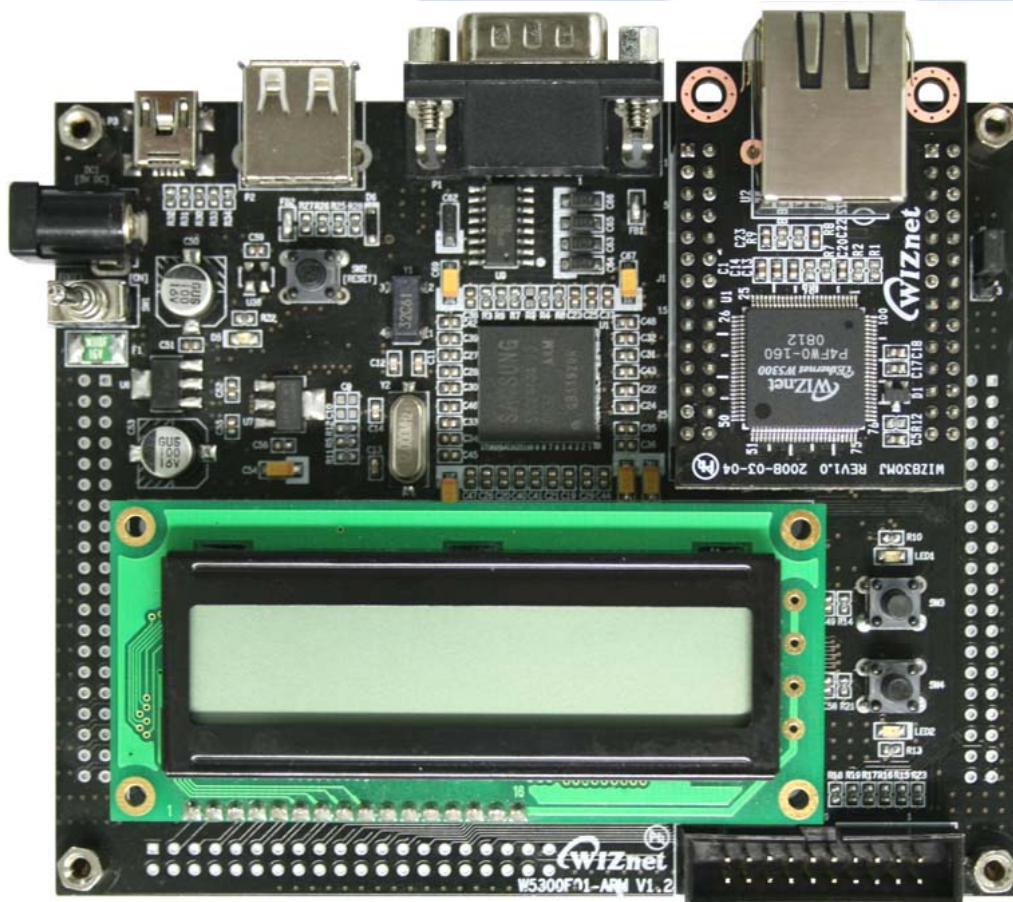


# W5300E01-ARM

## Cross Compiler User Manual

(Version1.0)



©2008 WIZnet Co., Ltd. All Rights Reserved.

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](http://www.wiznet.co.kr)). WIZnet Engineer will give an answer as soon as possible.

The screenshot shows the WIZnet website interface. At the top, there is a navigation bar with links for HOME, LOGIN, JOIN, CONTACT US, and language options (ENGLISH, CHINESE, JAPANESE, KOREAN). Below this is a main banner featuring a WIZnet W5300 chip and a WIZ600WI module. A sidebar on the left contains a menu with categories like PRODUCTS, TECHNOLOGY, TECHNICAL Q&A, LIBRARY, DISTRIBUTOR, PARTNERSHIP, BLOG, and ABOUT US. The main content area includes sections for RoHS compliance, NEWS LETTER, COMPANY OVERVIEW, NEW PRODUCT (highlighted with a red circle and 'Click' button), and DISTRIBUTOR. Below the banner, there is a secondary navigation bar with links for PRODUCTS, TECHNOLOGY, Q&A, LIBRARY, DISTRIBUTOR, PARTNERSHIP, BLOG, and ABOUT US. The Q&A section is prominently displayed, featuring a sidebar with Q&A, FAQ, and E-FORUM options. The main Q&A area contains a table of questions and answers.

Total : 2315 (1/116)				
NO	SUBJECT	NAME	DATE	HIT
2315	How to initialization TX_WR Pointer	Andria Ginting	2008-05-19	30
2314	ASRB=USB W3150A+	DOUG KHAN	2008-05-17	23
2313	re:ASRB=USB W3150A+	WIZnet	2008-05-21	3
2312	W5300 - Driver (Send function)	Ari Mendes dos Santos	2008-05-16	36
2311	why RX_WR_POINTER not return to zero	harry	2008-05-14	34
2310	UDP: Sn_RX_WR bug in W3150	Alex	2008-05-13	46
2309	UPDATE!!! UDP ERROR	Alex	2008-05-14	50
2308	recv size register does not return to ..	HARRY	2008-05-13	30
2307	re:recv size register does not return ..	WIZnet	2008-05-21	4

## COPYRIGHT NOTICE

Copyright 2008 WIZnet Co., Ltd. All Rights Reserved.

