

Cross-Compiler mini-Howto

Gerd Knorr and Joachim Nilsson krazel@goldbach.in-berlin.de, crash@vmlinux.org

August 30 1999, May 31, 2000

Sammanfattning

Introduction This is a short description how to build and use a cross compiler. In this brief tutorial we will guide you through the making of an Intel-to-SPARC cross compiler, the practical use of this is quite obvious to anyone who has got a beaten up SPARC SLC or ELC. Building the Linux kernel and various other applications on a Pentium Pro takes much less time than compiling directly on the SPARC.

This tutorial is also useful to anyone who is in need of a Intel x86 hosted cross compiler for the Power PC platform, i.e. when building applications for embedded use.

The contents of the tutorial can be seen as a brief introduction as well as a good starting point for further studies in the field.

1 Introduction

This is a short description on how to build and use a cross compiler. We've built Intel-to-SPARC and Intel-to-PPC cross compilers, because compiling kernels on a Pentium Pro takes *much* less time than compiling on an old SPARC SLC and is also sometimes impossible to do for an embedded system. Examples of both will be given.

See the list of available targets at the end.

2 Step one: build binutils

First, you need a cross-linker and cross-assembler for your target platform. Therefore, the first step is to compile the binutils. Binutils contain

If you have the Red Hat CD's offhand, you can pull the sources out of the rpm:

```
$ rpm2cpio /vol/cd2/SRPMS/binutils-2.9.1.0.4-2.src.rpm | \
cpio --extract
```

Otherwise you can download them from your favorite sun-site mirror <http://metalab.unc.edu/pub/Linux/MIRRORS.html>.

Now untar the binutils source code (`tar xvzf binutils-2.9.1.0.4.tar.gz`), change to into the new directory and configure the sources for cross-compiling:

```
$ ./configure --prefix=/usr/local --target=sparc-linux \
i486-redhat-linux
```

Type `make` to compile the whole thing now. Time for a break, this will take some time. When it is finished, install it (as root) with `make install`.

You should have the programs installed in `/usr/local/bin` now:

```
bogomips root /usr/local/bin# ll sparc-linux-*
-rwxr-xr-x  1 root    root      209428 Aug 16 10:20 sparc-linux-addr2line*
-rwxr-xr-x  2 root    root      210204 Aug 16 10:20 sparc-linux-ar*
-rwxr-xr-x  2 root    root      376860 Aug 16 10:20 sparc-linux-as*
```

```

-rwxr-xr-x 1 root root 21004 Aug 16 10:20 sparc-linux-c++filt*
-rwxr-xr-x 1 root root 36104 Aug 16 10:20 sparc-linux-gasp*
-rwxr-xr-x 2 root root 337620 Aug 16 10:20 sparc-linux-ld*
-rwxr-xr-x 2 root root 216516 Aug 16 10:20 sparc-linux-nm*
-rwxr-xr-x 1 root root 325244 Aug 16 10:20 sparc-linux-objcopy*
-rwxr-xr-x 1 root root 394076 Aug 16 10:20 sparc-linux-objdump*
-rwxr-xr-x 2 root root 210204 Aug 16 10:20 sparc-linux-ranlib*
-rwxr-xr-x 1 root root 195300 Aug 16 10:20 sparc-linux-size*
-rwxr-xr-x 1 root root 194860 Aug 16 10:20 sparc-linux-strings*
-rwxr-xr-x 2 root root 325244 Aug 16 10:20 sparc-linux-strip*

```

You should have a new directory called `/usr/local/sparc-linux` too:

```

bogomips root ~# ls /usr/local/sparc-linux
bin/  lib/

```

2.0.1 Step two: install include files and libraries

The new `/usr/local/sparc-linux` is the directory where the cross-compiler will look for libraries and includes first. So they should be installed there. I did it this way:

1. create and go to some temporary directory
2. unpack the glibc rpms there:

```

$ rpm2cpio /path/to/RPMS/glibc-2.0.7-17.sparc.rpm | \
cpio --extract --make-directories
$ rpm2cpio /path/to/RPMS/glibc-devel-2.0.7-17.sparc.rpm | \
cpio --extract --make-directories

```

3. copy the includes

```

$ cp -a usr/include /usr/local/sparc-linux

```

4. add the kernel headers

```

$ cp -a /usr/src/linux/include/linux \
/usr/local/sparc-linux/include/linux
$ cp -a /usr/src/linux/include/asm-sparc \
/usr/local/sparc-linux/include/asm

