
Embedded System Design

실습 3

Cho Yeongpil
Hanyang University



자료 복사

- ~~USB 파일 복사~~

- 블랙보드 다운로드

- 다음 페이지 설명에 따라 각 운영체제에 다운로드할 것을 추천함

- 웹 다운로드

- http://www-personal.umich.edu/~yjunpark/embedded_start_file_2017.zip

자료 복사

- 파일 및 폴더 리스트

- PL2303_Prolific_DriverInstaller (Windows) – [In Windows](#)
- toolchain-s5pc1xx.tar.gz - [In linux machine](#)
- u-boot-1.3.4.tar.gz - [In linux machine](#)
- vpos.bin - [In linux machine](#)
- vpos.zip - [In linux machine](#)

- 웹 다운로드

- 공유 폴더를 이용하여 Linux machine으로 이동시킬 수 있음



USB to Serial Driver 설치

- USB to Serial Driver 설치
- Windows의 경우
 - PL2303_Prolific_DriverInstaller_v1.10.0을 실행하여 설치
- MAC OS의 경우
 - PL-2303 Mac driver 검색 → 다운로드 및 설치
- 설치 후 재부팅



목차

1. U-Boot 소개
2. TFTP, ARM Cross compiler 설치
3. U-Boot 포팅
4. VPOS 부팅
5. 소스코드 디렉토리 및 파일 구조
6. 커널 포팅 준비



U-BOOT 소개



부트로더

- 부트로더(Bootloader)란?

- 시스템 하드웨어를 초기화
- 메모리에 OS 이미지를 적재시키고 OS 시작 루틴으로 분기
- 시스템에 전원이 공급되면 가장 먼저 실행되는 프로그램

- 부트로더의 위치

- 일반적으로 시스템 메모리의 물리 주소 0번지부터 위치
- 롬, 플래시롬, SRAM 등 정적인 메모리에 위치

- 부트로더의 기능

- 메모리 초기화
- 하드웨어 초기화



U-Boot

- **U-Boot란?**

- Universal Bootloader
- PowerPC와 ARM에 기반을 둔 임베디드 보드를 위한 부트로더
- 임베디드 보드에서 리눅스의 부트로더로 많이 사용
 - 데스크톱 리눅스에서는 grub 부트로더를 사용

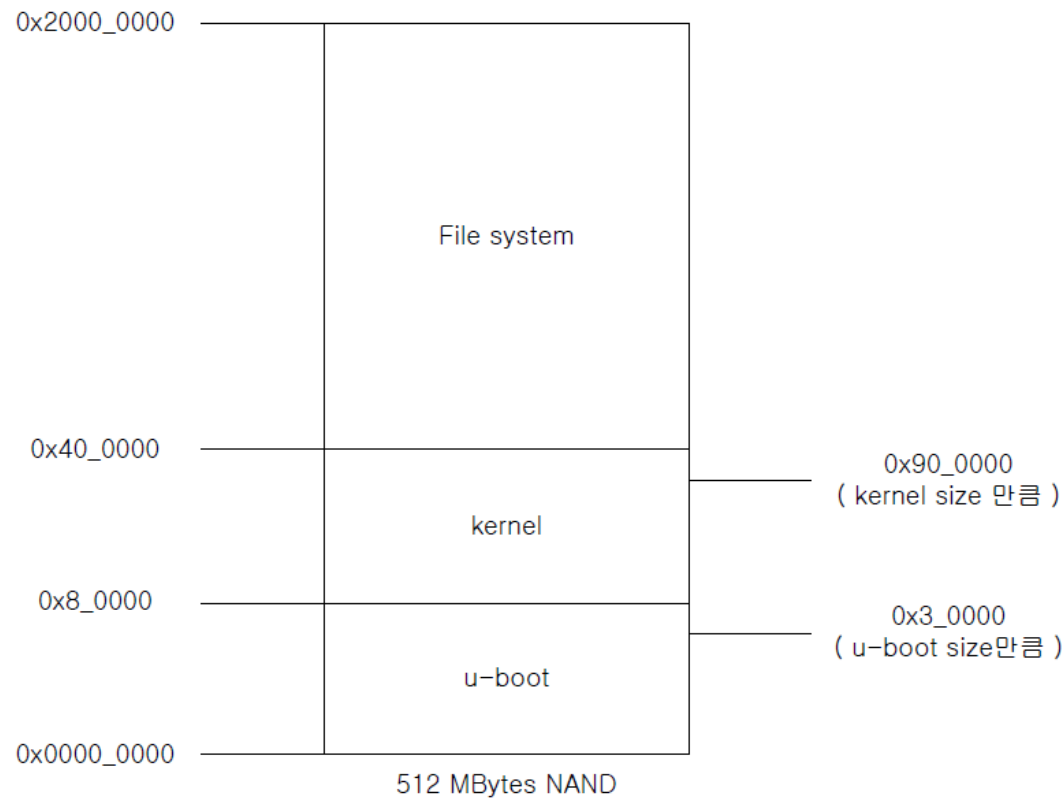
- **장점**

- 다양한 플랫폼에 포팅 가능(ARM, MIPS, x86 등)
- 코드가 깔끔하고 구조가 좋음
- 쉬운 환경설정



NAND Flash에서 U-boot와 커널의 위치

- SYS-LAB II 보드의 NAND Flash Address Map
 - Physical Address 기준



mDDR에서 U-boot와 커널의 위치

- **mDDR이란?**

- Double Data Rate Synchronous Dynamic RAM(SDRAM) for mobile computers
- 기존 DDR SDRAM보다 전력 소모가 적음
- SDRAM이란?
 - CPU의 빠른 버스 속도를 따라잡기 위해 만들어진 DRAM

- **U-boot**

- Physical address : 0x27e00000
- Virtual address : 0xc7e00000

- **Kernel**

- Physical address : 0x20008000
- Virtual address : 0xc0008000



TFTP, ARM CROSS COMPILER 설치



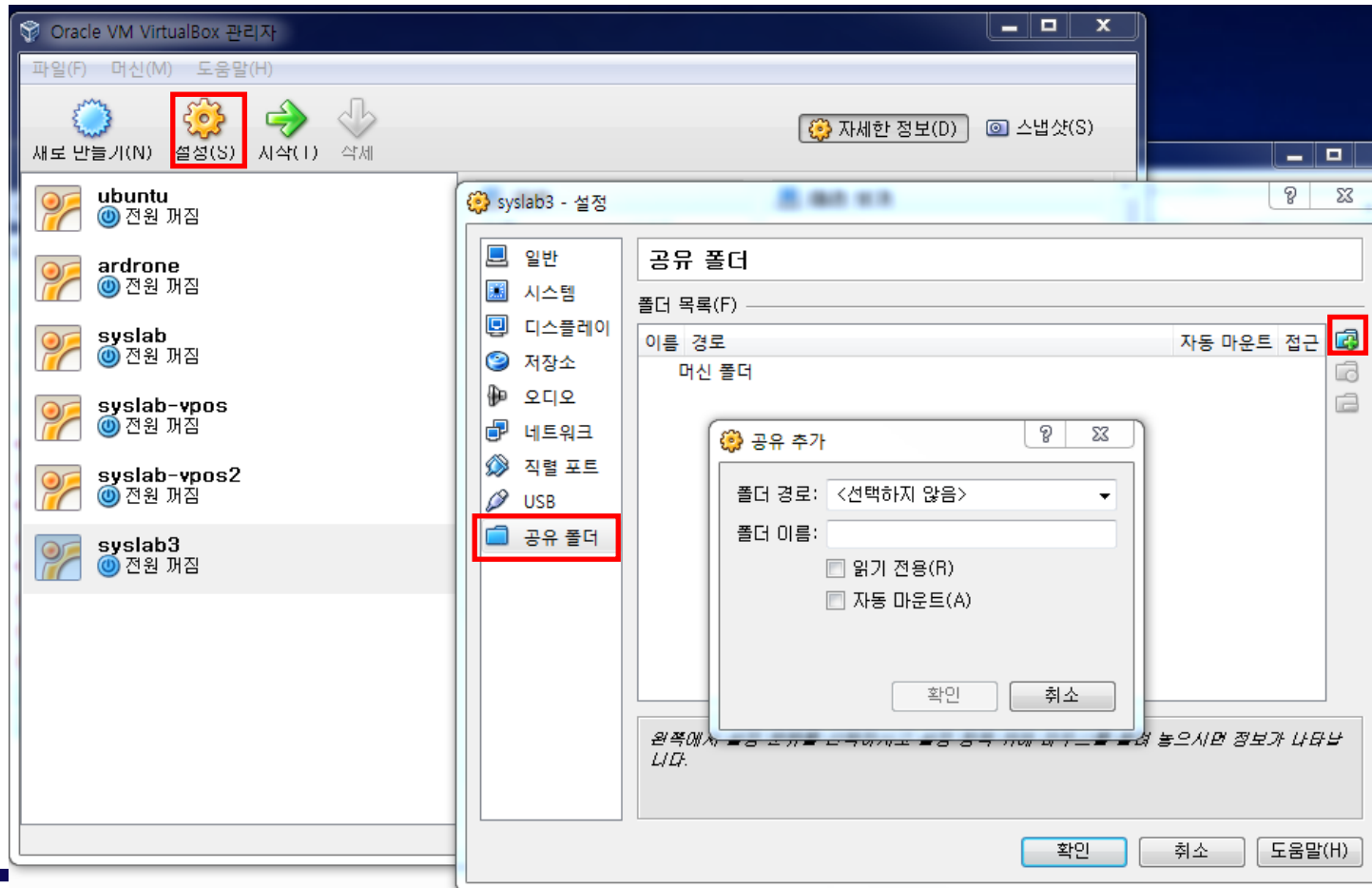
관련 파일

- **toolchain-s5pc1xx.tar.gz**
- **u-boot-1.3.4.tar.gz**
- **다운로드 옵션**
 - ~~— USB를 통해 직접 복사~~
 - Linux 머신에서 직접 다운로드
 - Windows에서 다운받은 뒤 공유폴더를 통해 공유
 - Optional



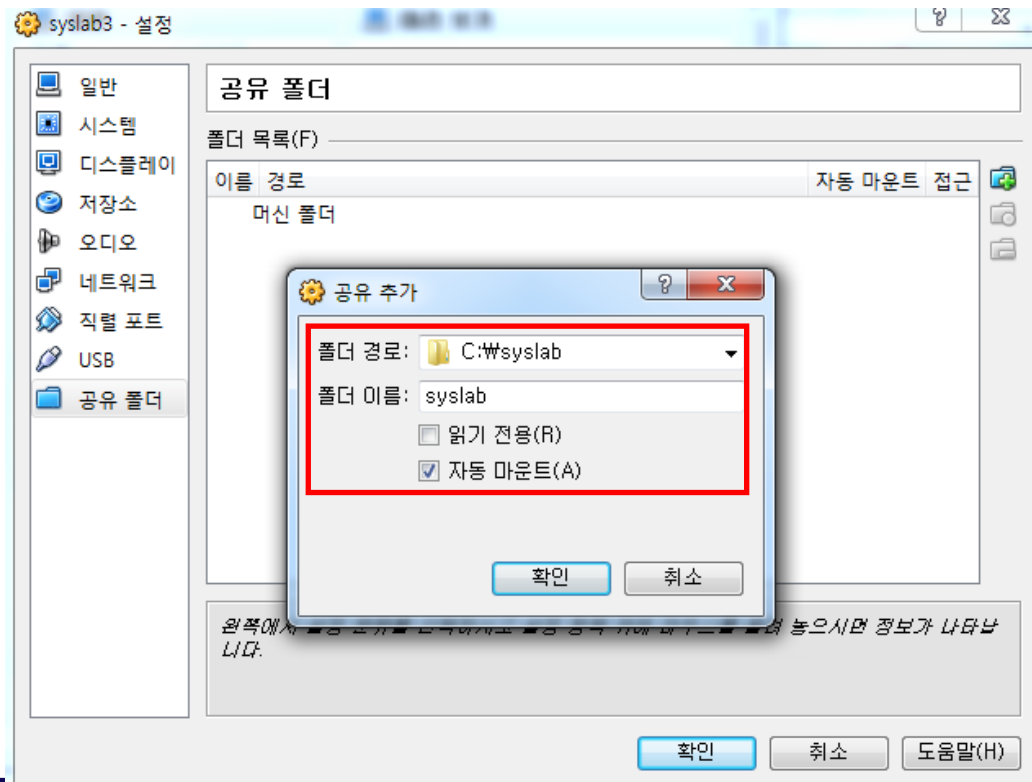
공유폴더 설정 (VirtualBox)

- 설정 → 공유 폴더



공유폴더 설정 (VirtualBox)

1. 폴더 경로 : '기타' 선택
2. c드라이브의 'syslab' 폴더 선택
3. 자동 마운트에 체크



공유폴더 확인(VirtualBox)

- Virtual Linux에서 공유할 폴더 생성

>> mkdir /home/hanyang/syslab

>> mount -t vboxsf syslab /home/hanyang/syslab

- mount -t vboxsf [윈도우 폴더] [리눅스 폴더]

```
root@hanyang-desktop: /home/hanyang# mkdir /home/hanyang/syslab
root@hanyang-desktop: /home/hanyang# mount -t vboxsf syslab /home/hanyang/syslab
root@hanyang-desktop: /home/hanyang#
```

- 공유 폴더 확인

>> ls /home/hanyang/syslab

```
root@hanyang-desktop: /home/hanyang# cd /home/hanyang/syslab/
root@hanyang-desktop: /home/hanyang/syslab# ls
Boot Binary Images      record_replay_temp_0204
Boot Loader_Kernel_Source  record_replay_temp_0206
QURIX-R                 record_replay_temp_0211
Root_File_System        record_replay_temp_0212
```

공유폴더 확인(VirtualBox)

- 자동으로 마운트하기

>> vi .bashrc

- 문서 마지막 줄에 “su” “mount -t vboxsf syslab/ /home/hanyang/syslab” 추가

```
alias la='ls -A'
alias l='ls -CF'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
    . /etc/bash_completion
fi

su
mount -t vboxsf syslab/ /home/hanyang/syslab/
```

