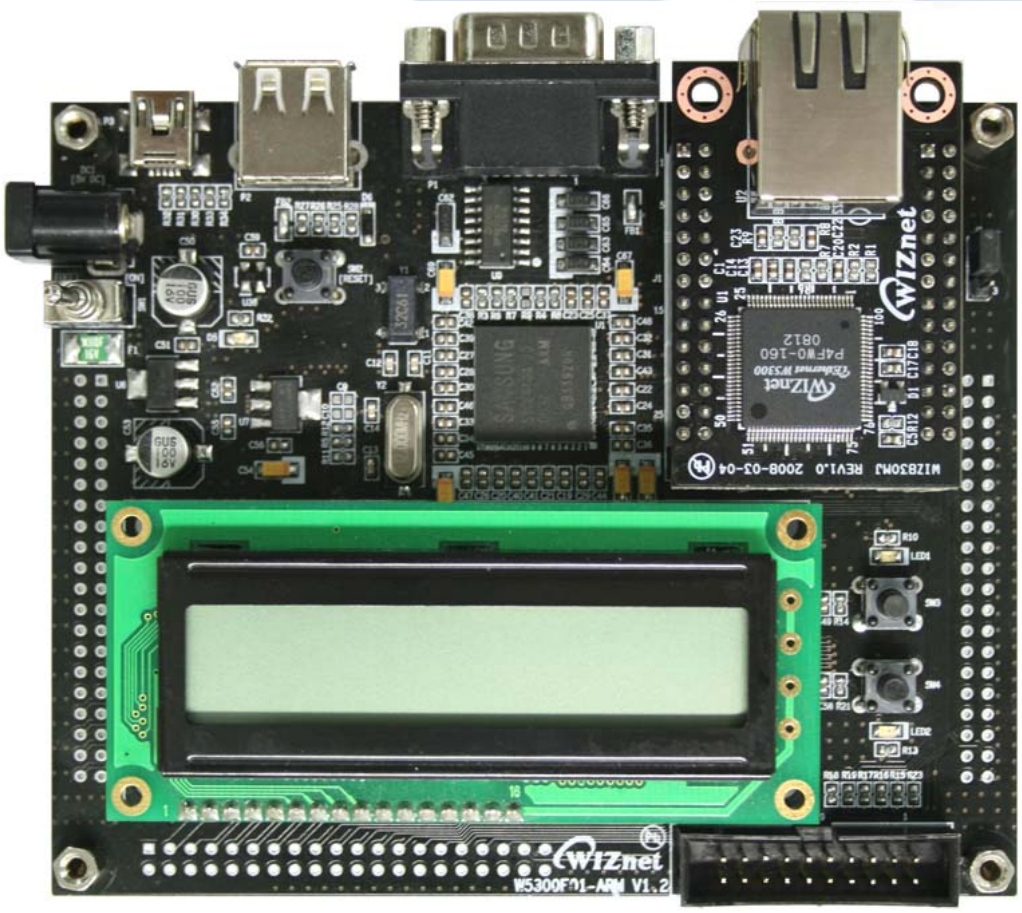
W5300E01-ARM

Cross Compiler User Manual

(Version1.0)



☞ For more information, visit our website at http://www.wiznet.co.kr

WIZnet's Online Technical Support

If you have something to ask about WIZnet Products, Write down your question on Q&A Board in WIZnet website (www.wiznet.co.kr). WIZnet Engineer will give an answer as soon as possible.

COPYRIGHT NOTICE

Technical Support: support@wiznet.co.kr

Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at http://www.wiznet.co.kr

Table of Contents

1 Introduction

Cross Compiler is used when user's development environment is different from the objective system. For example, when developing ARM-based embedded system, the user should write the source code at the PC, and compile the codes by using Cross Compiler. The compiled binary image operates not at the PC but at the ARM-based system.

1.1 Composition of Cross Compiler

There are several types of cross compiler – commercial compilers such as ADS/RVCT (provided by ARM) and GNU compiler. W5300E01-ARM provides GNU compiler. GNU compiler is composed of below

- *binutils*
  - Programming Tools for controlling various objective file format
- *gcc*
  - GNU Compiler
- *gibc*
  - Library used for cross compile

1.2 Before Starting

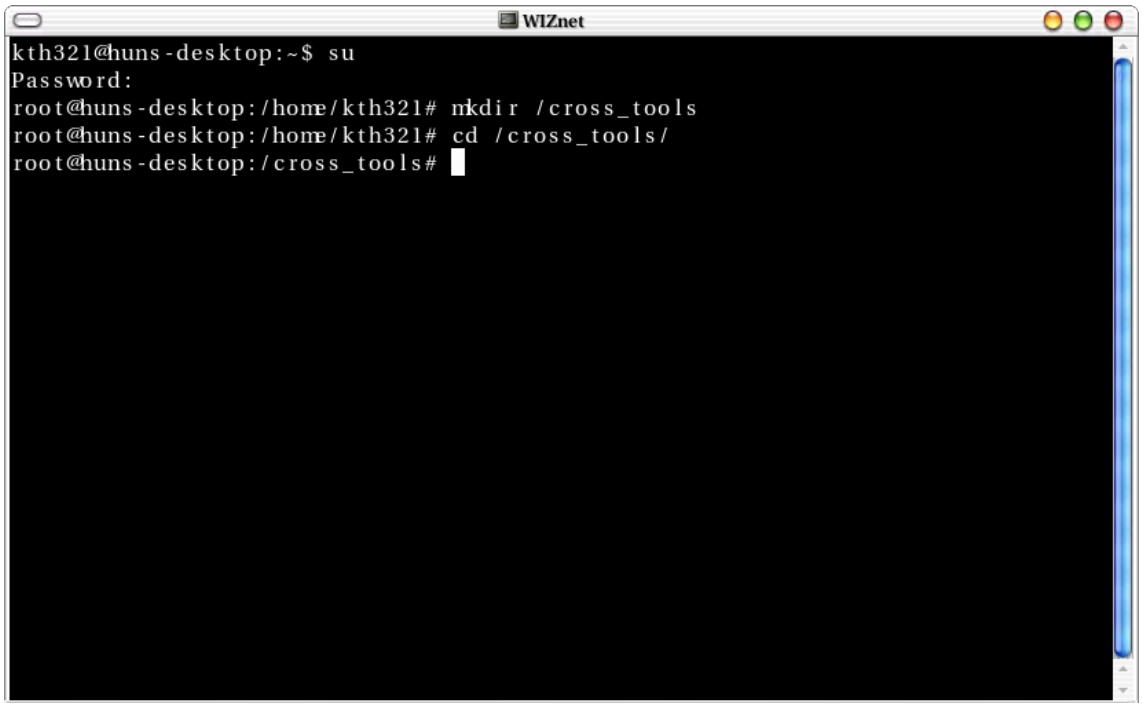
The arm-linux-gcc version of W5300E01-ARM is 3.4.3. So, for the cross compiling, 3.4.x version GCC is used. Please be sure to check version and install 3.4 version GCC.