Technical Support: [support@wiznet.co.kr](mailto:support@wiznet.co.kr)

Sales & Distribution: [sales@wiznet.co.kr](mailto:sales@wiznet.co.kr)

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

## Table of Contents

1	INTRODUCTION .....	5
1.1	COMPOSITION OF CROSS COMPILER .....	5
1.2	BEFORE STARTING .....	5
2	SOURCE CODE DOWNLOAD .....	5
3	LINUX KERNEL INSTALLATION .....	9
4	'BINUTILS' INSTALLATION .....	10
5	'GCC' INSTALLATION .....	11
6	'GLIBC' INSTALLATION .....	13
7	'GCC' RE-INSTALLATION .....	17
8	TEST .....	19

## 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
- *glibc*
  - 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***

```
WIZnet
kth321@huns-desktop:~$ su
Password:
root@huns-desktop:/home/kth321# mkdir /cross_tools
root@huns-desktop:/home/kth321# cd /cross_tools/
root@huns-desktop:/cross_tools#
```

② Download 'binutils-2.15' source code.

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

```
WIZnet
root@huns-desktop:/cross_tools# wget ftp://ftp.gnu.org/gnu/binutils/binutils-2.15.tar.bz2
--17:25:34--  ftp://ftp.gnu.org/gnu/binutils/binutils-2.15.tar.bz2
           => `binutils-2.15.tar.bz2.1'
Resolving ftp.gnu.org... 140.186.70.20
Connecting to ftp.gnu.org|140.186.70.20|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.      ==> PWD ... done.
==> TYPE I ... done.   ==> CWD /gnu/binutils ... done.
==> PASV ... done.    ==> RETR binutils-2.15.tar.bz2 ... done.
Length: 11,515,075 (11M) (unauthoritative)

100%[=====>] 11,515,075      1.39Ms   ETA 00:00
17:25:48 (971.77 KB/s) - `binutils-2.15.tar.bz2.1' saved [11515075]
root@huns-desktop:/cross_tools#
```

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>

```
WIZnet
root@huns-desktop:/cross_tools# wget ftp://ftp.gnu.org/gnu/gcc/gcc-3.4.3/gcc-3.4.3.tar.bz2
--08:44:31--  ftp://ftp.gnu.org/gnu/gcc/gcc-3.4.3/gcc-3.4.3.tar.bz2
              => `gcc-3.4.3.tar.bz2.1'
Resolving ftp.gnu.org... 140.186.70.20
Connecting to ftp.gnu.org|140.186.70.20|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.      ==> PWD ... done.
==> TYPE I ... done.   ==> CWD /gnu/gcc/gcc-3.4.3 ... done.
==> PASV ... done.    ==> RETR gcc-3.4.3.tar.bz2 ... done.
Length: 27,425,338 (26M) (unauthoritative)

100%[=====] 27,425,338  174.69K/s  ETA 00:00

08:47:48 (138.29 KB/s) - `gcc-3.4.3.tar.bz2.1' saved [27425338]

root@huns-desktop:/cross_tools#
```

④ Download 'glibc-2.3.3' source code.

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

```
WIZnet
root@huns-desktop:/cross_tools# wget ftp://ftp.gnu.org/gnu/glibc/glibc-2.3.3.tar.bz2
--08:54:29--  ftp://ftp.gnu.org/gnu/glibc/glibc-2.3.3.tar.bz2
              => `glibc-2.3.3.tar.bz2'
Resolving ftp.gnu.org... 140.186.70.20
Connecting to ftp.gnu.org|140.186.70.20|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.      ==> PWD ... done.
==> TYPE I ... done.   ==> CWD /gnu/glibc ... done.
==> PASV ... done.    ==> RETR glibc-2.3.3.tar.bz2 ... done.
Length: 13,053,127 (12M) (unauthoritative)

100%[=====] 13,053,127  1021.40K/s  ETA 00:00

08:54:46 (915.70 KB/s) - `glibc-2.3.3.tar.bz2' saved [13053127]

root@huns-desktop:/cross_tools#
```