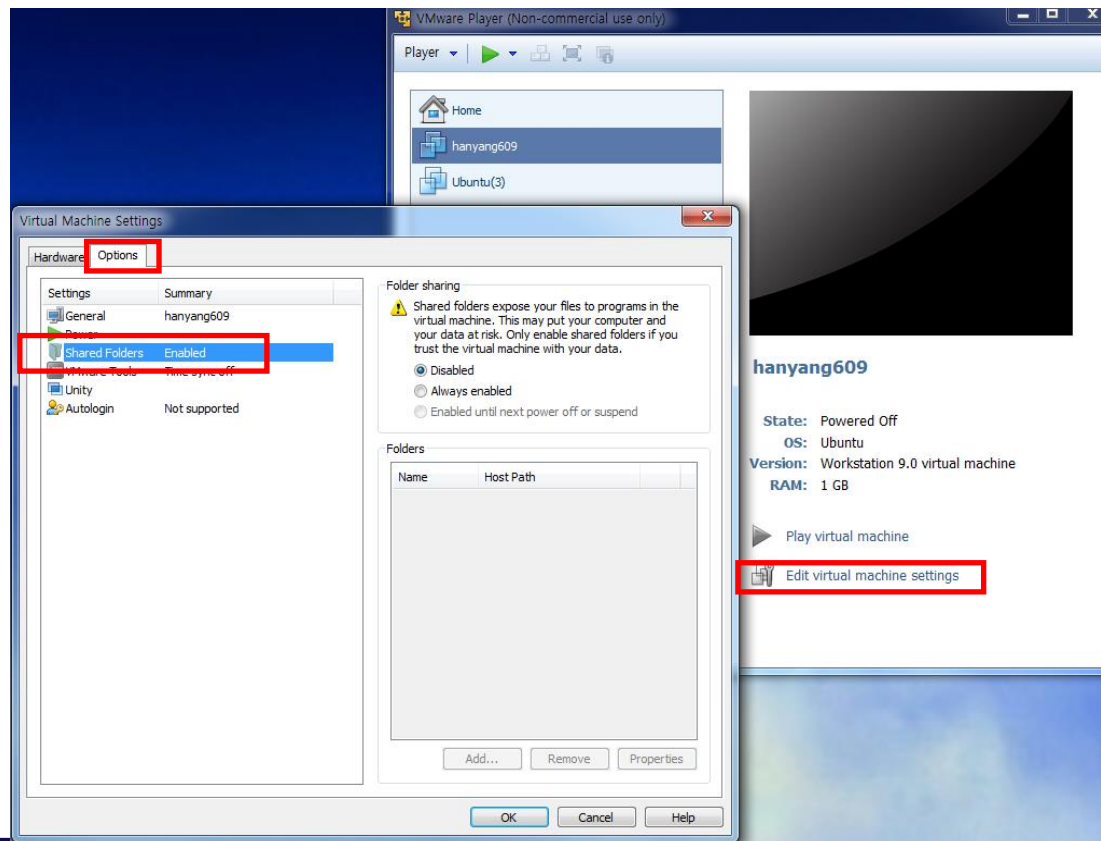
102, 1

바닥



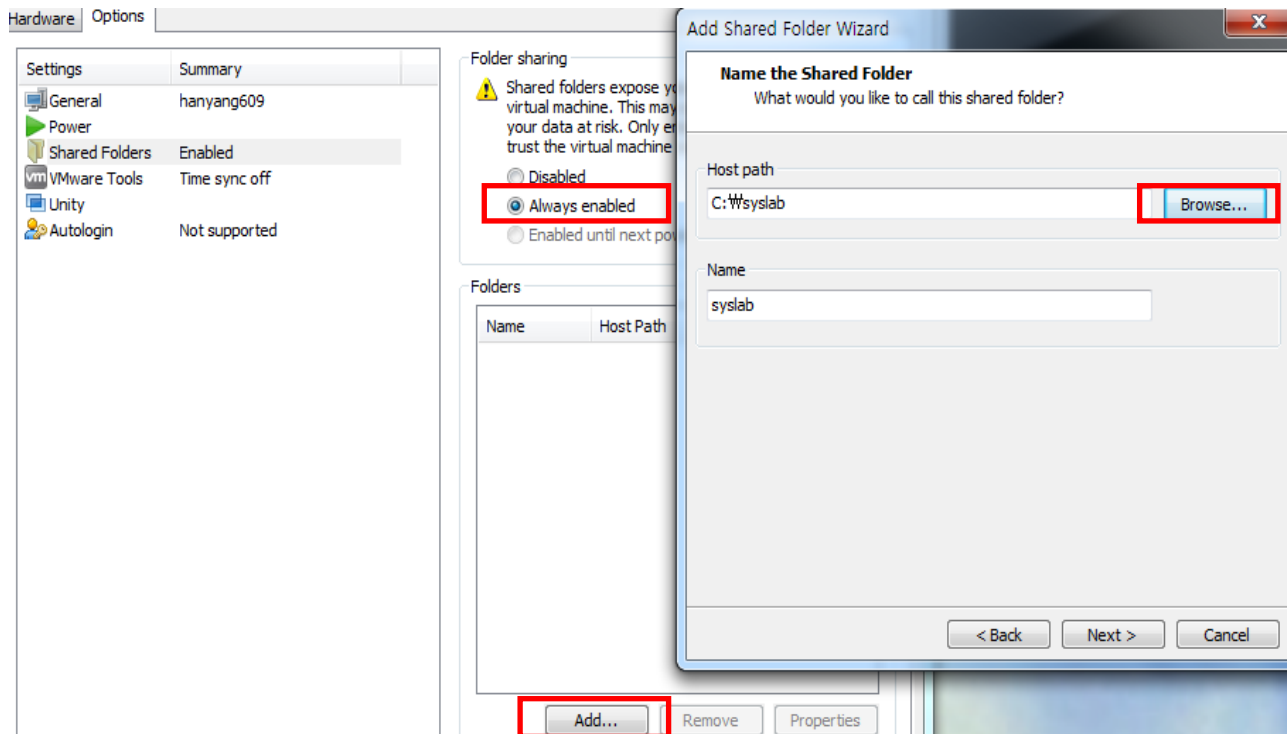
공유폴더 설정 (VMware)

1. 'Edit virtual machine settings' 클릭
2. Option 탭에서 Shared Folders를 클릭



공유폴더 설정 (VMware)

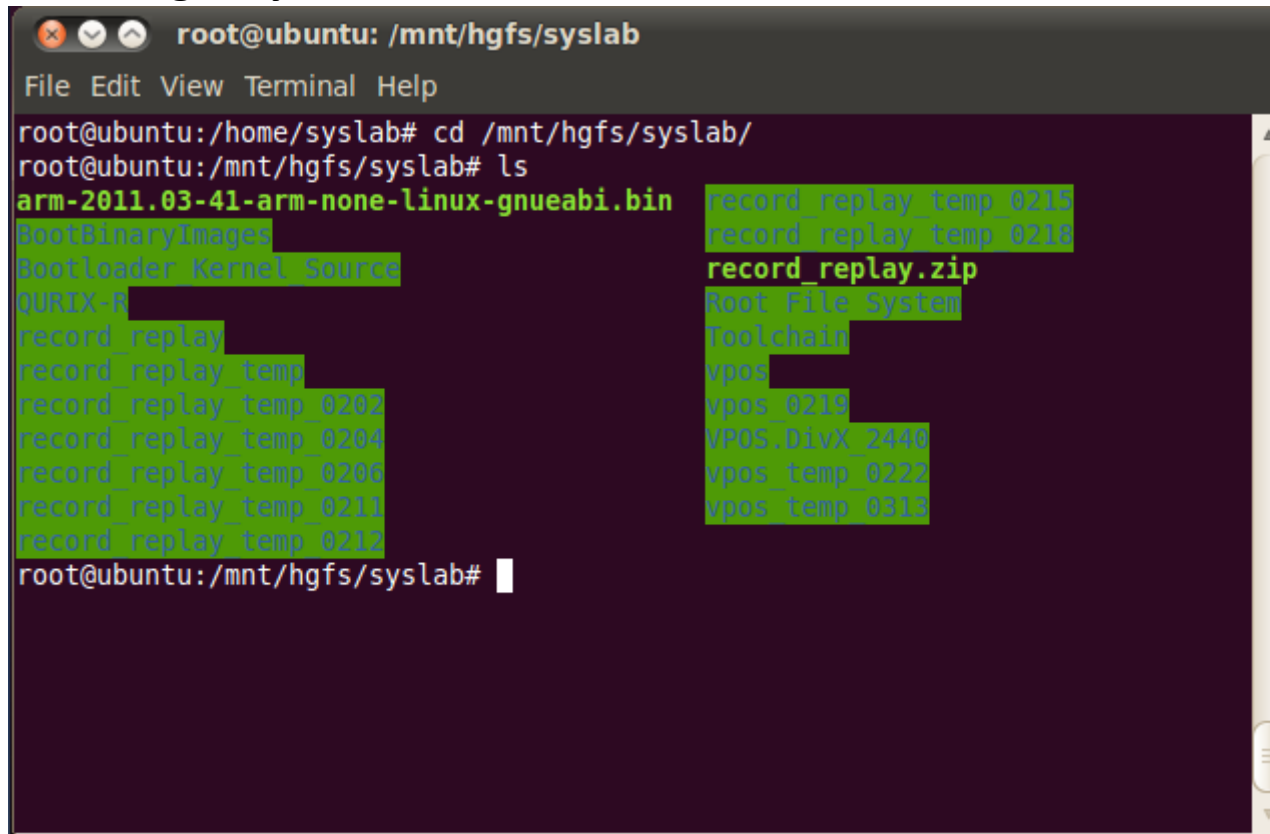
1. Always enabled를 선택
2. Add... 버튼을 클릭하고 Browse에서 'syslab' 폴더 선택



공유 폴더 확인(VMware)

- 공유폴더 위치 : /mnt/hgfs/[공유폴더]

>> ls /mnt/hgfs/syslab



A terminal window titled 'root@ubuntu: /mnt/hgfs/syslab' showing the output of the 'ls' command. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', and 'Help'. The terminal text is as follows:

```
root@ubuntu:/home/syslab# cd /mnt/hgfs/syslab/
root@ubuntu:/mnt/hgfs/syslab# ls
arm-2011.03-41-arm-none-linux-gnueabi.bin  record_replay_temp_0215
BootBinaryImages                          record_replay_temp_0218
Bootloader Kernel_Source                  record_replay.zip
QURIX-R                                   Root File System
record_replay                             Toolchain
record_replay_temp                        vpos
record_replay_temp_0202                   vpos_0219
record_replay_temp_0204                   VP0S.DivX 2440
record_replay_temp_0206                   vpos_temp_0222
record_replay_temp_0211                   vpos_temp_0313
record_replay_temp_0212
root@ubuntu:/mnt/hgfs/syslab#
```



업데이트 파일 받아오기

>> sudo apt-get update

>> sudo apt-get upgrade

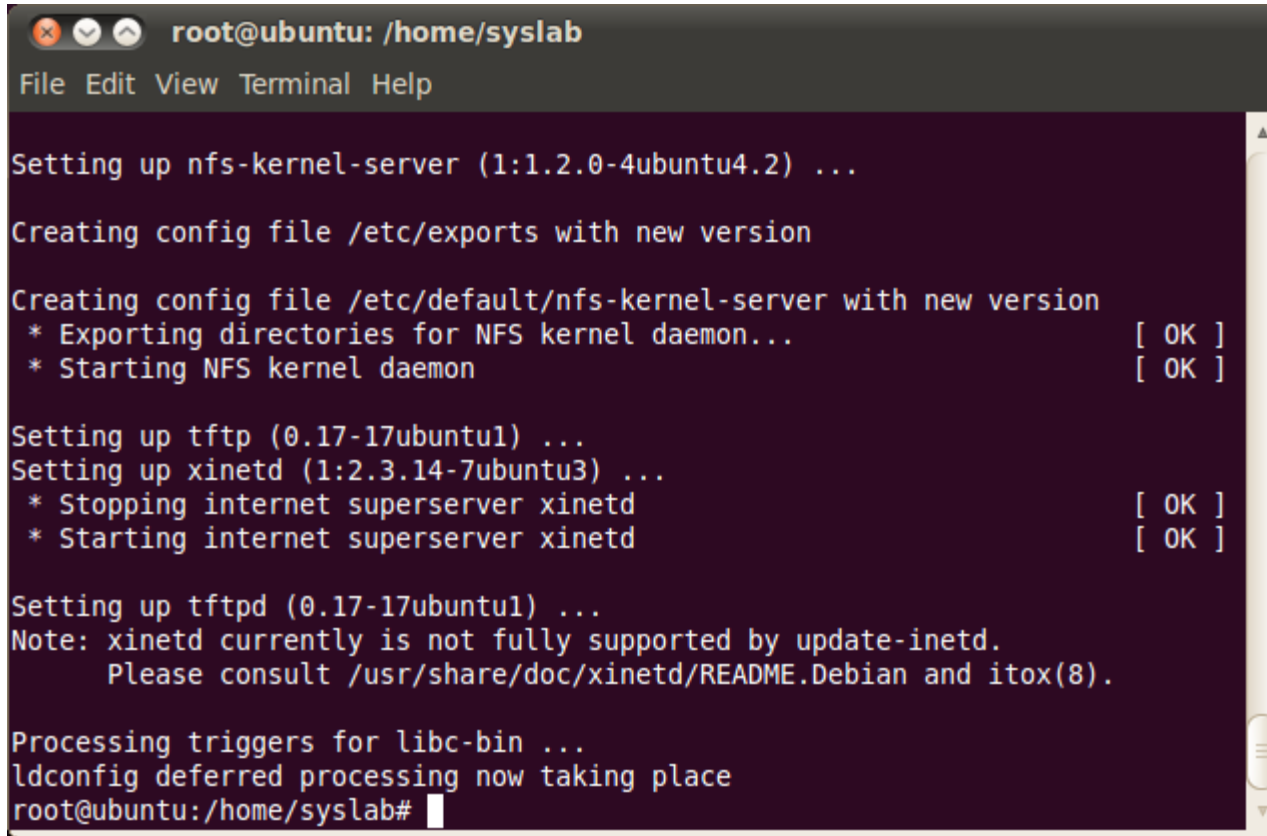
```
root@ubuntu: /home/syslab
File Edit View Terminal Help
Hit http://us.archive.ubuntu.com lucid/multiverse Sources
Get:6 http://us.archive.ubuntu.com lucid-updates/main Packages [670kB]
Get:7 http://security.ubuntu.com lucid-security/restricted Packages [2,867B]
Get:8 http://security.ubuntu.com lucid-security/main Sources [135kB]
Get:9 http://security.ubuntu.com lucid-security/restricted Sources [1,267B]
Get:10 http://security.ubuntu.com lucid-security/universe Packages [143kB]
Get:11 http://security.ubuntu.com lucid-security/universe Sources [44.5kB]
Get:12 http://security.ubuntu.com lucid-security/multiverse Packages [5,363B]
Get:13 http://security.ubuntu.com lucid-security/multiverse Sources [2,351B]
Get:14 http://us.archive.ubuntu.com lucid-updates/restricted Packages [4,630B]
]
Get:15 http://us.archive.ubuntu.com lucid-updates/main Sources [234kB]
Get:16 http://us.archive.ubuntu.com lucid-updates/restricted Sources [2,196B]
Get:17 http://us.archive.ubuntu.com lucid-updates/universe Packages [291kB]
Get:18 http://us.archive.ubuntu.com lucid-updates/universe Sources [108kB]
Get:19 http://us.archive.ubuntu.com lucid-updates/multiverse Packages [11.5kB]
]
Get:20 http://us.archive.ubuntu.com lucid-updates/multiverse Sources [5,819B]
Fetched 2,255kB in 8s (280kB/s)
Reading package lists... Done
root@ubuntu:/home/syslab#
```