From the next chapter, cross-compiling process is described step by step. As described in this manual, please install and make the compiler.

2 Source Code Download

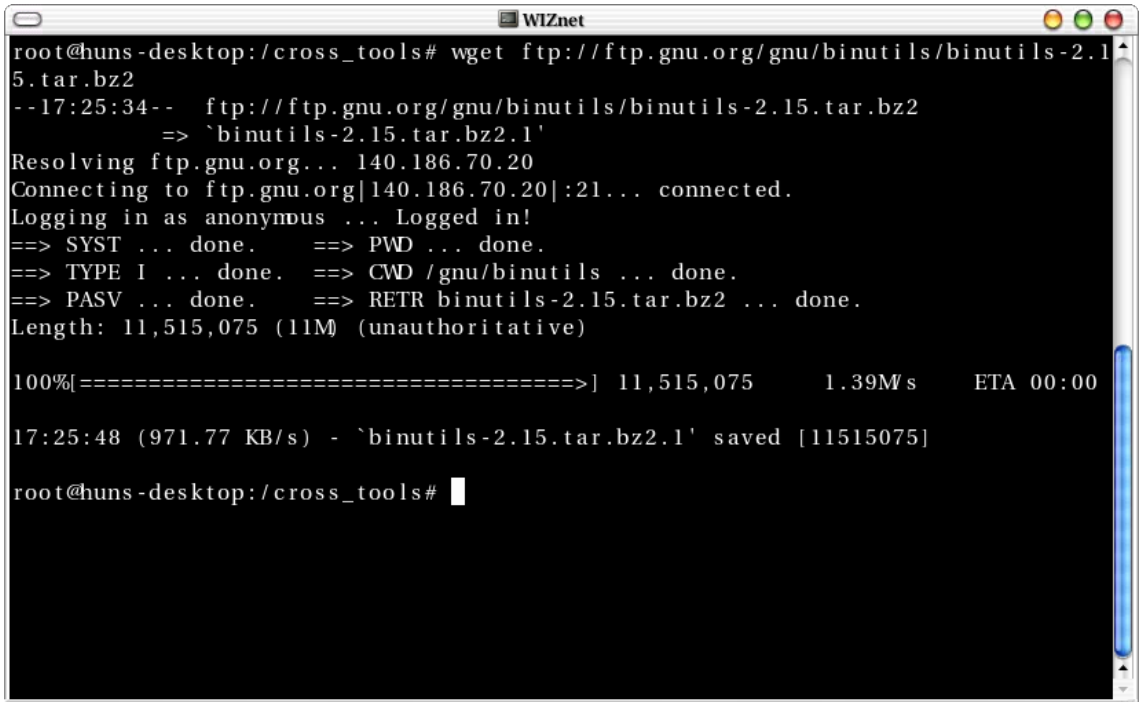
① Compiler installation requires root authority for installation of system composing factors. After logging in to root and create a directory as below.

- ***mkdir /cross_tools***
- ***cd /cross_tools***



② Download ‘binutils-2.15’ source code.

- ***wget ftp://ftp.gnu.org/gnu/binutils/binutils-2.15.tar.bz2***

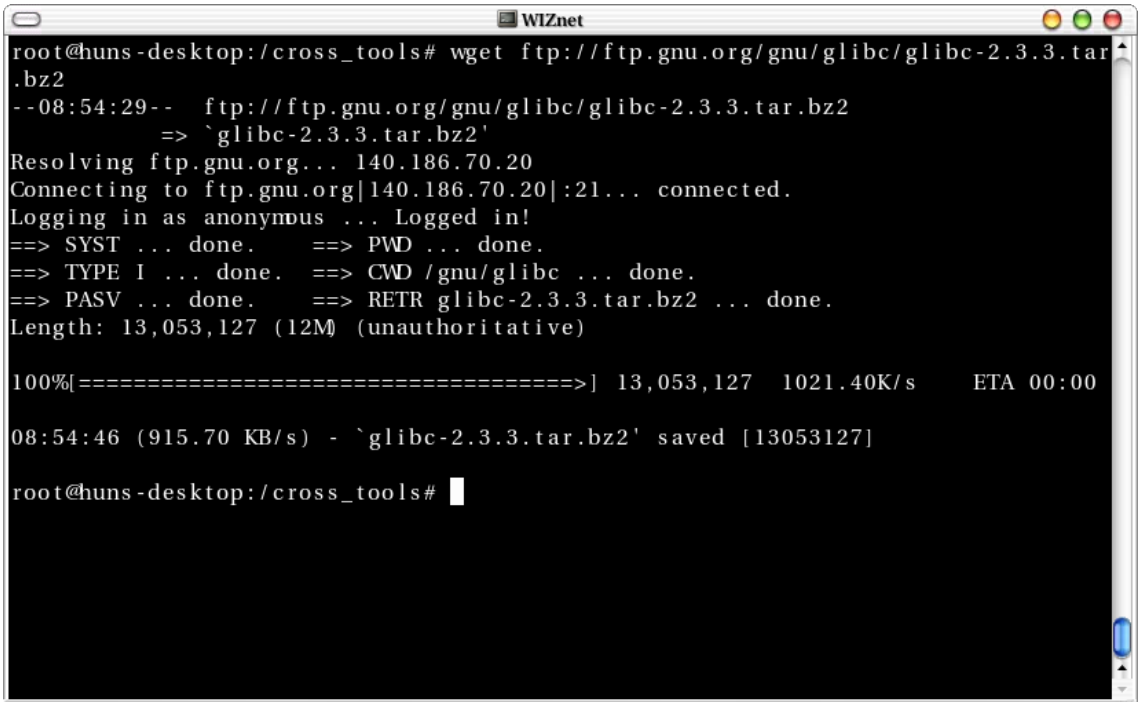


③ Download ‘gcc-3.4.3’ source code.

