**오퍼레이팅시스템 3주차**

**12180626 성시열**

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.

각 동작별 interrupt 번호가 있다. 함수를 실행할 때는 INT 128번이며 그 내에서 syscall 번호가 정해져있다. 각 함수별 system call number 특정 되어있다. 예를 들어 read 함수라면 컴파일러가 eax에 system call number를 저장하고 INT 128을 호출하고 컴파일한다. read함수의 인자는 ebx, ecx, edx에 각각 저장되어 전달된다.

- A packet has arrived : INT 42

- An application program calls scanf() : INT 128 / syscall 3

- A key is pressed : INT 33

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

- An application program calls printf() : INT 128 / syscall 4

- An application causes a page-fault error : INT 14

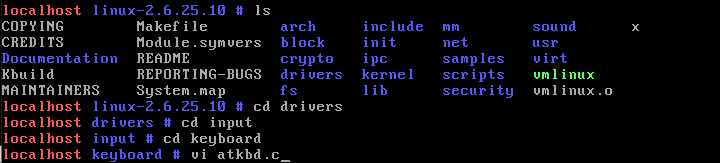
2) Change drivers/input/keyboard/atkbd.c as follows.

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

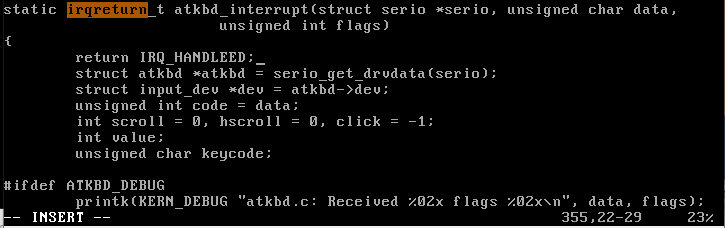
return IRQ\_HANDLED; // Add this at the first line

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

}

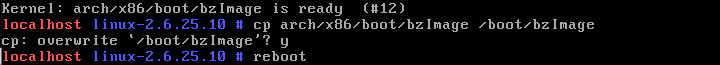


해당 경로를 따라 이동하였다.



해당 함수의 첫 줄에 return IRQ\_HANDELED; 를 추가하였다.

Recompile the kernel and reboot with it. What happens and why does this happen?



make bzImage를 통해 변경된 파일만 찾아 컴파일하였다. 생성된 리눅스파일을 복사 붙여넣기 하고 재부팅하였다.



MyLinux에 접속해본 결과 login 창에서 키보드 입력이 안되는 모습이다. 기존에는 키보드가 눌리면 해당 키보드가 누른 문자를 저장하고 display해준다. 하지만 위에서 수정한 함수 아무런 과정을 거치지 않고 곧장 IRQ\_HANDLED를 return 하기 때문에 실행되지 않는다.

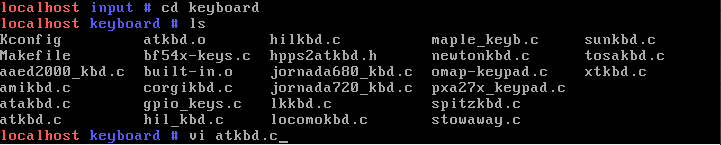
VirtualBox에서 머신 > 재시동을 통해 다시 프로그램을 실행하여 MyLinux2로 이동한다.

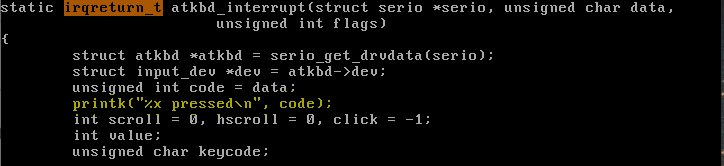
앞서 수정했던 코드를 원래 상태로 되돌린 뒤 My Linux로 이동하여 로그인한다.

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.

키보드 입력 -> 키보드가 irq1 라인을 통해 8259A 칩으로 신호를 보냄(각 주변장치 별로 연결된 line이 있음) -> 8259A 칩은 INT을 통해 CPU로 신호를 보냄 -> CPU는 ACK로 반응함 -> 8259A 칩은 INT 33번을 CPU로 보냄 -> cpu는 INT 33에 대한 행동을 실행함. -> 현재 프로세스를 중지하고 현재 상태를 저장함 -> 이후 keyboard press의 ISR1위치인 interrupt[1]로 이동(arch/x86/kernel/entry\_32.S에 위치함) -> keyboard interrupt를 처리하여 입력된 문자를 기억하고 스크린에 보여준다. -> reschedule

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.

