+0x180/0x199
[<c011ddb0>] release_console_sem+0x180/0x199
[<c011ddb0>] release_console_sem+0x180/0x199
[<c011e205>] printk+0x14/0x18
[<c05c6ee7>] mtrr_trim_uncached_memory+0x16f/0x17a
1,15 Top

```

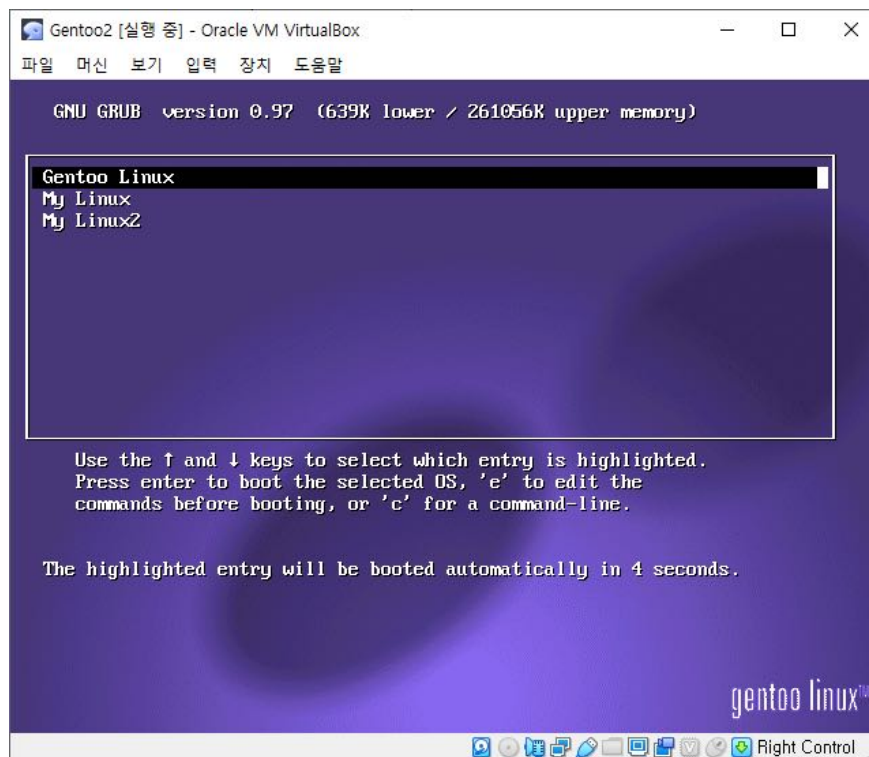
<My Linux의 부팅 메시지>



```
/*
 * Interrupts are still disabled. Do necessary setups, then
 * enable them
 */
lock_kernel();
tick_init();
boot_cpu_init();
page_address_init();
printk(KERN_NOTICE);
printk(linux_banner);
printk("hello from My Linux2\\n");
setup_arch(&command_line);
setup_command_line(command_line);
unwind_setup();
setup_per_cpu_areas();
smp_prepare_boot_cpu(); /* arch-specific boot-cpu hooks */

/*
 * Set up the scheduler prior starting any interrupts (such as the
 * timer interrupt). Full topology setup happens at smp_init()
 * time - but meanwhile we still have a functioning scheduler.
 */
sched_init();
"main.c" [converted] 859L, 20839C 536,2-9 62%
```

<'main.c'에 'printk("hello from My Linux2")'를 추가함>



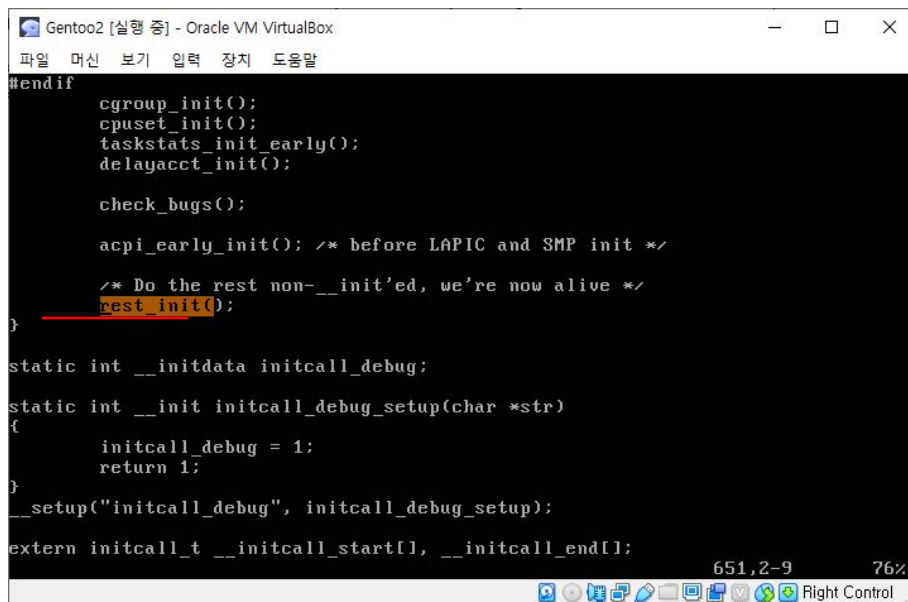
<My Linux2로 부팅,(My Linux2를 부팅하기 이전,
'cp/arch/x86/boot/bzImage /boot/bzImage2'해당 명령어를 실행함)>

