

0) What are the interrupt numbers for divide-by-zero exception, keyboard interrupt, and "read" system call?

divide by-zero's exception : 0

keyboard : 33

read system call : 128, system call number : 3

1) Following events will cause interrupts in the system. What interrupt number will be assigned to each event? For system call interrupt, also give the system call number.

- A packet has arrived
- An application program calls scanf()
- A key is pressed
- An application causes a divide-by-zero error
- An application program calls printf()
- An application causes a page-fault error
- A user tries to remove a file

A packet has arrived : 42

An application program calls scanf() : 128, 3

A key is pressed : 33

divide by-zero's exception : 0, system call number : 4

An application causes a page-fault error : 14

A user tries to remove a : 33/128, system call number : 3

2) Change drivers/input/keyboard/atkbd.c as follows.

```
static irqreturn_t atkbd_interrupt(...){
    return IRQ_HANDLED; // Add this at the first line
    .....
}
```

Recompile the kernel and reboot with it. What happens and why does this happen? Show the sequence of events that happen when you hit a key in a normal Linux kernel (as detail as possible): hit a key => keyboard controller sends a signal through IRQ line 1 =>etc. Now with the changed Linux kernel show which step in this sequence has been modified and prevents the kernel to display the pressed key in the monitor.

```
        return code;
}

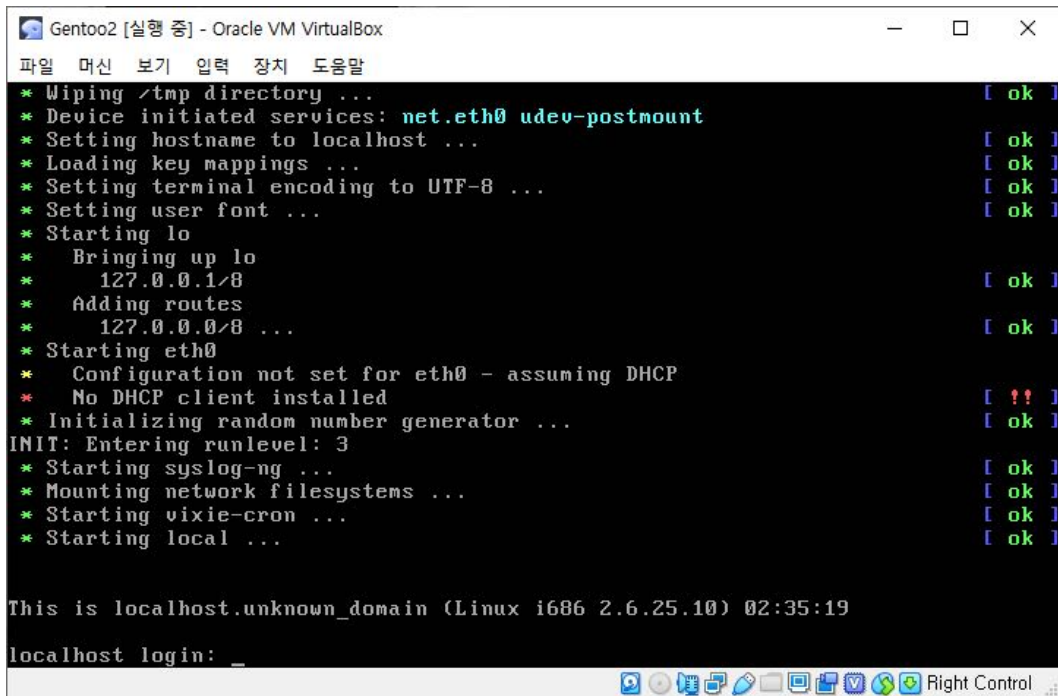
/*
 * atkbd_interrupt(). Here takes place processing of data received from
 * the keyboard into events.
 */

static irqreturn_t atkbd_interrupt(struct serio *serio, unsigned char data,
                                   unsigned int flags)
{
    return IRQ_HANDLED;
    struct atkbd *atkbd = serio_get_drvdata(serio);
    struct input_dev *dev = atkbd->dev;
    unsigned int code = data;
    int scroll = 0, hscroll = 0, click = -1;
    int value;
    unsigned char keycode;

#ifdef ATKBD_DEBUG
    printk(KERN_DEBUG "atkbd.c: Received %02x flags %02x\n", data, flags);
#endif

#ifdef __i386__ && !defined (__x86_64__)
    "atkbd.c" [converted] 1470L, 36794C                                     355,20-27      23%
```

<‘atkbd_interrupt()’에 ‘return IRQ_HANDLED’문을 추가하였다.>



```
* Wiping /tmp directory ... [ ok ]
* Device initiated services: net.eth0 udev-postmount
* Setting hostname to localhost ... [ ok ]
* Loading key mappings ... [ ok ]
* Setting terminal encoding to UTF-8 ... [ ok ]
* Setting user font ... [ ok ]
* Starting lo
*   Bringing up lo
*     127.0.0.1/8 [ ok ]
*   Adding routes
*     127.0.0.0/8 ... [ ok ]
* Starting eth0
*   Configuration not set for eth0 - assuming DHCP
*   No DHCP client installed [ !! ]
*   Initializing random number generator ... [ ok ]
INIT: Entering runlevel: 3
* Starting syslog-ng ... [ ok ]
* Mounting network filesystems ... [ ok ]
* Starting vixie-cron ... [ ok ]
* Starting local ... [ ok ]

This is localhost.unknown_domain (Linux i686 2.6.25.10) 02:35:19
localhost login: _
```

