**오퍼레이팅시스템 3주차 Homework**

**002분반 12180626 성시열**

0) Boot sequence.

Power ON을 하게 되면 cs=0xF000, ip=0xFFF0인 상태로 시작되고 BIOS가 가장 먼저 실행된다.

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

첫번째 프로그램인 BIOS가 실행될 때, 그 위치는 FFFFFFF0이다.

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

BIOS에서 시스템을 초기화하고, 부팅디스크의 첫번째 메모리 sector에서 GRUB을 찾는다.

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

리눅스 실행파일은 /boot/bzImage에 위치해있다.

GRUB로 리눅스 실행파일의 위치를 파악하는 방법은 /boot/grub/grub.conf를 통해 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

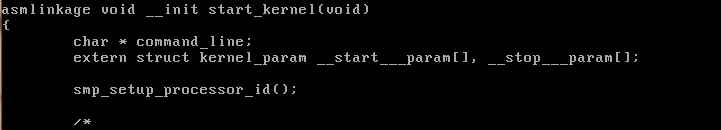
# reboot -- reboot the system with this new kernel and select "My Linux"

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

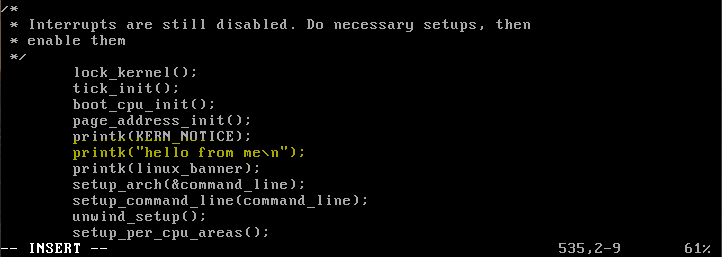
(select “My Linux”)

# dmesg > x

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



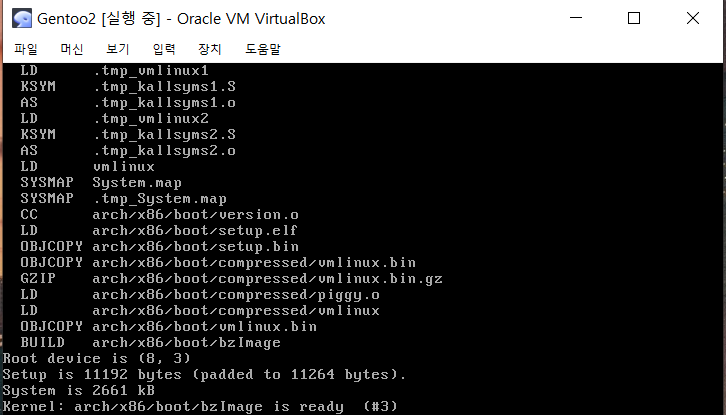
Top 디렉토리에서 init 디렉토리로 이동한 뒤 main.c에서 start\_kernel 함수를 찾았다.



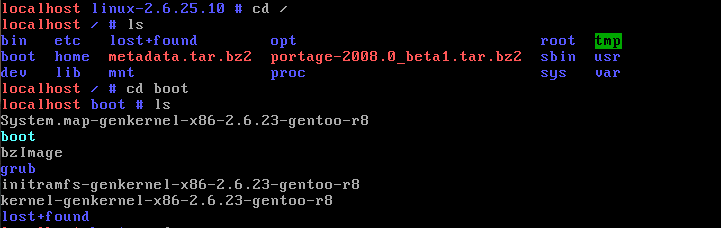
문제에서 주어진대로 printk(linux\_banner); 상단에 printk문 하나를 추가하였다.



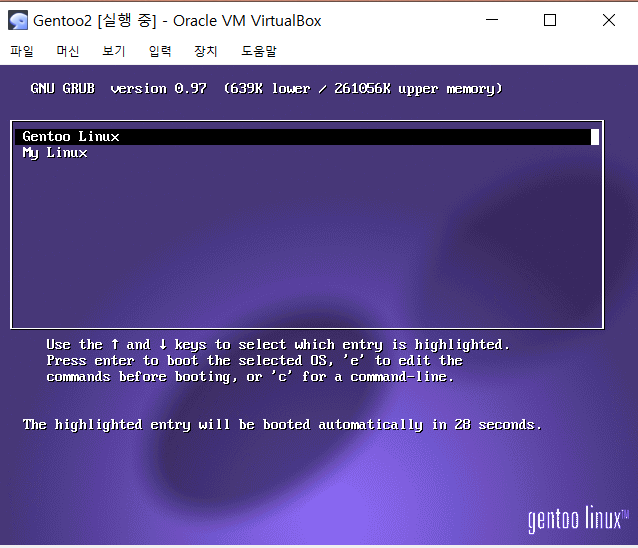
Top 디렉토리로 다시 이동한뒤 make bzImage(boot zip Image)를 통해 kernel을 다시 컴파일한다.