리눅스 기반 개발환경 설치 및 설정

- Package 설치

- >> apt-get install nfs-kernel-server tftp tftpd xinetd



```
root@ubuntu: /home/syslab
File Edit View Terminal Help

Setting up nfs-kernel-server (1:1.2.0-4ubuntu4.2) ...
Creating config file /etc/exports with new version
Creating config file /etc/default/nfs-kernel-server with new version
* Exporting directories for NFS kernel daemon... [ OK ]
* Starting NFS kernel daemon [ OK ]

Setting up tftp (0.17-17ubuntu1) ...
Setting up xinetd (1:2.3.14-7ubuntu3) ...
* Stopping internet superserver xinetd [ OK ]
* Starting internet superserver xinetd [ OK ]

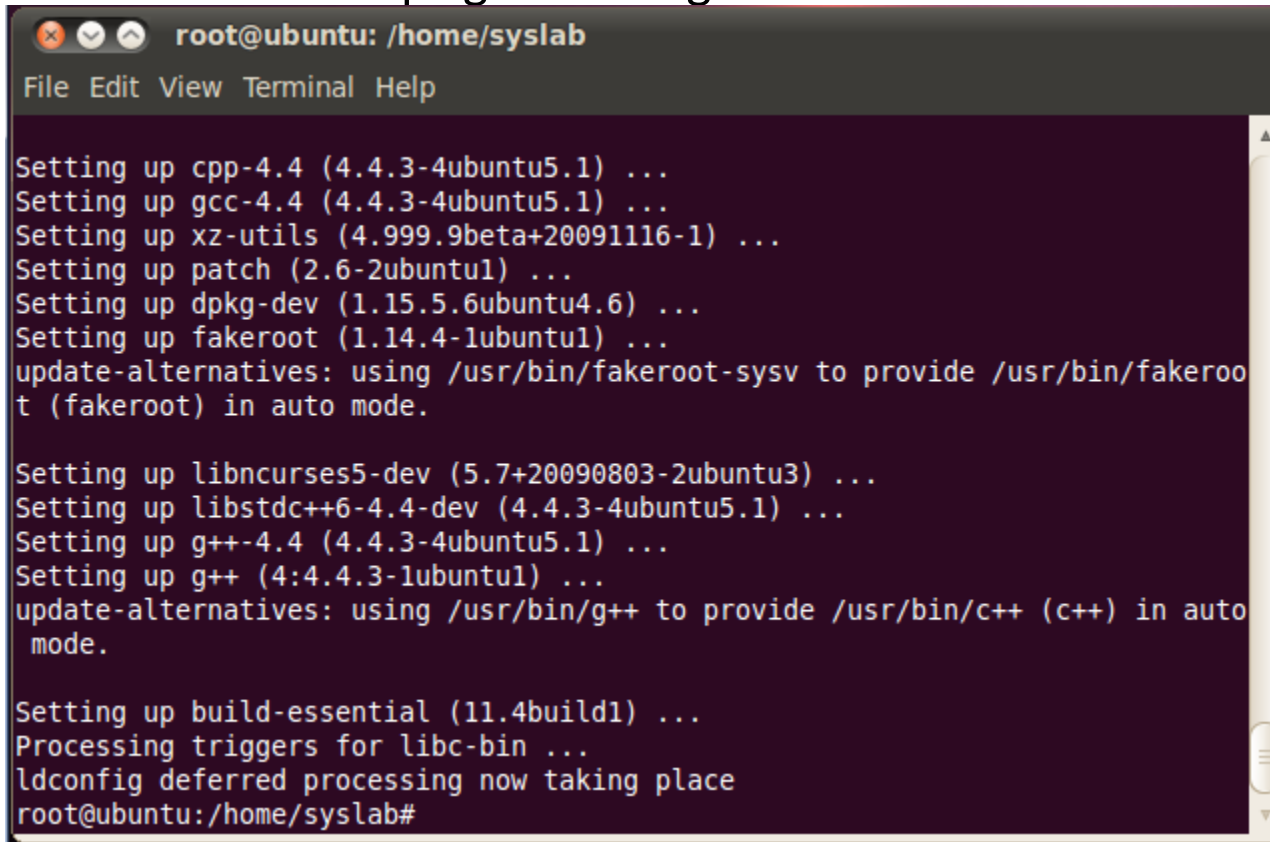
Setting up tftpd (0.17-17ubuntu1) ...
Note: xinetd currently is not fully supported by update-inetd.
Please consult /usr/share/doc/xinetd/README.Debian and itox(8).

Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
root@ubuntu:/home/syslab#
```



리눅스 기반 개발환경 설치 및 설정

- >> apt-get install libncurses5 libncurses5-dev build-essential
- For 64bit OS? >> apt-get install gcc-multilib



```
root@ubuntu: /home/syslab
File Edit View Terminal Help

Setting up cpp-4.4 (4.4.3-4ubuntu5.1) ...
Setting up gcc-4.4 (4.4.3-4ubuntu5.1) ...
Setting up xz-utils (4.999.9beta+20091116-1) ...
Setting up patch (2.6-2ubuntu1) ...
Setting up dpkg-dev (1.15.5.6ubuntu4.6) ...
Setting up fakeroot (1.14.4-1ubuntu1) ...
update-alternatives: using /usr/bin/fakeroot-sysv to provide /usr/bin/fakeroot (fakeroot) in auto mode.

Setting up libncurses5-dev (5.7+20090803-2ubuntu3) ...
Setting up libstdc++6-4.4-dev (4.4.3-4ubuntu5.1) ...
Setting up g++-4.4 (4.4.3-4ubuntu5.1) ...
Setting up g++ (4:4.4.3-1ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode.

Setting up build-essential (11.4build1) ...
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
root@ubuntu:/home/syslab#
```



TFTP

- TFTP란?

- Trivial File Transfer Protocol
- FTP처럼 파일을 전송하기 위한 프로토콜
 - FTP보다 더 단순한 방식으로 전송
- 임베디드 시스템에서 운영체제 업로드로 주로 사용됨

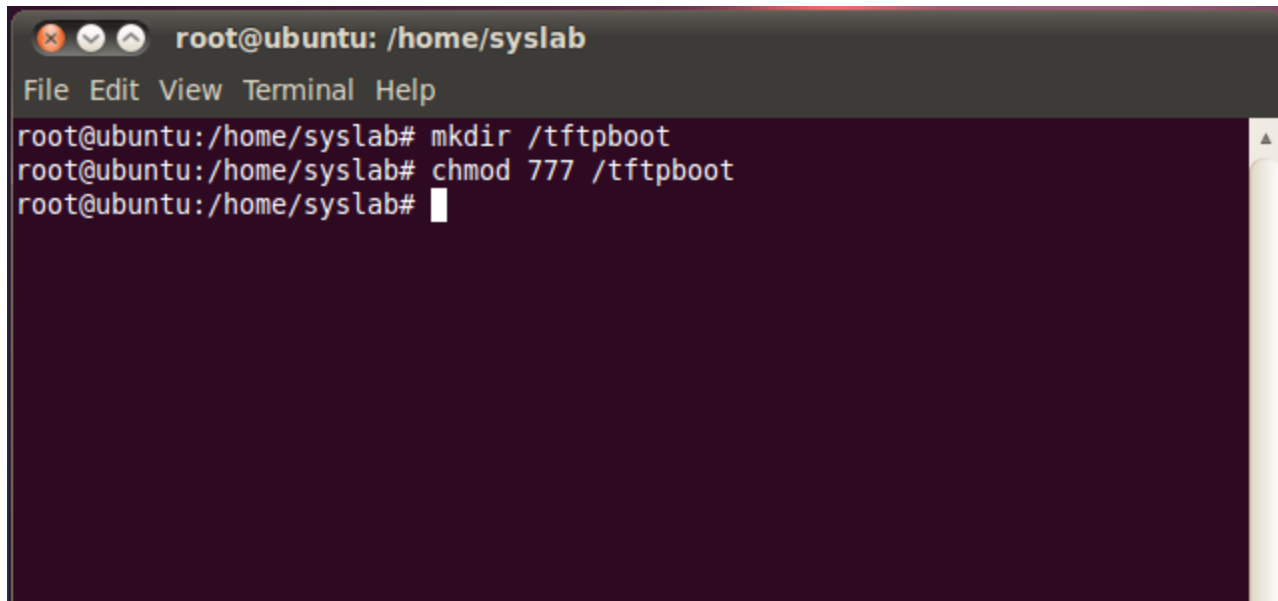
+ 구현이 간단

- 데이터 전송 중 데이터가 손실될 수 있음

TFTP 설정

- TFTP 설정

- tftpboot 폴더 만들기
 - >> `mkdir /tftpboot`
- tftpboot 폴더 권한 설정
 - >> `chmod 777 /tftpboot`

A terminal window titled 'root@ubuntu: /home/syslab' with a menu bar 'File Edit View Terminal Help'. The terminal shows three commands being executed: 'mkdir /tftpboot', 'chmod 777 /tftpboot', and a prompt 'root@ubuntu:/home/syslab#' with a cursor.

```
root@ubuntu: /home/syslab
File Edit View Terminal Help
root@ubuntu:/home/syslab# mkdir /tftpboot
root@ubuntu:/home/syslab# chmod 777 /tftpboot
root@ubuntu:/home/syslab#
```

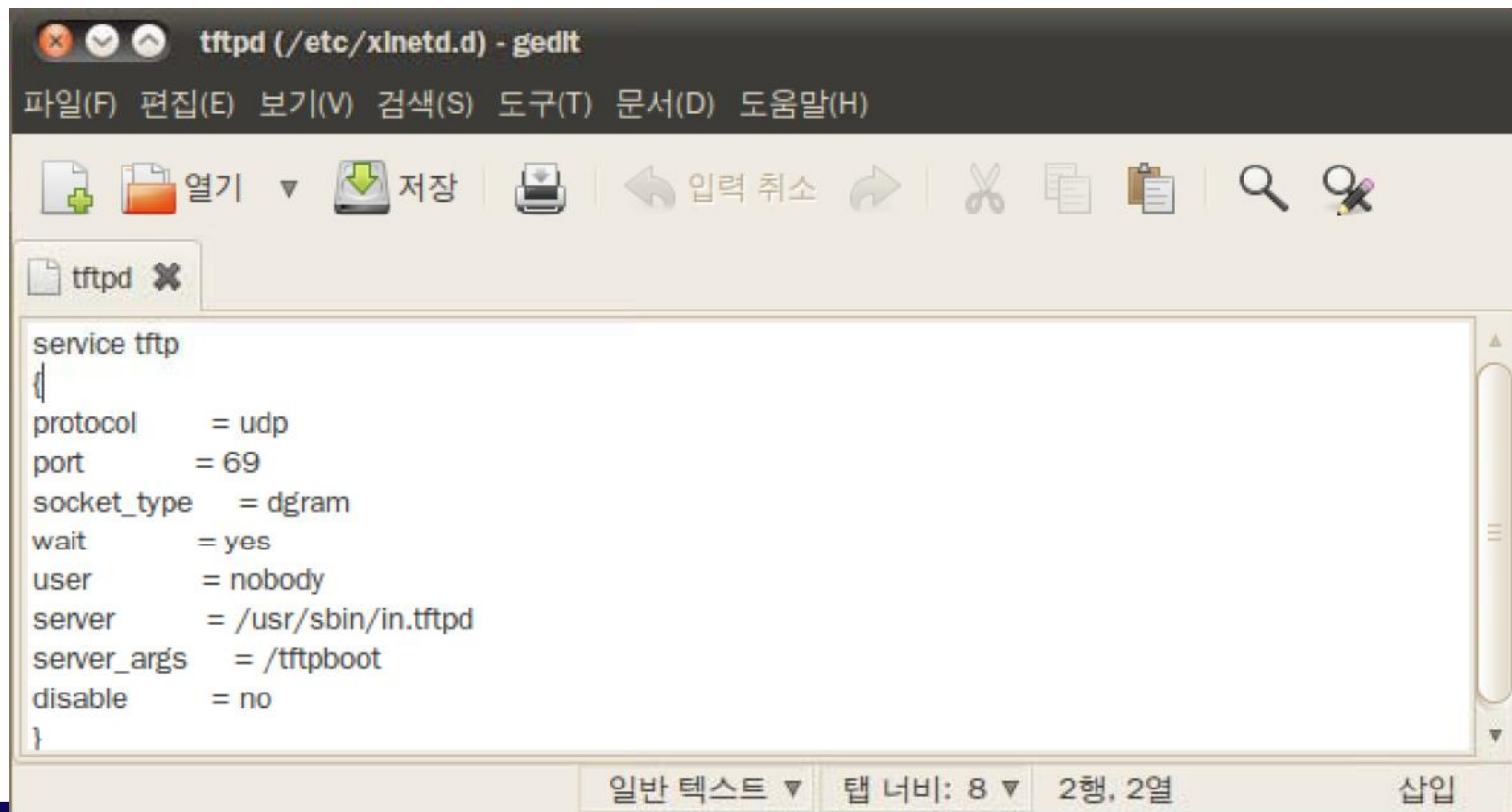


TFTP 설정

- TFTP 환경 설정

>> gedit /etc/xinetd.d/tftpd

- 아래 그림과 같이 작성 후 저장

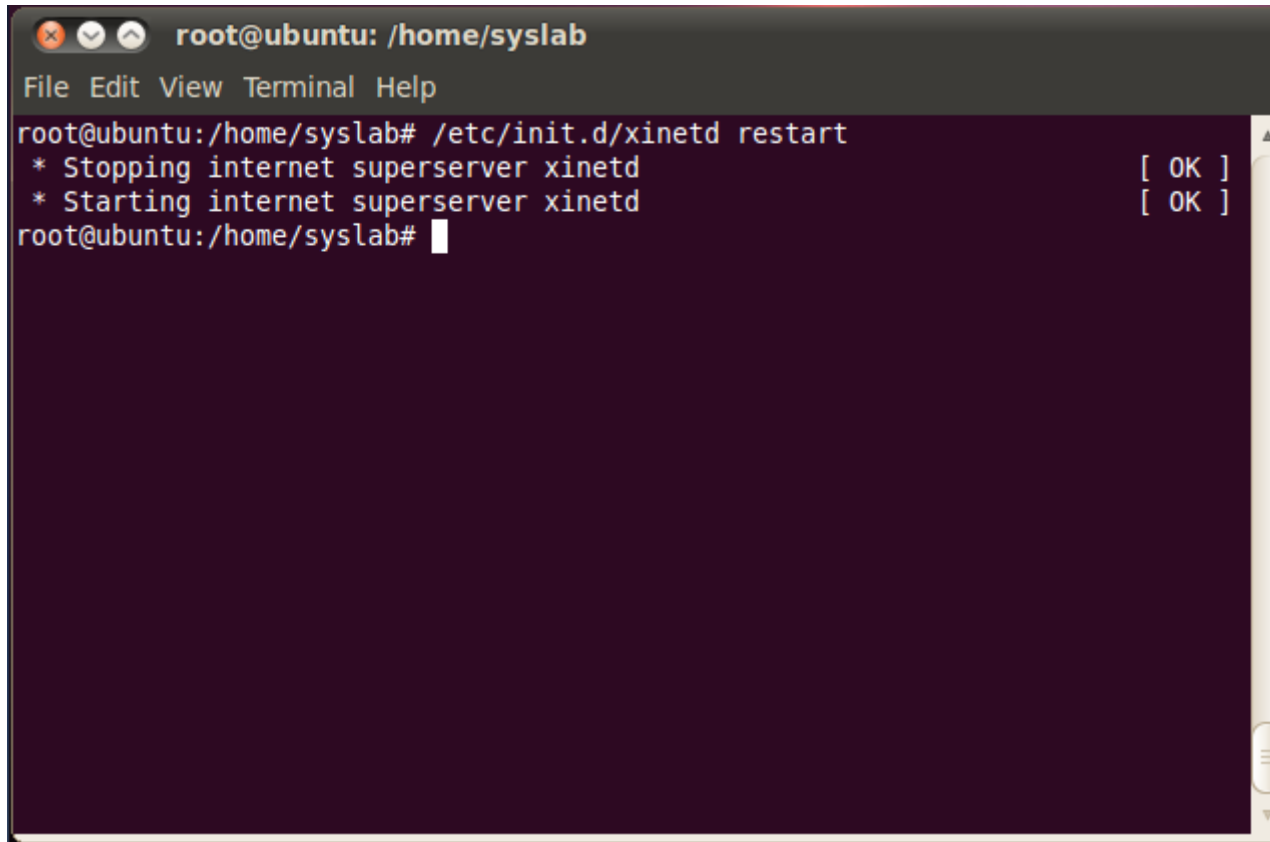


```
service tftp
{
protocol      = udp
port         = 69
socket_type  = dgram
wait         = yes
user         = nobody
server       = /usr/sbin/in.tftpd
server_args  = /tftpboot
disable      = no
}
```