- ***wget ftp://ftp.gnu.org/gnu/gcc/gcc-3.4.3/gcc-3.4.3.tar.bz2***

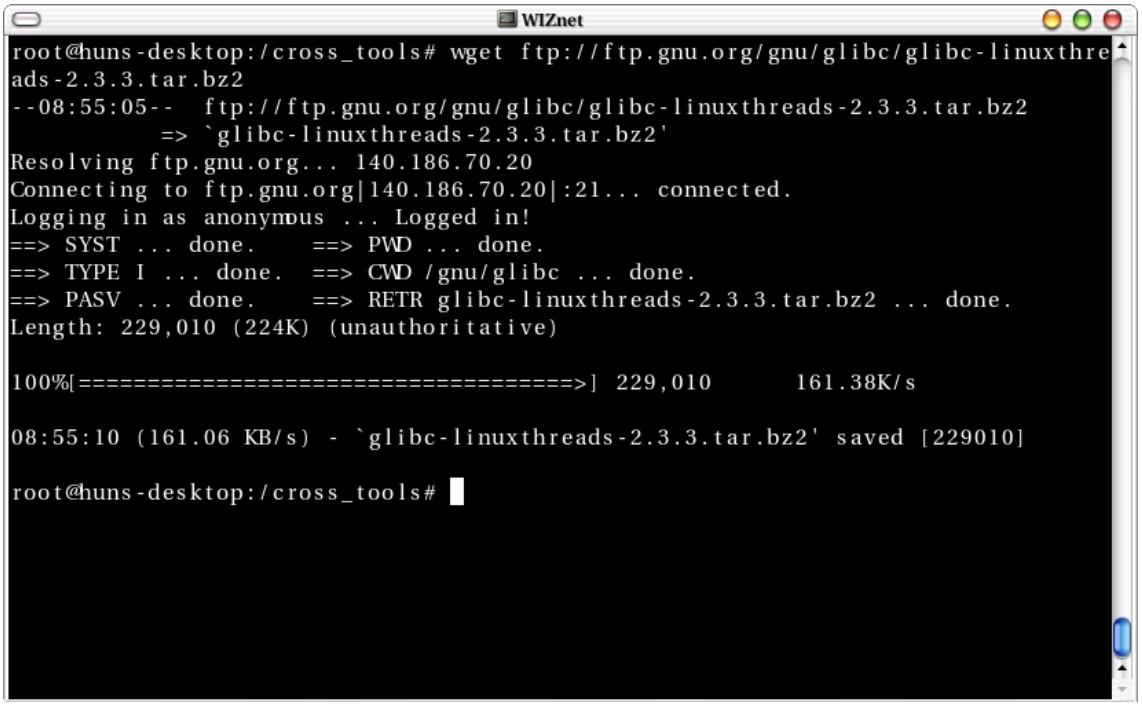
④ Download ‘glibc-2.3.3’ source code.

- ***wget ftp://ftp.gnu.org/gnu/glibc/glibc-2.3.3.tar.bz2***



⑤ Download ‘glibc-linuxthreads-2.3.3’ source code. ‘linuxthreads’ is the library to be included in ‘glibc’.

- ***wget ftp://ftp.gnu.org/gnu/glibc/glibc-linuxthreads-2.3.3.tar.bz2***



3 Linux Kernel Installation

The reason for installing Linux kernel source is to use the header file of the Linux kernel. Before installing tool chain, install the Linux kernel codes provided by W5300E01-ARM, and create 'linux/version.h' file.

① Mount the CD in the package of W5300E01-ARM, and install the kernel source code for W5300E01-ARM. If CD-ROM is automatically mounted, the mount directory can be different according to release version.

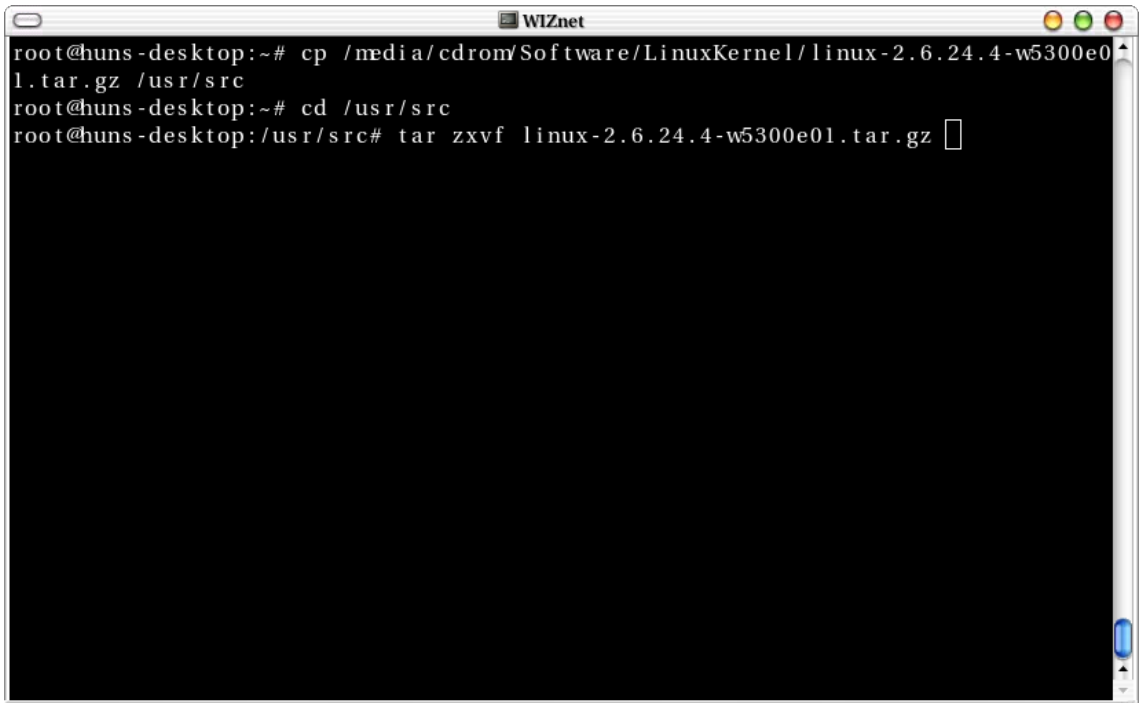
- ***mount /dev/cdrom /media/cdrom***

② Copy the kernel source into the '/usr/src' directory.

- ***cp /media/cdrom/Softare/LinuxKernel/linux-2.6.24.4-w5300e01.tar.gz /usr/src/***

③ Move to '/usr/src' directory, and extract the Linux kernel source file.

- ***cd /usr/src***
- ***tar zxvf linux-2.6.24.4-w5300e01.tar.gz***
- ***cd linux-2.6.24.4-w5300e01***



④ Create header file for compiler and symbolic link

- ***make prepare***

⑤ In order to simplify the directory name, make the symbolic link.

- ***cd ..***
- ***ln –s linux-2.6.24.4-w5300e01 linux***

4 'binutils' Installation

① Go to the working directory and extract the 'binutils' file.

