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

(system call table : arch/x86/kernel/syscall\_table\_32.S)

-divide-by-zero exception INT 0

-keyboard interrupt INT 33

-“read” system call INT 128

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

INT 42

- An application program calls scanf()

scanf를 부르면 read로 변형되고 read가 interrupt를 부르기 때문에 read의 interrupt number인 INT 128.(sys 3)

- A key is pressed

INT 33

- An application causes a divide-by-zero error

INT 0

- An application program calls printf()

printf를 부르면 write로 변형되고 write가 interrupt를 부르기 때문에 write의 interrupt number인 INT 128. (sys 4)

- An application causes a page-fault error

INT 14

- A user tries to remove a file

유저가 파일을 제거하기 위해서 rm f1을 치게 되는데, 이 때 키보드 인터럽트인 INT 33이 불러진다.



파일을 지우기 위해서 system call interrupt number인 INT 128을 부르게 되고, 이것이 system call number로는 235가 된다.

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

static irqreturn\_t atkbd\_interrupt(....){

return IRQ\_HANDLED; // Add this at the first line

.............

}

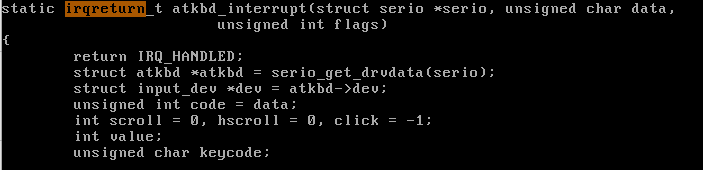
Recompile the kernel and reboot with it. What happens and why does this happen? Show the sequence of events that happen when you hit a key in a normal Linux kernel (as detail as possible): hit a key => keyboard controller sends a signal through IRQ line 1 => ......etc. Now with the changed Linux kernel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

키를 누를 때 일어나는 일:

키를 누름 => interrupt 발생시킴 => keyboard controller가 IRQ line 1을 통해서 시그널 보냄 => 그 시그널을 받은 장치는 CPU의 interrupt 입력 단자로 signal을 보냄 => CPU가 ACK 출력단자를 불러서 interrupt 번호 물어봄 -> CPU가 signal이 들어온 IRQ 번호를 basic register에 전함 => interrupt 33이 발생한걸 알아챔 =>CPU가 cs, eip, eflag를 저장한 후 IDT[33]으로 점프함 => ISR1에서 현재 프로그램이 가지고있는 모든 레지스터를 다 저장함 => ISR2를 부름 => ISR2에서 interrupt 처리 => iret로 interrupt이 처음 발생한 지점으로 돌아감

키보드 누름 => irq 1(8259A칩의 1번 단자)에 키보드 컨트롤러가 시그널 보냄 => 8259A는 33을 cpu에 전달 =>cpu는 INT 33이 발생했다고 판단 => CPU가 cs, eip, eflag (주 레지스터들)를 저장한 후 IDT[33]으로 점프함 => ISR1에서 현재 프로그램이 가지고있는 모든 레지스터를 다 저장함 => ISR2를 부름 => ISR2에서 interrupt 처리 => iret로 interrupt이 처음 발생한 지점으로 돌아감

drivers/input/keyboard/atkbd.c



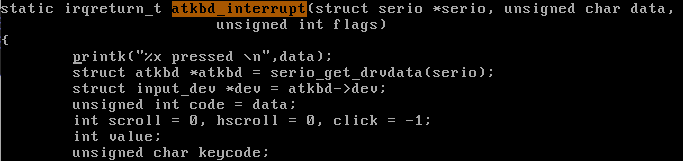


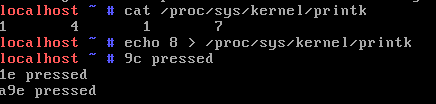
원래는 return IRQ\_HANDLED가 함수의 끝부분에 있어서 글자가 출력된 다음에 리턴을 하게 되는데, 그 전에 리턴을 해버리면 irqreturn의 함수 동작은 글자를 출력하기 전에 끝나게 된다. 이 상태로 저장하고 재부팅하면 로그인 할 때 글자가 안눌려져서 로그인이 불가능해지게 된다.

(원래 있던 IRQ\_HANDLED을 지우면 안되고 그냥 바로 추가해줘야함)

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.

drivers/input/keyboard/atkbd.c

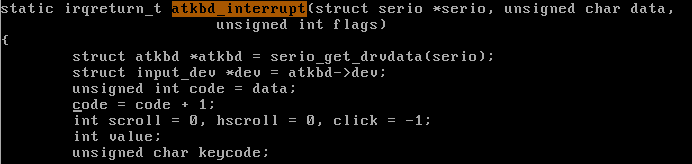




cat /proc/sys/kernel/printk를 이용해서 log level들을 출력했다. 이 중 첫번째 숫자가 console log level이고 두번째 숫자가 current printk log level인데, log level은 숫자가 작으면 작을수록 높은 것이기 때문에 console의 log level이 printk의 log level보다 높다는걸 알 수 있다. 만약 printk가 정상적으로 출력되게 하려면 printk의 log level이 더 높아야하기 때문에 echo를 사용해 console log level을 8로 낮춰줌으로써 printk의 log level을 더 높게해서 printk가 출력되게 해줬다. 그 결과, echo 명령어를 누르고 엔터를 누르고 뗄 때 엔터의 스캔코드 값인 1c에서 80을 더한 9c가 출력됐고, 그 뒤에는 a를 눌러봤는데 a는 스캔코드상 1e이기 때문에 1e가 출력됐고 a가 화면에 표시된 다음 키보드 키가 release 됐기 때문에 1e에서 80을 더한 9e가 출력됐다.