TFTP 설정

- Xinetd 재실행

>> /etc/init.d/xinetd restart

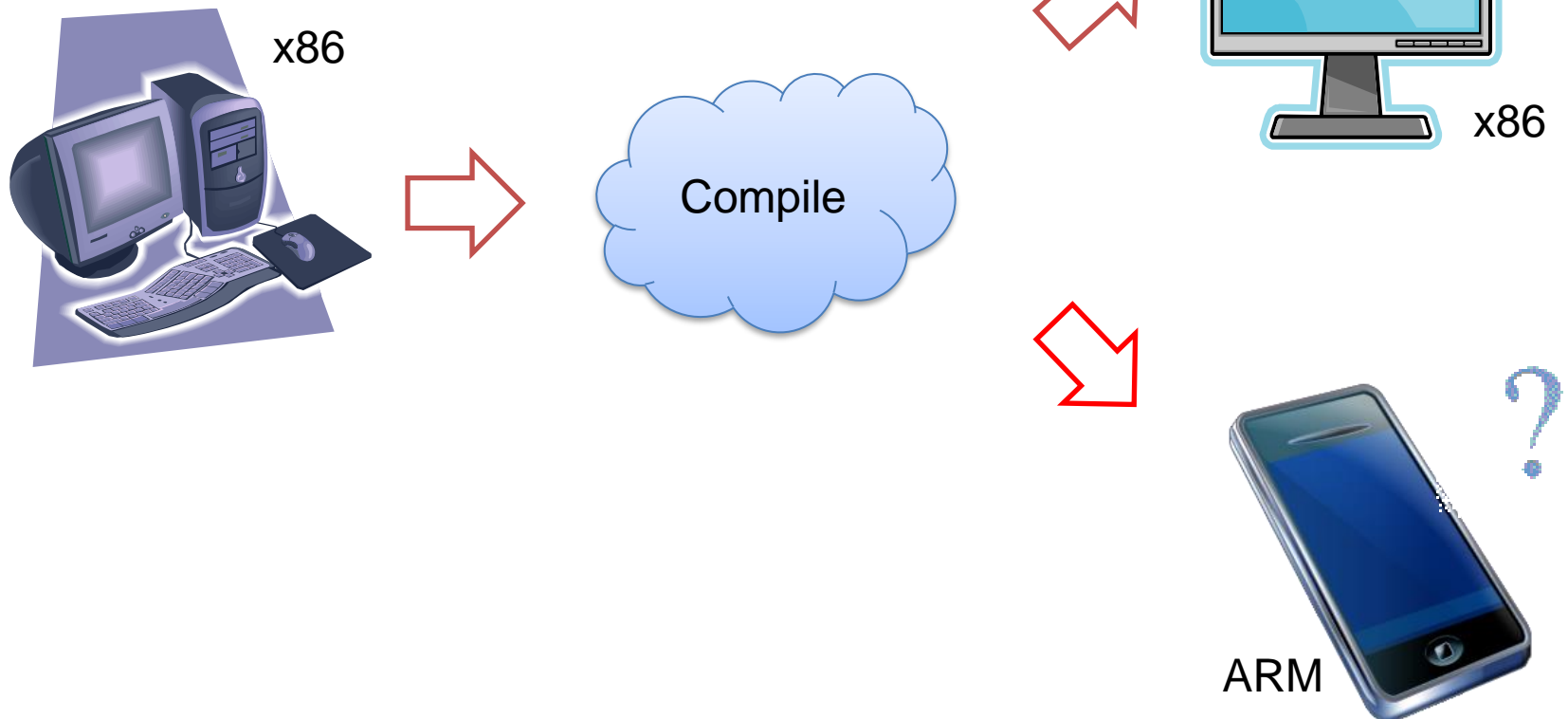


```
root@ubuntu: /home/syslab
File Edit View Terminal Help
root@ubuntu:/home/syslab# /etc/init.d/xinetd restart
* Stopping internet superserver xinetd [ OK ]
* Starting internet superserver xinetd [ OK ]
root@ubuntu:/home/syslab#
```



크로스 컴파일러(Cross Compiler)

- 크로스 컴파일러의 필요성



크로스 컴파일러(Cross Compiler)

- **크로스 컴파일러의 필요성**

- 통상의 컴퓨터(x86)에서는 컴파일과 실행을 동일한 기계(x86)로 함
- 임베디드 시스템(ARM)에서는 컴파일을 호스트 컴퓨터(x86)에서 함
 - 일반적인 컴파일러를 쓸 경우 x86의 기계어로 번역되어 ARM에서 실행할 수 없음

- **크로스 컴파일러란?**

- 프로그램을 컴파일러가 수행되고 있는 컴퓨터의 기계어로 번역하는 것이 아니라, 다른 기종에 맞는 기계어로 번역하는 컴파일
- Ex) x86 환경에서 ARM 프로세서의 기계어로 번역

- **Toolchain**

- 원하는 임베디드 시스템의 소프트웨어 개발을 하기 위한 호스트 컴퓨터의 컴파일 환경
- 크로스 컴파일러 포함



ToolChain 설치

- ToolChain 설치

- 제공 받은 ToolChain 복사

```
>> cd /opt
```

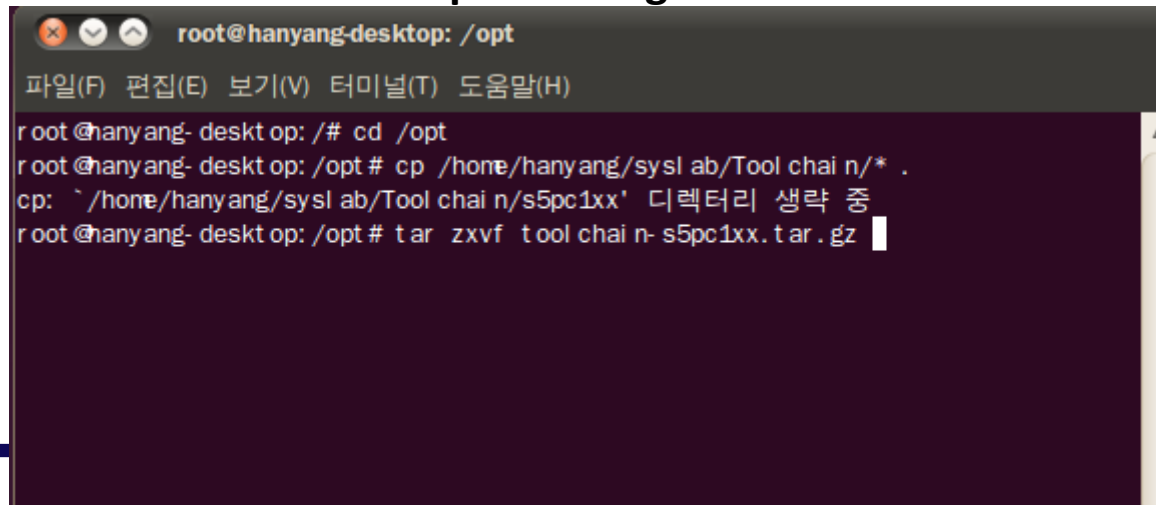
```
>> cp /home/hanyang/syslab/toolchain-s5pc1xx.tar.gz ./ (VirtualBox)
```

```
>> cp /mnt/hgfs/syslab/toolchain-s5pc1xx.tar.gz ./ (Vmware)
```

```
>> cp /home/($username)/Download/toolchain ./
```

- 압축 풀기

```
>> tar zxvf toolchain-s5pc1xx.tar.gz
```



```
root@hanyang-desktop: /opt
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
root@hanyang-desktop: /# cd /opt
root@hanyang-desktop: /opt # cp /home/hanyang/syslab/Toolchain/* .
cp: `/home/hanyang/syslab/Toolchain/s5pc1xx' 디렉터리 생략 중
root@hanyang-desktop: /opt # tar zxvf toolchain-s5pc1xx.tar.gz
```

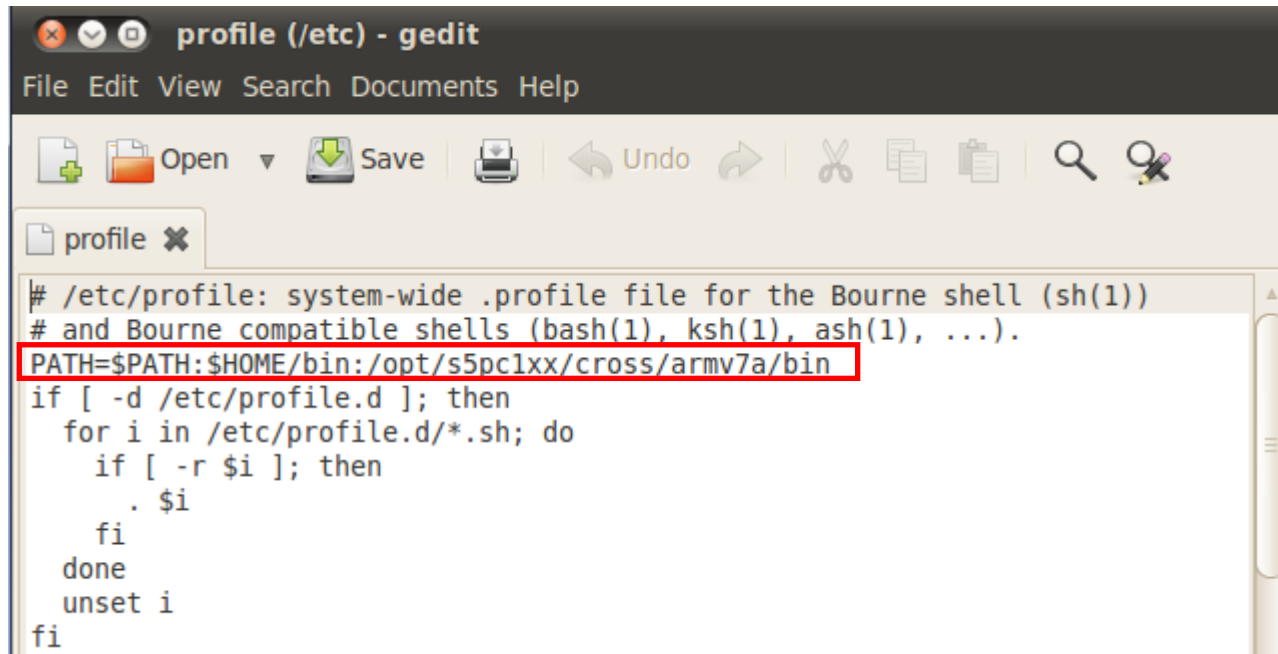


ToolChain 설치

- 크로스컴파일러 환경 변수 설정

>> gedit /etc/profile

- PATH=\$PATH:\$HOME/bin:/opt/s5pc1xx/cross/armv7a/bin 를 추가



```
profile (/etc) - gedit
File Edit View Search Documents Help

profile x
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).
PATH=$PATH:$HOME/bin:/opt/s5pc1xx/cross/armv7a/bin
if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
```

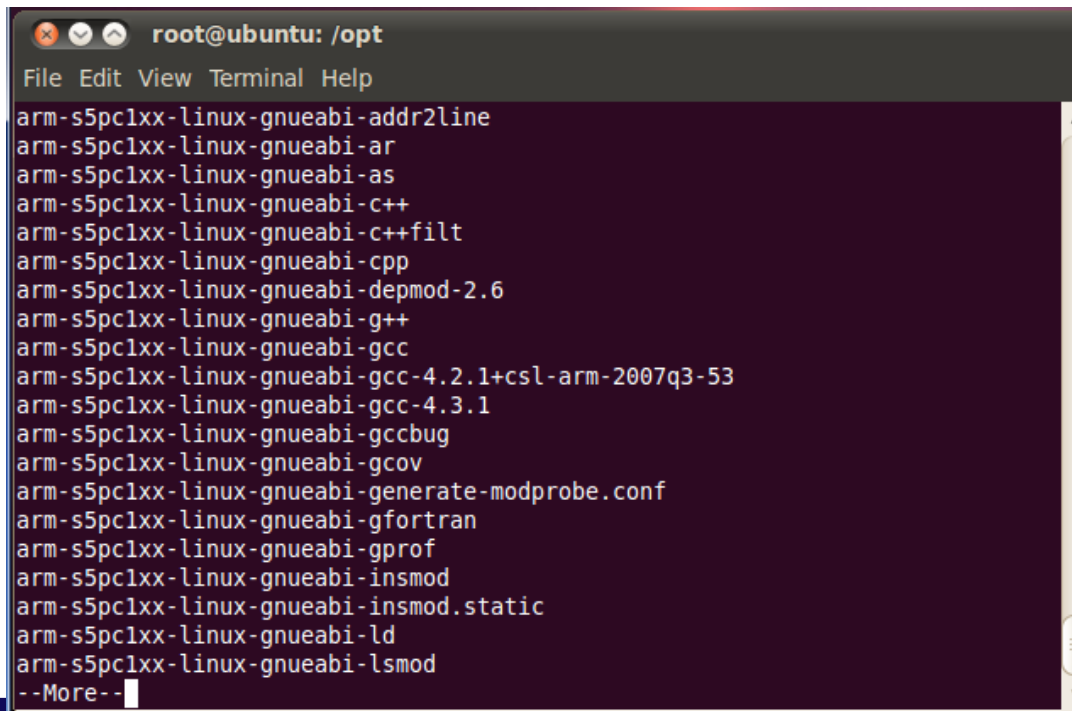
ToolChain 설치

- 수정된 사항을 적용

>> source /etc/profile

- 크로스 컴파일러가 적용이 되었는지 확인

- 셸에 'arm'을 입력하고 Tap키를 2번 누름



```
root@ubuntu: /opt
File Edit View Terminal Help
arm-s5pc1xx-linux-gnueabi-addr2line
arm-s5pc1xx-linux-gnueabi-ar
arm-s5pc1xx-linux-gnueabi-as
arm-s5pc1xx-linux-gnueabi-c++
arm-s5pc1xx-linux-gnueabi-c++filt
arm-s5pc1xx-linux-gnueabi-cpp
arm-s5pc1xx-linux-gnueabi-depmod-2.6
arm-s5pc1xx-linux-gnueabi-g++
arm-s5pc1xx-linux-gnueabi-gcc
arm-s5pc1xx-linux-gnueabi-gcc-4.2.1+csl-arm-2007q3-53
arm-s5pc1xx-linux-gnueabi-gcc-4.3.1
arm-s5pc1xx-linux-gnueabi-gccbug
arm-s5pc1xx-linux-gnueabi-gcov
arm-s5pc1xx-linux-gnueabi-generate-modprobe.conf
arm-s5pc1xx-linux-gnueabi-gfortran
arm-s5pc1xx-linux-gnueabi-gprof
arm-s5pc1xx-linux-gnueabi-insmod
arm-s5pc1xx-linux-gnueabi-insmod.static
arm-s5pc1xx-linux-gnueabi-ld
arm-s5pc1xx-linux-gnueabi-lsmod
--More--
```



U-BOOT 포팅



U-boot 컴파일

- U-boot 파일을 복사 후 압축 풀기

```
>> cp /home/hanyang/syslab/u-boot-1.3.4.tar.gz /home/($username)/  
      (VirtualBox)
```

```
>> cp /mnt/hgfs/syslab/u-boot-1.3.4.tar.gz /home/($username)/  
      (VMware)
```

```
>> cp /home/($username)/Download/u-boot-1.3.4.tar.gz /home/hanyang/  
      (Blackboard download)
```

```
>> cd /home/hanyang
```

```
>> tar zxvf u-boot-1.3.4.tar.gz
```

```
u-boot-1.3.4/libfdt/fdt.c  
u-boot-1.3.4/rules.mk  
u-boot-1.3.4/lib_m68k/  
u-boot-1.3.4/lib_m68k/time.c  
u-boot-1.3.4/lib_m68k/cache.c  
u-boot-1.3.4/lib_m68k/traps.c  
u-boot-1.3.4/lib_m68k/Makefile  
u-boot-1.3.4/lib_m68k/board.c  
u-boot-1.3.4/lib_m68k/bootm.c  
u-boot-1.3.4/lib_m68k/interrupts.c  
u-boot-1.3.4/ppc_config.mk  
u-boot-1.3.4/mips_config.mk  
u-boot-1.3.4/CHANGELOG-before-U-Boot-1.1.5  
u-boot-1.3.4/u-boot.srec  
root@ubuntu:/home/syslab#
```