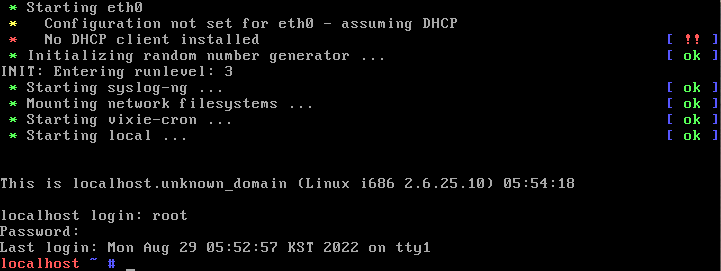
make bzImage를 통해 실행된 결과의 모습이다. arch/x86/boot/bzImage에 생성된 새로운 커널을 부팅 위치인 /boot/bzImage에 복사해주기 위해 cp arch/x86/boot/bzImage /boot/bzImage를 사용한다. overwrite를 허용한다.



실행 후 /boot를 찾아간 결과 bzImage가 존재함을 확인하였다.



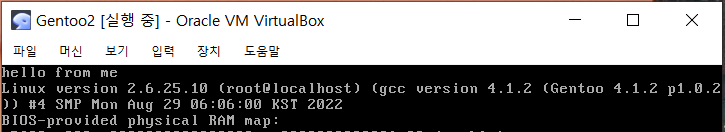
reboot를 하면 재부팅이 되며 해당 창으로 이동함을 알 수 있다. My Linux를 선택한다.



My Linux에 접속하며, root를 입력하고 Enter를 눌러 로그인한다.



dmesg > x를 통해 부팅 메시지를 x에 저장하고 x를 확인한다.



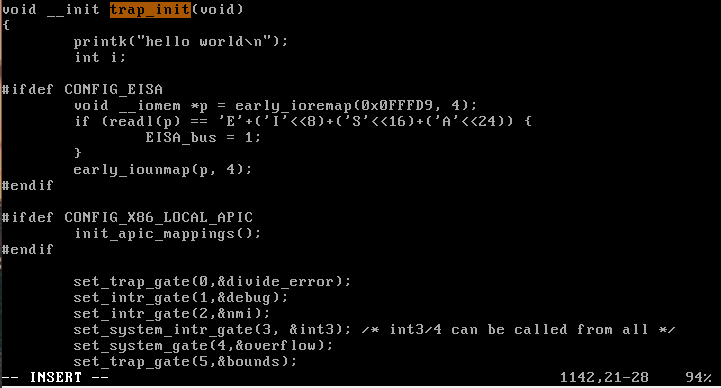
hello from me가 추가됨을 확인하였다.

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():

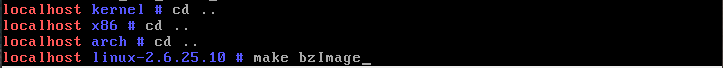
top디렉토리에서 grep -nr “trap\_init” \* | more 를 통해 trap\_init 함수가 정의된 곳을 파악하였다.



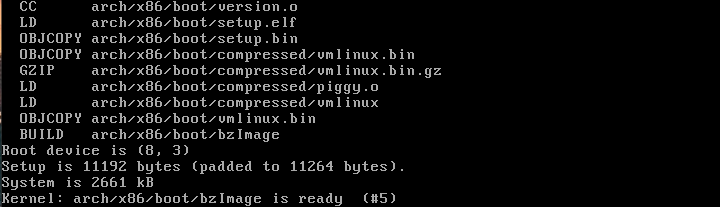
해당 파일로 이동하였다.



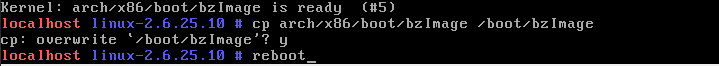
해당 파일의 trap\_init 함수가 정의된 곳으로 가서 printk(“hello world\n”);을 추가하였다.