- ***cd /cross_tools/***
- ***tar jxvf binutils-2.15.tar.bz2***
- ***cd binutils-2.15***

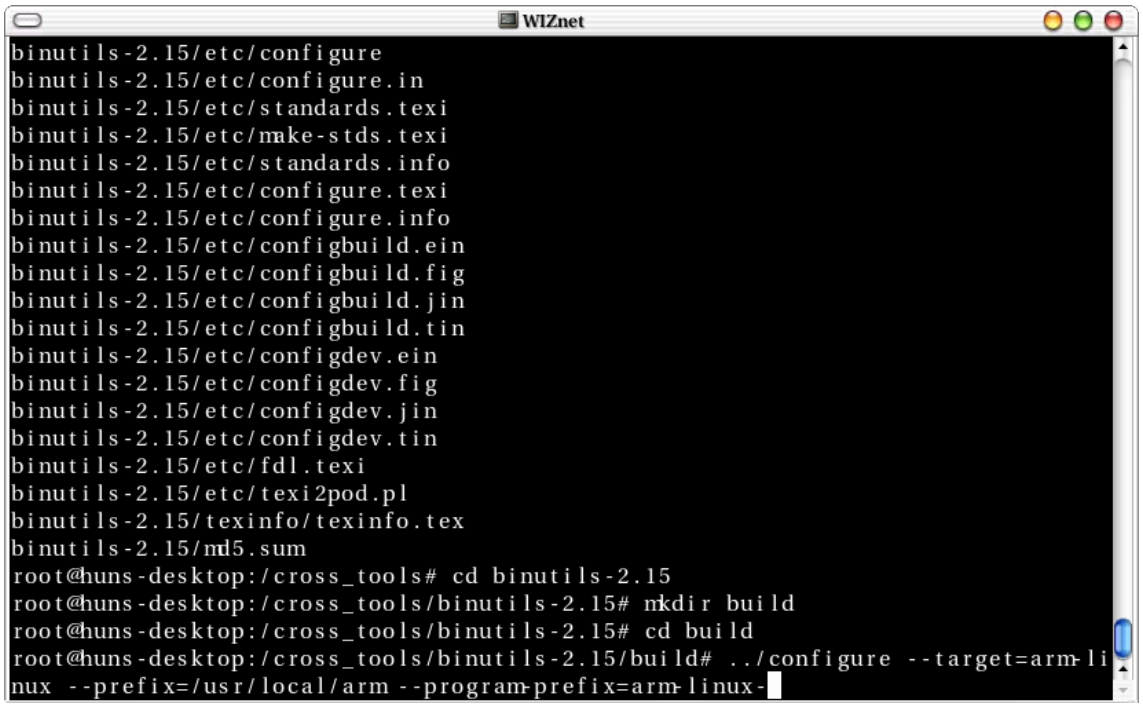
② Make the directory for 'binutils' compile.

- ***mkdir build***
- ***cd build***

③ Configure the compiling environment.

In order to prevent overwriting the existing compiler, designate the installation directory as '/usr/local/arm'.

- ***../configure –target=arm-linux –prefix=/usr/local/arm –program-prefix=arm-linux-***



④ Compile and install.

- ***make***
- ***make install***

⑤ Export the directory to the PATH.

- ***export PATH=/usr/local/arm/bin:$PATH***

5 'gcc' Installation

Install the 'gcc' for compiling 'glibc' library and linux kernel. After installing 'glibc', the 'gcc' for developing Embedded software should be re-installed.

① Move to working directory, and extract 'gcc' comperession.

- ***cd /cross_tools/***
- ***tar jxvf gcc-3.4.3.tar.bz2***
- ***cd gcc-3.4.3***

② **Modify 'gcc/config/arm/t-linux'** file. (line 3)

- < Before >

```
TARGET_LIBGCC2_CFLAGS = -fomit-frame-pointer -fPIC
```

- < After >

```
TARGET_LIBGCC2_CFLAGS = -fomit-frame-pointer –fPIC –Dinhibit_libc
–D__gthr_posix_h
```

③ **Add some contents to 'gcc/config/arm/t-linux'**.

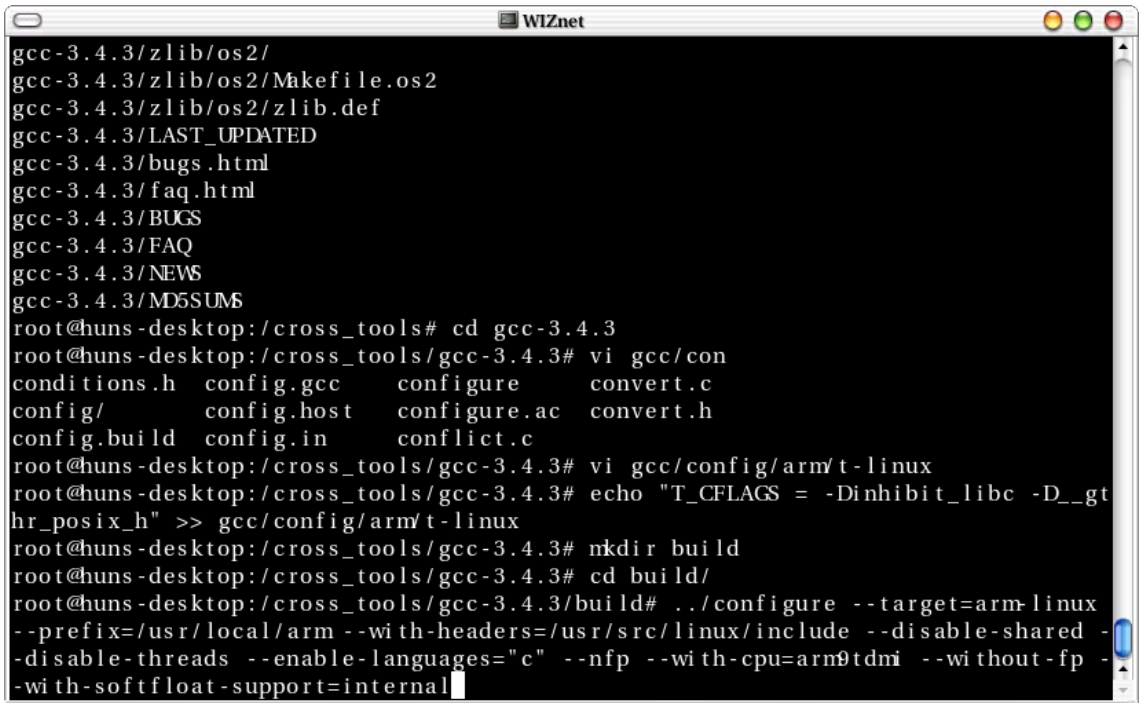
- ***echo "T_CFLAGS = -Dinhibit_libc –D__gthr_posix_h" >> gcc/config/arm/t-linux***

④ Make the directory for 'gcc' compiling.

