Homework

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

Divide-by-zero exception : 0
keyboard interrupt : 33
read : 128, 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
42

- An application program calls scanf()
128, 3

- A key is pressed
33

- An application causes a divide-by-zero error
0

- An application program calls printf()
128, 4

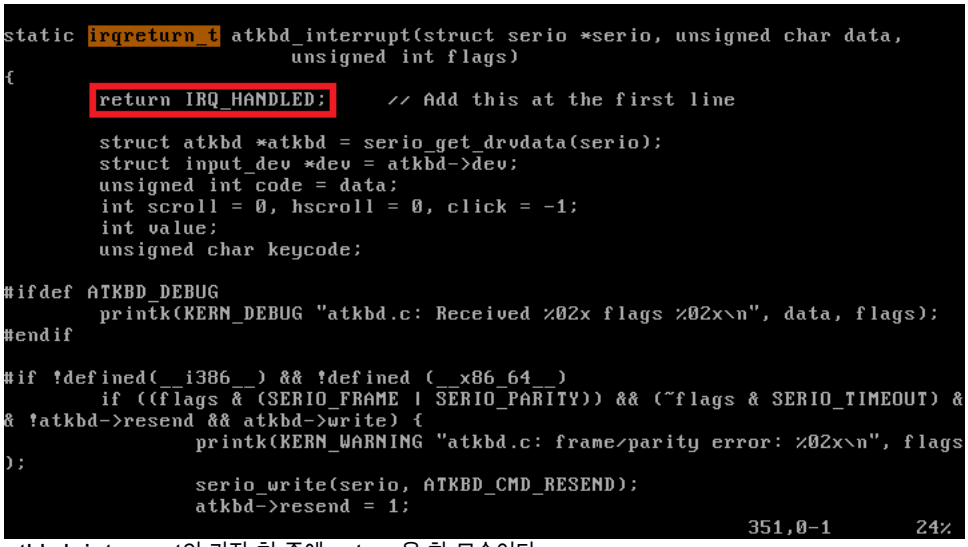
- An application causes a page-fault error
14

- A user tries to remove a file
33
128, 3

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

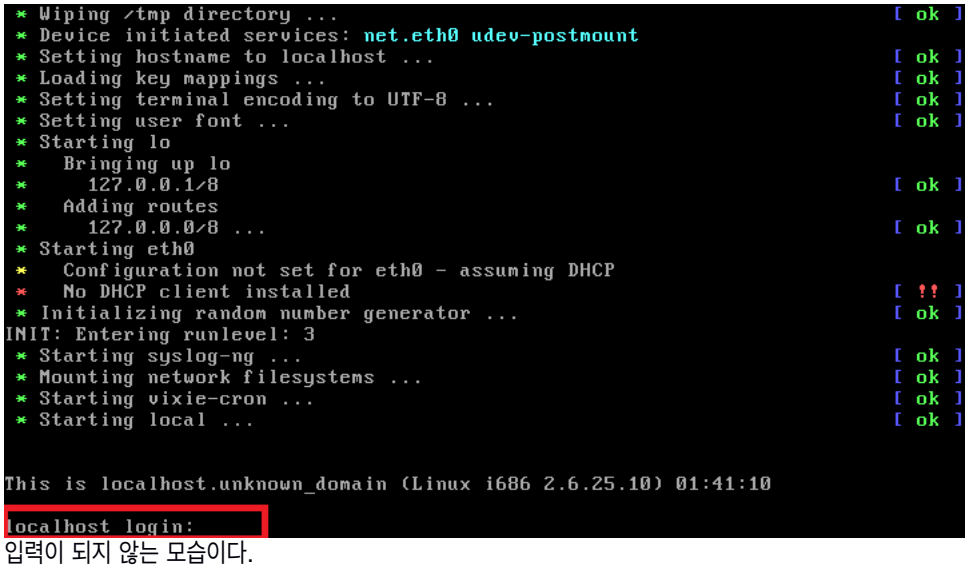
```
static irqreturn_t atkbd_interrupt(....){
   return IRQ_HANDLED;  // Add this at the first line
   .............
}
```