Top 디렉토리로 이동한 뒤 make bzImage를 입력한다.

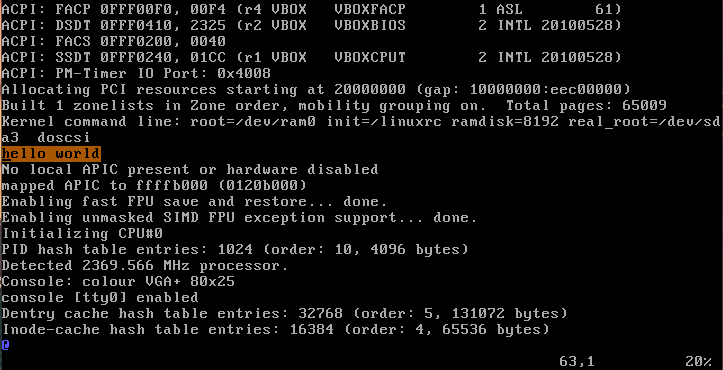


arch/x86/boot/bzImage에 커널이 준비된 모습이다.



cp arch/x86/boot/bzImage /boot/bzImage를 통해 새로운 커널을 부팅이 되는 주소에 복사한다. 오버라이트를 허용하고 재부팅한다.

앞서처럼 MyLinux를 선택해준다.

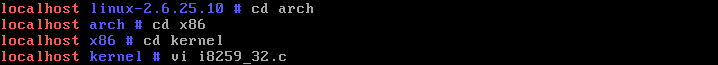


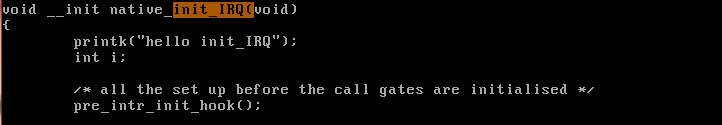
부팅 메시지를 확인해본 결과 hello world가 들어간 것을 알 수 있다.

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().

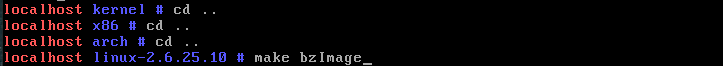


grep -nr init\_IRQ \* | more을 통해 init\_IRQ가 정의된 경로를 파악한다.

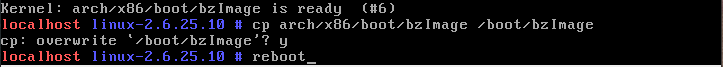




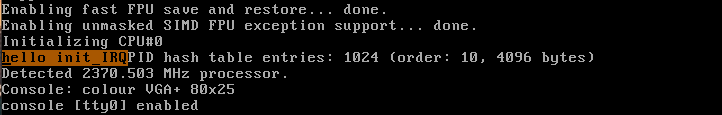
경로를 따라 디렉토리를 이동한 뒤 init\_IRQ 함수 정의에서 printk 문을 한 줄 추가한다.



Top 디렉토리로 이동한 뒤 make bzImage를 실행한다.

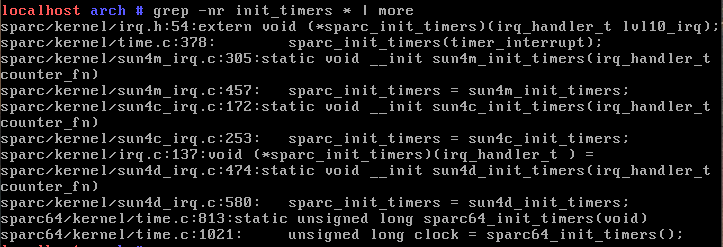


실행한 뒤 새로운 커널을 부팅 실행하는 위치에 복사 붙여넣기한다. 오버라이팅을 허용하고 재부팅한다.



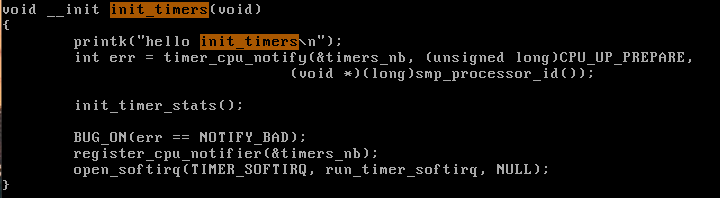
부팅 메시지를 확인해본 결과 hello init\_IRQ가 존재함을 확인하였다.

init\_timers



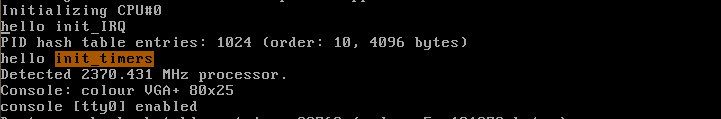


grep을 활용하여 init\_timers를 찾는다.