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>

```
root@huns-desktop: /cross_tools# wget ftp://ftp.gnu.org/gnu/glibc/glibc-linuxthreads-2.3.3.tar.bz2
--08:55:05-- ftp://ftp.gnu.org/gnu/glibc/glibc-linuxthreads-2.3.3.tar.bz2
=> `glibc-linuxthreads-2.3.3.tar.bz2'
Resolving ftp.gnu.org... 140.186.70.20
Connecting to ftp.gnu.org[140.186.70.20]:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.      ==> PWD ... done.
==> TYPE I ... done.   ==> CWD /gnu/glibc ... done.
==> PASV ... done.     ==> RETR glibc-linuxthreads-2.3.3.tar.bz2 ... done.
Length: 229,010 (224K) (unauthoritative)

100%[=====] 229,010      161.38K/s

08:55:10 (161.06 KB/s) - `glibc-linuxthreads-2.3.3.tar.bz2' saved [229010]

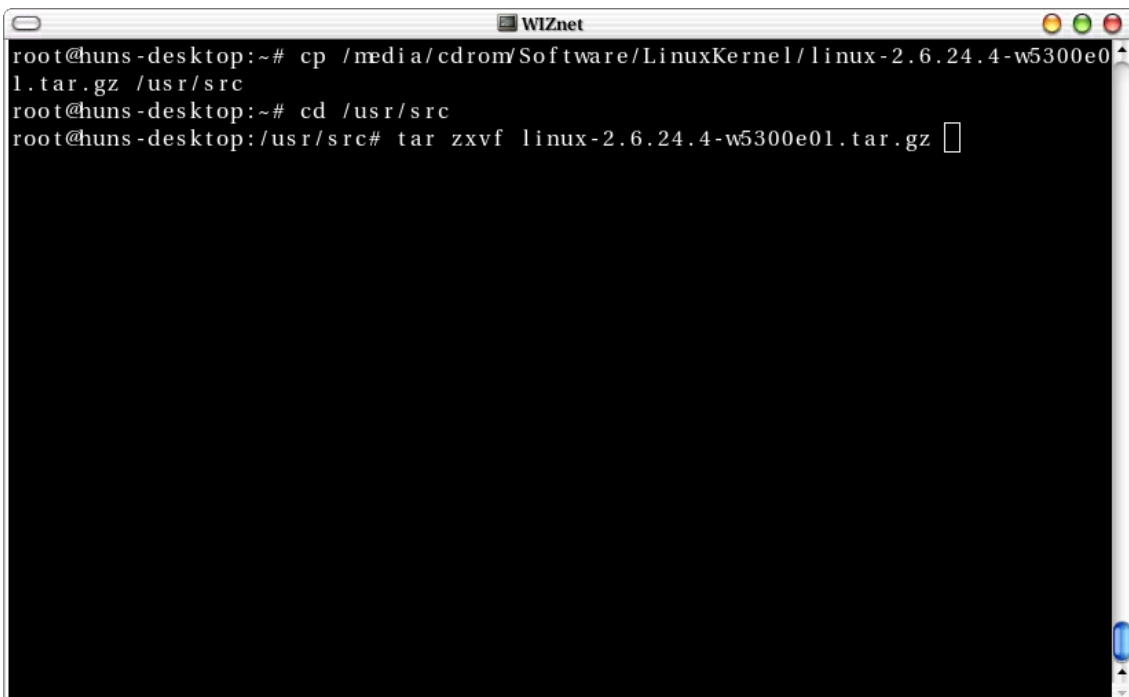
root@huns-desktop: /cross_tools#
```



### 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/Software/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`**



```
root@huns-desktop:~# cp /media/cdrom/Software/LinuxKernel/linux-2.6.24.4-w5300e01.tar.gz /usr/src
root@huns-desktop:~# cd /usr/src
root@huns-desktop:/usr/src# tar zxvf linux-2.6.24.4-w5300e01.tar.gz
```

- 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`**

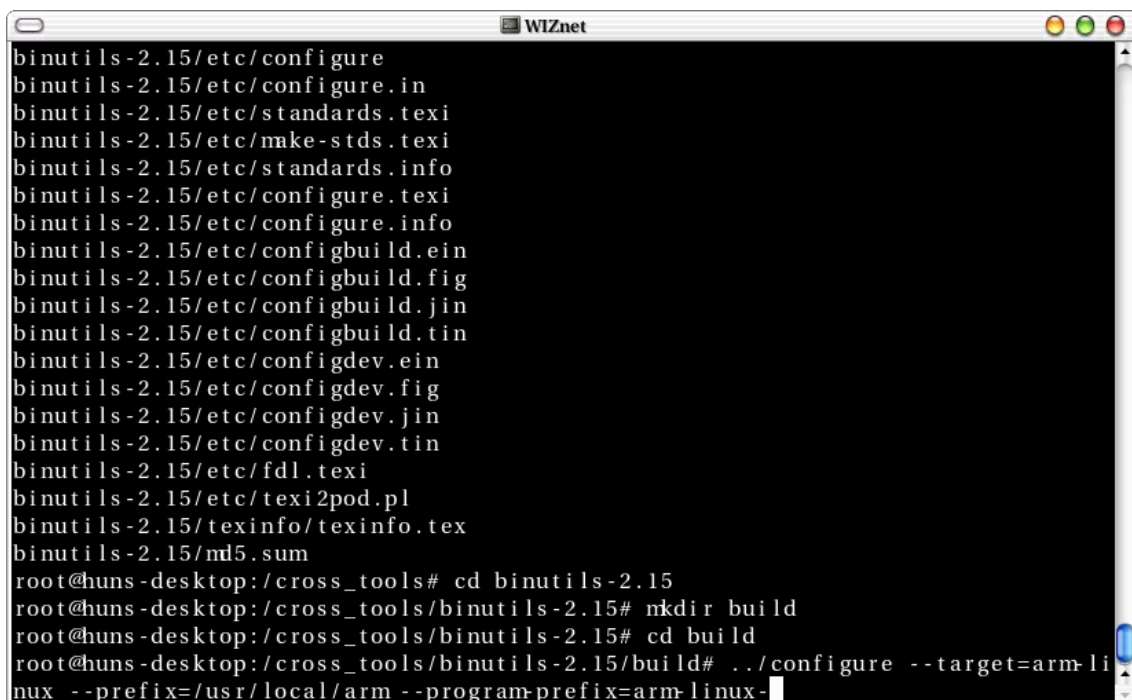
- ② Make the directory for 'binutils' compile.