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.



```
static irqreturn_t atkbd_interrupt(struct serio *serio, unsigned char data,
                                   unsigned int flags)
{
        return IRQ_HANDLED;     // Add this at the first line

        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__)
        if ((flags & (SERIO_FRAME | SERIO_PARITY)) && (~flags & SERIO_TIMEOUT) &
& !atkbd->resend && atkbd->write) {
                printk(KERN_WARNING "atkbd.c: frame/parity error: %02x\n", flags
);
                serio_write(serio, ATKBD_CMD_RESEND);
                atkbd->resend = 1;
                                                              351,0-1       24%
```

atkbd_interrupt의 가장 첫 줄에 return을 한 모습이다.



```
 * 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) 01:41:10

localhost login:
```

입력이 되지 않는 모습이다.
원래는 key가 입력이 되면 key stroke interrupt인 33이 발생한다.
IRQ는 1번이므로 해당 함수를 호출한다. 여기서 함수는 atkbd_interrupt() 이므로 이 함수가 호출된다. 여기에서 key input에 해당하는 값을 출력하고 다시 돌아가는 역할이다. 여기서는 keyboard 입력으로 interrupt가 발생하여 atkbd_interrupt가 호출되었지만 바로 return을 하기 때문에 아무 것도 하지 못하고 끝나므로, 아무 일도 일어나지 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 conole 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.)

```
/*
 * 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__)
                                                              357,0-1       23%
```

입력된 char인 data를 출력하는 것을 추가하였다.

```
localhost linux-2.6.25.10 # cat /proc/sys/kernel/printk
1       4       1       7
localhost linux-2.6.25.10 # echo 8 > /proc/sys/kernel/printk
localhost linux-2.6.25.10 # cat /proc/sys/kernel/printk
8       4       1       7
localhost linux-2.6.25.10 #
```

우선순위를 바꿔준 모습이다. 우선순위를 바꿔주지 않으면 실제 화면에 출력이 되지 않는다.

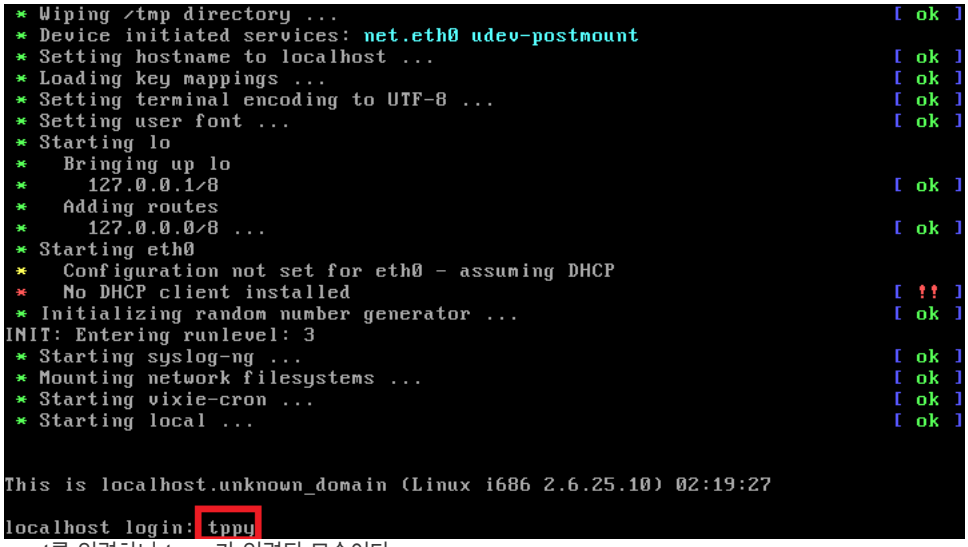
```
ca0 pressed
12 pressed
eae pressed
11 pressed
w21 pressed
f91 pressed
a1 pressed
92 pressed
2e pressed
c1e pressed
a1f pressed
sae pressed
9e pressed
9f pressed
1f pressed
s9f pressed
38 pressed
38 pressed
38 pressed
b8 pressed
38 pressed
54 pressed
d4 pressed
b8 pressed
```

key를 누를 때 마다 출력이 되는 모습이다. 누를 때 뿐만 아니라 땔 때도 또한 출력이 된다.

