1.0) Download Virtualbox from Internet and install.

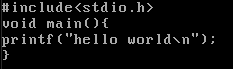
1.1) Download Gentoo virtualbox files (gentoo.zip) from the I-class and un-compress it.

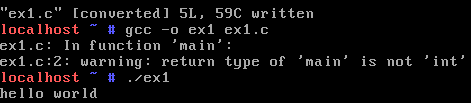
1.2) Run VirtualBox, and click File>Import and go to the gentoo directory. Select Gentoo2.ovf. Uncheck USB controller. Select Import. This will install Gentoo Linux on your virtual box. You can export your gentoo to your usb and bring it home later. In the directory, vmdk is the hard disk image, ovf file has the VM configuration, and mf file contains UUID's of the VM. UUID stands for Universal Unique IDentifier.

1.3) Run Gentoo. If you have an error, run VirutalBox as administrator and try again. For USB error, simply disable usb controller in "Setting" tab. For Hyper-V error (Raw-mode is unavailable), turn off Hyper-V feature in control panel>program and feature>window feature. Select My Linux. Login as root and hit Enter for the password prompt. If VirtualBox still cannot open the session, you need to look at the log file (right click on Gentoo VM and select “log file”) and see what is the error and fix it.

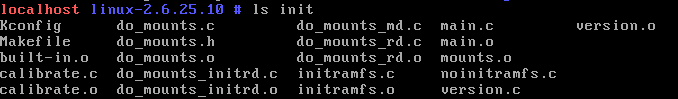
(In some cases, you may need to download and install virtualbox extension package.)

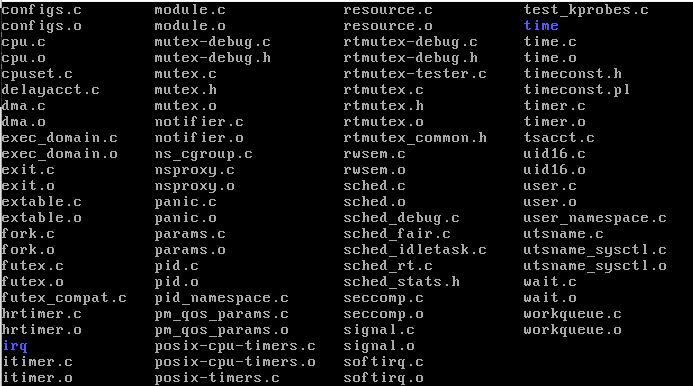
1.4) Make a simple program that displays "hello world". Compile and run it.

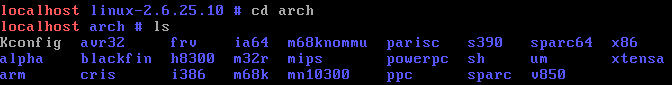




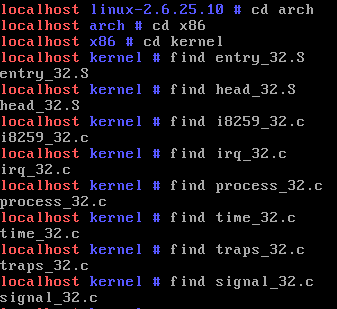
2) Go to linux-2.6.25.10 directory and find all the files referred in Section 1 such as main.c, fork.c, entry\_32.S, etc.

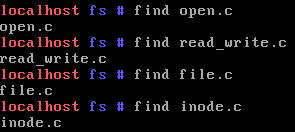


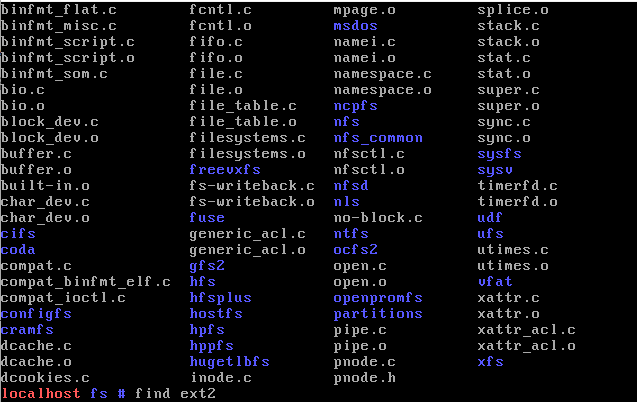


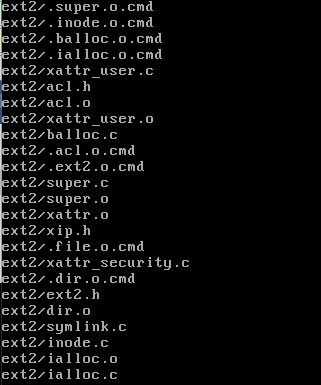












ls 명령어를 사용해 찾아낸 파일들은 빨간색 네모로 표시했고, ls 명령어로 표시되는 파일이 많아서 못찾은 파일들은 다 find 명령어로 찾아냈다.