U-boot 컴파일

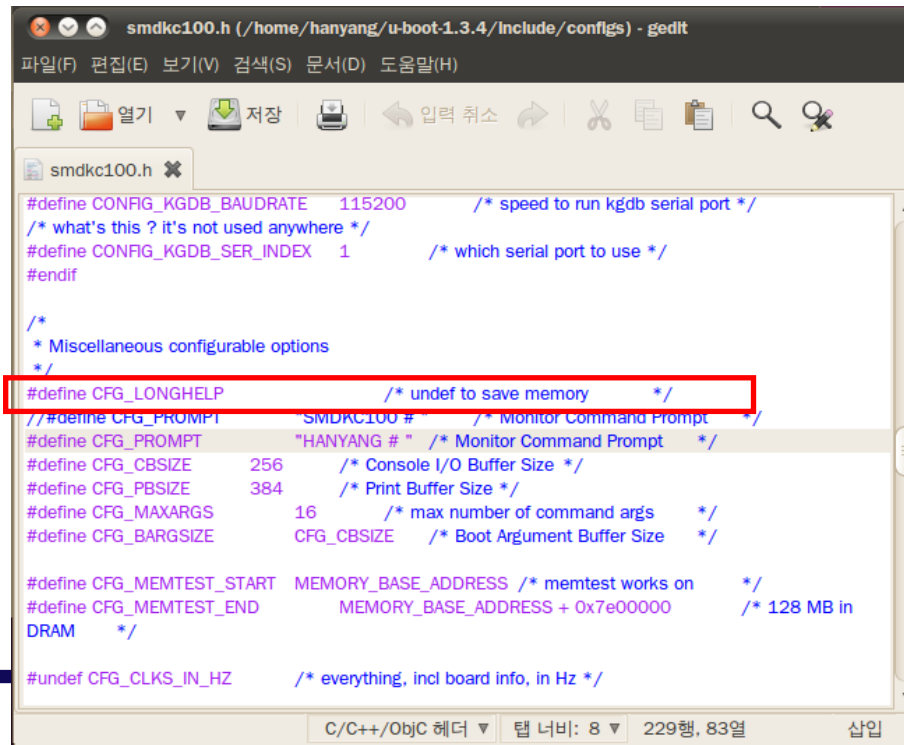
- U-boot가 제대로 설치되었는지 확인하기 위해 코드를 수정

>> gedit /home/hanyang/u-boot-1.3.4/include/configs/smdkc100.h

- 229번째 줄 CFG_PROMPT의 문자열을 수정

define CFG_PROMPT "SL2_C100 #" → "Student_ID #"

ex) 2019XXXXXX



```
smdkc100.h (/home/hanyang/u-boot-1.3.4/include/configs) - gedit
파일(F) 편집(E) 보기(V) 검색(S) 문서(D) 도움말(H)

smdkc100.h
#define CONFIG_KGDB_BAUDRATE 115200 /* speed to run kgdb serial port */
/* what's this ? it's not used anywhere */
#define CONFIG_KGDB_SER_INDEX 1 /* which serial port to use */
#endif

/*
 * Miscellaneous configurable options
 */
#define CFG_LONGHELP /* undef to save memory */
// #define CFG_PROMPT "SMDKC100 #" /* Monitor Command Prompt */
#define CFG_PROMPT "HANYANG #" /* Monitor Command Prompt */
#define CFG_CBSIZE 256 /* Console I/O Buffer Size */
#define CFG_PBSIZE 384 /* Print Buffer Size */
#define CFG_MAXARGS 16 /* max number of command args */
#define CFG_BARGSIZE CFG_CBSIZE /* Boot Argument Buffer Size */

#define CFG_MEMTEST_START MEMORY_BASE_ADDRESS /* memtest works on */
#define CFG_MEMTEST_END MEMORY_BASE_ADDRESS + 0x7e00000 /* 128 MB in
DRAM */

#undef CFG_CLKS_IN_HZ /* everything, incl board info, in Hz */

C/C++/ObjC 헤더 ▼ 탭 너비: 8 ▼ 229행, 83열 삽입
```



U-boot 컴파일

- Makefile 확인

>> gedit /home/hanyang/u-boot-1.3.4/Makefile

- 144번째 줄 #CROSS_COMPILE = arm-linux- 를 주석 처리
[arm-s5pc1xx-linux-gnueabi- 크로스 컴파일러 사용]

Hanyang University
Division of Computer Science & Engineering

The logo of Hanyang University, featuring a circular emblem with the university's name in Korean and English, and the year 1939.

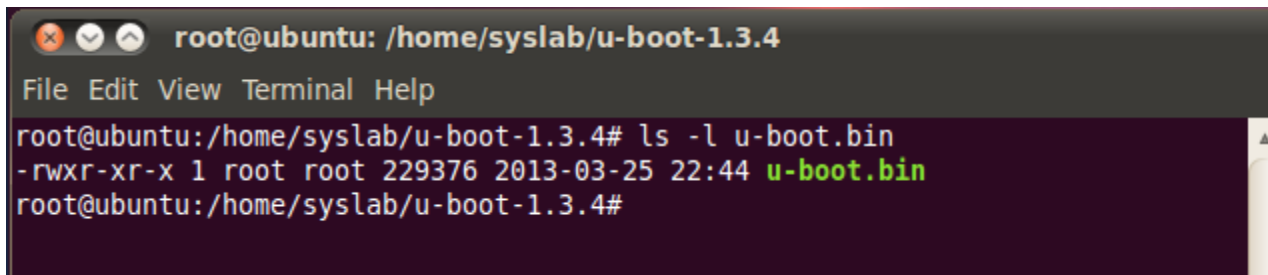
U-boot 컴파일

- **U-boot make**

```
>> cd /home/($username)/u-boot-1.3.4  
>> source /etc/profile  
>> make clean  
>> make clobber  
>> make smdkc100_config  
>> make
```

- **U-boot.bin 파일이 생성되었는지 확인**

```
>> ls -l u-boot.bin
```



```
root@ubuntu: /home/syslab/u-boot-1.3.4  
File Edit View Terminal Help  
root@ubuntu:/home/syslab/u-boot-1.3.4# ls -l u-boot.bin  
-rwxr-xr-x 1 root root 229376 2013-03-25 22:44 u-boot.bin  
root@ubuntu:/home/syslab/u-boot-1.3.4#
```



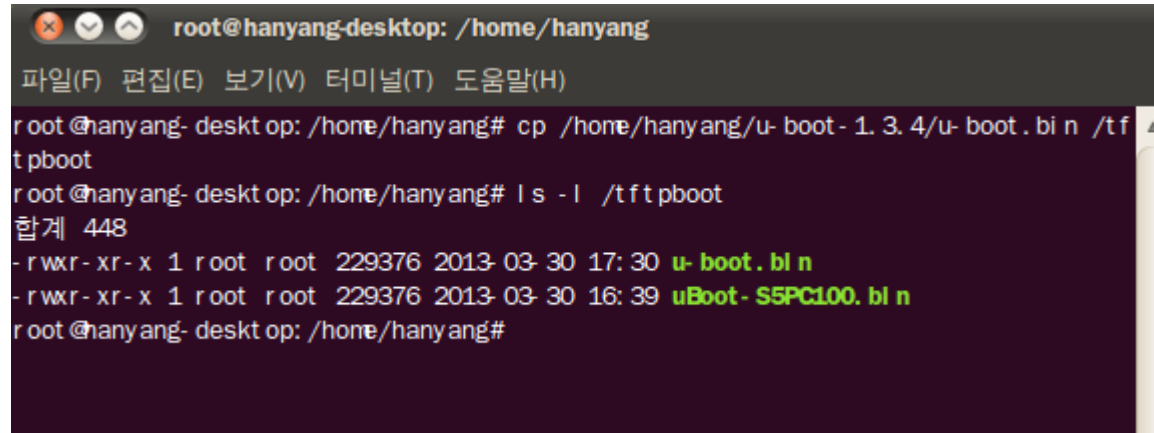
U-boot 컴파일

- u-boot.bin을 /tftpboot 폴더로 복사

```
>> cp /home/hanyang/u-boot-1.3.4/u-boot.bin /tftpboot
```

- u-boot.bin 파일이 복사되었는지 확인

```
>> ls -l /tftpboot
```

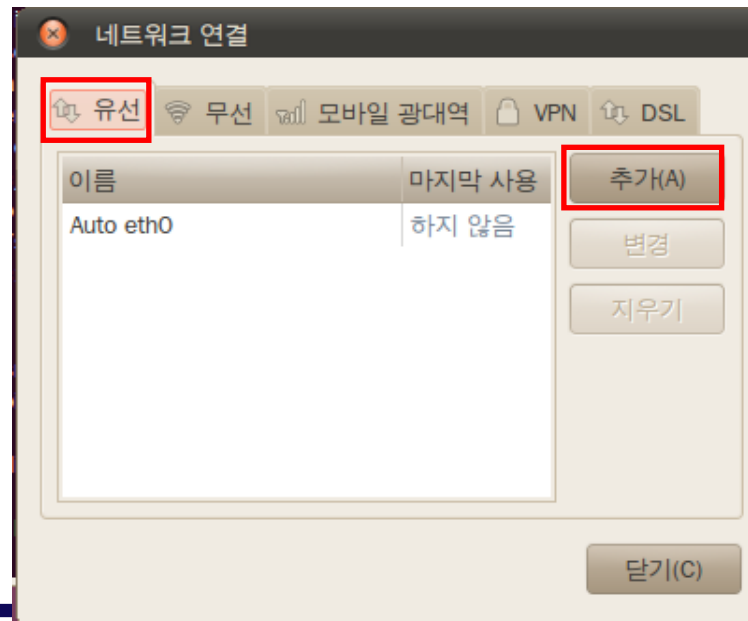


```
root@hanyang-desktop: /home/hanyang
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
root@hanyang-desktop: /home/hanyang# cp /home/hanyang/u-boot-1.3.4/u-boot.bin /tftpboot
root@hanyang-desktop: /home/hanyang# ls -l /tftpboot
합계 448
-rwxr-xr-x 1 root root 229376 2013-03-30 17:30 u-boot.bin
-rwxr-xr-x 1 root root 229376 2013-03-30 16:39 uBoot-S5PC100.bin
root@hanyang-desktop: /home/hanyang#
```



Network 설정

- Network 설정
 - 리눅스 IP 설정
- 시스템 → 기본 설정 → 네트워크 연결
 - 유선 탭에서 추가를 클릭



Network 설정

- Auto eth1 편집

- IPv4 설정

- 방식 : 수동
 - 주소 : 166.104.146.5
 - 넷마스크 : 255.255.255.0
 - 게이트웨이 : 166.104.146.1

- 적용 버튼 클릭

Auto eth1 편집

연결 이름(N): Auto eth1

☒ 자동으로 연결(A)

유선 802.1x 보안 IPv4 설정 IPv6 설정

방식(M): 수동

주소

주소	넷마스크	게이트웨이	추가(A)
166.104.146.5	255.255.255.0	166.104.146.1	삭제(D)

DNS 서버:

검색 도메인(S):

DHCP 클라이언트 ID:

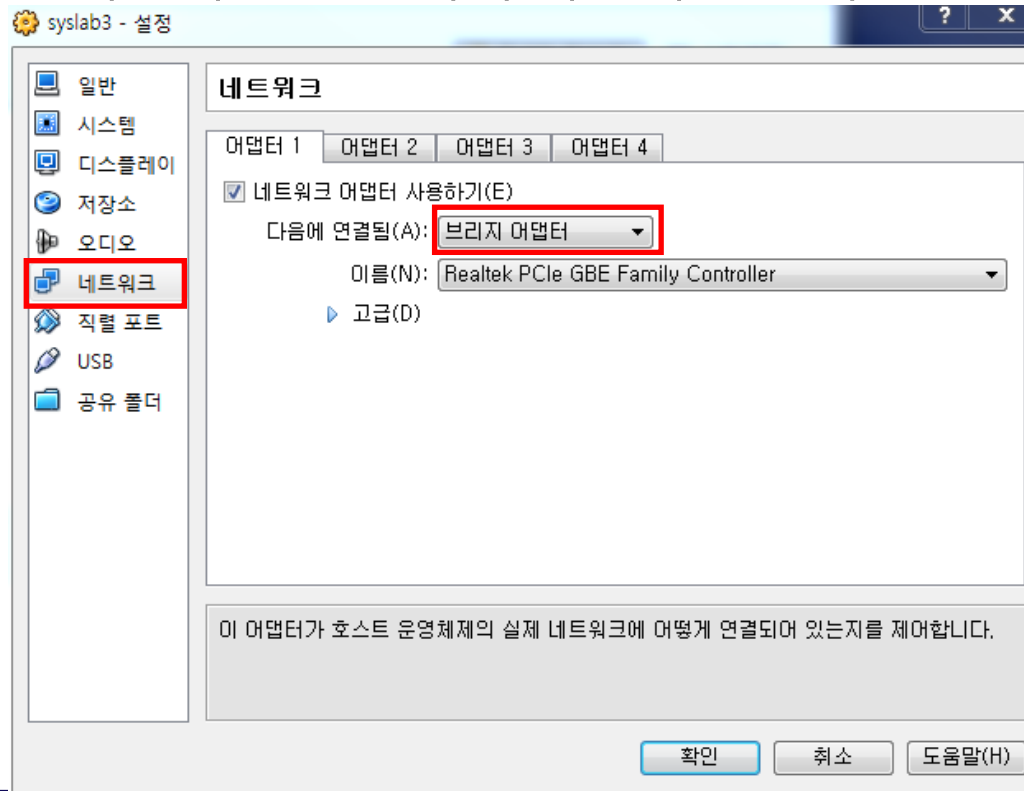
경로(R)...