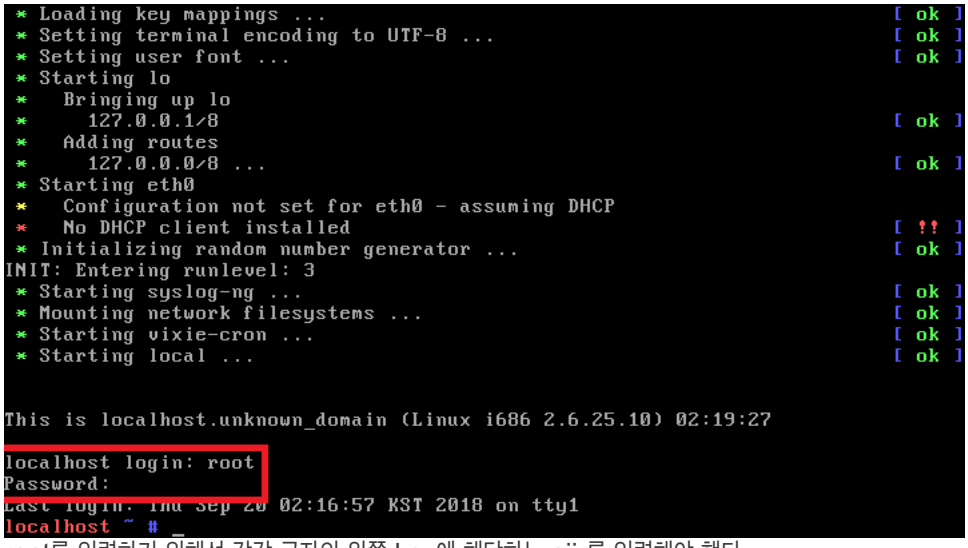
4) Change the kernel such that it displays the next character in the keyboard scancode table. For example, when you type "root", the monitor would display "tppy". How can you log in as root with this kernel?

```
 * 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 + 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) {
                printk(KERN_WARNING "atkbd.c: frame/parity error: %02x\n", flags
);
                serio_write(serio, ATKBD_CMD_RESEND);
                                                              357,29-36     23%
```

code를 저장하는 과정에서 1을 더하여 저장하게 하였다.



root를 입력하니 tppy가 입력된 모습이다.



root를 입력하기 위해선 각각 글자의 왼쪽 key에 해당하는 eiir를 입력해야 했다.
그리고 return 또한 return의 왼쪽 key에 해당하는]를 누르니 login이 정상적으로 되었다.

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);

                // do things

                if(exit_mydelay==1)
                        break;
        }
}

static int __init kernel_init(void * unused)
{
        lock_kernel();
        /*
         * init can run on any cpu.
         */
                                                              806,6-13      94%
```

init/main.c의 kernel_init 앞에 강의노트에 있던 mydelay 변수와 함수를 추가한 모습이다. 해당 함수는 exit_mydelay라는 변수가 1이 되지않으면 무한 loop을 돌게 되고, 1로 바뀌면 break 되는 함수이다.

```
        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 (!ramdisk_execute_command)
                ramdisk_execute_command = "/init";

        if (sys_access((const char __user *) ramdisk_execute_command, 0) != 0) {
                ramdisk_execute_command = NULL;
                prepare_namespace();
        }
                                                              843,1-8       98%
```

그리고 kernel_init함수에서 do_basic_setup 이후에 방금 만든 mydelay 함수를 호출하는 모습이다.

```
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
        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) {
                                                              362,19-33     24%
```

drivers/input/keyboard/atkbd.c에서 atkbd_interrupt함수 부분이다. 우선 다른 곳에서 선언된 exit_mydely를 사용하기 위해 extern 으로 선언해주었다. 그리고 ‘s’ key의 scancode인 0x1f인 경우 exit_mydelay의 값을 1로 변경하게 코드를 추가한 모습이다.