<리컴파일을 'My Linux'로 재부팅을 하여 로그인을 시도하였지만, 키를 눌렀음에도 아무런 문자가 출력되지 않는다.>

키보드에 키들을 누르면, 하드웨어 인터럽트가 발생한다.(interrupt number : 33)해당 인터럽트가 발생되면, 이 인터럽트를 처리하는 'atkbd_interrupt()'함수가 호출된다. 즉, 키를 누를 때 해당 함수가 호출되어 우리가 누르는 키들이 화면에 출력되는 것이다. 또한, 키를 누를 때뿐만 아니라, 키를 누르고 때는 순간에서도 해당 함수가 호출된다.

그러므로 해당 함수 선언 첫 부분에 return 문을 선언해버리면, 해당 문자를 출력시켜주는 나머지 코드들이 실행되지 않기 때문에, 로그인 화면에서 아무런 문자가 출력되지 않는다.

3) Change the kernel such that it prints "x pressed" for each key pressing, where x is the scan code of the key. After you change the kernel and reboot it, do followings to see the effect of your changing.

```
# echo 8 > /proc/sys/kernel/printk
```

/proc/sys/kernel/printk shows the console log level, default log level, min and max log level.

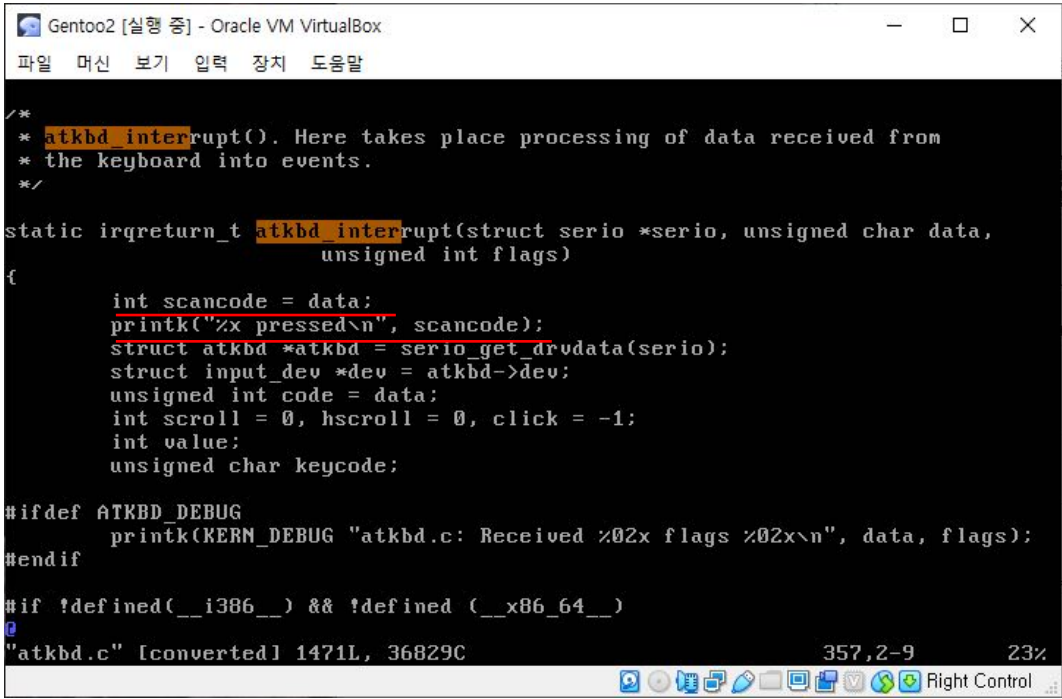
```
# cat /proc/sys/kernel/printk
```

```
1 4 1 7
```

The above means the console log level is 1, default printk log level is 4, and min console log level is 1 and default console log level is 7. Lower log level means higher priority. Since default log level has lower priority than console log level, using printk() will not show the message on the console. We change the console log level to lowest level so that printk() will be able to display message on the console.

```
# echo 8 > /proc/sys/kernel/printk
```

Above will set console log level to 8 which means all printk() message will appear on the console from now on. (Note the files in /proc file system are not real files. They are generated dynamically when needed.)