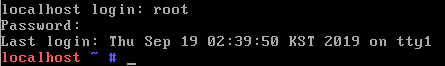
여기에서 %x는 16진수이고, %d는 10진수이다.

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?

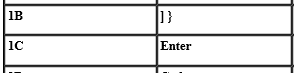


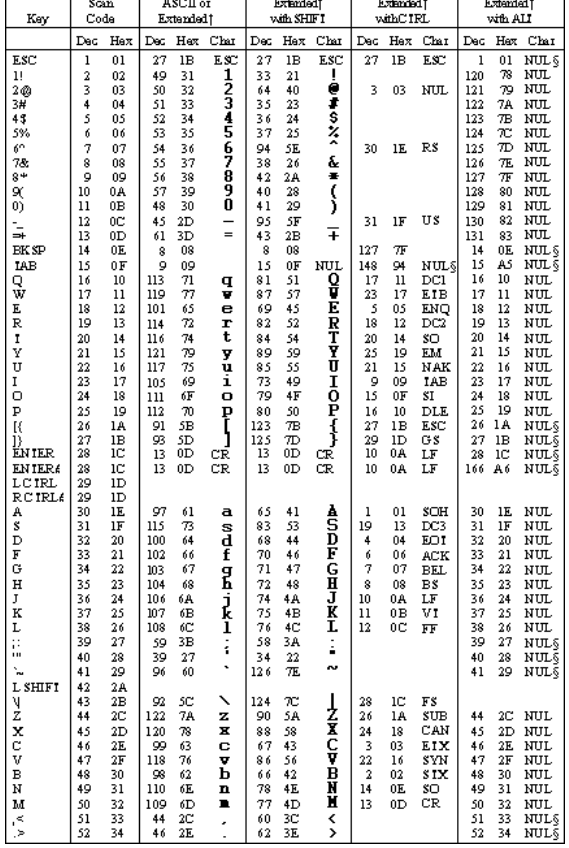


다음 캐릭터를 출력하게 하기 위해서 data값과 같은 code값에 1만큼 추가해줬다. 그 결과, root를 입력했을 때 tppy가 출력됐다.



원하는 글자의 스캔코드에서 1을 빼면 그것은 키보드 상에서 친 글자의 바로 왼쪽에 있는 글자가 된다. 따라서 root를 치고싶으면 각각 글자의 바로 왼쪽에 있는 글자인 eiir을 차례대로 치면 된다.



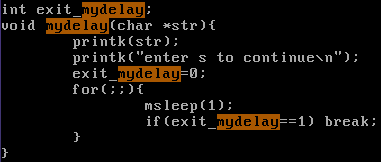


(즐겨찾기에 키보드 스캔코드 (hex로 비교) 넣어둠)

엔터는 스캔코드상 1c인데, 여기서 1을 뺀 값은 1b가 되기 때문에 엔터를 치고 싶으면 ]를 치면 된다. 이렇게 해서 로그인에 성공하게 됐다.

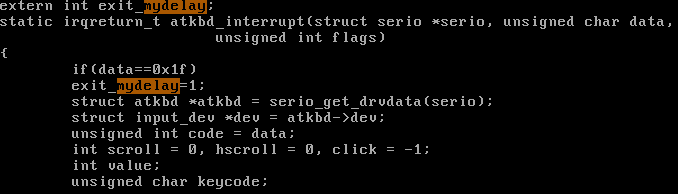
5) Define a function "mydelay" in init/main.c which whenever called will stop the booting process until you hit 's'. Call this function after do\_basic\_setup() function call in kernel\_init() in order to make the kernel stop and wait for 's' during the booting process. You need to modify atkbd.c such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

init/main.c





drivers/input/keyboard/atkbd.c



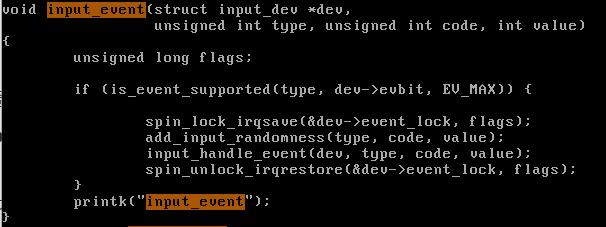
main.c에서의 mydelay 함수의 코드를 보면 exit\_mydelay가 1이 되기 전까진 계속해서 무한루프를 돌게 돼있다. s를 눌러야 부팅 프로세스가 진행되게 하려면 키보드 입력을 받는 irqreturn 함수에서 s를 칠 때 exit\_mydelay가 1로 바뀌어지도록 해야한다. exit\_mydelay는 외부함수이기 때문에 extern을 사용해서 정의해줬고, s는 아스키코드로 0x1f이기 때문에 data의 값이 0x1f일 때 exit\_mydelay의 값이 1이 되도록 해주었다.



그 결과, enter s to continue가 출력된 다음에 s를 쳤을 때 부팅 프로세스가 계속 진행됐다.

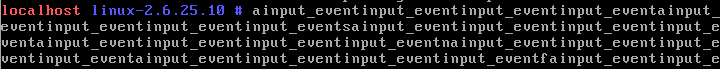
6) Which function call in atkbd\_interrupt() actually displays the pressed key in the monitor?

driver/input/input.c



drivers/input/keyboard/atkbd.c





input.c 파일에서 printk(“input\_event”)를 넣고 atkbd.c 파일의 atkbd\_interrupt() 함수에서 input\_event 함수 전에 printk(“a”)를 넣었다. 그리고 echo 8 > /proc/sys/kernel/printk를 해준 결과, 알파벳 하나를 치면 그 알파벳이 출력되기 이전에 a와 수많은 input\_event들이 출력됐다. 따라서 눌려진 키를 실제로 출력하는 함수는 input\_event 함수라는걸 알 수 있었다.