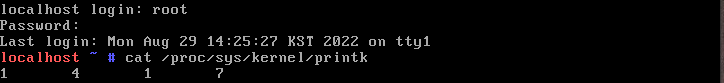




atkbd.c로 이동하여 표시된 부분과 같이 printk문을 추가한다.



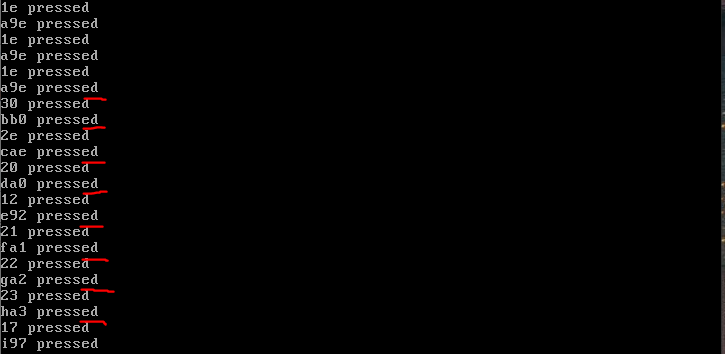
홈 디렉토리로 이동한 뒤 make bzImage를 통해 recomplie하고 다음의 과정을 진행한 뒤 reboot 한다.



로그인 후 log level을 확인해본 결과 1 4 1 7 임을 확인하였다.

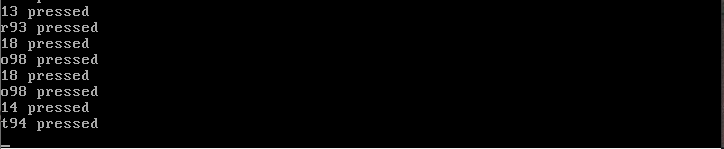


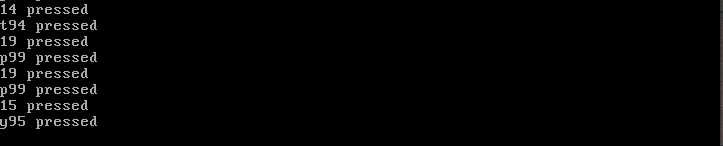
echo 8 > /proc/sys/kernel/printk을 통해 log level을 수정하였다.



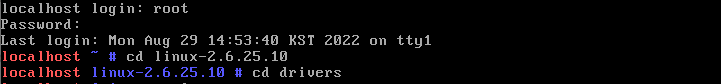
abcdefhgi까지 입력해보았다. 구분선을 기준으로 누른 a를 누른 순간 1e, a를 뗀 순간 9e가 출력되었다. 다른 문자도 같은 원리로 출력된다.

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?



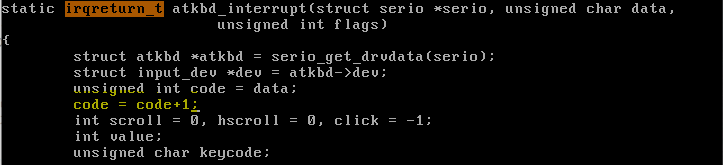


root를 입력할 때 tppy가 입력되려면 입력 시 code가 1씩 차이남을 확인하였다.



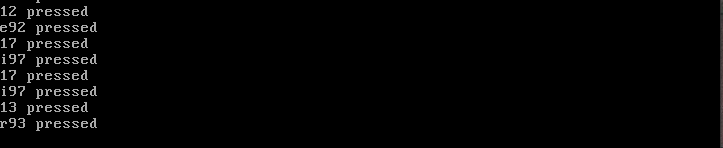
강제 재부팅을 진행한 뒤 My Linux2로 이동하여 코드를 수정해준다.



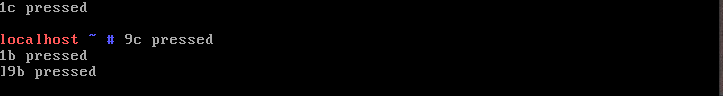


이를 해결하기 위해 atkbd.c로 이동하여 code에서 1을 추가해주는 코드로 수정하였다.

이를 통해 로그인이 방법을 고려해보았다. 알파벳 순서상 순서 차이가 발생한다고 판단하여 id인 root를 출력하려면 eiir을 입력해야 할 것으로 예상된다.



입력해본 결과 root에서 1을 뺀 값임을 알 수 있다.