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

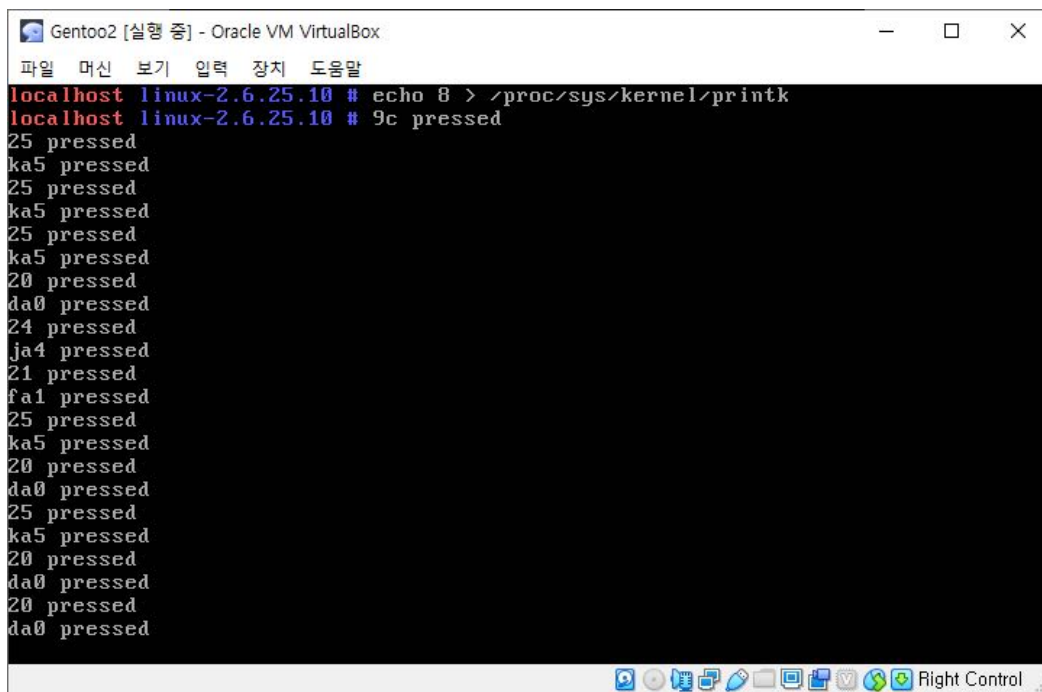
/*
 * atkbd_interrupt(). Here takes place processing of data received from
 * the keyboard into events.
 */

static irqreturn_t atkbd_interrupt(struct serio *serio, unsigned char data,
                                   unsigned int flags)
{
    int scancode = data;
    printk("%x pressed\n", scancode);
    struct atkbd *atkbd = serio_get_drvdata(serio);
    struct input_dev *dev = atkbd->dev;
    unsigned int code = data;
    int scroll = 0, hscroll = 0, click = -1;
    int value;
    unsigned char keycode;

#ifdef ATKBD_DEBUG
    printk(KERN_DEBUG "atkbd.c: Received %02x flags %02x\n", data, flags);
#endif

#if !defined(__i386__) && !defined(__x86_64__)
0
"atkbd.c" [converted] 1471L, 36829C                                     357,2-9      23%
Right Control
```

<atkbd_interrupt()함수 내에 키를 누를 때마다 scancode를 출력하는 코드를 추가함>



The screenshot shows a terminal window titled "Gentoo2 [실행 중] - Oracle VM VirtualBox". The terminal output is as follows:

```
localhost linux-2.6.25.10 # echo 8 > /proc/sys/kernel/printk
localhost linux-2.6.25.10 # 9c pressed
25 pressed
ka5 pressed
25 pressed
ka5 pressed
25 pressed
ka5 pressed
20 pressed
da0 pressed
24 pressed
ja4 pressed
21 pressed
fa1 pressed
25 pressed
ka5 pressed
20 pressed
da0 pressed
25 pressed
ka5 pressed
20 pressed
da0 pressed
20 pressed
da0 pressed
```

The terminal window has a menu bar with "파일", "머신", "보기", "입력", "장치", and "도움말". At the bottom, there is a taskbar with various icons and a "Right Control" button.

<printk로그 레벨을 변경한 후, 키를 누를 때마다 해당 scancode가 출력됨>

4) Change the kernel such that it displays the next character in the keyboard scancode table. For example, when you type "root", the monitor would display "tppy". How can you log in as root with this kernel?

```

static irqreturn_t atkbd_interrupt(struct serio *serio, unsigned char data,
                                unsigned int flags)
{
    struct atkbd *atkbd = serio_get_drvdata(serio);
    struct input_dev *dev = atkbd->dev;
    unsigned int code = data+1;
    int scroll = 0, hscroll = 0, click = -1;
    int value;
    unsigned char keycode;

#ifdef ATKBD_DEBUG
    printk(KERN_DEBUG "atkbd.c: Received %02x flags %02x\n", data, flags);
#endif

    if (!defined(__i386__) && !defined(__x86_64__))
        if ((flags & (SERIO_FRAME | SERIO_PARITY)) && (~flags & SERIO_TIMEOUT) &
            & !atkbd->resend && atkbd->write) {
        @
    }
}

```

<unsigned int 자료형인 'code' 변수를 'data+1'로 수정함>