- **`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-`**



```
binutils-2.15/etc/configure
binutils-2.15/etc/configure.in
binutils-2.15/etc/standards.texi
binutils-2.15/etc/make-stds.texi
binutils-2.15/etc/standards.info
binutils-2.15/etc/configure.texi
binutils-2.15/etc/configure.info
binutils-2.15/etc/configbuild.ein
binutils-2.15/etc/configbuild.fig
binutils-2.15/etc/configbuild.jin
binutils-2.15/etc/configbuild.tin
binutils-2.15/etc/configdev.ein
binutils-2.15/etc/configdev.fig
binutils-2.15/etc/configdev.jin
binutils-2.15/etc/configdev.tin
binutils-2.15/etc/fdl.texi
binutils-2.15/etc/textipod.pl
binutils-2.15/texinfo/texinfo.tex
binutils-2.15/md5.sum
root@huns-desktop:/cross_tools# cd binutils-2.15
root@huns-desktop:/cross_tools/binutils-2.15# mkdir build
root@huns-desktop:/cross_tools/binutils-2.15# cd build
root@huns-desktop:/cross_tools/binutils-2.15/build# ../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' compression.

- ***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***

```
gcc-3.4.3/zlib/os2/
gcc-3.4.3/zlib/os2/Makefile.os2
gcc-3.4.3/zlib/os2/zlib.def
gcc-3.4.3/LAST_UPDATED
gcc-3.4.3/bugs.html
gcc-3.4.3/faq.html
gcc-3.4.3/BUGS
gcc-3.4.3/FAQ
gcc-3.4.3/NEWS
gcc-3.4.3/MD5SUMS
root@huns-desktop:/cross_tools# cd gcc-3.4.3
root@huns-desktop:/cross_tools/gcc-3.4.3# vi gcc/con
conditions.h  config.gcc  configure  convert.c
config/      config.host  configure.ac  convert.h
config.build  config.in    conflict.c
root@huns-desktop:/cross_tools/gcc-3.4.3# vi gcc/config/arm/t-linux
root@huns-desktop:/cross_tools/gcc-3.4.3# echo "T_CFLAGS = -Dinhibit_libc -D_gt
hr_posix_h" >> gcc/config/arm/t-linux
root@huns-desktop:/cross_tools/gcc-3.4.3# mkdir build
root@huns-desktop:/cross_tools/gcc-3.4.3# cd build/
root@huns-desktop:/cross_tools/gcc-3.4.3/build# ../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**

```
root@huns-desktop:/cross_tools/gcc-3.4.3/build# export PATH=/usr/local/arm/bin:$
PATH
root@huns-desktop:/cross_tools/gcc-3.4.3/build# arm-linux-gcc -v
Reading specs from /usr/local/arm/lib/gcc/arm-linux/3.4.3/specs
Configured with: ../configure --target=arm-linux --prefix=/usr/local/arm --with-
headers=/usr/src/linux/include --disable-shared --disable-threads --enable-langu
ages=c --nfp --with-cpu=arm9tdmi --without-fp --with-softfloat-support=internal
Thread model: single
gcc version 3.4.3
root@huns-desktop:/cross_tools/gcc-3.4.3/build#
```

## 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 'linuxthreads' compression 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 l, 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 l, 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' compile.