☐ 모든 사용자가 사용 가능

취소(C) 적용

Network 설정

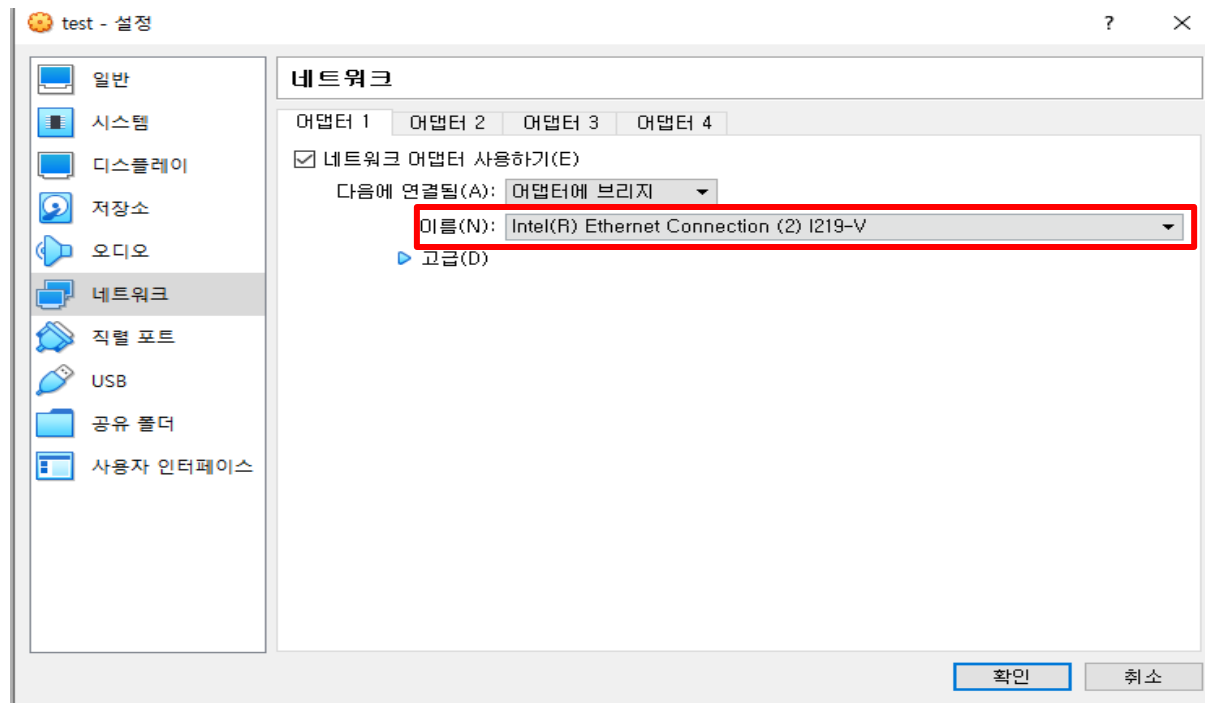
- Virtual machine의 네트워크 설정 (Virtual Box)
 - 리눅스 종료
 - 설정 → 네트워크 → 브리지 어댑터를 선택



Network 설정

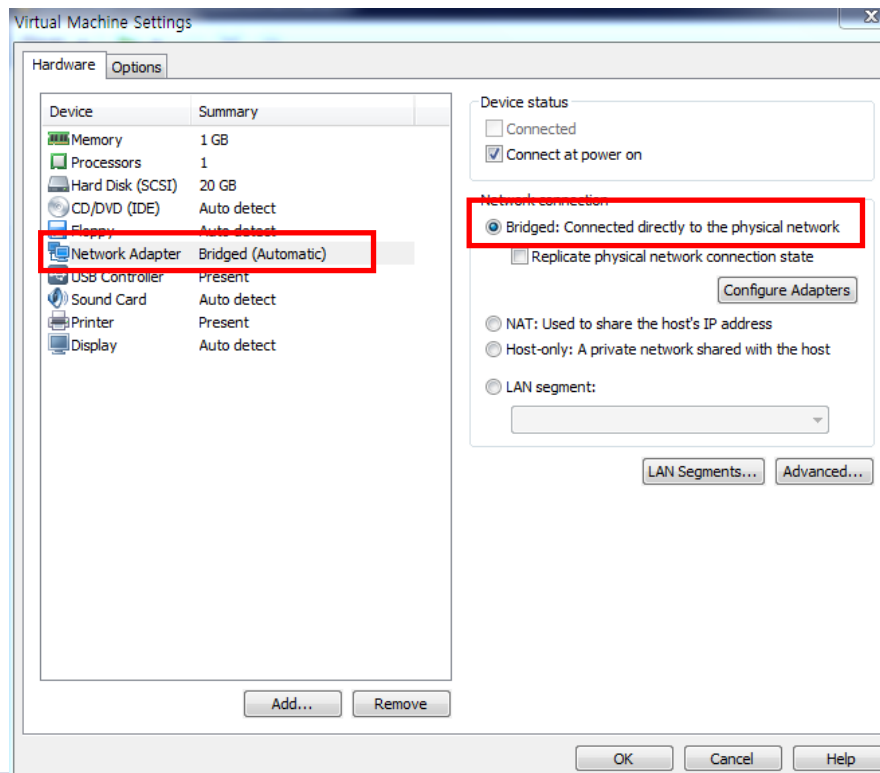
- 주의 사항

- 연결되어 있는 장치가 Ethernet인지 확인
- Wifi는 절대로 안됨



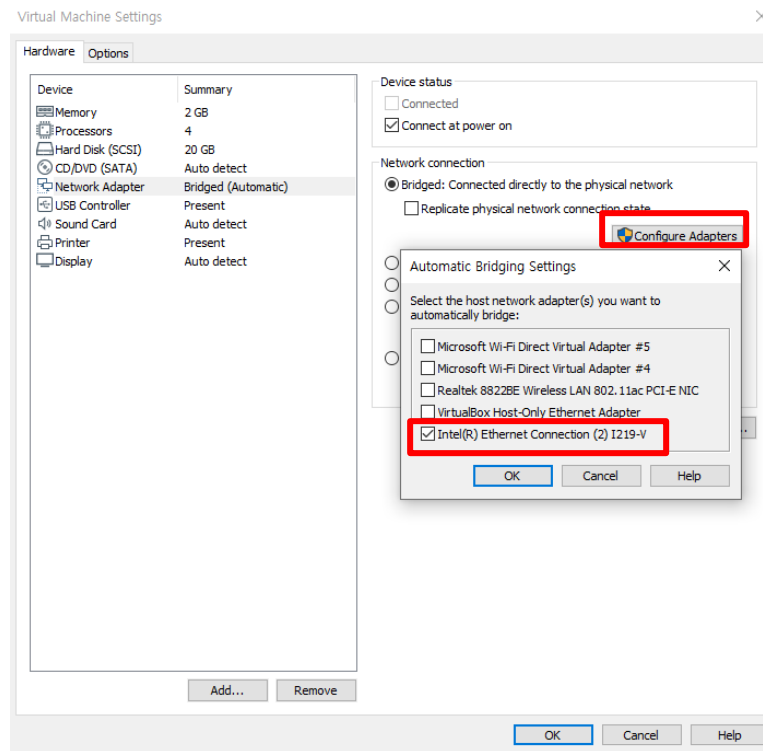
Network 설정

- Virtual machine의 네트워크 설정 (VMware)
 - 리눅스 종료
 - Hardware 탭 → Network Adapter → Bridged에 체크



Network 설정

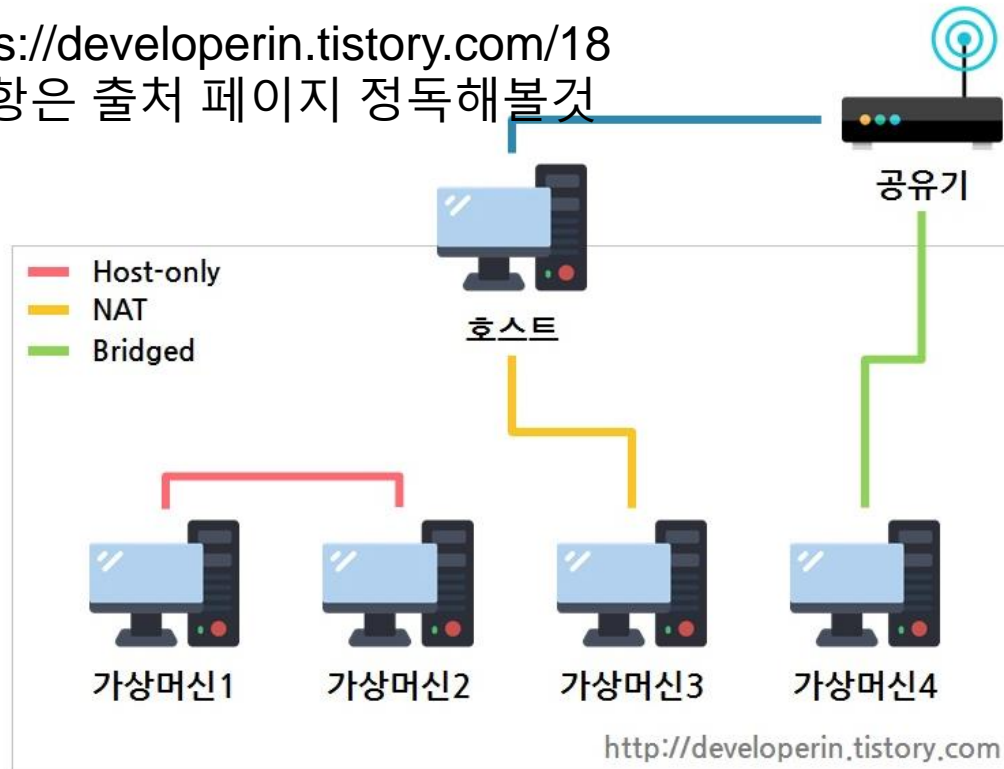
- 브릿지 어댑터 설정 변경 (VMware)
 - Ethernet 관련 장치를 제외하고 모두 체크해제



What is NAT? Bridge?

- NAT : 호스트 PC 통해 통신
- Bridge : 네트워크 장치를 직접 사용

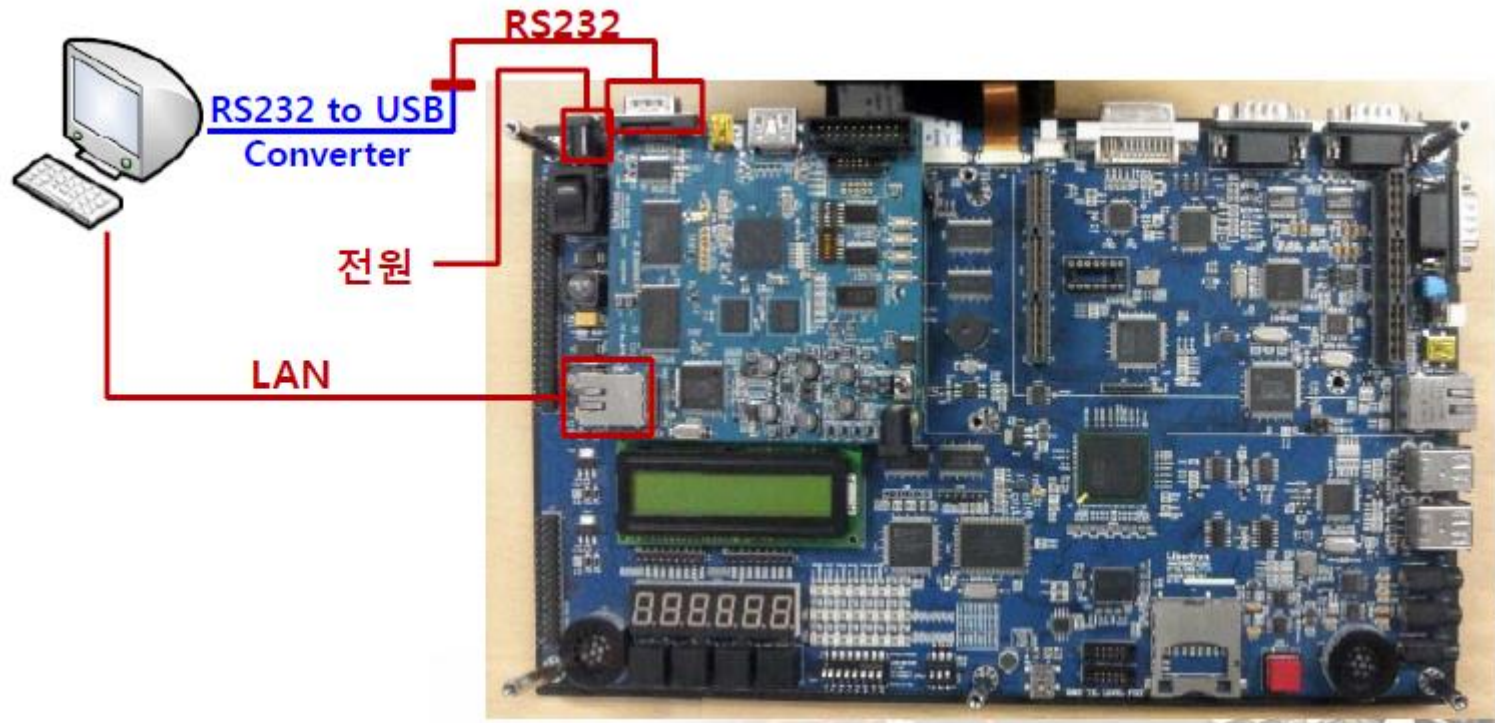
출처 : <https://developerin.tistory.com/18>
자세한 사항은 출처 페이지 정독해볼것



Network 연결

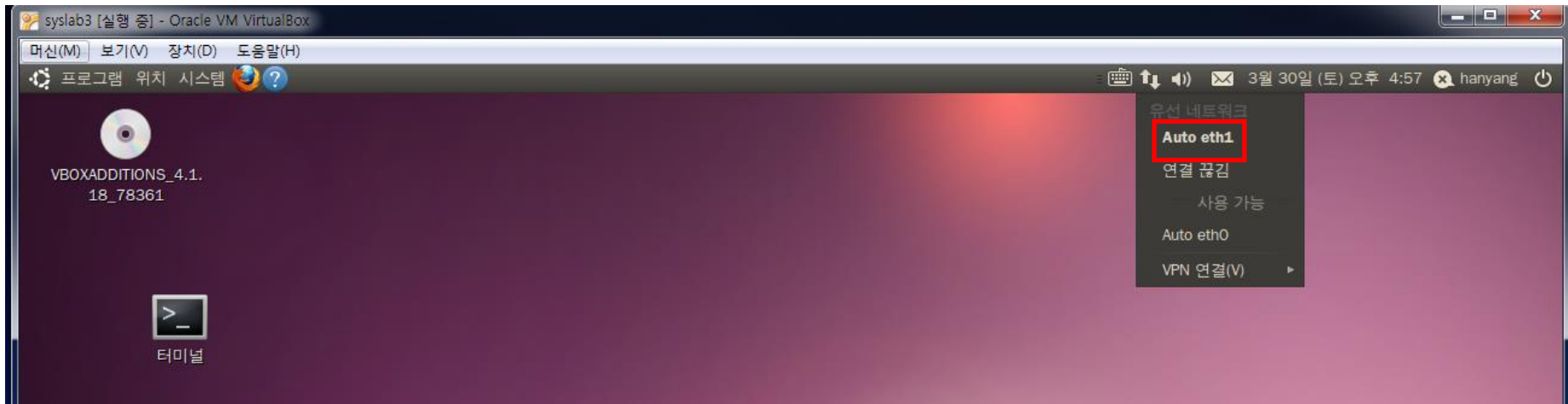
- 보드 연결

- 시리얼 케이블을 pc에 연결 (Minicom에 사용)
- 랜선을 pc에 연결 (TFTP에 사용)