해당 파일로 가서 printk 문을 추가한다.

top 디렉토리 이동 후 make bzImage 및 cp, reboot를 하고 My Linux를 선택한다.



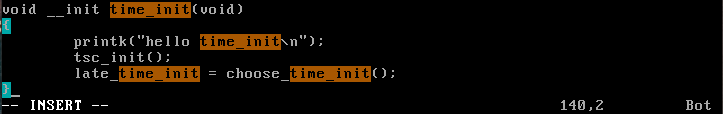
부팅메세지를 확인한다.

time\_init()

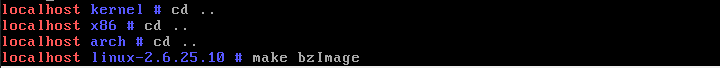
top 디렉토리/arch에서 grep을 이용하였다.

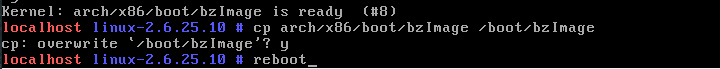


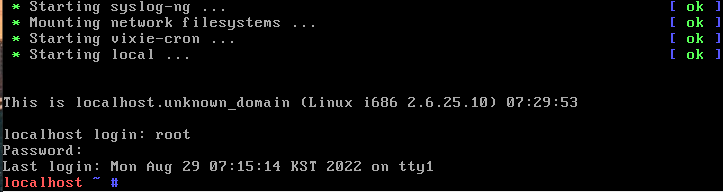




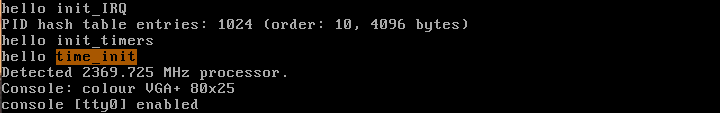
해당 위치로 이동하여 printk문을 추가한다.











hello time\_init이 추가된 모습이다.

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

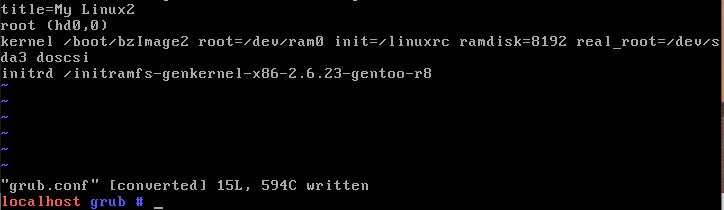
initrd /initramfs-genkernel-x86-2.6.23-gentoo-r8

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

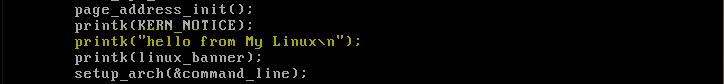
and save it in /boot/bzImage. Now go back to start\_kernel() and change it to

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

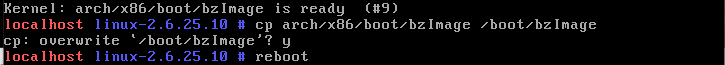
Check if you have different boot message with different Linux.



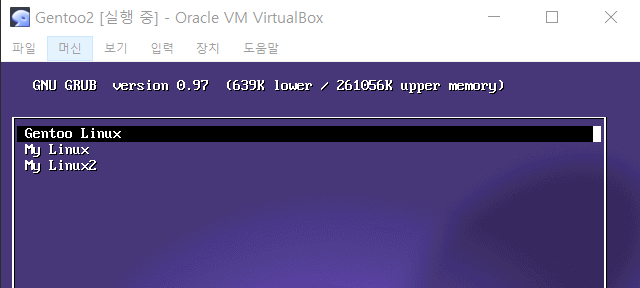
/boot/grub로 이동하여 grub.conf 파일을 확인하였다. 기존에 작성된 내용 아래 My Linux2라는 title을 가진 another Linux selection을 만든다.