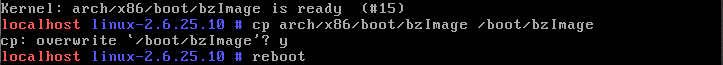


비밀번호는 enter를 입력해야 하므로 enter의 code를 확인해본 결과 1c임을 확인하였다.

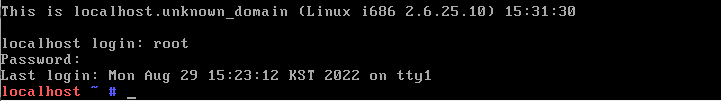
그러므로 1b에 해당하는 부분을 찾아본 결과 ] 문자임을 확인하였다.



홈 디렉토리로 이동한 뒤 recompile 하였다.

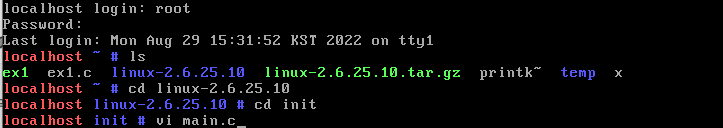


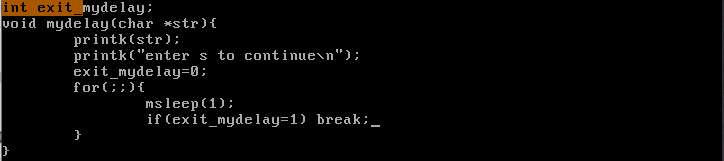
Recompile 한 뒤 재부팅하였다.

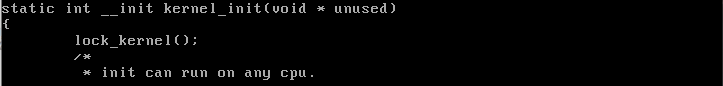


My Linux로 접속하여 로그인해보았다. eiir 후 ], ]해본 결과 로그인이 가능하였다.

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'.

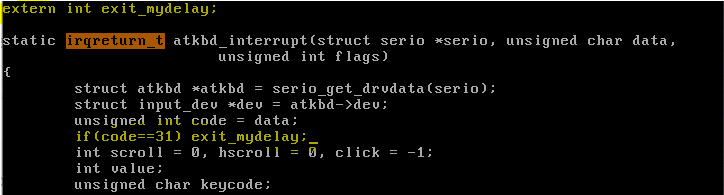




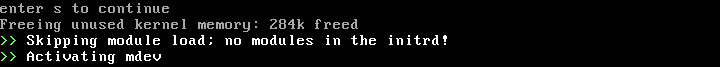




Main.c로 이동한 뒤 코드를 위와 같이 수정하였다.



Atkbd.c로 이동한뒤 코드를 위와 같이 수정하였다. extern은 외부 변수를 가져오는 것으로 예상된다.

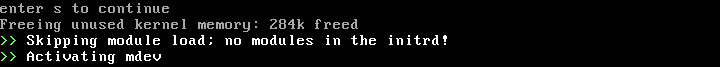


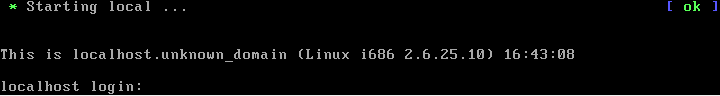
recompile 후 reboot 해본 결과 enter s to continue가 출력된 것을 확인하였다. 출력은 되었으나 시스템 진행 상 차이는 없음을 확인하였다.

5-1) Add mydelay before do\_basic\_setup(). What happens and why?



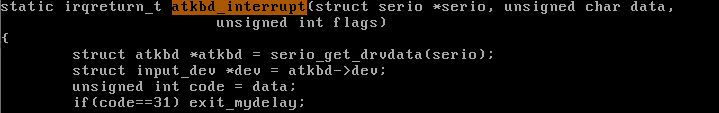
main.c에서 해당 코드를 추가해주었다.

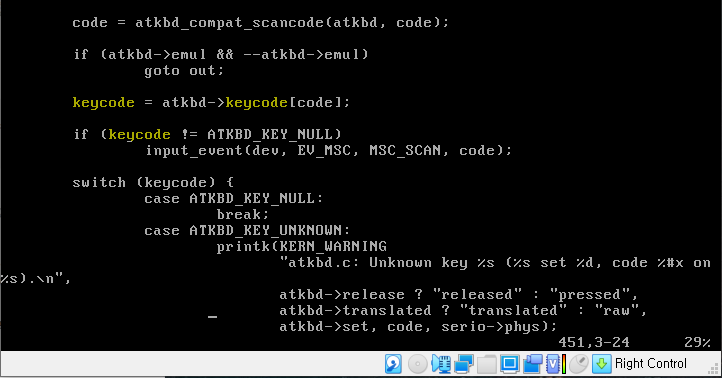




recompile 후 reboot 해본 결과 login 창에서 다른 키보드를 입력하면 인식이 안 되었고, s를 누른 뒤 키보드 입력이 되는 것을 확인하였다. mydelay를 통해 s 입력 전까지 키보드가 통제되었었는데, 이를 setup 직전에 추가해줌으로써 로그인 창에서 키보드가 인식 안 됨을 확인하였다.