➤ **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**

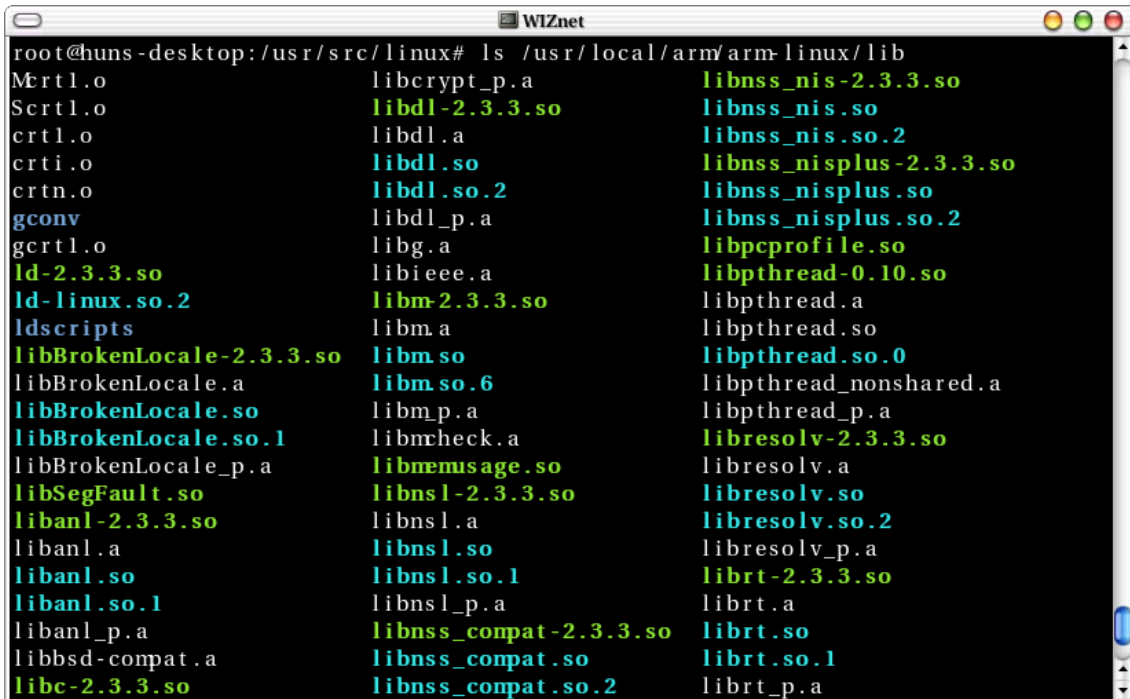
➤ Before

```
"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**