```

* Wiping /tmp directory ... [ ok ]
* Setting hostname to localhost ... [ ok ]
* Loading key mappings ... [ ok ]
* Setting terminal encoding to UTF-8 ... [ ok ]
* Setting user font ... [ ok ]
* Starting lo
*   Bringing up lo
*   127.0.0.1/8 [ ok ]
*   Adding routes
*   127.0.0.0/8 ... [ ok ]
* Starting eth0
*   Configuration not set for eth0 - assuming DHCP
*   No DHCP client installed [ !? ]
* Initializing random number generator ... [ ok ]
INIT: Entering runlevel: 3
* Starting syslog-ng ... [ ok ]
* Mounting network filesystems ... [ ok ]
* Starting vixie-cron ... [ ok ]
* Starting local ... [ ok ]

This is localhost.unknown_domain (Linux i686 2.6.25.10) 07:03:45

localhost login: tppy

```

<'root'를 타이핑하니 화면에 'tppy'가 출력됨,
'root'문자열을 화면에 출력하려면 'eiir'를 타이핑해야함.>

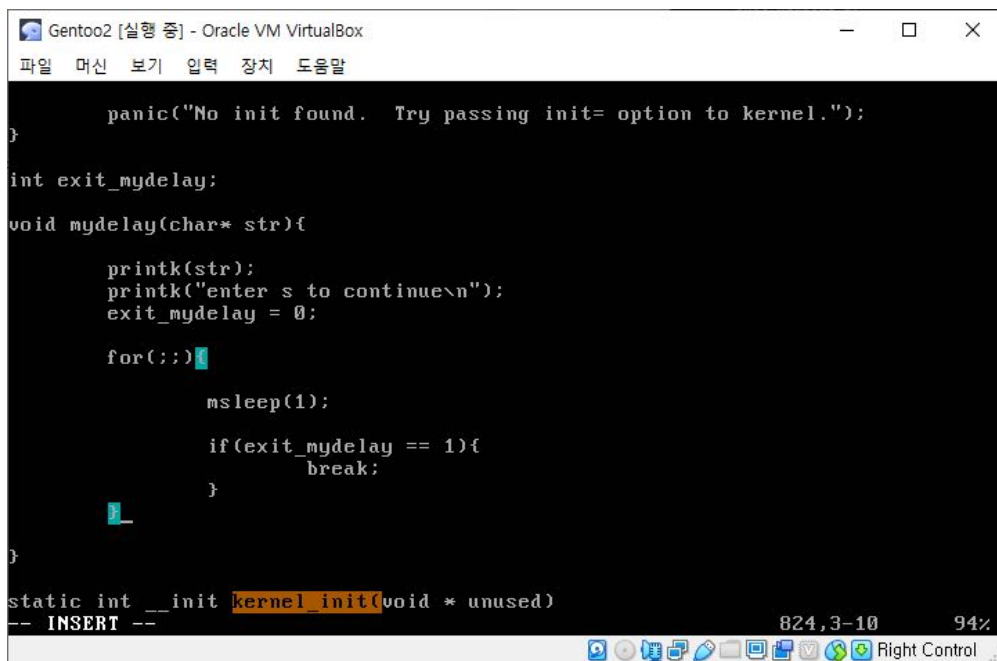
5) Define a function "mydelay" in init/main.c which whenever called will stop the booting process until you hit 's'. Call this function after do_basic_setup() function call in kernel_init() in order to make the kernel stop and wait for 's' during the booting process. You need to modify atkbd.c such that it changes exit_mydelay to 1 when the user presses 's'.

init/main.c

```
.....
int exit_mydelay;    // define a global variable
void mydelay(char *str){
    printk(str);
    printk("enter s to continue\n");
    exit_mydelay=0; // init to zero
    for(;;){ // and wait here until the user press 's'
        msleep(1); // sleep 1 micro-second so that keyboard interrupt ISR
                    // can do its job
        if (exit_mydelay==1) break; // if the user press 's', break
    }
}
void kernel_init(){
    .....
    do_basic_setup();
    mydelay("after do basic setup in kernel_init\n"); // wait here
    .....
}
```

drivers/input/keyboard/atkbd.c