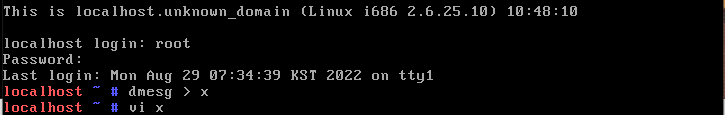
init 디렉토리로 이동한 뒤 main.c 파일에 들어가 수정한다.



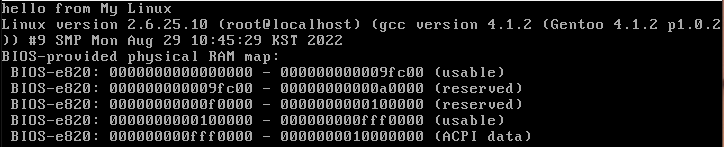
Top 디렉토리로 이동하여 make bzImage를 하고 복사하여 부팅 위치에 붙여넣기한다. 재부팅한다.



추가한 My Linux2가 생성된 모습. 일단 My Linux를 선택한다.

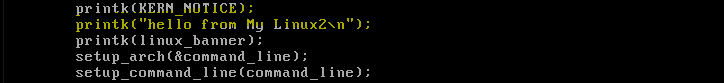


부팅 메시지를 확인한다.

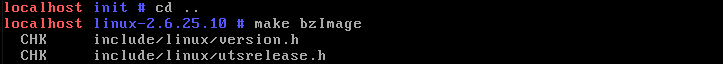


부팅 메시지 가장 위에 hello from My Linux가 출력됨을 확인하였다.

이번엔 My Linux2를 출력해보자.

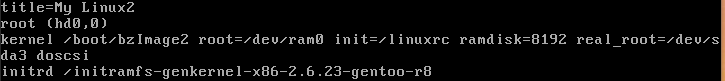


앞서처럼 init 디렉토리의 main.c 파일로 이동하여 printk문을 수정한다.



Top 디렉토리로 이동한 뒤 make bzImage를 실행한다. 이후 복사를 진행하고 재부팅한 뒤

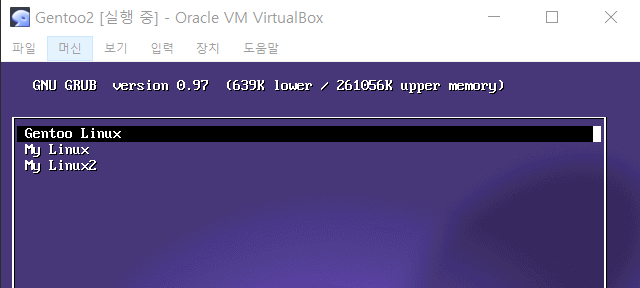
My Linux2로 접속하였으나, 에러가 발생하였다. 코드를 수정하기 위해 Linux1으로 접속하여 # cd /boot/grub , vi grub.conf를 들어가보았다.



다시 확인해본 결과 오타는 없었다.



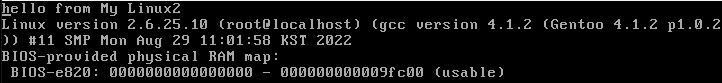
save it in /boot/bzImage2를 통해 My Linux2의 부팅 장소가 다르다는 것을 파악하고 cp를 다시 진행하였다. 이후 재부팅 하였다. 앞선 방식과 다른 점은 오버라이팅이 아니었다.



해당 화면이 떴고, My Linux2로 접속하였다.

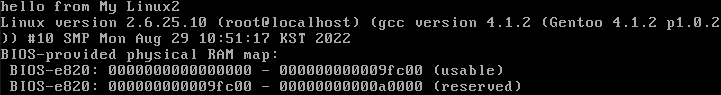


부팅 메시지를 확인하였다.



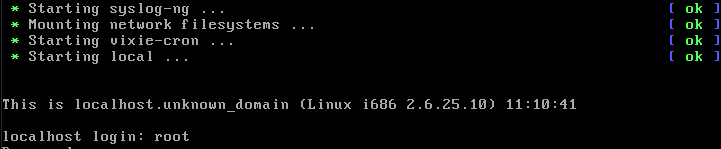
My Linux2에서는 부팅 메시지 상단에 hello from My Linux2가 출력됨을 알 수 있다.

호기심에 선택한 버전 별로 출력문이 다르게 나오는 것인지 궁금하여 reboot 후 My Linux로 접속하여 부팅 메시지를 확인하였다.