```

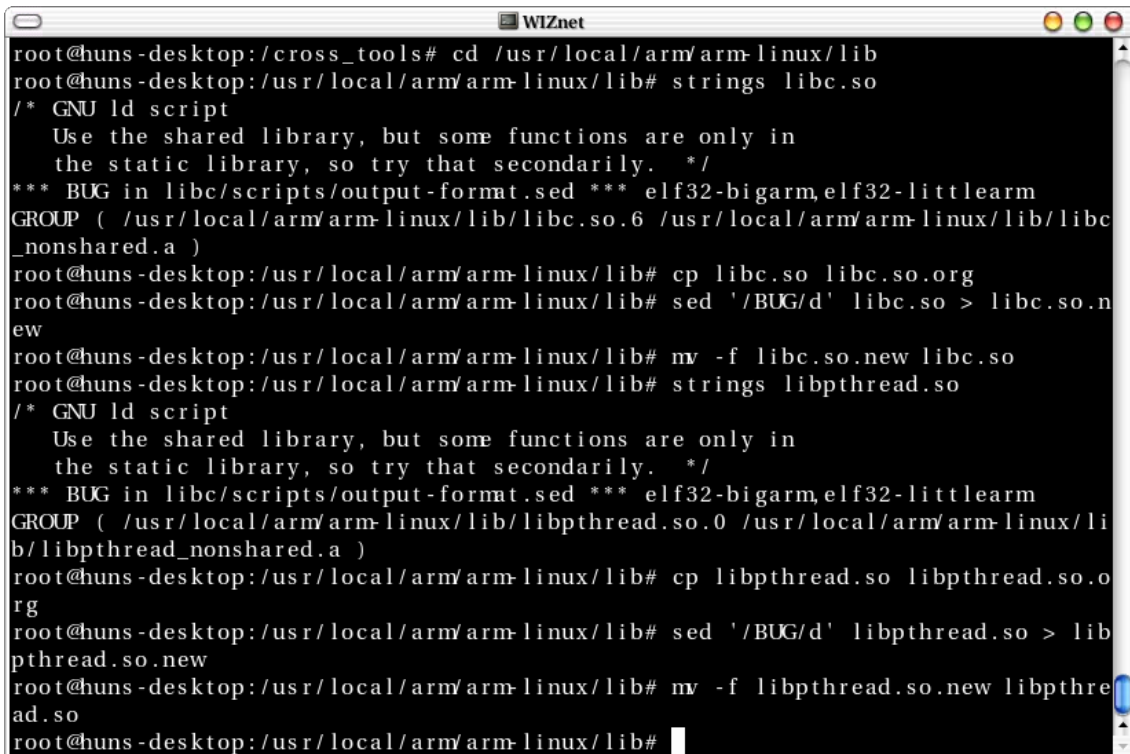
root@huns-desktop: /usr/src/linux# ls /usr/local/arm/arm-linux/lib
Mrt1.o          libcrypt_p.a      libnss_nis-2.3.3.so
Sert1.o         libdl-2.3.3.so    libnss_nis.so
crt1.o          libdl.a           libnss_nis.so.2
crti.o          libdl.so          libnss_nisplus-2.3.3.so
crtn.o          libdl.so.2        libnss_nisplus.so
gconv           libdl_p.a         libnss_nisplus.so.2
gcrt1.o         libg.a            libpcprofile.so
ld-2.3.3.so     libieee.a         libpthread-0.10.so
ld-linux.so.2  libm-2.3.3.so     libpthread.a
ldscripts      libm.a            libpthread.so
libBrokenLocale-2.3.3.so libm.so           libpthread.so.0
libBrokenLocale.a libm.so.6         libpthread_nonshared.a
libBrokenLocale.so libm_p.a          libpthread_p.a
libBrokenLocale.so.1 libmcheck.a       libresolv-2.3.3.so
libBrokenLocale_p.a libmemusage.so    libresolv.a
libSegFault.so  libnsl-2.3.3.so   libresolv.so
libanl-2.3.3.so libnsl.a          libresolv.so.2
libanl.a        libnsl.so         libresolv_p.a
libanl.so       libnsl.so.1       librt-2.3.3.so
libanl.so.1     libnsl_p.a        librt.a
libanl_p.a      libnss_compat-2.3.3.so librt.so
libbsd-compat.a libnss_compat.so  librt.so.1
libc-2.3.3.so   libnss_compat.so.2 librt_p.a
  
```



## 7 'gcc' Re-Installation

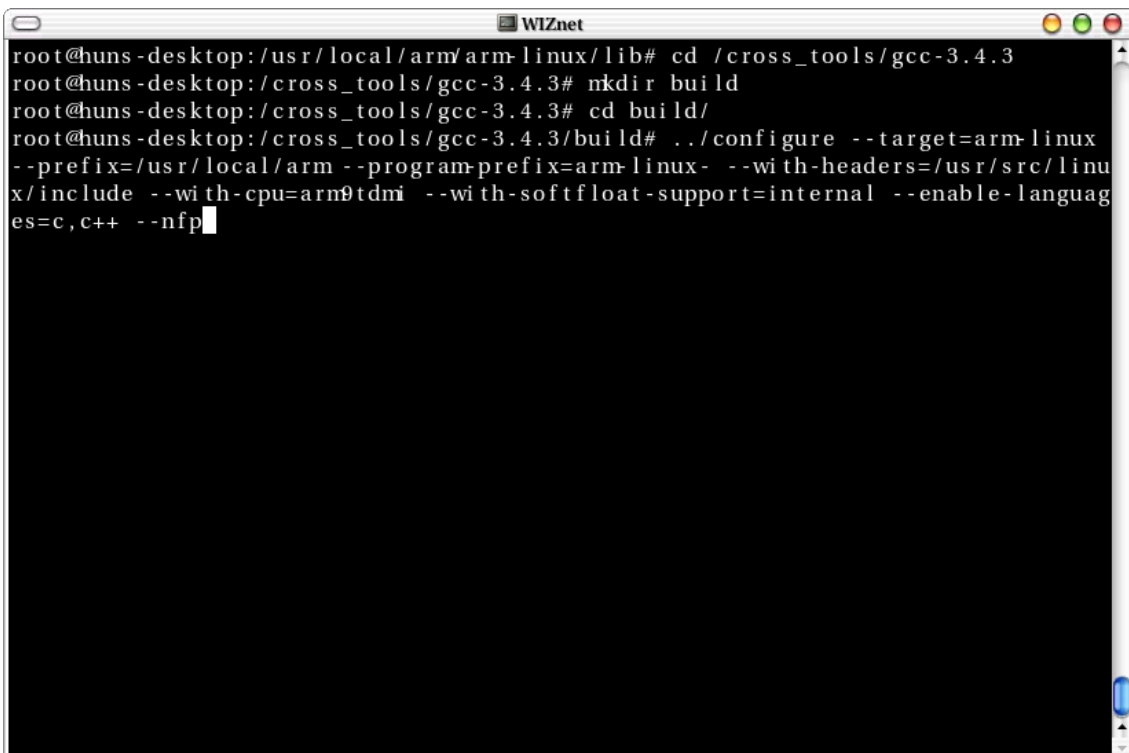
Until now, we have prepared the development environment for cross compiling. If we re-install 'gcc', cross-compiler installation is completed.

- 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**