```
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
PNP: PS/2 Controller [PNP0303:PS2K,PNP0f03:PS2M] at 0x60,0x64 irq 1,12
serio: i8042 KBD port at 0x60,0x64 irq 1
serio: i8042 AUX port at 0x60,0x64 irq 12
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.com
cpuidle: using governor ladder
cpuidle: using governor menu
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
```

부팅 시 정상적으로 멈춘 모습이다. s 키를 누르기 전까지는 더 이상 진행이 되지 않는다.

```
PNP: PS/2 Controller [PNP0303:PS2K,PNP0f03:PS2M] at 0x60,0x64 irq 1,12
serio: i8042 KBD port at 0x60,0x64 irq 1
serio: i8042 AUX port at 0x60,0x64 irq 12
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.com
cpuidle: using governor ladder
cpuidle: using governor menu
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!
```

s key를 누르니 정상적으로 booting이 진행되는 모습이다.

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

```
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__)
        if ((flags & (SERIO_FRAME | SERIO_PARITY)) && (~flags & SERIO_TIMEOUT) &
& !atkbd->resend && atkbd->write) {
                printk(KERN_WARNING "atkbd.c: frame/parity error: %02x\n", flags
);
                serio_write(serio, ATKBD_CMD_RESEND);
                atkbd->resend = 1;
                goto out;
        }
                                                              360,11-18     24%
```

어디서 입력된 char가 출력되는지 알아보기 위해 순차적으로 따라가 보았다.
우선 입력된 scan code가 들어있던 data가 인수로 들어온다.
그리고 그 data가 code로 들어가게 된다.

```
                        goto out;
                case ATKBD_RET_ERR:
                        atkbd->err_count++;
#ifdef ATKBD_DEBUG
                        printk(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:
                                                                  432,1-8       29%
```

code는 atkbd_compat_scancode 함수를 통하여 다시 code로 저장되었다.
keycode라는 무언가 또 저장이 된 모습이다.

```
        switch (keycode) {
                case ATKBD_KEY_NULL:
                        break;
                case ATKBD_KEY_UNKNOWN:
                        printk(KERN_WARNING
                                "atkbd.c: Unknown key %s (%s set %d, code %#x on
%s).\n",
                                atkbd->release ? "released" : "pressed",
                                atkbd->translated ? "translated" : "raw",
                                atkbd->set, code, serio->phys);
                        printk(KERN_WARNING
                                "atkbd.c: Use 'setkeycodes %s%02x <keycode>' to m
ake it known.\n",
                                code & 0x80 ? "e0" : "", code & 0x7f);
                        input_sync(dev);
                        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:
                                                                  441,0-1       30%
```

keycode를 인자로 둔 switch문을 발견하였다.
case를 보니 KEY_UNKNOWN을 보게 되었다. 무언가 입력이 되었을 경우를 나누어 놓은 것 같다.

```
                        break;
                case ATKBD_SCR_CLICK:
                        click = !atkbd->release;
                        break;
                case ATKBD_SCR_LEFT:
                        hscroll = -1;
                        break;
                case ATKBD_SCR_RIGHT:
                        hscroll = 1;
                        break;
                default:
                        break;
        _               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;
                                                                478,1-8       32%
```

default가 일반적인 경우인 것 같으므로 여기에 break문을 추가해보았다.

```
 * 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) 06:38:20

localhost login:
```

입력을 했으나 반응을 하지 않는다. 그러므로 방금 찾았던 부분이 입력과 관련된 부분일 것이다.

```
                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);
                                                                495,2-16      32%
```

해당 부분을 보니 굉장히 복잡했다. 뭔가 많이 호출이 되지만 해당 함수들을 찾아가 보아도 더 이상 찾는 것은 무리였다. 대략 이 부분에서 입력된 것이 출력되는 것을 알 수 있었다.