GDC Cross-Compiler

From OSDev Wiki

This page is a work in progress and may thus be incomplete. Its content may be changed in the near future.

Difficulty level

🟩⬜⬜⬜
Beginner

If you reached here, you should have enough knowledge about cross-compilers, how they work. and so on. If you don't, check out GCC Cross-Compiler.

Contents

Requirements

We assume you have a host system with a working GCC installation. If you are not using a bash shell, you might have to modify some of the command lines below. If you have just installed the basic Cygwin package, you have to run the setup.exe again and install the following packages:

- GCC
- Make
- Flex
- Bison

You will also have to install the following (using your system's package management):

- GNU GMP (http://gmplib.org/) (libgmp-devel on Cygwin, libgmp3-dev on apt-based systems, dev-libs/gmp on Gentoo)
- GNU MPFR (http://www.mpfr.org/) (libmpfr-devel on Cygwin, libmpfr-dev on apt-based systems, dev-libs/mpfr on Gentoo)
- MPC (http://www.multiprecision.org/) (libmpc-devel on Cygwin, libmpc-dev on apt-based systems, dev-libs/mpc on Gentoo)

Step 1 - Bootstrap

We will build a toolset running on your host that can turn source code into object files for your target system.

We need the binutils and the GCC packages from http://ftp.gnu.org/gnu/. Make sure you downloaded exactly GCC ver 4.9.x (or 3.x, if you like old stuffs). Download them to /usr/src (or whereever you think appropriate), and unpack them. Also, we need gdc package from https://github.com/D-Programming-GDC/GDC/tree/gdc-4.9. Current version is 0.24

Preparation

```
export PREFIX=/usr/local/cross
export TARGET=i586-elf
cd /usr/src
mkdir build build/binutils build/gcc
```

binutils

```
cd /usr/src/build/binutils
/usr/src/binutils-x.xx/configure --target=$TARGET --prefix=$PREFIX --disable-nls
make all
make install
```

gcc

This needs some work out. First, unpack gcc, and we got /usr/src/gcc-x.x.x Go to /usr/src/gcc-x.x.x/gcc, and unpack gdc.This will create a subdirectory named "d".

```
cd /usr/src/gcc-x.x.x
./gcc/d/setup-gcc.sh
```

Then go to /usr/src/gcc-x.x.x/gcc/d, open d-lang.cc and look at line 188. You'll see this

```
const char* cygwin_d_os_versym = D_OS_VERSYM;
```

Change D_OS_VERSYM to whatever you want, since it is a string, like "MyOS". But do **not** set it to NULL, or your compiler will segfault. Forum Post by Wilkie (http://forum.osdev.org/viewtopic.php?f=15&t=18914&start=47%7C)

Now, you can build GCC.

```
cd /usr/src/build-gcc
export PATH=$PATH:$PREFIX/bin
../gcc-x.x.x/configure --target=$TARGET --prefix=$PREFIX --disable-nls \
    --enable-languages=c,d,c++ --without-headers
make all-gcc
make install-gcc
```

Step 2 -

Before you can build a kernel, you must have a kind of D Runtime. At least you must have a correct object.d file.

TO BE CONTINUED...