```
.....
extern int exit_mydelay; // declare as extern since it is defined in main.c
static irqreturn_t atkbd_interrupt(...){
    .....
    // detect 's' key pressed and change exit_mydelay
    .....
}
```



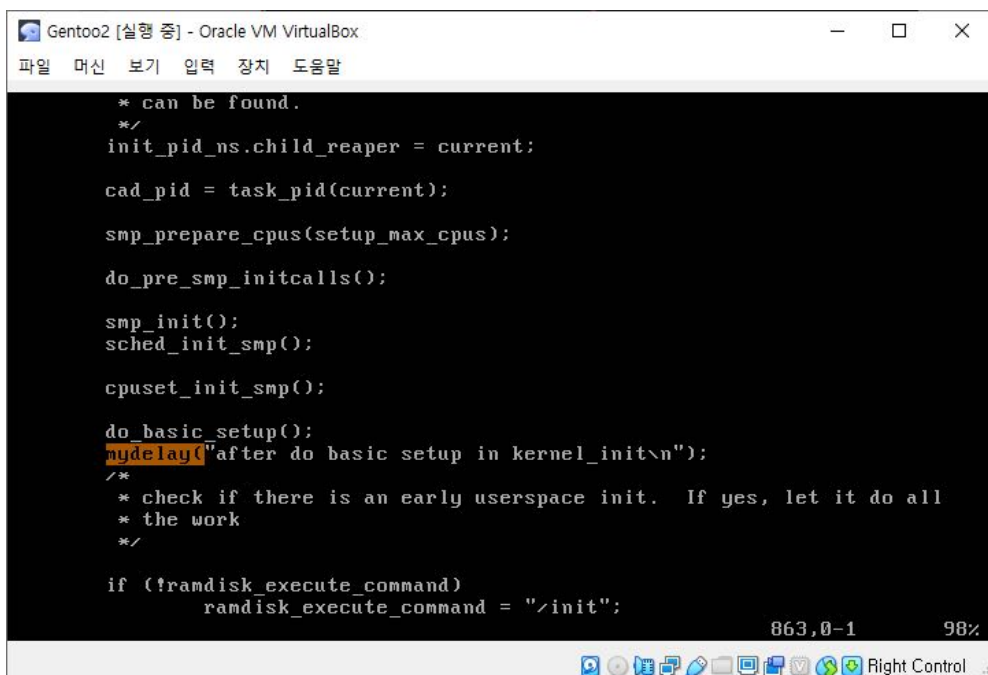
```
panic("No init found. Try passing init= option to kernel.");
}
int exit_mydelay;
void mydelay(char* str){
    printk(str);
    printk("enter s to continue\n");
    exit_mydelay = 0;

    for(;;){
        msleep(1);

        if(exit_mydelay == 1){
            break;
        }
    }
}

static int __init kernel_init(void * unused)
-- INSERT --
```

<'kernel_init()'함수 전 'mydelay()'함수를 정의함>



```
* can be found.
*/
init_pid_ns.child_reaper = current;

cad_pid = task_pid(current);

smp_prepare_cpus(setup_max_cpus);

do_pre_smp_initcalls();

smp_init();
sched_init_smp();

cpuset_init_smp();

do_basic_setup();
mydelay("after do basic setup in kernel_init\n");
/*
 * check if there is an early userspace init. If yes, let it do all
 * the work
 */

if (!randisk_execute_command)
    randisk_execute_command = "/init";
```

<'do_basic_setup()'함수를 호출한 부분 뒤에 'mydelay()'함수를 호출함>


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

*/
* atkbd_interrupt(). Here takes place processing of data received from
* the keyboard into events.
*/

extern int exit_mydelay;

static irqreturn_t atkbd_interrupt(struct serio *serio, unsigned char data,
                                unsigned int flags)
{
    struct atkbd *atkbd = serio_get_drvdata(serio);
    struct input_dev *dev = atkbd->dev;
    unsigned int code = data;

    if(code == 0x1f){
        exit_mydelay = 1;
    }

    int scroll = 0, hscroll = 0, click = -1;
    int value;
    unsigned char keycode;

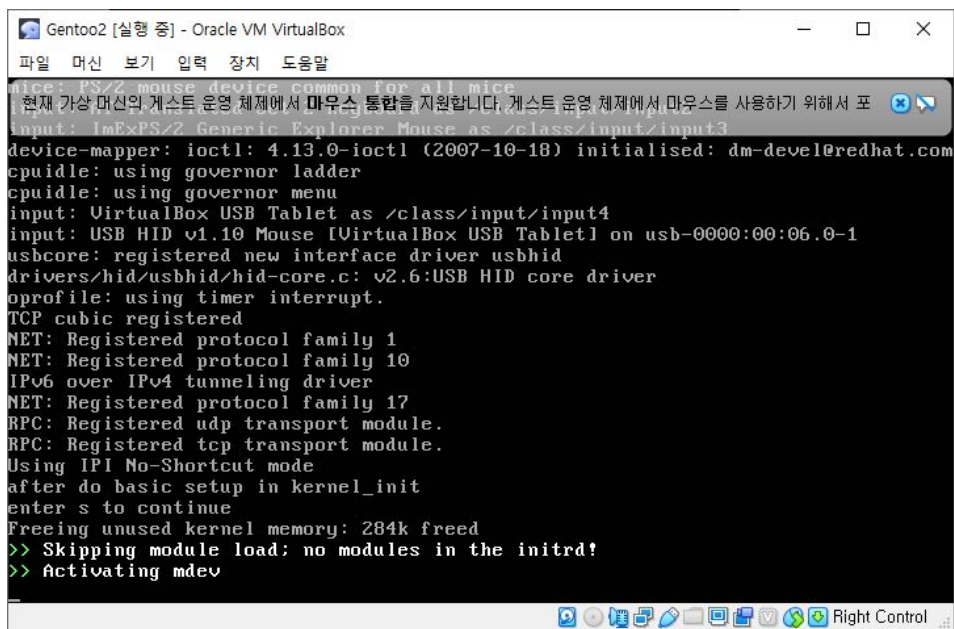
#ifdef ATKBD_DEBUG
-- INSERT --
363,3-17 23%
```

<main.c에서 정의된 exit_mydelay 변수를 사용하기 위해 extern으로
exit_mydelay 변수를 선언하였고, 'code'가 '0x1f'('s'의 스캔코드)라면,
exit_mydelay의 값이 1로 변경되도록
수정함>