```

5. copy the libraries

```

$ cp -a lib/* /usr/local/sparc-linux/lib
$ cp -a usr/lib/* /usr/local/sparc-linux/lib

```

6. remove the temporary directory

Now we have to fix a few things. There are some broken symbolic links, because we have everything in a single directory and not split into `/lib` and `/usr/lib`. I am too lazy to do this by hand and did it with a small script:

```

$ cd /usr/local/sparc-linux/lib
$ ls -l | grep "../../lib" | sed 's|../../lib/||' | \
awk '{ print "ln -sf", $11, $9 }' | tee fixit
$ sh fixit
$ rm fixit

```

Another pitfall is the `libc.so`. This isn't a real shared library, but a linker script (simple text, use your favorite editor for editing). The paths in that file must be fixed there too, using your hosts' `/lib/libc.so` will not work for cross-compiling...

2.0.2 Step three: The compiler itself

I used `egcs` here, again with the sources from the Red Hat CD. Same procedure as with the `binutils`: pull the sources out of the rpm, untar them, change to the new `egcs-1.0.2` directory.

The rpm came along with a few patches, so I applied them before continuing:

```
$ for file in ../egcs*patch; do patch -p1 < $file; done
```

Again, configure it as cross-compiler:

```
$ ./configure --prefix=/usr/local --target=sparc-linux \
--with-gnu-ld --with-gnu-as i486-redhat-linux
```

Type `make` to compile it. If you got the include- and library installation (see above) right, it should compile completely out-of-the-box. Install your new cross-compiler (as root) with `make install`.

2.0.3 Step four: test if it did work (optional)

As every responsible hacker you have the source code for the well know "hello world" program on your hard disk, haven't you? If not, it is time to write one now. If you are lazy, you can download GNU hello from any GNU mirror site or `ftp://ftp.gnu.org`. OK, test it:

Intel box:

```
bogomips kraxel ~/src# uname -a
Linux bogomips.isdn.cs.tu-berlin.de 2.1.115 #5-vger \
      Fri Aug 14 21:54:36 CEST 1998 i686 unknown
bogomips kraxel ~/src# sparc-linux-gcc -o hello hello.c
bogomips kraxel ~/src# file hello
hello: ELF 32-bit MSB executable, SPARC, version 1
bogomips kraxel ~/src#
```

SPARC box:

```
eibau kraxel ~/src# uname -a
Linux eibau.isdn.cs.tu-berlin.de 2.1.115 #2-vger \
      Sun Aug 16 00:56:16 CEST 1998 sparc unknown
eibau kraxel ~/src# ./hello
hello world!
eibau kraxel ~/src#
```

Cool, test successful. You have a full-featured cross-compiler now.

2.0.4 Additional libraries

If you are going to cross-compile some packages, they might miss some libraries. Just copy them (both includes and libraries of course) to `/usr/local/sparc-linux/[include|lib]`. Or cross-compile them too :-)

2.0.5 Cross-compiling your Linux-kernel

As you can see ...

```
eibau kraxel ~# cat /proc/version
Linux version 2.1.115 (kraxel@bogomips.isdn.cs.tu-berlin.de) \
    (gcc version egcs-2.90.27 980315 (egcs-1.0.2 release)) \
    #2-vger Sun Aug 16 00:56:16 CEST 1998
eibau kraxel ~#
```

...my SPARC runs a kernel which is cross-compiled on the Intel box.

Cross-compiling your kernel is easy. Because I don't like typing long command lines, I've defined a alias:

```
bash$ alias smake='make ARCH=sparc CROSS_COMPILE=sparc-linux-'
```

Compiling isn't much different from a normal compile, just use the new alias instead of the normal make:

```
bash$ smake config
[ ... answer all the questions here ... ]
bash$ smake clean dep && smake boot modules
```

That's all. Happy compiling!

A Supported platforms in GCC 2.95.2

- a29k-amd-udi
- arm-aout
- arm-coff
- h8300-hms
- hppa1.1-hp-proelf
- i386-aout
- i386-coff
- i386-elf
- i960-coff
- m32r-elf
- m68k-aout
- m68k-coff
- mips-ecoff
- mips-elf
- mips64-elf
- powerpc-eabi
- sh-hms
- sparc-aout
- sparclite-aout
- sparclite-coff
- sparc64-elf