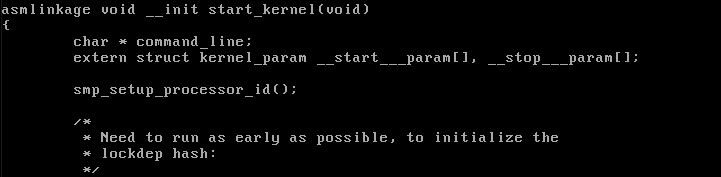


My Linux2가 나오는 것을 알 수 있었다. 이러한 결과로 선택한 버전에 따라 나오는 것이 아닌, start\_kernel() 함수에 printk문을 추가한 뒤 make bzImage로 생성한 파일을 복사한 시점에 따라 출력된다는 것을 확인하였다.

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



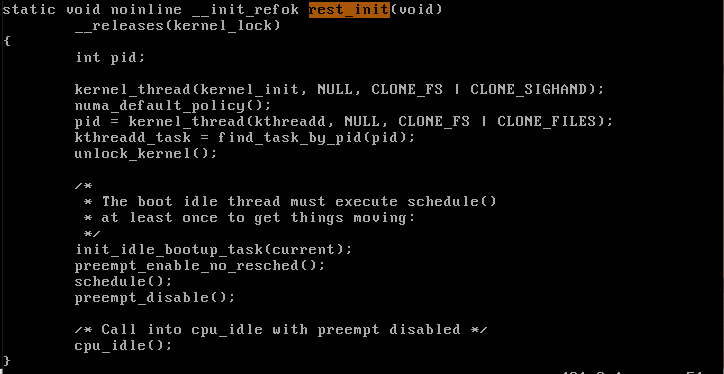
Reboot를 통해 부팅의 과정에서 부팅 메시지가 모두 출력되고 난 뒤 가장 마지막에 login이 출력되면서 사용자의 입력을 기다리는 것을 알 수 있었다.



먼저 start\_kernel()을 찾아보았다. Start\_kernel에는 부팅 때 실행되는 함수들이 존재하였다. (printk 추가를 통해 확인해온 점이다.)



start\_kernel()의 가장 마지막 부분에는 rest\_init() 함수가 존재하였다.

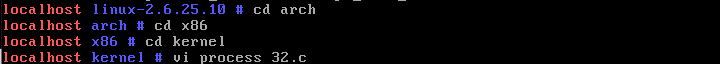


grep -nr rest\_init \* | more을 통해 확인해본 결과 같은 파일은 main.c 내에 정의 되어있었다.

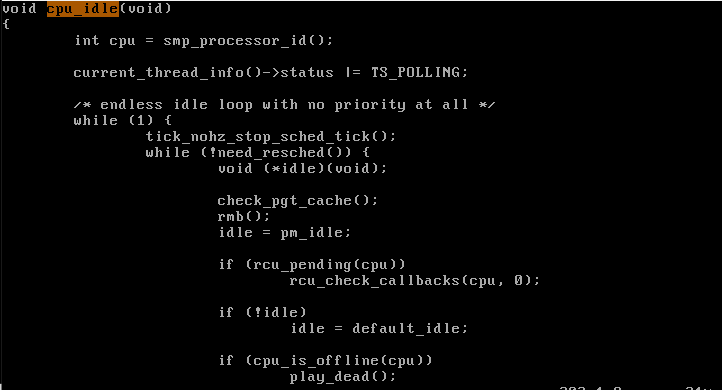
해당 함수의 가장 아래를 보면 cpu\_idle() 함수가 호출 됨을 파악하였다.

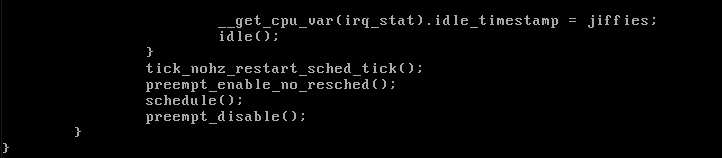


grep -nr cpu\_idle \* | more을 통해 확인해본 결과 해당 위치에 정의가 있음을 확인하였다.



해당 위치로 이동하여 process\_32.c 를 확인하였다.





해당 코드는 while(1){ }을 통해 무한루프를 발생시키는 코드이며, 사용자로부터 아이디를 입력받을 때까지 tick 하며 cpu가 한 단계씩 실행되고 있을 것으로 예상된다.