```
root@huns-desktop:/cross_tools# cd /usr/local/arm/arm-linux/lib
root@huns-desktop:/usr/local/arm/arm-linux/lib# strings libc.so
/* GNU ld script
Use the shared library, but some functions are only in
the static library, so try that secondarily. */
*** BUG in libc/scripts/output-format.sed *** elf32-bigarm,elf32-littlearm
GROUP ( /usr/local/arm/arm-linux/lib/libc.so.6 /usr/local/arm/arm-linux/lib/libc
_nonshared.a )
root@huns-desktop:/usr/local/arm/arm-linux/lib# cp libc.so libc.so.org
root@huns-desktop:/usr/local/arm/arm-linux/lib# sed '/BUG/d' libc.so > libc.so.n
ew
root@huns-desktop:/usr/local/arm/arm-linux/lib# mv -f libc.so.new libc.so
root@huns-desktop:/usr/local/arm/arm-linux/lib# strings libpthread.so
/* GNU ld script
Use the shared library, but some functions are only in
the static library, so try that secondarily. */
*** BUG in libc/scripts/output-format.sed *** elf32-bigarm,elf32-littlearm
GROUP ( /usr/local/arm/arm-linux/lib/libpthread.so.0 /usr/local/arm/arm-linux/li
b/libpthread_nonshared.a )
root@huns-desktop:/usr/local/arm/arm-linux/lib# cp libpthread.so libpthread.so.o
rg
root@huns-desktop:/usr/local/arm/arm-linux/lib# sed '/BUG/d' libpthread.so > lib
pthread.so.new
root@huns-desktop:/usr/local/arm/arm-linux/lib# mv -f libpthread.so.new libpthre
ad.so
root@huns-desktop:/usr/local/arm/arm-linux/lib#
```

- ④ Create a directory for 'gcc' compiling.
  - **`cd /cross_tools/gcc-3.4.3`**
  - **`mkdir build`**
  - **`cd build`**
- ⑤ Configure compile environment.
  - **`../configure --target=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`**



```
root@huns-desktop: /usr/local/arm/arm-linux/lib# cd /cross_tools/gcc-3.4.3
root@huns-desktop: /cross_tools/gcc-3.4.3# mkdir build
root@huns-desktop: /cross_tools/gcc-3.4.3# cd build/
root@huns-desktop: /cross_tools/gcc-3.4.3/build# ../configure --target=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'

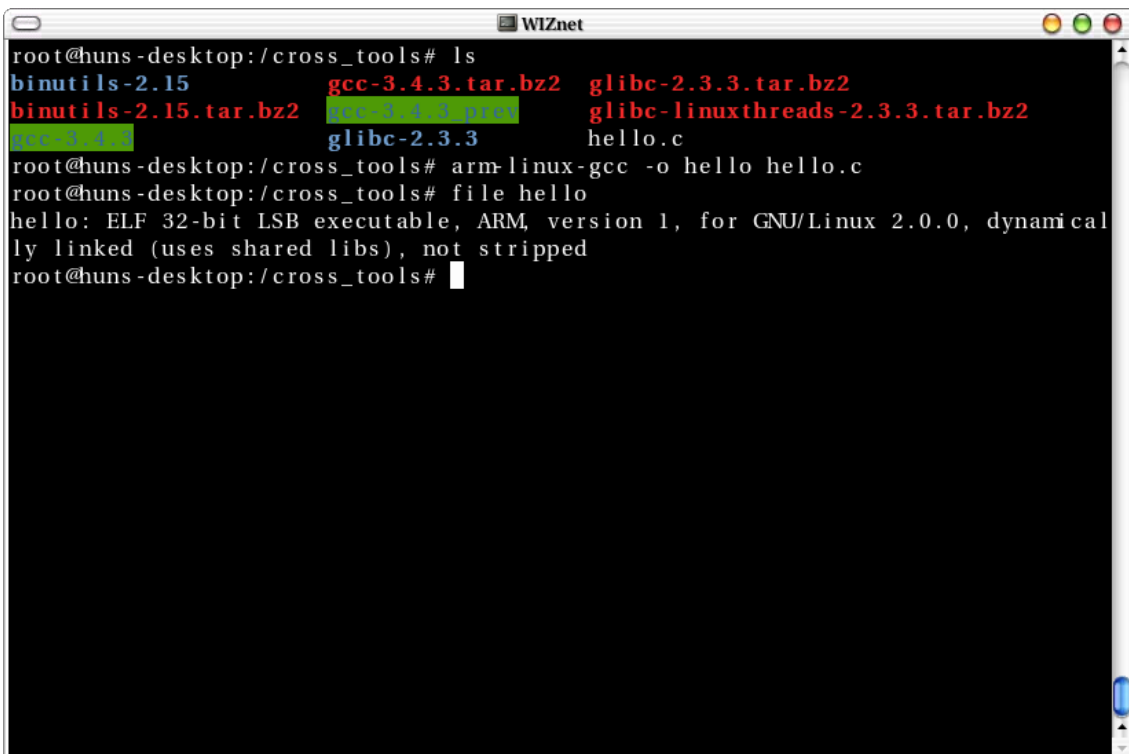
```
#include <stdio.h>

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***

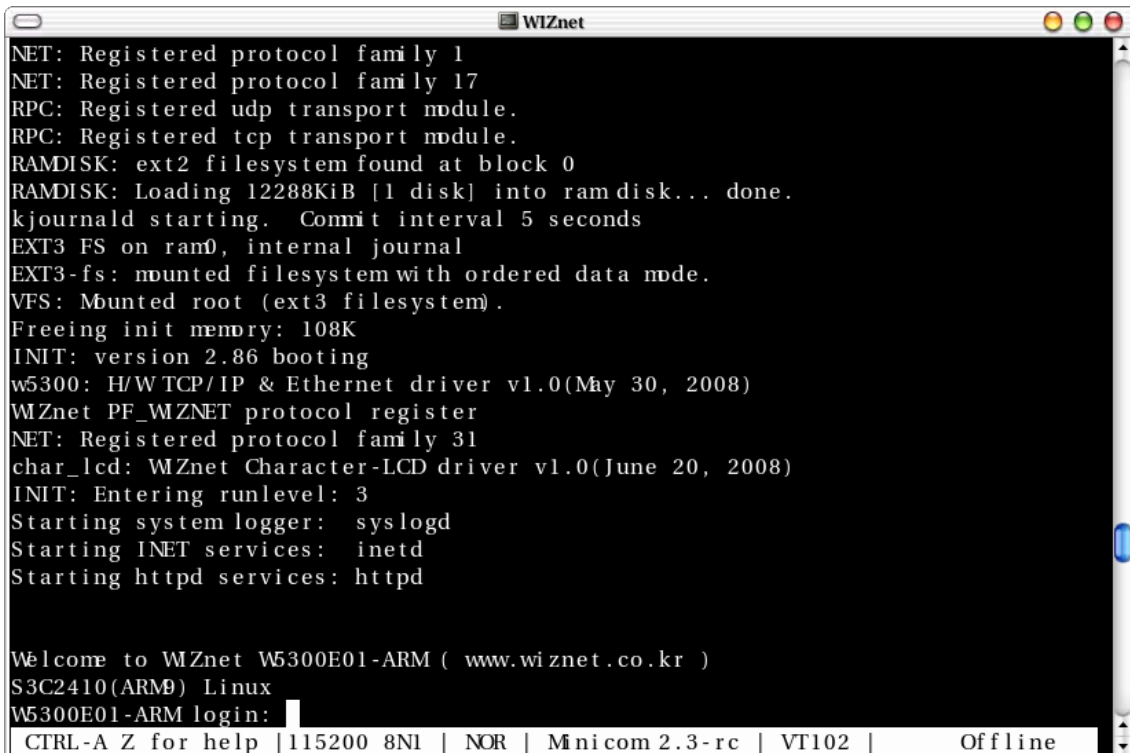
➤ ***file hello***



