j a e g e r . m o r p h e u s . n e t

How to build an i386->Sparc Linux cross-compiler using GCC

This tutorial is several years old. I do not support it any longer but I've left it here as I still get a lot of requests for it.

What follows is a somewhat out-of-date, yet still relevant in concept, at least, description of the steps needed to build a cross-compiler for a Sparc Linux system, hosted on an i386 Linux system. This was written several years ago originally, and thus the systems mentioned herein are somewhat old, but the idea is still the same, and the steps to get you there should be similar enough that this howto will still be useful.

The systems used here were a Red Hat Linux 5.1 and 5.2 machine, respectively, but you should be able to produce a cross-compiler on any new version of the same or on nearly any other unix where gcc is well-liked.

Thanks go to Gerd Knorr, the author of this cross-compiler mini-howto, for some of the information I used to develop my own howto. The rest of it comes from the CrossGCC FAQ.

And so it begins...

When compiling GCC, it's a good idea to compile it in a directory separate from your source tree, so I created the following build directories in a temporary location for cross-compiling:

```
.~# mkdir cross
.~# mkdir cross/build-binutils
.~# mkdir cross/build-gcc
```

In order to facilitate compiles, you can set a couple environment variables to useful values. This is not necessary, just speeds things up. If you don't set them here, you need to specify them on the various configure/compile command lines mentioned hereafter.

```
.~# TARGET=sparc-linux
.~# PREFIX=/usr/local
```

That done, we need to first compile the binary utilities (as, ld, ar, etc.) for use with gcc (and others.) The binutils package is available from any GNU mirror. I'm using version 2.9.1 for the purposes of this howto.

Uncompress the binutils package file (binutils-2.9.1.tar.gz) into the **cross/** directory, cd to the temporary **build** directory, and configure/compile binutils:

```
.~# cd cross
.~/cross# tar zxvf /path/to/binutils-2.9.1.tar.gz
.~/cross# cd build-binutils
.~/cross/build-binutils# ../binutils-2.9.1/configure --target=$TARGET --prefix=$PREFIX -v
.~/cross/build-binutils# make all
.~/cross/build-binutils# make install
```

If you look in **$PREFIX/bin** now, you should find the sparc-linux compiled binutils, prefixed with '**sparc-linux-**'. Also, if you look in **$PREFIX/sparc-linux**, you should find two directories, **bin** and **lib**.

Once the binutils package is compiled and installed, we need to install Sparc Linux libraries and headers for the compiler and compiled programs to use on the host machine. The files we need are contained in the **glibc** and **glib-devel** RPMS in a Red Hat distribution, so we'll just extract the files we need from those two RPMS. They can be downloaded from any Red Hat Linux mirror with the correct version of Red Hat you're using on the target machine. If you're using another distribution, this will vary, and it's up to you to figure out what files you need, but this should be a useful guideline.

```
./usr/local/sparc-linux# cd ~/cross
.~/cross# rpm2cpio glibc-2.0.7-29.sparc.rpm | cpio --extract --make-directories
```

```
29571 blocks
.~/cross# rpm2cpio glibc-devel-2.0.7-29.sparc.rpm | cpio --extract --make-directories
31257 blocks
```

Copy the includes to **$PREFIX/sparc-linux/**:

```
.~/cross# cp -a usr/include $PREFIX/sparc-linux
```

Copy the kernel headers to **$PREFIX/sparc-linux/include** (need the headers from the kernel source):

```
.~/cross# cp -a /usr/src/linux/include/linux $PREFIX/sparc-linux/include/linux
.~/cross# cp -a /usr/src/linux/include/asm-sparc $PREFIX/sparc-linux/include/asm
```

Next, we copy the libraries from the glibc packages we extracted previously:

```
.~/cross# cp -a lib/* $PREFIX/sparc-linux/lib
.~/cross# cp -a usr/lib/* $PREFIX/sparc-linux/lib
```

With that done, it's safe to remove the glibc RPMs and the directories we created by extracting them, namely **etc/**, **lib/**, and **usr/**. **DON'T confuse this with /etc, /lib, and /usr!**

The glibc installation contained in the aforementioned RPMs will be expecting to find some of its files in different places than we installed them, since we're putting what is normally a spread out installation into one directory, **$PREFIX/linux/lib/**. We need to fix some symlinks. The following script, from the aforementioned cross-compiler mini-howto, does the job nicely:

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

On top of that, **libc.so** is also broken. It's not a binary executable, it's a shell script, which we need to edit. Change it from this:

```
/* GNU ld script
Use the shared library, but some functions are only in
the static library, so try that secondarily. */
GROUP ( /lib/libc.so.6 /usr/lib/libc_nonshared.a )
```

To this:

```
/* GNU ld script
Use the shared library, but some functions are only in
the static library, so try that secondarily. */
GROUP ( /usr/local/sparc-linux/lib/libc.so.6 /usr/local/sparc-linux/lib/libc_nonshared.a )
```

Moving right along, we need to actually compile the cross-compiler. Go back to the **cross/** directory and untar gcc there. It's also available from any GNU mirror, and we're using version 2.8.1 for this howto.

```
./usr/local/sparc-linux/lib# cd ~/cross
.~/cross# tar zxvf gcc-2.8.1.tar.gz
.~/cross# cd build-gcc
.~/cross/build-gcc# ../gcc-2.8.1/configure --target=$TARGET --prefix=$PREFIX -v
.~/cross/build-gcc# make all
.~/cross/build-gcc# make install
```

If everything so far has been done correctly, gcc will compile cleanly and be ready to go. Bingo, cross-compiler is complete. :) Compile a test program, hello world or whatever on your host machine, send it over to the target machine, and run it. If it works, you did it right. :)

Now, in order to use this cross-compiler to compile a linux kernel (which was my original intent, and the reason for this howto), the author of the aforementioned cross-compiler mini-howto suggests the following alias to simplify the process:

```
.~# alias smake='make ARCH=sparc CROSS_COMPILE=sparc-linux-'
```

So, to compile your kernel, you'd do something like this:

```
./usr/src/linux# smake config && smake dep && smake clean && smake vmlinux && smake modules
```

Copy the resulting kernel and modules over to the sparc, fix up the boot loader, and you're ready to go. :)