```
Gentoo2 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
Linux version 2.6.25.10 (root@localhost) (gcc version 4.1.2 (Gentoo 4.1.2 p1.0.2
)) #9 SMP Mon Sep 16 12:12:15 KST 2019
hello from My Linux2
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
 BIOS-e820: 000000000009fc00 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000f0000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000000fff0000 (usable)
 BIOS-e820: 0000000000fff0000 - 0000000010000000 (ACPI data)
 BIOS-e820: 00000000fec00000 - 00000000fec01000 (reserved)
 BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
 BIOS-e820: 00000000fffc0000 - 0000000010000000 (reserved)
WARNING: strange, CPU MTRRs all blank?
-----[ cut here ]-----
WARNING: at arch/x86/kernel/cpu/mtrr/main.c:696 mtrr_trim_uncached_memory+0x16f/
0x17a()
Modules linked in:
Pid: 0, comm: swapper Not tainted 2.6.25.10 #9
[<c011d61f>] warn_on_slowpath+0x40/0x4f
[<c011ddb0>] release_console_sem+0x180/0x199
[<c011ddb0>] release_console_sem+0x180/0x199
[<c011ddb0>] release_console_sem+0x180/0x199
[<c011e205>] printk+0x14/0x18
[<c05c6ee7>] mtrr_trim_uncached_memory+0x16f/0x17a
"y" [converted] 294L, 13681C
1,1 Top
Right Control
```

<My Linux2의 부팅 메시지>

5) Where is CPU at the end of the boot sequence when it prints "login" and waits for the user login? Explain your reasoning.

답변 : 'start_kernel()'함수의 마지막 부분에는 'rest_init()'함수를 호출한다. 해당 'rest_init()'함수 선언의 마지막 부분에선 'cpu_idle()'함수를 호출한다. 즉, "login" 문자열을 호출한 부분에선 CPU가 IDLE상태가 된다.



```
#endif
cgroup_init();
cpuset_init();
taskstats_init_early();
delayacct_init();

check_bugs();

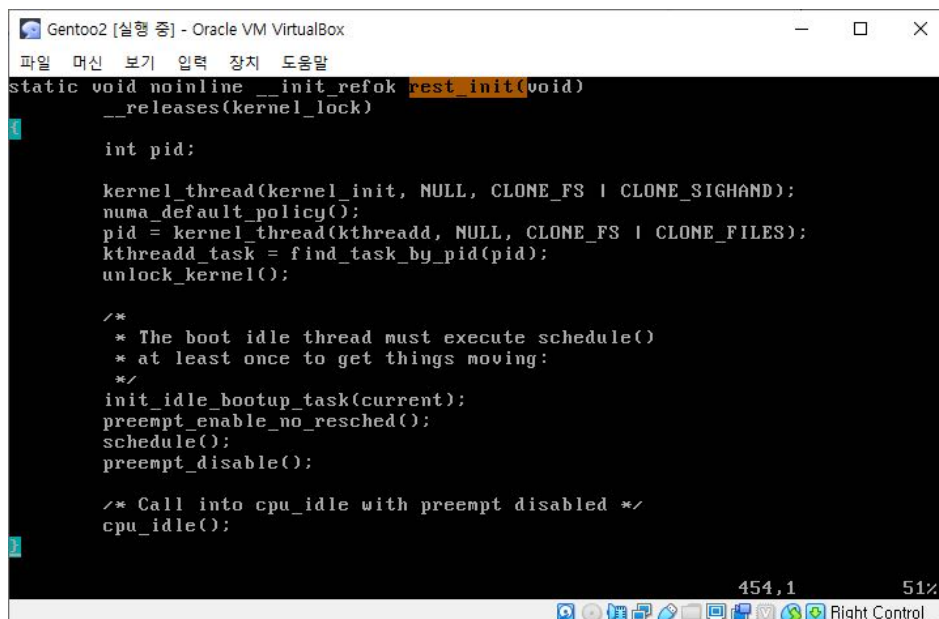
acpi_early_init(); /* before LAPIC and SMP init */