```
root@huns-desktop: /cross_tools# ls
binutils-2.15      gcc-3.4.3.tar.bz2  glibc-2.3.3.tar.bz2
binutils-2.15.tar.bz2  gcc-3.4.3-proc    glibc-linuxthreads-2.3.3.tar.bz2
gcc-3.4.3        glibc-2.3.3      hello.c
root@huns-desktop: /cross_tools# arm-linux-gcc -o hello hello.c
root@huns-desktop: /cross_tools# file hello
hello: ELF 32-bit LSB executable, ARM, version 1, for GNU/Linux 2.0.0, dynamically linked (uses shared libs), not stripped
root@huns-desktop: /cross_tools#
```

- 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***



```
WIZnet
NET: Registered protocol family 1
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 12288KiB [1 disk] into ram disk... done.
kjournald starting. Commit interval 5 seconds
EXT3 FS on ram0, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem).
Freeing init memory: 108K
INIT: version 2.86 booting
w5300: H/W TCP/IP & Ethernet driver v1.0(May 30, 2008)
WIZnet PF_WZNET protocol register
NET: Registered protocol family 31
char_lcd: WIZnet Character-LCD driver v1.0(June 20, 2008)
INIT: Entering runlevel: 3
Starting system logger: syslogd
Starting INET services: inetd
Starting httpd services: httpd

Welcome to WIZnet W5300E01-ARM ( www.wiznet.co.kr )
S3C2410(ARM9) Linux
W5300E01-ARM login:
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.3-rc | VT102 | Offline
```

- 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

```
WIZnet
NET: Registered protocol family 17
RP+-----[Select one or more files for upload]-----+
RP|Directory: /cross_tools
RA| [..]
RA| [binutils-2.15]
kj| [gcc-3.4.3]
EX| [gcc-3.4.3_prev]
EX| [glibc-2.3.3]
VF| binutils-2.15.tar.bz2
Fr| gcc-3.4.3.tar.bz2
IN| glibc-2.3.3.tar.bz2
w5| glibc-linuxthreads-2.3.3.tar.bz2
W | hello
NE| hello.c
ch|
IN|
St|
St|
St|
|
| ( Escape to exit, Space to tag )
We+-----+
S3C2410(ARM) Linux
W5 [Goto] [Prev] [Show] [Tag] [Untag] [Okay]
[root@W5300E01-ARM ~]$
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.3-rc | VT102 | Offline
```

- ⑤ Execute 'hello' program at the 'W5300E01-ARM' board.
  - chmod 755 hello
  - ./hello

```
WIZnet
[root@W5300E01-ARM ~]$ chmod 755 hello
[root@W5300E01-ARM ~]$ ./hello
Hello WIZnet
[root@W5300E01-ARM ~]$
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.3-rc | VT102 | Offline
```