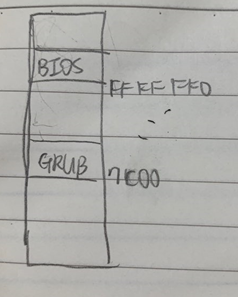
0) Boot sequence.

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

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

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



BIOS는 ROM area의 FFFFFF0 위치에 있고, GRUB는 0007c00의 위치에 있다. linux의 실행파일의 위치는 /boot/bzImage에 있고, GRUB는 /boot/grub/grub.conf를 보고 실행파일이 /boot/bzImage에 있는걸 알 수 있다.

1) Simple modification of the kernel.

Add

printk("hello from me\n");

after

printk(linux\_banner);

in start\_kernel(). Go to the Linux top directory and compile the kernel and replace the boot image. Reboot with this new kernel, and run dmesg to see if the kernel is printing "hello from me".

# cd linux-2.6.25.10

# cd init

# vi main.c

....... modify start\_kernel()

# cd .. – go back to the Linux top directory

# make bzImage -- recompile the kernel

# cp arch/x86/boot/bzImage /boot/bzImage

--copy the new kernel image to boot location

# reboot -- reboot the system with this new kernel and select "My Linux"

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

(select “My Linux”)

# dmesg > x

# vi x -- now check if we can see our new message





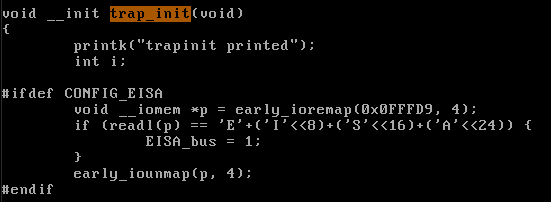


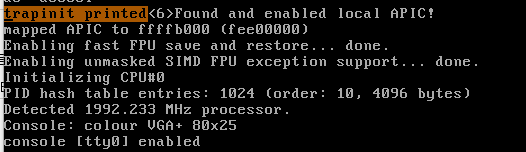
Main.c에서 linux\_banner 다음에 hello from me를 추가해줬고 그 결과 부팅의 첫 줄 다음에 hello from me가 출력돼었다.

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():

# grep -nr “trap\_init” \* | more

grep를 사용해서 trap\_init 함수를 검색한 뒤 trap\_init 함수가 있는 곳에 가서 printk를 썼다. printk가 출력되는 곳이 실제로 trap\_init 함수가 불러지는 곳이다.

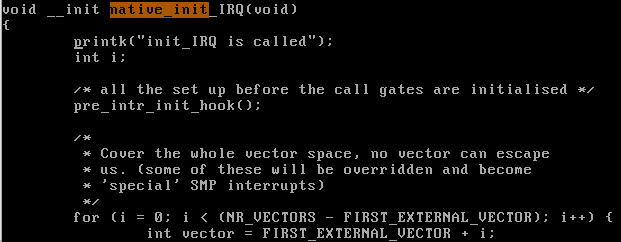


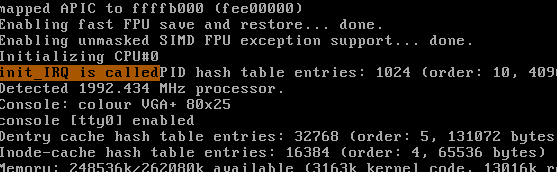


arch/x86/kernel/traps\_32.c 에서 printk를 썼더니 printk에 쓴 내용이 그대로 부팅 메시지에 출력됐다.

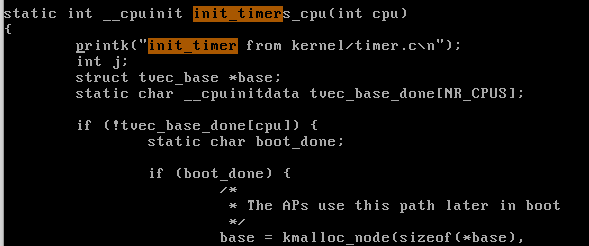
3) Find also the exact locations of init\_IRQ() and insert some printk() in the beginning of init\_IRQ() to confirm (actually you insert it in native\_init\_IRQ). Do the same thing for init\_timers() and time\_init().

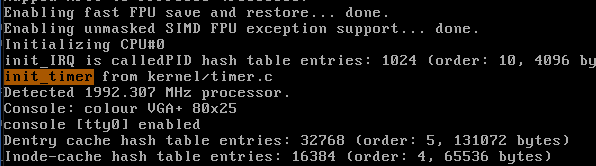
나머지 함수들도 2번 문제와 같이 grep을 쓴 후 함수 안에 printk를 써서 부팅 메시지를 확인해보는 방법으로 찾아냈다.



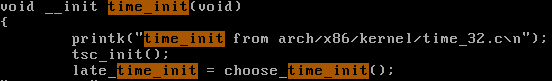


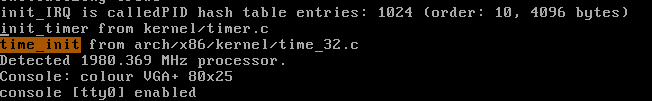
arch/x86/kernel/i8259\_32.c (init\_IRQ)





kernel/timer.c (init\_timers)





arch/x86/kernel/time\_32.c (time\_init)

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".