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

현재 가상 머신의 게스트 운영 체제에서 마우스 통합을 지원합니다. 게스트 운영 체제에서 마우스를 사용하기 위해서 포
mice: PS/2 mouse device common for all mice
input: AT Translated Set 2 keyboard as /class/input/input2
input: ImExPS/2 Generic Explorer Mouse as /class/input/input3
device-mapper: ioctl: 4.13.0-ioctl (2007-10-18) initialised: dm-devel@redhat.co
cpuidle: using governor ladder
cpuidle: using governor menu
input: VirtualBox USB Tablet as /class/input/input4
input: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-1
usbcore: registered new interface driver usbhid
drivers/hid/usbhid/hid-core.c: v2.6:USB HID core driver
oprofile: using timer interrupt.
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 10
IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
Using IPI No-Shortcut mode
after do basic setup in kernel_init
enter s to continue
```

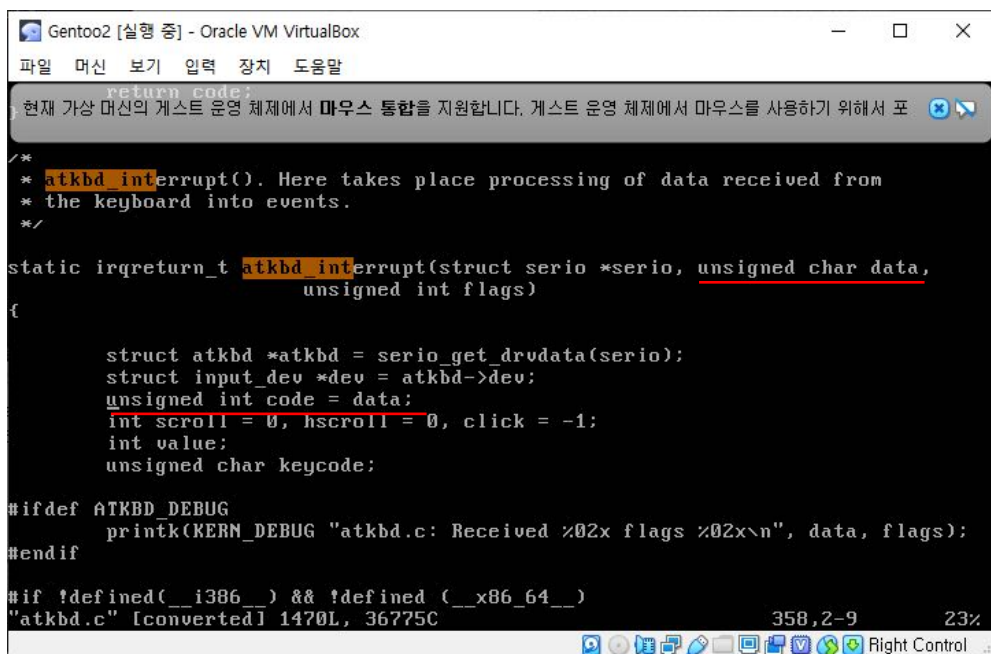
<부팅도중 's'를 누르라는 문장이 출력됨>



```
mice: PS/2 mouse device common for all mice
현재 가상 머신의 게스트 운영 체제에서 마우스 통합을 지원합니다. 게스트 운영 체제에서 마우스를 사용하기 위해서 포
input: ImExPS/2 Generic Explorer Mouse as /class/input/input3
device-mapper: ioctl: 4.13.0-ioctl (2007-10-18) initialised: dm-devel@redhat.com
cpuidle: using governor ladder
cpuidle: using governor menu
input: VirtualBox USB Tablet as /class/input/input4
input: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-1
usbcore: registered new interface driver usbhid
drivers/hid/usbhid/hid-core.c: v2.6:USB HID core driver
oprofile: using timer interrupt.
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 10
IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
Using IPI No-Shortcut mode
after do basic setup in kernel_init
enter s to continue
Freeing unused kernel memory: 284k freed
>> Skipping module load: no modules in the initrd!
>> Activating mdev
```

<'s'를 누른 후, 부팅이 다시 진행됨>

6) Which function call in `atkbd_interrupt()` actually displays the pressed key in the monitor?



```
return code;
현재 가상 머신의 게스트 운영 체제에서 마우스 통합을 지원합니다. 게스트 운영 체제에서 마우스를 사용하기 위해서 포

/*
 * atkbd_interrupt(). Here takes place processing of data received from
 * the keyboard into events.
 */