- ***mkdir build***
- ***cd build***

⑤ Configure the environment for compile.

- ***../configure --target=arm-linux --prefix=/usr/local/arm ***
  ***--with-headers=/usr/src/linux/include --disable-shared --disable-threads ***
  ***--enable-languages="c" –nfp --with-cpu=arm9tdmi --without-fp ***
  --with-softfloat-support=internal

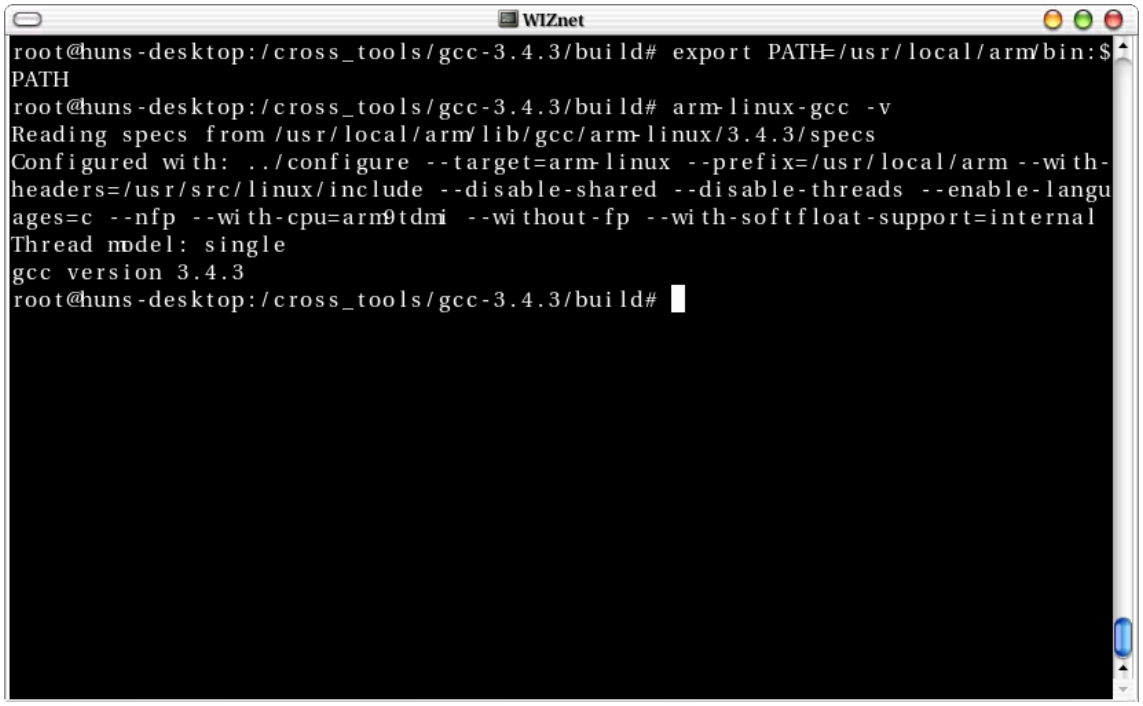


⑥ Compile and install.

- ***make***
- ***make install***

⑦ Check the version.

- ***arm-linux-gcc –v***



6 ‘glibc’ Installation

① Move to working directory and extract ‘glibc’ compression.

- ***cd /cross_tools/***
- ***tar jxvf glibc-2.3.3.tar.bz2***

② Extract ‘linuxhreads’ compresstion at the ‘glibc’ directory.

- ***tar –C glibc-2.3.3 –jxvf glibc-linuxthreads-2.3.3.tar.bz2***
- ***cd glibc-2.3.3***

③ Modify **‘Makeconfig’** file. (line 514)

- Before

```
gnulib := -lgcc –lgcc_eh
```

- After

```
gnulib := -lgcc
```

④ Modify **‘csu/Makefile’** file.(line 107)

- Before

```
CFLAGS-initfini.s = -g0 –fPIC –fno-inline-functions
```

- After

```
CFLAGS-initfini.s = -O1 –g0 –fPIC –fno-inline-functions
```

⑤ Modify **‘linuxthreads/Makefile’** file.(line 104)

- Before

```
CFLAGS-pt-initfini.s = -g0 –fPIC –fno-inline-functions
```

- After

```
CFLAGS-pt-initfini.s = -O1 –g0 –fPIC –fno-inline-functions
```

⑥ Modify **‘sysdeps/generic/framestate.c’** file. (line 44)

- Before

```
frame_state_for = fallback_frame_state_for;
```

➢ After

```
#ifndef __USING_SJLJ_EXCEPTIONS__
frame_state_for = fallback_frame_state_for;
#else
frame_state_for = abort;
#endif
```

⑦ Modify **'sysdeps/arm/machine-gmon.h'** file. (line 35 ~ 38)

➢ Before

```
static void mcount_internal (u_long frompc, u_long selfpc);

#define _MCOUNT_DECL(frompc, selfpc) \
static void mcount_internal (u_long frompc, u_long selfpc)
```

➢ After

```
#define _MCOUNT_DECL(frompc, selfpc) \
void mcount_internal (u_long frompc, u_long selfpc)
```

⑧ Modify **'sysdeps/unix/sysv/linux/arm/ioperm.c'** file. (line 98 ~ 104)

➢ Before

```
static int
init_iosys (void)
{
  char systype[256];
  int I, n;
  static int iobase_name[] = { CTL_BUS, BUS_ISA, BUS_ISA_PORT_BASE };
  static int ioshift_name[] = { CTL_BUS, BUS_ISA, BUS_ISA_PORT_SHIFT };
```

- After

```
#include <linux/version.h>
static int
init_iosys (void)
{
   char systype[256];
   int I, n;
#if LINUX_VERSION_CODE < 132119
   static int iobase_name[] = { CTL_BUS, BUS_ISA, BUS_ISA_PORT_BASE };
   static int ioshift_name[] = { CTL_BUS, BUS_ISA, BUS_ISA_PORT_SHIFT };
#else
   static int iobase_name[] = { CTL_BUS, CTL_BUS_ISA, BUS_ISA_PORT_BASE };
   static int ioshift_name[] = { CTL_BUS, CTL_BUS_ISA, BUS_ISA_PORT_SHIFT };
#endif
```

⑨ Create a directory for 'glibc' compilie.

- ***mkdir build***
- ***cd build***

⑩ Install compile environment.