# cd /boot/grub

# vi grub.conf

.......move the cursor to "title=My Linux" and copy 4 lines there (with 4yy)

........move the cursor to the last line and paste the 4 lines (with p)

........change this new kernel as below: Linux => Linux2, bzImage=>bzImage2, the rest same

……………….

title=My Linux2

root (hd0,0)

kernel /boot/bzImage2 root=/dev/ram0 init=/linuxrc ramdisk=8192 real\_root=/dev/sda3 doscsi

initrd /initramfs-genkernel-x86-2.6.23-gentoo-r8

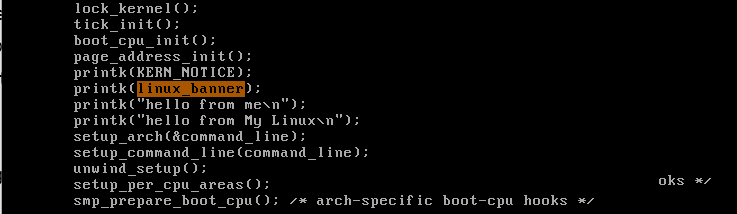
Go to start\_kernel() and add "hello from My Linux" and recompile the kernel

and save it in /boot/bzImage. Now go back to start\_kernel() and change it to

"hello from My Linux2", recompile the kernel and save it in /boot/bzImage2.

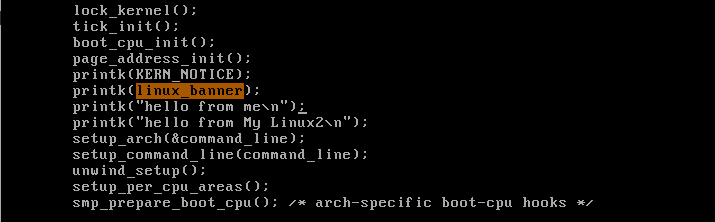
Check if you have different boot message with different Linux.





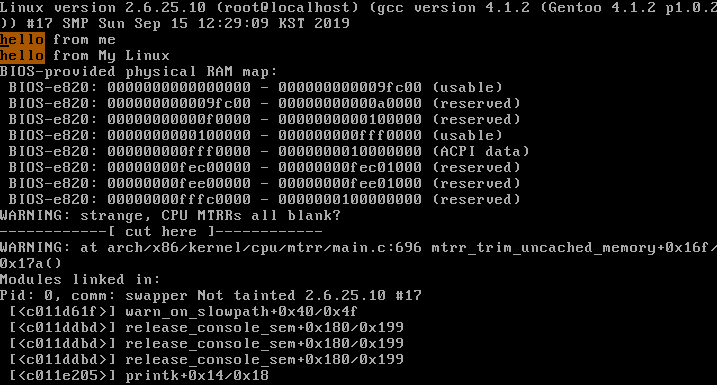


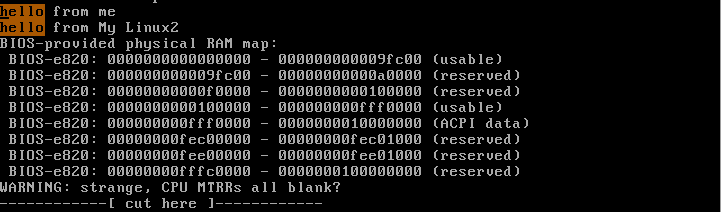
먼저 미리 써뒀던 hello from me 다음줄에 hello from My Linux를 추가한 후 /boot/bzImage에 저장시켰다.





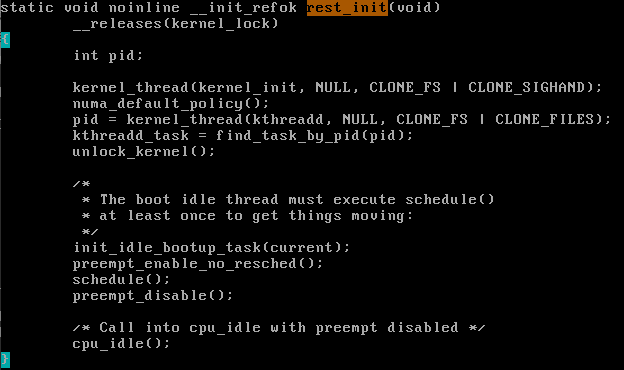
hello from My Linux를 hello from My Linux2로 바꾼 뒤 /boot/bzImage2에 저장시켰다.

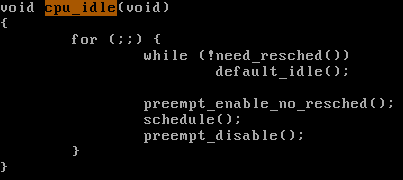




그 결과, My Linux의 부팅 메시지에는 hello from My Linux가 출력됐고, My Linux2의 부팅메세지에는 hello from My Linux2가 출력됐다.

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





login이란 메시지는 rest\_init 함수가 무언가를 부르면 그 무언가가 출력해내는 메시지다. 모든 작업이 다 끝나고 나면 idle mode로 전환하여 cpu\_idle을 실행하게 되는데, 사용자가 무언가를 입력하기 전까진 여기에서 무한루프를 돌기 때문에 CPU는 여기에 있다고 할 수 있다.