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





해당 함수 하단에 호출 부분을 확인해보았다. 실제 모니터에 눌린 키를 화면에 호출하는 기능이 무엇인지 모르겠지만 keycode 가 많이 핵심 변수임은 파악하였다.

6-1) What are the interrupt numbers for divide-by-zero exception, keyboard interrupt, and "read" system call? Where is ISR1 and ISR2 for each of them (write the exact code location)? Show their code, too.

|  |  |  |  |
| --- | --- | --- | --- |
|  | INT number | ISR 1 | ISR 2 |
| divide-by-zero | 0 | divide\_error | do\_divide\_error |
| keyboard interrupt | 33 | interrupt[1] | atkbd\_interrupt |
| system call “read” | 128(syscall 3) | system\_call | sys\_read |

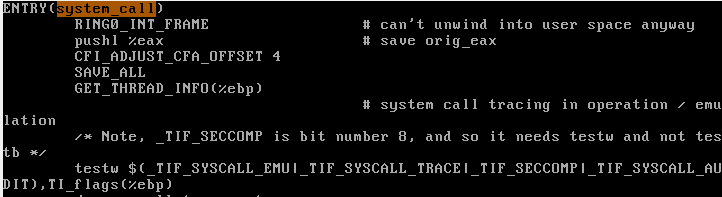
상황에 따른 interrupt number, ISR 1, ISR 2를 표로 정리해보았다.

ISR 1 의 코드는 arch/x86/kernel/entry\_32.S에 존재한다.



해당 위치로 이동하여 코드를 확인하였다.

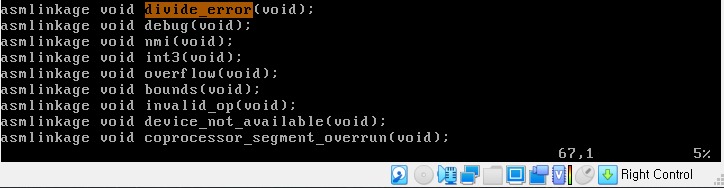




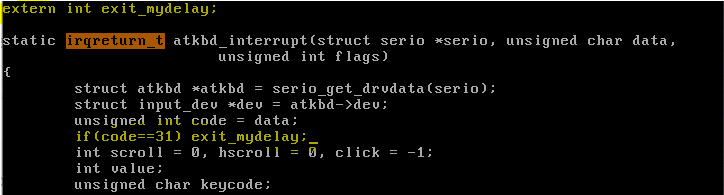
검색을 통해 코드를 파악해보았고, interrupt[1]은 어떻게 찾아야할지 감이 오지 않았다.

이후 ISR 2의 위치와 해당 코드들이다.

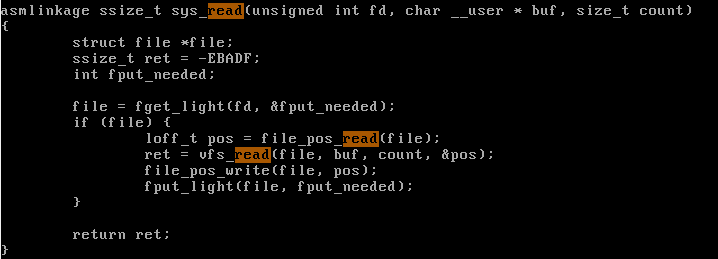
|  |  |  |
| --- | --- | --- |
|  | ISR 2 | 코드 위치 |
| divide-by-zero | do\_divide\_error | arch/x86/kernel/traps\_32.c |
| keyboard interrupt | atkbd\_interrupt | drivers/input/keyboard/atkbd.c |
| system call “read” | sys\_read | fs/read\_write.c |



실제 파일은 compile할 때 생성된다.



keyboard interrupt는 위에서 계속 수정해왔던 코드이다.



read\_write.c에서도 read에 해당하는 부분이 매우 많았다.