- ***CC=arm-linux-gcc ../configure –host=arm-linux –build=i686-pc-linux-gnu ***
  ***--prefix=/usr/local/arm/arm-linux –with-headers=/usr/src/linux/include ***
  --enable-add-ons=linuxthreads –enable-shared

⑪ Perform compile. If you get some error at the**'csu/version-info.h'** file during compiling, open & modify the file and re-compile it. (line 1~4)

- ***make***
- Before

```
"Compiled on a Linux >>2.6.24-16-generic<< system on 2008-10-14
"
"Available extensions:
"
```

- After

```
"Compiled on a Linux >>2.6.24-16-generic<< system on 2008-10-14"
"Available extensions:"
```

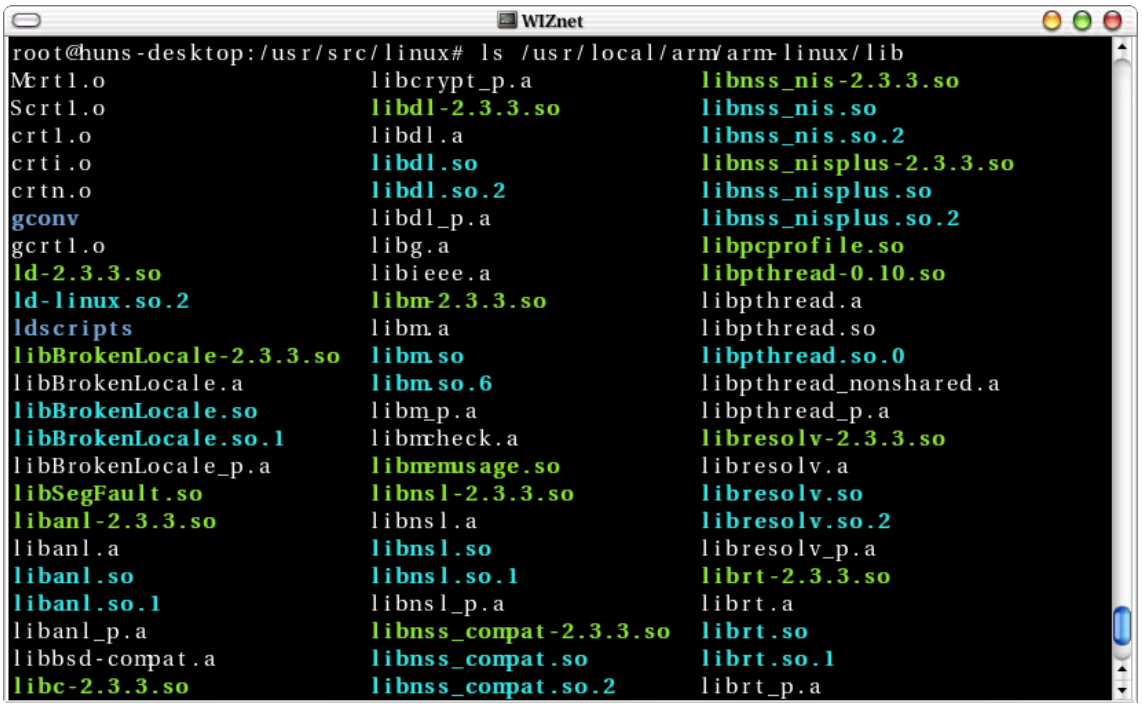
- ***make***

⑫ Install compiled ‘glibc’ library.

- ***make install***

⑬ Check installed ‘glibc’ library.

- ***ls /usr/local/arm/arm-linux/lib***



7 'gcc' Re-Installation

Until now, we have prepared the development environment for cross compiling. If we re-install 'gcc', cross-compiler installation is completed.

① Delete the 'gcc' source code used before..

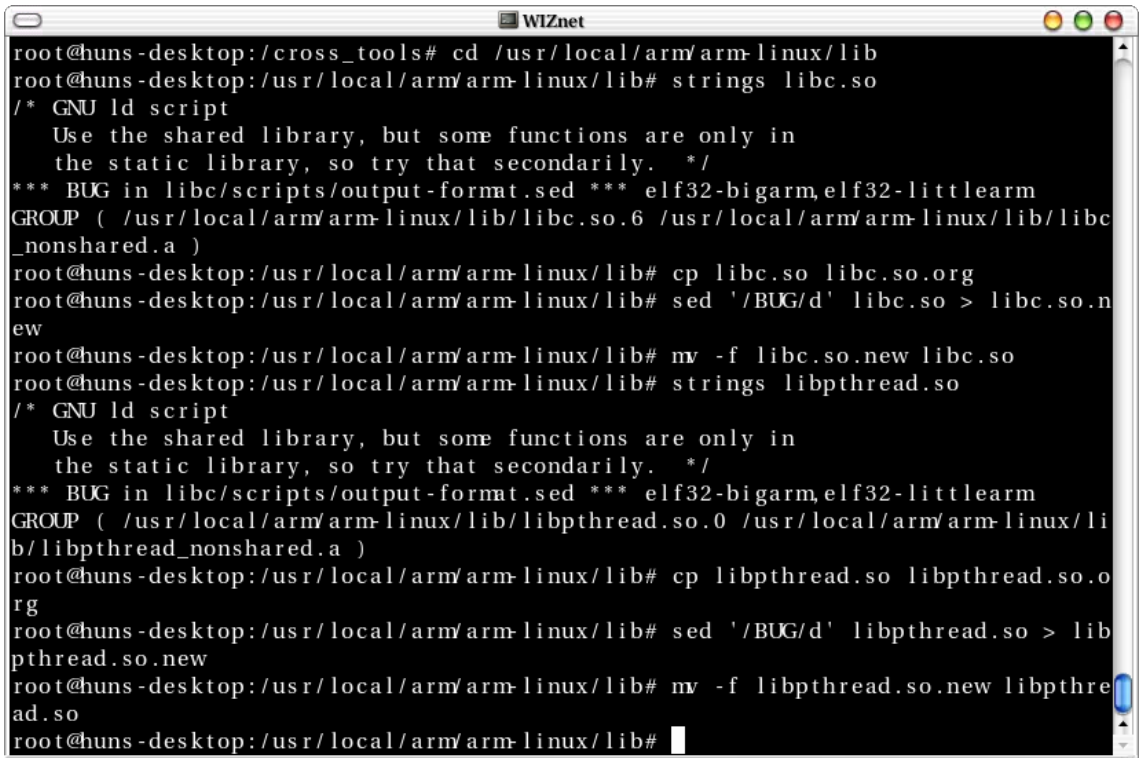
- ***cd /cross_tools/***
- ***rm –rf gcc-3.4.3***

② Extract 'gcc' source code.

- ***tar jxvf gcc-3.4.3.tar.bz2***

③ Below modification should be done to prevent library dependency.