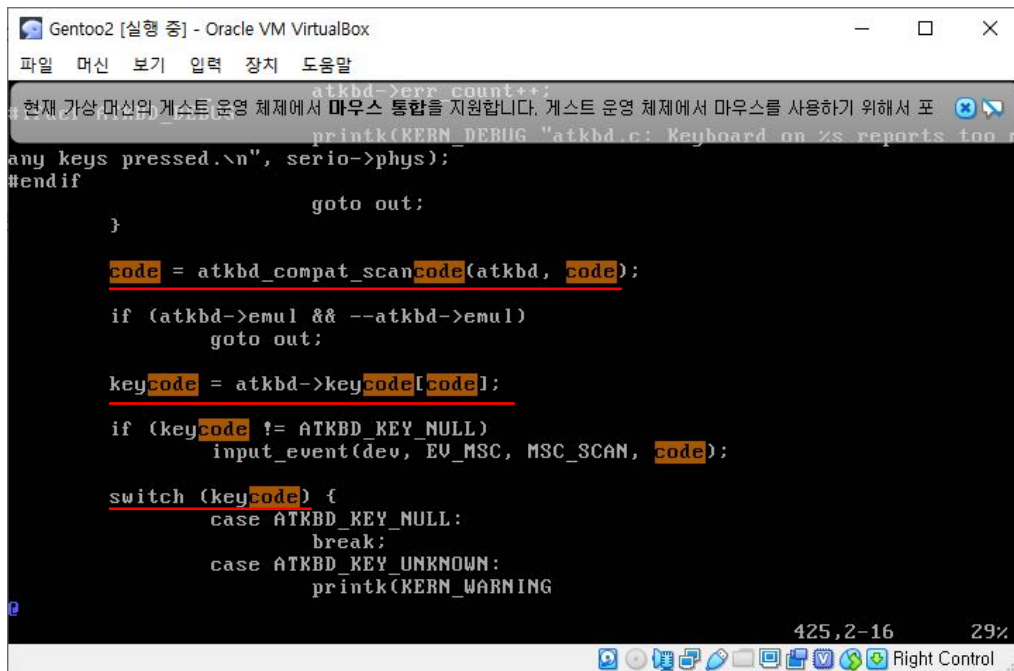
static irqreturn_t atkbd_interrupt(struct serio *serio, unsigned char data,
                                   unsigned int flags)
{
    struct atkbd *atkbd = serio_get_drvdata(serio);
    struct input_dev *dev = atkbd->dev;
    unsigned int code = data;
    int scroll = 0, hscroll = 0, click = -1;
    int value;
    unsigned char keycode;

#ifdef ATKBD_DEBUG
    printk(KERN_DEBUG "atkbd.c: Received %02x flags %02x\n", data, flags);
#endif

#if !defined(__i386__) && !defined (__x86_64__)
"atkbd.c" [converted] 1470L, 36775C
358,2-9 23%
```

<atkbd_interrupt()함수선언 초기 부분>

해당 함수의 'data' 매개변수에 char 자료형이 들어온다. 해당 변수가 char 자료형인 것을 토대로, 우리가 키보드로 입력하는 키 값들이 해당 'data' 매개변수에 들어가는 것을 예측할 수 있다. 그리고 해당 data 매개변수의 인자는 'code'라는 int 자료형 변수에 저장된다. 해당 code 변수에는 우리가 입력하는 키의 스캔코드가 저장되는 것처럼 보인다.



```
atkbd->err_count++;
현재 가상 머신의 게스트 운영 체제에서 마우스 통합을 지원하지 않습니다. 게스트 운영 체제에서 마우스를 사용하기 위해서 포
printf(KERN_DEBUG "atkbd.c: Keyboard on %s reports too m
any keys pressed.\n", serio->phys);
#endif
        goto out;
    }

    code = atkbd_compat_scancode(atkbd, code);

    if (atkbd->emul && --atkbd->emul)
        goto out;

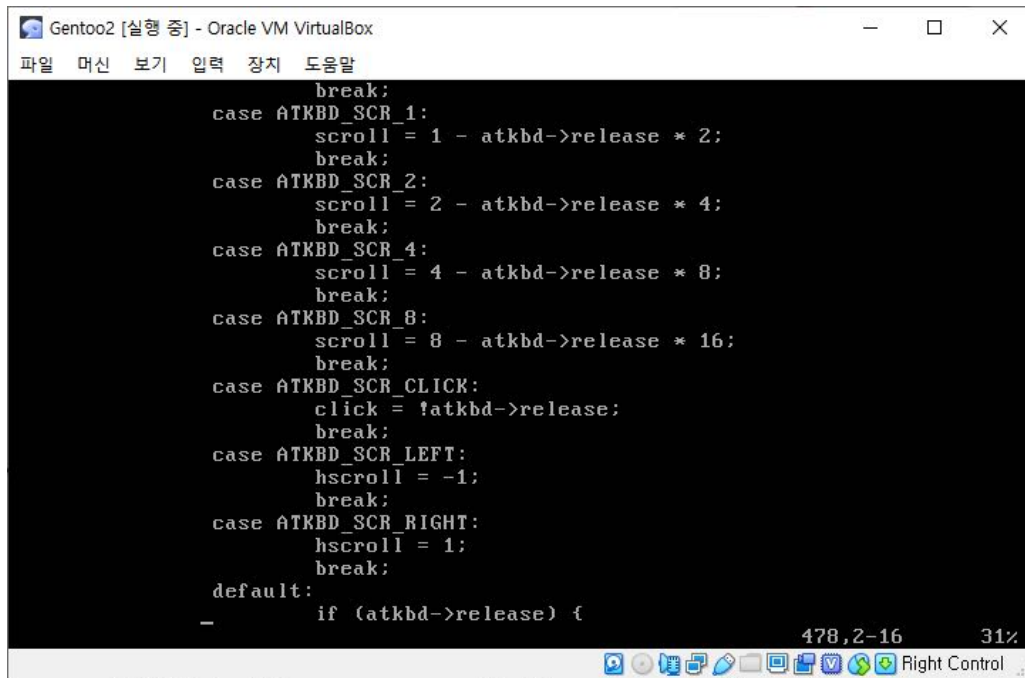
    keycode = atkbd->keycode[code];