3) Find the location of start\_kernel().

To find a string "start\_kernel", go to the linux top directory (linux-2.6.25.10) and do

$ grep -nr "start\_kernel" \* | more

Use "space" to move down the screen, "q" to exit

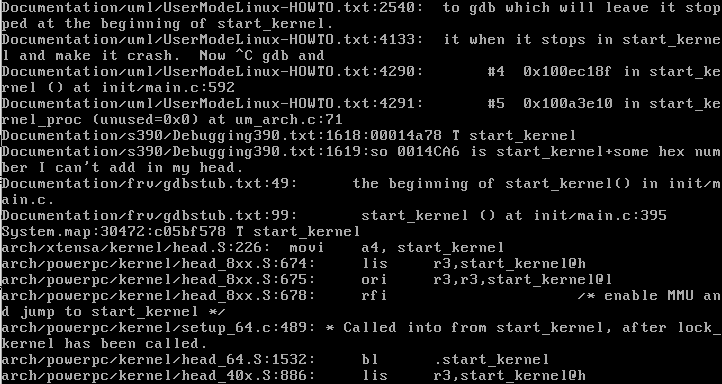
Once you found the file that has "start\_kernel", use "vi" to read the file.

In vi, type "/start\_kernel" to search for the first instance of "start\_kernel".

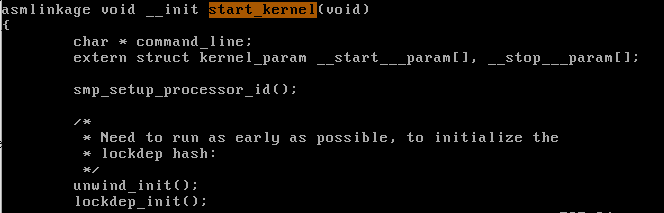
For the next string, simply type "/". Repeat "/" until you find the start\_kernel() function.

Use "j" to mode down the cursor, "k" to move up, "^f" to move one screen down, "^b" to move

up one screen up.

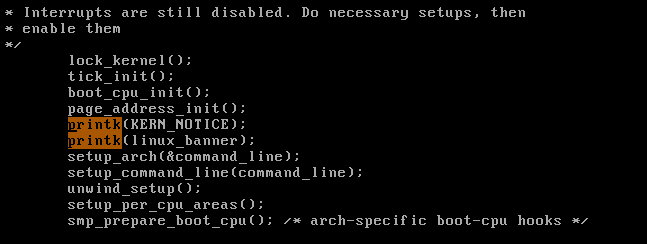


$grep -nr “start\_kernel” \* | more 를 사용해서 start\_kernel() 함수가 init의 main.c에 있는걸 발견했다.

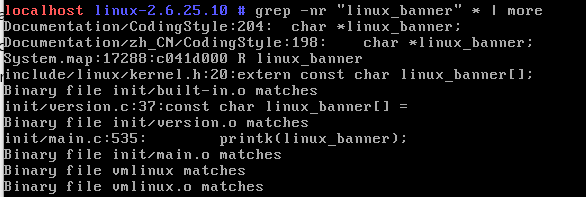


main.c 파일을 열고 나서 /start\_kernel을 사용해 start\_kernel을 검색해서 start\_kernel 함수를 찾았다. start\_kernel 함수는 리눅스가 처음 부팅할 때 시작되는 함수이고, 여기에서부터 본격적으로 초기작업이 시작된다.

4) start\_kernel() is the first C function run by Linux. Predict what will be the first message appearing on the screen by analyzing start\_kernel(). Note that "printk()" (not "printf") is the function to print something in the kernel. Confirm your prediction with "dmesg > x" and "vi x". The kernel remembers the booting message in the system buffer and dmesg displays the content of this buffer to the screen. "dmesg > x" will send the booting message to file x. With "vi x" you can look at the file x.