/* Do the rest non-__init'ed, we're now alive */
rest_init();
}

static int __initdata initcall_debug;
static int __init initcall_debug_setup(char *str)
{
    initcall_debug = 1;
    return 1;
}
__setup("initcall_debug", initcall_debug_setup);

extern initcall_t __initcall_start[], __initcall_end[];
```

<'start_kernel()'선언 내 'rest_init()'함수 호출>



```
static void noinline __init_refok rest_init(void)
__releases(kernel_lock)
{
    int pid;

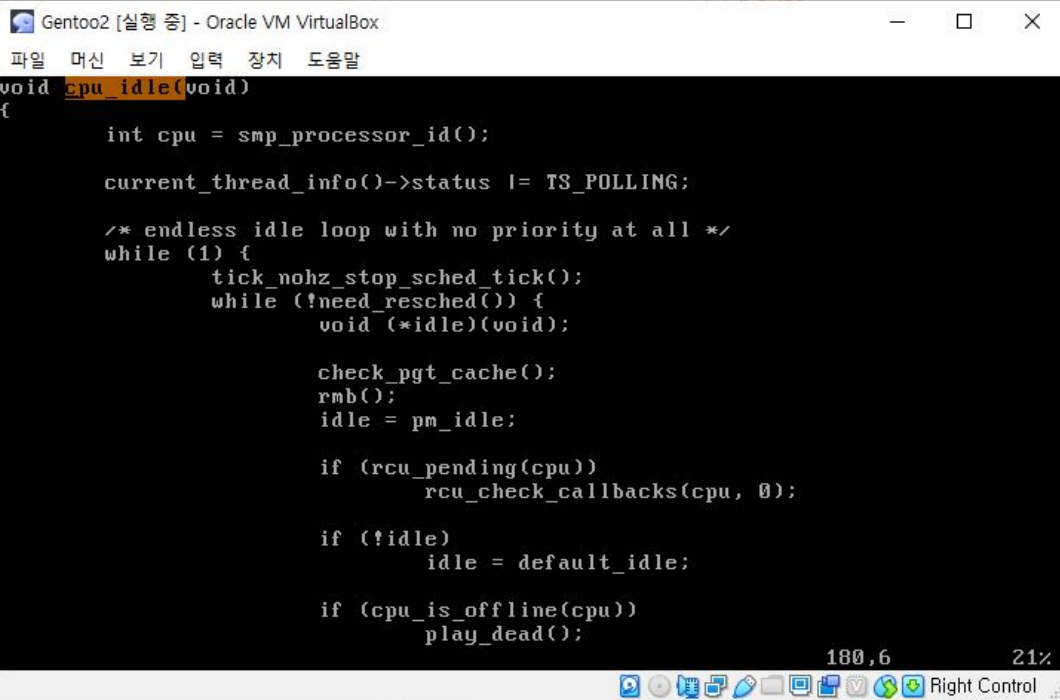
    kernel_thread(kernel_init, NULL, CLONE_FS | CLONE_SIGHAND);
    numa_default_policy();
    pid = kernel_thread(kthreadd, NULL, CLONE_FS | CLONE_FILES);
    kthreadd_task = find_task_by_pid(pid);
    unlock_kernel();

    /*
     * The boot idle thread must execute schedule()
     * at least once to get things moving:
     */
    init_idle_bootup_task(current);
    preempt_enable_no_resched();
    schedule();
    preempt_disable();

    /* Call into cpu_idle with preempt disabled */
    cpu_idle();
}
```

<'rest_init()'함수 선언부분 내 'cpu_idle()'함수 호출>

밑 사진의 'cpu_idle()'함수 선언처럼, CPU가 idle모드가 되면 'while(1)'의 무한 루프가 실행 될 것이다.



```
void cpu_idle(void)
{
    int cpu = smp_processor_id();

    current_thread_info()->status |= TS_POLLING;

    /* endless idle loop with no priority at all */
    while (1) {
        tick_nohz_stop_sched_tick();
        while (!need_resched()) {
            void (*idle)(void);

            check_pgt_cache();
            rmb();
            idle = pm_idle;

            if (rcu_pending(cpu))
                rcu_check_callbacks(cpu, 0);

            if (!idle)
                idle = default_idle;

            if (cpu_is_offline(cpu))
                play_dead();
        }
    }
}
```

180.6 21%

Right Control