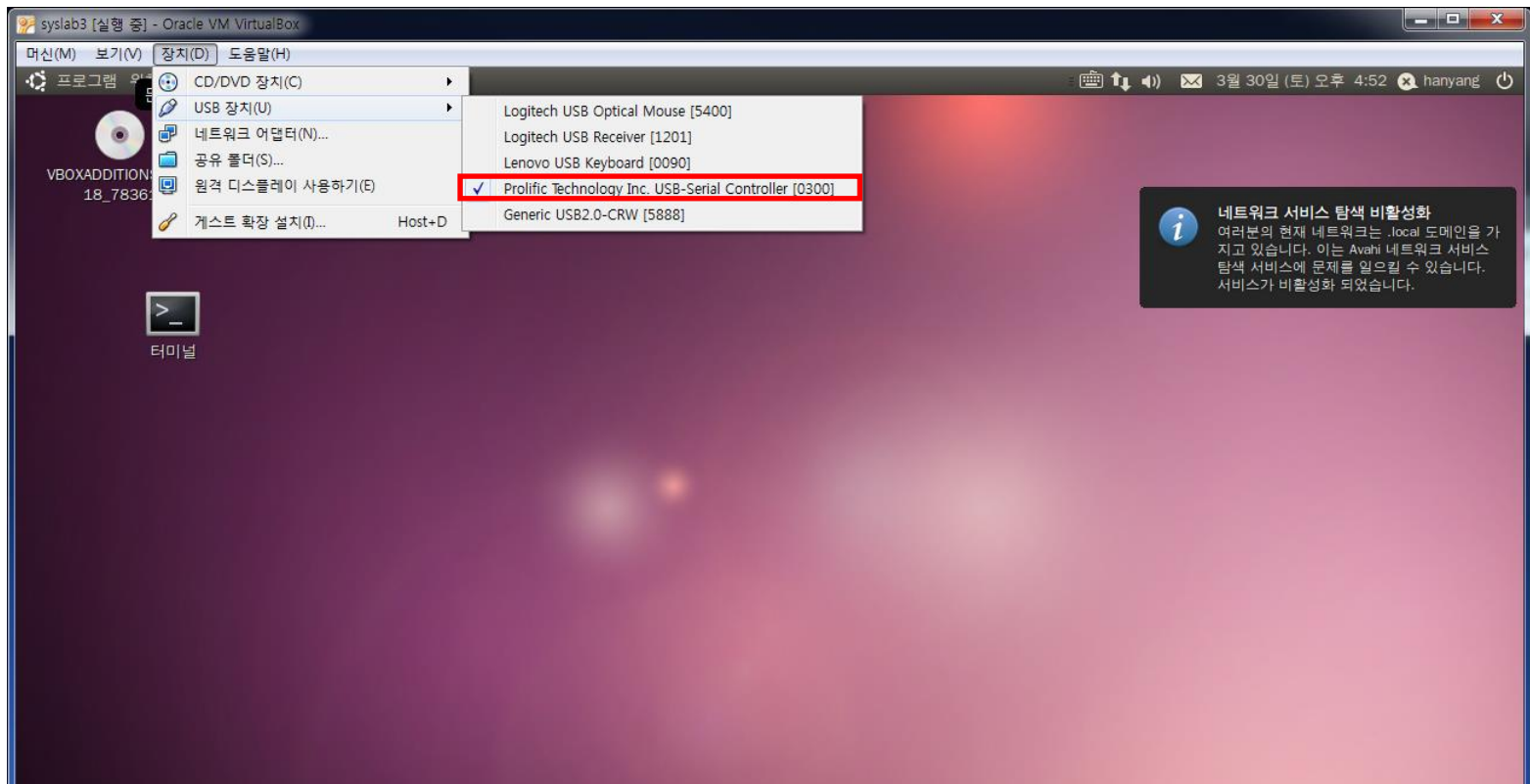
Network 연결

- 리눅스 재실행 후 네트워크를 'Auto eth1'로 연결



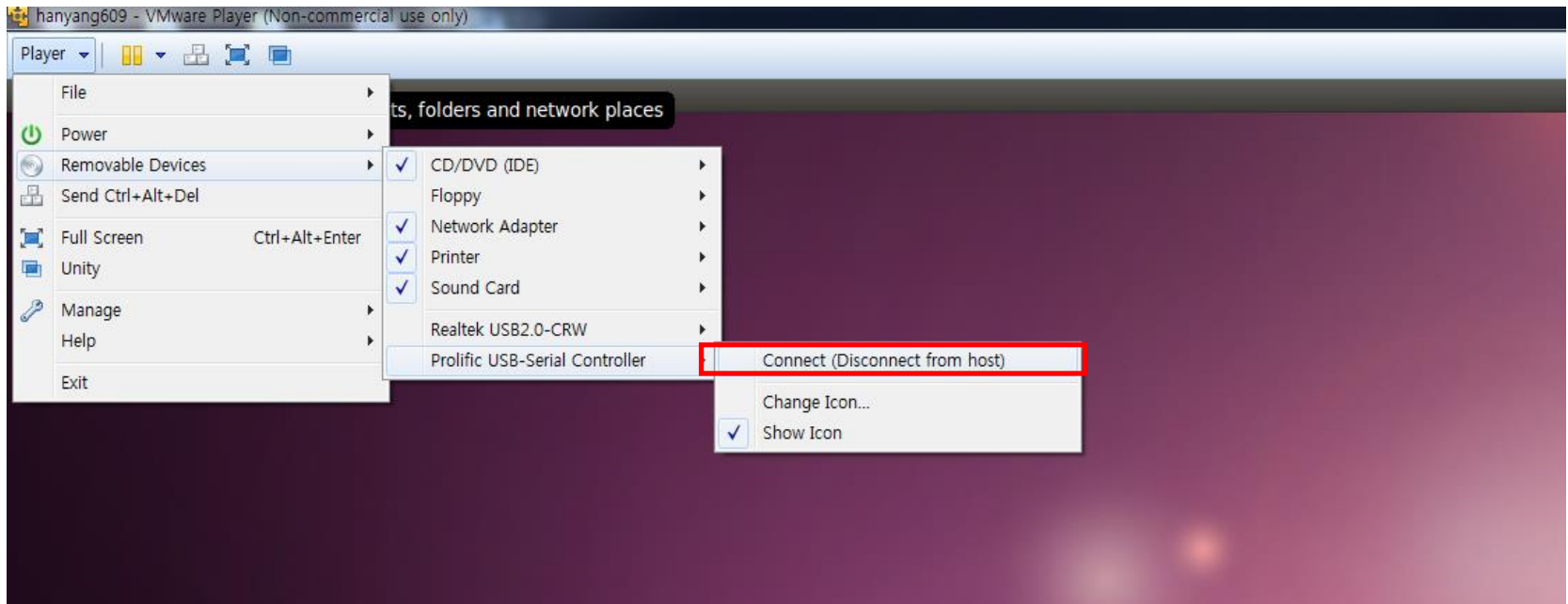
시리얼 케이블 연결

- 장치 → USB 장치 (Virtual Box)
 - ‘Prolific Technology Inc. USB-Serial Controller 클릭(체크)



시리얼 케이블 연결

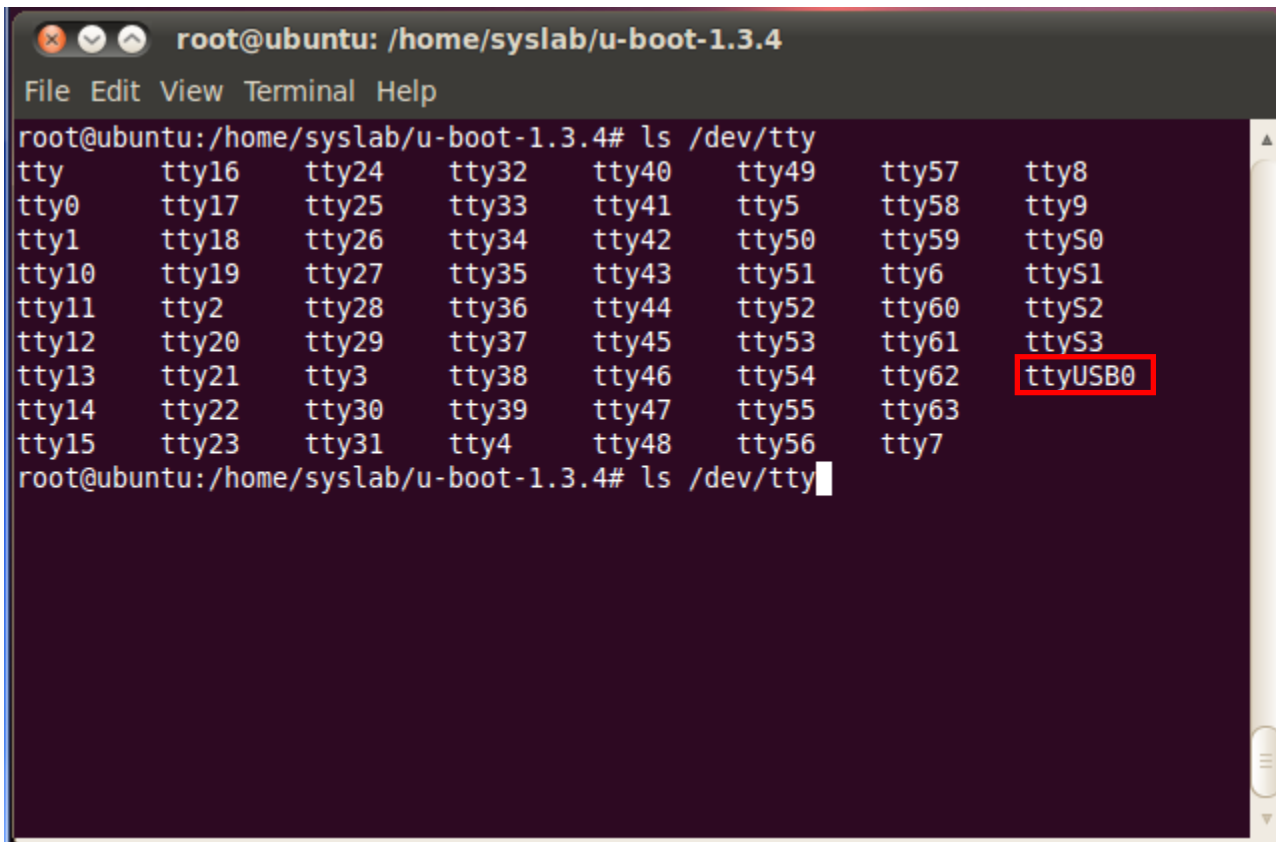
- Player → Removable Devices (VMware)
 - ‘Prolific USB-Serial Controller → Connect 클릭



시리얼 케이블 연결

- 보드가 연결되었는지 확인

>> ls /dev/tty + Tap 2번



```
root@ubuntu: /home/syslab/u-boot-1.3.4
File Edit View Terminal Help
root@ubuntu:/home/syslab/u-boot-1.3.4# ls /dev/tty
tty      tty16   tty24   tty32   tty40   tty49   tty57   tty8
tty0     tty17   tty25   tty33   tty41   tty5     tty58   tty9
tty1     tty18   tty26   tty34   tty42   tty50   tty59   ttyS0
tty10    tty19   tty27   tty35   tty43   tty51   tty6     ttyS1
tty11    tty2     tty28   tty36   tty44   tty52   tty60   ttyS2
tty12    tty20   tty29   tty37   tty45   tty53   tty61   ttyS3
tty13    tty21   tty3     tty38   tty46   tty54   tty62   ttyUSB0
tty14    tty22   tty30   tty39   tty47   tty55   tty63
tty15    tty23   tty31   tty4     tty48   tty56   tty7
```

Minicom

- 미니컴 설정 (반드시 root계정으로)

>> minicom -s

- Serial port setup에서 Serial Device와 Hardware Flow Control을 아래와 같이 바꾼다 (키보드 a키와 f키를 누르면 해당 항목으로 이동)

```
root@hanyang-desktop: /home/hanyang
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)

+---- [ configuration ] ----+
| File names and paths      |
| File transfer protocols   |
| Serial port setup       |
| Modem and dialing         |
| Screen and keyboard       |
| Save setup as dfl         |
| Save setup as..          |
| Exit                      |
| Exit from Minicom         |
+----+

111
```

```
root@ubuntu: /home/syslab/u-boot-1.3.4
File Edit View Terminal Help

+-----+
| A -   Serial Device       : /dev/ttyUSB0 |
| B - Lockfile Location    : /var/lock     |
| C - Callin Program       :               |
| D - Callout Program      :               |
| E - Bps/Par/Bits         : 115200 8N1    |
| F - Hardware Flow Control : No         |
| G - Software Flow Control : No          |
+-----+

Change which setting?

+-----+
| Screen and keyboard      |
| Save setup as dfl       |
| Save setup as..         |
| Exit                    |
| Exit from Minicom       |
+-----+
```



Minicom

- 미니컴 설정
 - ‘Save setup as dfl’로 미니컴 설정을 저장
- 미니컴으로 보드에 접속
 - (1) >> minicom
 - (2) >> minicom -s 후 ‘Exit’

```
+----- [ configuration ] -----+
| File names and paths             |
| File transfer protocols          |
| Serial port setup               |
| Modem and dialing               |
| Screen and keyboard             |
| Save setup as dfl               |
| Save setup as..                 |
| Exit                             |
| Exit from Mini com              |
+-----+-----+-----+-----+
```

U-boot 포팅

- 보드에 전원을 키고 Enter 키를 누름
- 아래와 같은 화면이 나오면 보드에 접속 성공
 - 화면이 안 뜨면 리셋 버튼을 눌러 리셋

```
root@hanyang-desktop: /home/hanyang
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
Welcome to mini com 2.4

OPTI ONs: 118n
Compiled on Jan 25 2010, 06:49:09.
Port /dev/ttyUSB0

Press CTRL-A Z for help on special keys

0

U-Boot 1.3.4 (Mar 27 2013 - 22:06:45) for SL2_C100

CPU:      S5PC100@666MHz
        Fck = 1332MHz, Hck = 166MHz, Pck = 66MHz, Serial = PCLK
Board:    SL2_C100
DRAM:     256 MB
Flash:    1 MB
NAND:     512 MB
In:       serial
Out:      serial
Err:      serial
Hit any key to stop autoboot: 0
SL2_C100 #
```



U-boot 포팅

- 네트워크 환경 설정

- 아래 U-boot 명령으로 호스트 PC의 IP와 타겟보드의 IP를 설정한다

setenv ipaddr 166.104.146.10 : 타겟보드 IP 주소 저장

setenv serverip 166.104.146.5 : 리눅스 IP 주소

setenv gatewayip 166.104.146.1 : 게이트웨이 주소

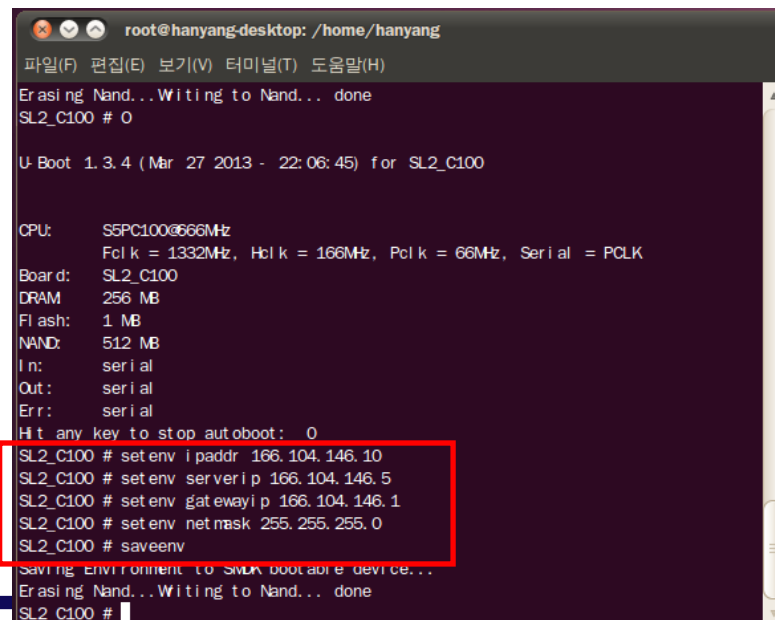
setenv netmask 255.255.255.0 : 넷마스크 주소

- 환경 설정 저장

saveenv

- 환경 변수 설정 확인

printenv



```
root@hanyang-desktop: /home/hanyang
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
Erasing Nand...Writing to Nand... done
SL2_C100 # 0

U-Boot 1.3.4 (Mar 27 2013 - 22:06:45) for SL2_C100

CPU:      S5PC100@666MHz
          Fclk = 1332MHz, Hclk = 166MHz, Pclk = 66MHz, Serial = PCLK
Board:    SL2_C100
DRAM:     256 MB
Flash:    1 MB
NAND:     512 MB
In:       serial
Out:      serial
Err:      serial
Hit any key to stop autoboot: 0
SL2_C100 # setenv ipaddr 166.104.146.10
SL2_C100 # setenv serverip 166.104.146.5
SL2_C100 # setenv gatewayip 166.104.146.1
SL2_C100 # setenv netmask 255.255.255.0
SL2_C100 # saveenv
Saving Environment to SMC bootable device...
Erasing Nand...Writing to Nand... done
SL2_C100 #
```



U-boot 포팅

- TFTP를 이용하여 u-boot write
 - Network를 이용하여 tftp로 u-boot를 nand에 write
- ```
tftp c0008000 u-boot.bin
nand erase 0 60000
nand write c0008000 0 40000
```

```
root@hanyang-desktop: /home/hanyang
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
Out: serial
Err: serial
Hit any key to stop autoboot: 0
SL2_C100 # tftp c0008000 u-boot.bin
smc911x: initializing
smc911x: detected LAN9217 controller
smc911x: phy initialized
smc911x: MAC 00:40:5c:26:0a:5b
TFTP from server 166.104.146.5; our IP address is 166.104.146.10
Filename 'u-boot.bin'.
Load address: 0xc0008000
Loading: #####
done
Bytes transferred = 229376 (0x38000)
SL2_C100 # nand erase 0 60000

NAND erase: device 0 offset 0x0, size 0x60000
Erasing at 0x40000 -- 100% complete.
OK
SL2_C100 # nand write c0008000 0 40000

NAND write: device 0 offset 0x0, size 0x40000
262144 bytes written: OK
SL2_C100 #
```

Load, Erase, Write  
연산이 제대로 됐는지  
반드시 확인할 것!!