부팅할 때 나오는 첫 메시지를 찾기 위해서 main.c 파일에서 printk를 검색해보았다. printk(KERN\_...) 라고 써져있는 것은 숫자이기 때문에 logging level setting에 사용되지 화면에 찍히진 않는다. printk(KERN\_...)이 끝난 바로 다음 printk는 printk(linux\_banner); 가 된다. 따라서 linux\_banner이란 배열 안에 저장돼 있는 것이 첫 메시지이기 때문에 linux-2.6.25.10으로 돌아가 grep을 이용해서 linux\_banner이 어디에 있는지 찾아냈다.

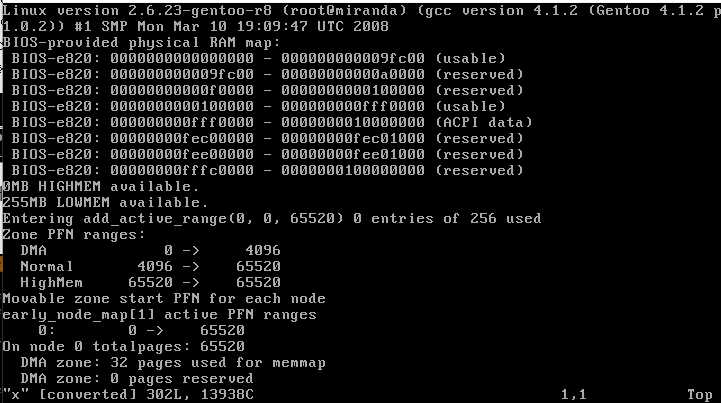


linux-2.6.25.10 에서 grep으로 linux\_banner를 검색해본 결과, version.c 파일 안에 linux\_banner이 정의돼있는걸 발견했다.



version.c 파일을 열어서 linux\_banner를 검색해본 결과, 부팅할 때 첫 메시지는 “Linux version ~~~ (~~@~~) (~~) ~~”의 형태로 된다는걸 알 수 있었다.





예측이 맞는지 확인하기 위해서 dmesg > x를 사용해서 부팅 메시지가 x라는 파일로 가게 한 다음에 vi x를 사용해서 부팅메세지를 보았다.



이 메시지가 부팅 메시지의 첫 줄인데, 나의 예상과 같은 결과가 나온걸 알 수 있다.

5) Find the location of the following functions called in start\_kernel() and briefly explain your guessing about what each function is doing (but do not copy the whole code).

trap\_init(), init\_IRQ(), sched\_init(), time\_init(), console\_init(), mem\_init(),

rest\_init()

arch/xtensa/kernel/traps.c 에 있는 trap\_init 함수를 보았을 때, trap\_init 함수는 주로 벡터, 핸들러 등의 기본값을 세팅하는 역할을 하는 것 같다.

arch/m68k/mac/macints.c 에 있는 mac\_init\_IRQ 함수에 있는 설명을 보면, init\_IRQ 함수는 주로 인터럽트를 세팅하거나 클리어하는 함수인 것 같다.

arch/m68k/mvme147/config.c 에 있는 mvme147\_sched\_init 함수를 보면, sched\_init 함수는 클럭의 세팅과 연관이 돼있다. arch/m68k/mvme16x/config.c의 함수를 봐도 역시 타이머 세팅과 관련있는걸 볼 수 있다.

arch/xtensa/kernel/time.c 의 함수를 보며, time\_init 함수는 리눅스 타이머 인터럽트를 초기화해주는 역할을 하고, arch/s390/kernel/time.c의 함수를 보면 xtime을 설정하는 역할도 한다.

arch/alpha/kernel/setup.c 에 console\_init()에 관한 설명이 있는데, 그 설명에 따르면 console\_init()가 실행되는 동안 console device가 등록된다고 써있다.

arch/powerpc/mm/pgtable\_32.c에 나오는 mem\_init()에 관한 설명에 따르면, mem\_init 함수는 high memory를 세팅한다고 한다.

init/main.c를 봐보면 rest\_init 함수는 부팅 프로세스와 관련이 있는걸 알 수 있다.

6) Why can't we use "printf" instead of "printk" in Linux kernel?

Printf는 C 라이브러리에 있는 함수지만 커널은 C 라이브러리와 연관되지 않는다. 그렇기 때문에 커널에서는 printk를 사용해야만 커널에서 버퍼로 내용을 보내고 버퍼에서 화면으로 내용을 다시 보내게 돼서 화면에 무언가가 찍히게 된다.