    if (keycode != ATKBD_KEY_NULL)
        input_event(dev, EV_MSC, MSC_SCAN, code);

    switch (keycode) {
        case ATKBD_KEY_NULL:
            break;
        case ATKBD_KEY_UNKNOWN:
            printk(KERN_WARNING
425,2-16 29%
```

<code 변수 업데이트, keycode 변수 업데이트>

code 변수가 'atkbd_compat_scancode()' 함수의 return 값으로 업데이트 되고, 해당 code 변수의 값을 활용하여 생성된 값이 'keycode' 변수에 저장된다. 이후, keycode 변수가 인자로 활용되는 switch문이 등장한다.

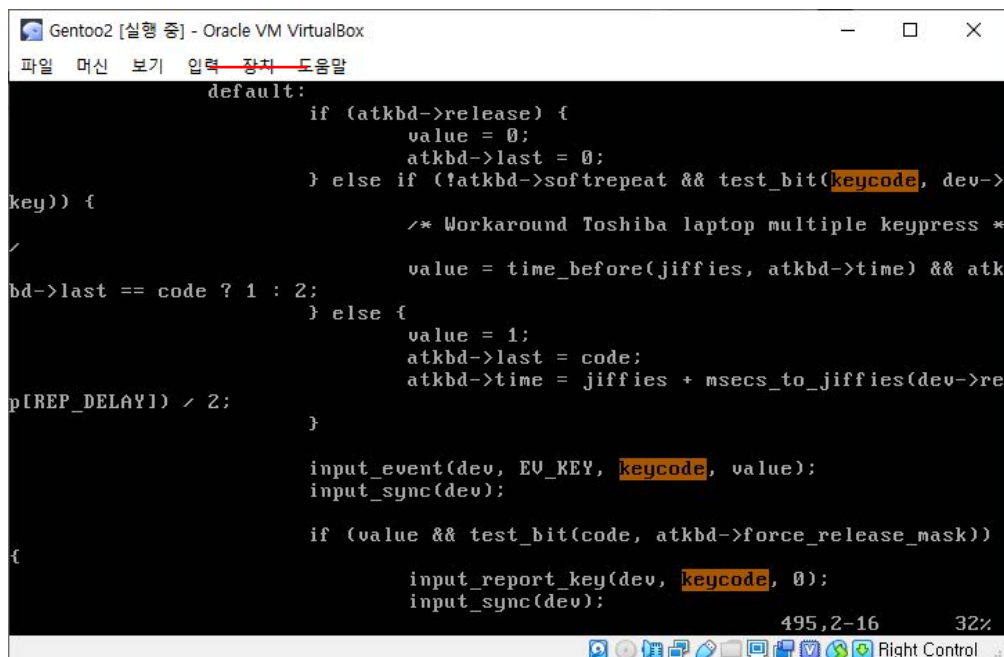


```
break;
case ATKBD_SCR_1:
    scroll = 1 - atkbd->release * 2;
    break;
case ATKBD_SCR_2:
    scroll = 2 - atkbd->release * 4;
    break;
case ATKBD_SCR_4:
    scroll = 4 - atkbd->release * 8;
    break;
case ATKBD_SCR_8:
    scroll = 8 - atkbd->release * 16;
    break;
case ATKBD_SCR_CLICK:
    click = !atkbd->release;
    break;
case ATKBD_SCR_LEFT:
    hscroll = -1;
    break;
case ATKBD_SCR_RIGHT:
    hscroll = 1;
    break;
default:
    if (atkbd->release) {

```

<Switch문 내 case 선언 부분>

일반적이지 않은 키눌림 현상이 있을 때는 case문에 선언된 것들이 실행되고 일반적인 키눌림 현상이 있을 때는 default문에 선언된 것들이 실행되어, 모니터에 우리가 입력하는 키들이 출력되는 것처럼 보인다.



```
default:
    if (atkbd->release) {
        value = 0;
        atkbd->last = 0;
    } else if (!atkbd->softrepeat && test_bit(keycode, dev->
key)) {
        /* Workaround Toshiba laptop multiple keypress */
        value = time_before(jiffies, atkbd->time) && atk
bd->last == code ? 1 : 2;
    } else {
        value = 1;
        atkbd->last = code;
        atkbd->time = jiffies + msecs_to_jiffies(dev->re
p[REP_DELAY]) / 2;
    }

    input_event(dev, EV_KEY, keycode, value);
    input_sync(dev);

    if (value && test_bit(code, atkbd->force_release_mask))
    {
        input_report_key(dev, keycode, 0);
        input_sync(dev);

```

<Switch문 내 default 선언 부분>