# U-boot 포팅

- 보드 리셋 후 Enter 키를 누름
  - ‘StudentID #’ 으로 변경되었는지 확인

```
root@hanyang-desktop: /home/hanyang
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)

NAND erase: device 0 offset 0x0, size 0x60000
Erasing at 0x40000 -- 100% complete.
OK
SL2_C100 # nand write c0008000 0 40000

NAND write: device 0 offset 0x0, size 0x40000
262144 bytes written: OK
SL2_C100 # 0

U-Boot 1.3.4 (Mar 30 2013 - 16:39:36) for SL2_C100

CPU: S5PC100@666MHz
Clock = 1332MHz, Hclk = 166MHz, Pclk = 66MHz, Serial = PCLK
Board: SL2_C100
DRAM: 256 MB
Flash: 1 MB
NAND: 512 MB
In: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0
HANYANG #
```

- U-boot write 완료

---

# VPOS 부팅





# vpos 부팅 준비

- **이미지 파일 다운로드**

- vpos.bin 파일 이용
- 윈도우 공유폴더(C:\syslab)에 vpos.bin 복사
- 블랙보드를 통해 직접 다운로드

- **공유폴더에서 /tftpboot 파일로 복사**

- 리눅스 공유폴더에서 /tftpboot로 vpos.bin 복사

>> cp /home/(\$username)/syslab/vpos.bin /tftpboot (VirtualBox)

>> cp /mnt/hgfs/syslab/vpos.bin /tftpboot (VMware)

>> cp /home/(\$username)/Download/vpos.bin /tftpboot  
(BlackBoard)



# Minicom

- 미니컴으로 보드에 접속

(1) >> minicom

(2) >> minicom -s 후 'Exit'

```
+-----[configuration]-----+
| Filenames and paths |
| File transfer protocols |
| Serial port setup |
| Modem and dialing |
| Screen and keyboard |
| Save setup as df1 |
| Save setup as.. |
| Exit |
| Exit from Minicom |
+-----+-----+-----+-----+
```

# VPOS 커널 다운로드 (NAND 미사용)

- 본 실습 수업에서는 NAND에 저장하지 않고 SDRAM에 직접 write
- TFTP를 이용하여 kernel write
  - Network를 이용하여 tftp로 vpos kernel을 SDRAM에 write

```
setenv bootcmd tftp c0008000 vpos.bin\;bootm c0008000
saveenv
```
  - 부팅 명령어

```
boot
```



# VPOS 커널 다운로드 (NAND 사용)

- TFTP를 이용하여 kernel write

- Network를 이용하여 tftp로 vpos kernel을 nand에 write

- # tftp c0008000 vpos.bin

- # nand erase 80000 400000

- # nand write c0008000 80000 400000

- # setenv bootcmd nand read c0008000 80000 300000\;bootm  
c0008000

- # saveenv

- 부팅 명령어

- # boot

- 알아만 둘 것



# VPOS 부팅

- VPOS 부팅 화면

```
Board: SL2_C100
DRAM 256 MB
Flash: 1 MB
NAND: 512 MB
In: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0
HANYANG # boot

NAND read: device 0 offset 0x80000, size 0x300000
3145728 bytes read: OK
Boot with zImage

Starting kernel ...

* QURI X version 3.0 xx/10/2012 *

Race condition value = 1191214

Shell >
```

# VPOS의 스레드

- **VPOS에서 스레드 실행**
  - Shell에 명령어를 입력하면 shell은 명령어를 해석
  - 해당 명령어 처리 루틴을 thread로 생성
  - Ready 큐에 적재시키고 스케줄러를 호출해 thread 실행
- **VPOS의 명령어**
  - ls : 명령어 목록을 보여줌
  - help : 명령어 목록을 보여줌
  - debug : Debug용 명령어
  - temp : 스레드 2개로 context switching 실험
  - thread : 스레드 2개로 context switching 실험 (무한루프)
- **명령어를 하나씩 입력하여 UART, TIMER, INTERRUPT가 제대로 동작하는지 확인하세요**



---

# 소스코드 디렉토리 및 파일 구조



---

# VPOS 커널 코드 배포

- 소스코드 파일 다운로드
  - vpos.zip파일 이용
- 압축을 풀고 공유 폴더로 복사





# 소스코드 디렉토리 및 파일

- **hal/cpu**
  - CPU Architecture에 의존적인 코드를 작성하고 관리
    - HAL\_arch\_startup.S : 커널 초기화 소스코드 파일
    - hal\_swi\_handler.c : 소프트웨어 인터럽트 핸들러 소스코드 파일
    - vpos\_kernel-ld-script : 링커 스크립트
- **hal/include**
  - CPU와 I/O 관련 레지스터들의 주소나 매크로를 저장
    - vh\_io\_hal.h : I/O 관련 레지스터 주소나 매크로를 저장
- **hal/io**
  - I/O 관련 코드를 작성하고 관리
    - serial.c : UART 관련 소스코드 파일
    - timer.c : TIMER 관련 소스코드 파일
    - led.c : LED 관련 소스코드 파일



# 소스코드 디렉토리 및 파일

- **kernel**
  - 커널 관련 소스 코드를 작성하고 관리하는 디렉토리
    - exception\_handler.c : exception 처리 관련 소스코드 파일
    - kernel\_start.c : 커널의 C코드 진입 소스코드 파일
    - machine\_init.c : 필요한 하드웨어를 초기화하는 소스코드 파일
- **include**
  - 헤더파일을 관리하는 디렉토리
- **images**
  - 컴파일 후 실행 이미지들이 존재하는 디렉토리
- **objs**
  - 컴파일 후 오브젝트 파일이 적재되는 디렉토리



---

# 커널 포팅 준비



---

# VPOS 커널을 포팅하기 위한 준비

1. 커널 컴파일 + 커널 이미지를 RAM에 적재
2. STARTUP code 구현
3. UART 설정
4. TIMER 설정
5. Hardware Interrupt Handler 구현
  - (1) UART Interrupt
  - (2) Timer Interrupt
6. Software Interrupt Entering/Leaving Routine 구현



# 커널 이미지를 RAM에 적재

- U-Boot를 통해 커널 이미지를 RAM에 적재

```
setenv bootcmd tftp c0008000 vpos.bin\;bootm
c0008000
```

- 명령어 설명

- tftp c0008000 vpos.bin
  - TFTP를 통해 VPOS 커널 이미지를 RAM의 c0008000번지에 적재
- bootm c0008000
  - c0008000번지에 저장된 이미지로 부팅



# 왜 c0008000번지인가?

- mDDR(SDRAM)에서 커널의 위치
  - Physical address : 0x20008000
  - Virtual address : 0xc0008000
- U-Boot에서는
  - MMU를 사용하므로 virtual address를 사용하여 메모리에 접근
- VPOS 커널에서는
  - MMU를 사용하지 않으므로 physical address를 사용하여 메모리에 접근



# ctags & cscope

- vpos/ 폴더에서 ctags와 cscope 설정
- ctags
  - 소스 코드 분석
    - >> ctags -R
  - 사용법
    - 함수가 정의된 파일로 이동
      - Ctrl + ]
    - 이전 위치로 이동
      - Ctrl + t
- cscope
  - 설정 방법
    - >> cscope -R
    - >> 사용법
      - cscope 실행
        - >> cscope
      - cscope 종료
        - Ctrl + D



# Makefile 확인

- Makefile에서 Cross Compiler 설정

- 디렉토리 : vpos/

- >> vi Makefile

```
.EXPORT_ALL_VARIABLES:

TOPDIR := $(shell if ["$$PWD" != ""]; then echo $$PWD; else pwd; fi)

DIRS = kernel hal/cpu hal/io shell objs
CROSTOOL = arm-s5pc1xx-linux-gnueabi-
CC = $(CROSTOOL)gcc
OC = $(CROSTOOL)objcopy
LD = $(CROSTOOL)ld

INCLUDE = -I. -I$(TOPDIR)/include -I$(TOPDIR)/hal/include -I$(TOPDIR)/fs/include
 -I/opt/s5pc1xx/cross/armv7a/lib/gcc/arm-s5pc1xx-linux-gnueabi/4.2.1/include

CFLAGS = -g -O0 -Wall -Wstrict-prototypes -fPIC -msoft-float -nostdinc -nostart
 files -nostdlib -march=armv5 -fno-builtin $(INCLUDE)

OCFLAGS = -Obinary -R.note -R.comment -R.stab -R.stabstr -S

all:
 for i in $(DIRS) ; do make -C $$i || exit $? ; done
 dd if=images/vpos_kernel_binary of=images/vpos.bin bs=1k conv=sync
```



# Linker Script 수정

- Linker Script 수정

- >> vi hal/cpu/vpos\_kernel-ld-script
- 'SECTIONS' 첫 줄을 ". = 0x20008000;"로 변경

```
OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
OUTPUT_ARCH(arm)
ENTRY(vh_VPOS_STARTUP)
SECTIONS
{
 . = 0x20008000;

 . = ALIGN(4);
 .text : { *(.text) }

 . = ALIGN(4);
 .rodata : { *(.rodata) }

 . = ALIGN(4);
 .data : { *(.data) }

 . = ALIGN(4);
 .got : { *(.got) }

 . = ALIGN(4);
 .bss : { *(.bss) }
}
```

# Linker Script

- **Linker Script란?**

- 링커는 오브젝트 파일을 조합하여 하나의 실행 파일을 만듦
- Linker Script는 링킹 과정에서 링커의 동작을 제어

- **Linker Script 분석**

- OUTPUT\_FORMAT("elf32-littlearm", ...)
  - ELF32의 little endian으로 코드를 생성
- OUTPUT\_ARCH(arm)
  - Binary를 실행할 수 있는 CPU architecture로 ARM을 지정
- ENTRY(vh\_VPOS\_STARTUP)
  - 프로그램의 시작을 가리킴
- SECTION {
  - 출력파일의 메모리 레이아웃을 설명



# Linker Script

- Linker Script 분석

- SECTION {}

- 프로그램의 각 섹션을 정의
      - 프로그램의 전체 영역과 각 섹션이 어디에 저장될지를 결정
      - 각 섹션이 저장될 위치를 가리키는 위치 카운터를 지정
        - 특별 심볼 ‘.’을 사용
        - 출력 섹션의 주소를 표시
- ex) . = 0x20008000;



# 실행 파일 분석하기

- **objdump**

- 실행 파일에 대한 정보를 출력하는 리눅스 명령어
- 옵션으로 -h를 사용하면 실행 파일의 섹션 헤더에서 요약된 정보를 출력

>> cd images

>> objdump -h vpos\_kernel-elf32

```
Sections:
Idx Name Size VMA LMA File off Algn
 0 .text 0000c130 20008000 20008000 00008000 2**4
 CONTENTS, ALLOC, LOAD, READONLY, CODE
 1 .rodata 00000a9c 20014130 20014130 00014130 2**2
 CONTENTS, ALLOC, LOAD, READONLY, DATA
 2 .data 0000011c 20014bcc 20014bcc 00014bcc 2**2
 CONTENTS, ALLOC, LOAD, DATA
 3 .got.plt 0000000c 20014ce8 20014ce8 00014ce8 2**2
 CONTENTS, ALLOC, LOAD, DATA
 4 .data.rel.local 0000000c 20014cf4 20014cf4 00014cf4 2**2
 CONTENTS, ALLOC, LOAD, DATA
 5 .data.rel 00000084 20014d00 20014d00 00014d00 2**2
 CONTENTS, ALLOC, LOAD, DATA
 6 .data.rel.ro.local 00000070 20014d84 20014d84 00014d84 2**2
 CONTENTS, ALLOC, LOAD, DATA
 7 .got 00000110 20014df4 20014df4 00014df4 2**2
 CONTENTS, ALLOC, LOAD, DATA
 8 .bss 000fb284 20014f04 20014f04 00014f04 2**2
 ALLOC
 9 .tbss 00000004 20110188 20110188 00018188 2**2
 ALLOC, THREAD_LOCAL
```



# Shell Script

- 커널 컴파일 후 이미지를 /tftpboot로 복사
  1. make clean
  2. make
  3. cp images/vpos.bin /tftpboot
- Shell Script 만들기
  - >> vi rr.sh
  - 아래와 같이 입력 후 저장

```
#!/bin/bash
source /etc/profile
make clean
make
cp images/vpos.bin /tftpboot
~
~
~
~
~
```



# 커널 컴파일

- Shell Script를 실행하여 자동으로 컴파일 후 /tftpboot로 복사

>> ./rr.sh

```
make[1]: Leaving directory `/home/bhsong/vpos_student/shell'
make[1]: Entering directory `/home/bhsong/vpos_student/objs'
arm s5pc1xx-linux-gnueabi-ld -v -T../hal/cpu/vpos_kernel.ld -script -Bstatic *.o
-o vpos_kernel -elf32 -L /opt/s5pc1xx/cross/armv7a/lib/gcc/arm s5pc1xx-linux-gnueabi/4.2.1 -L /opt/s5pc1xx/staging/armv7a-s5pc1xx-linux-gnueabi/usr/lib -lgcc -ld
GNU ld (Linux/GNU Binutils) 2.18.50.0.7.20080502
arm s5pc1xx-linux-gnueabi-objcopy -O binary -R .note -R .comment -R .stab -R .st
abstr -S vpos_kernel-elf32 vpos_kernel_binary
mv vpos_kernel_binary vpos_kernel-elf32 ../images
make[1]: Leaving directory `/home/bhsong/vpos_student/objs'
dd if=images/vpos_kernel_binary of=images/vpos.bin bs=1k conv=sync
46+1 레코드 들어옴
47+0 레코드 나감
48128 바이트 (48 kB) 복사됨, 0.000138079 초, 349 MB/초
date
2013. 04. 05. (금) 20:24:37 KST
root@bhsong-desktop: /home/bhsong/vpos_student #
```

- U-Boot를 통해 커널을 RAM에 적재하고 부팅



---

# 보고서 제출

- 보고서

- 저번주 강의와 제출파일 이름 및 양식은 동일하게
- 부트로더, 크로스 컴파일에 대해 조사하고 A4 1장 내외로 정리
- 수정한 U-boot 실행화면 및 VPOS 부팅 화면도 첨부 (p.54, p.60)



---

# 제출 방법

- 제출 방법
  - 워드나 한글로 작성
  - 문서 제목에 학번과 이름을 적을 것
- 블랙보드를 통해 제출
- 마감일
  - 5/17일 자정 전까지



---

수고하셨습니다.