- ***cd /usr/local/arm/arm-linux/lib***
- ***strings libc.so***
- ***cp libc.so libc.so.org***
- ***sed '/BUG/d' libc.so > libc.so.new***
- ***mv –f libc.so.new libc.so***
- ***strings libpthread.so***
- ***cp libpthread.so libpthread.so.org***
- ***sed '/BUG/d' libpthread.so > libpthread.so.new***
- ***mv –f libpthread.so.new libpthread.so***

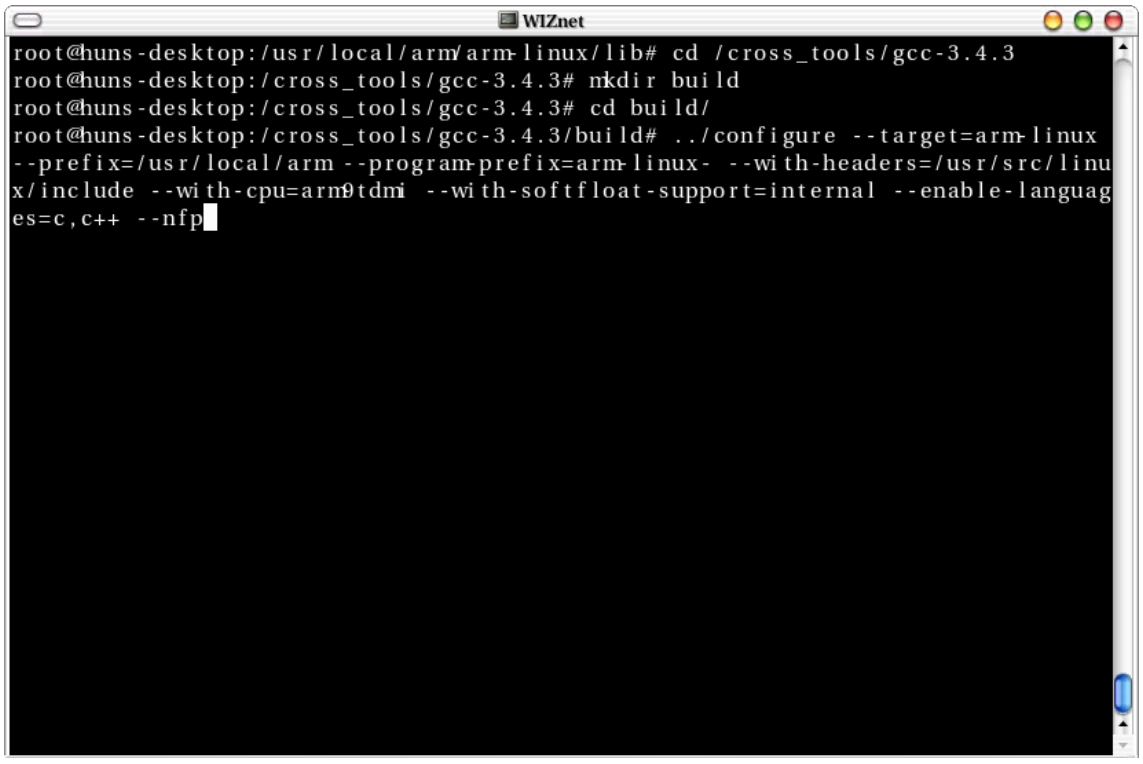


④ Create a directory for ‘gcc’ compiling.

- ***cd /cross_tools/gcc-3.4.3***
- ***mkdir build***
- ***cd build***

⑤ Configure compile environment.

- ***../configure –taeget=arm-linux –prefix=/usr/local/arm –program-prefix=arm-linux- ***
  ***--with-headers=/usr/src/linux/include –with-cpu=arm9tdmi ***
  --with-softfloat-support=internal –enable-languages=c,c++ --nfp



⑥ Compile and install

- ***make***
- ***make install***

8 Test

① For the testing, we will make example program ‘Hello WIZnet’.

➢ Write ‘/cross_tools/hello.c’

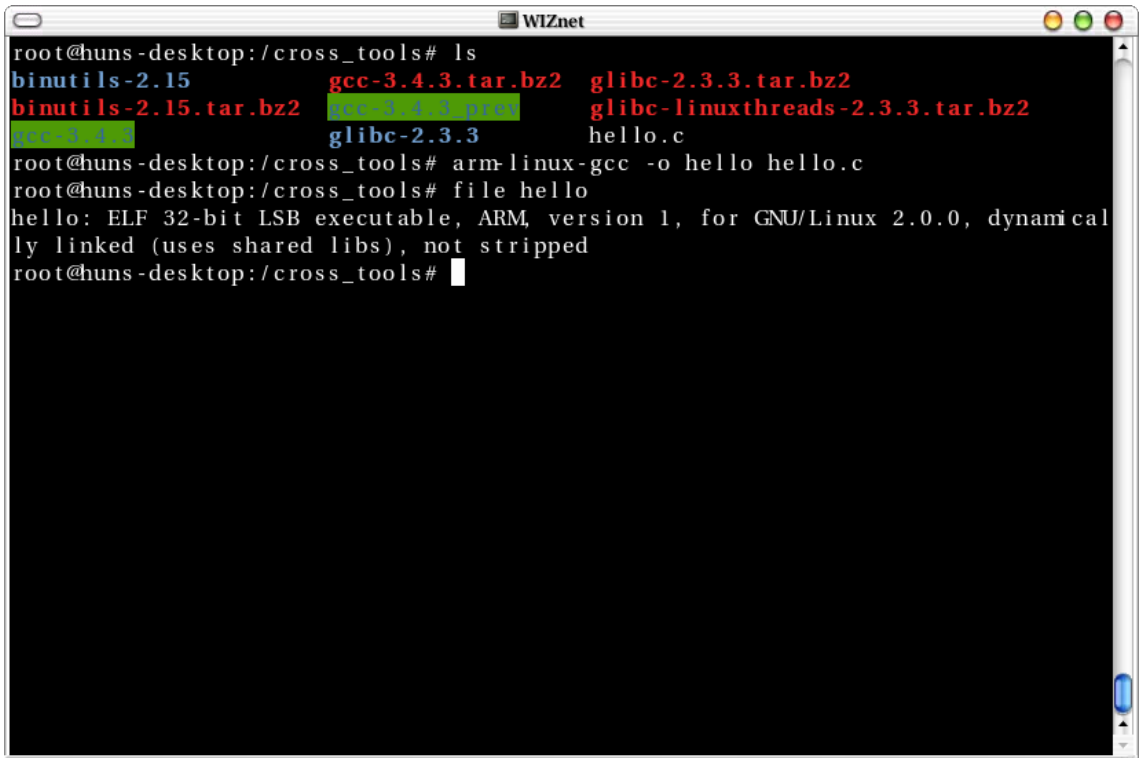
```
#include <stdio.h>


Int main(void)
{
   printf(“Hello WIZnet\n”);
   return 0;
}
```

② With newly created cross-compiler, compile ‘hello.c’ source code.

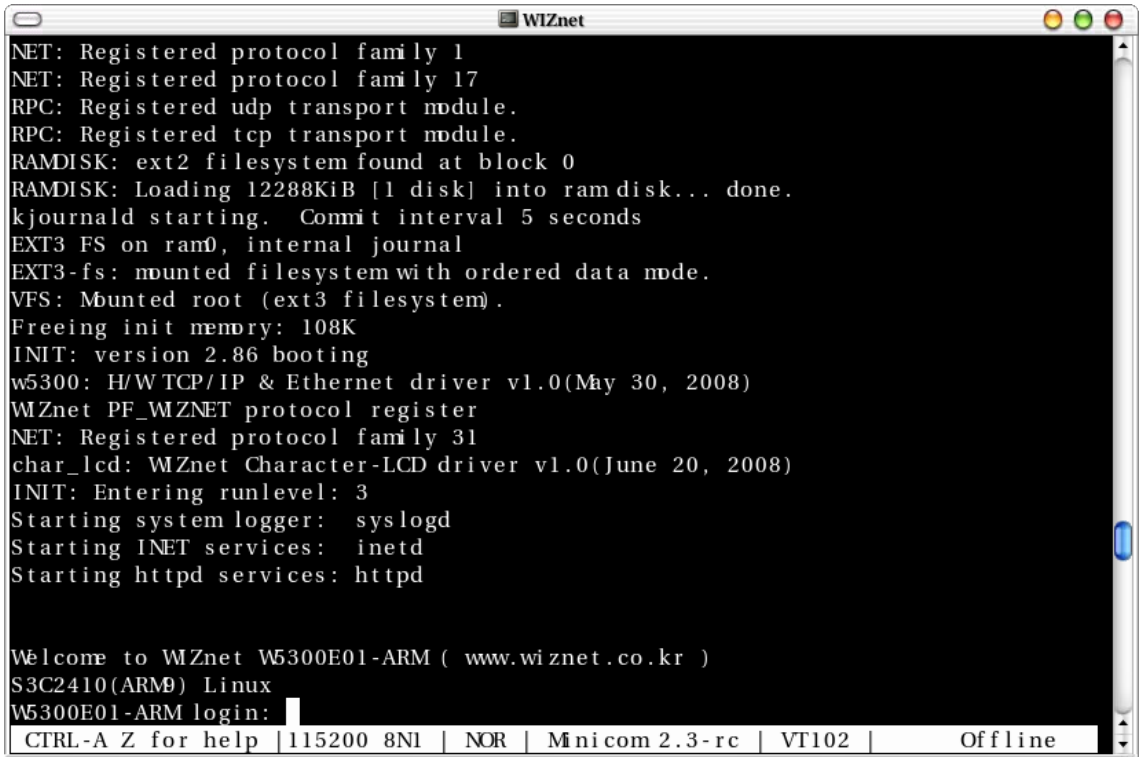
➢ ***arm-linux-gcc –o hello hello.c***

➢ ***file hello***



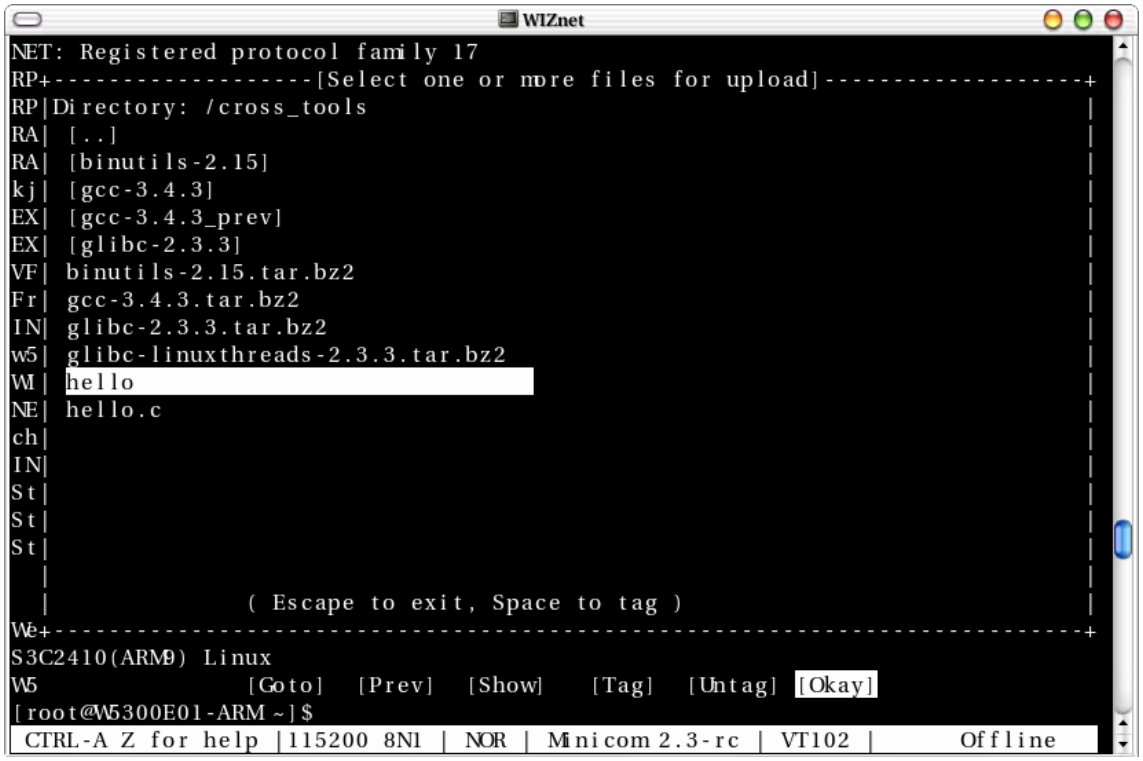
③ After connecting ‘W5300E01-ARM’ board and PC, execute ‘minicom’. After executing ‘minicom’, supply the power to the board. For the detailed information about ‘minicom’ configuration, refer to ‘W5300E01-ARM User Manual’, chapter ‘4.1.2 Booting Check at the Linux’.

➢ ***minicom***



④ If booting is completed, log in to ‘root’ and transmit the ‘hello’ program to serial. W5300

- root
- ‘Ctrl’ + ‘a’,’s’
- Select ‘Zmodem’
- Move to ‘/cross_tools’
- Select ‘hello’ file by using space key.
- Transmit the file from PC to ‘W5300E01-ARM’ by pressing enter key



⑤ Execute ‘hello’ program at the ‘W5300E01-ARM’ board.

- chmod 755 hello
- ./hello