Using newer version of GCC and binutils with the FreeBSD Ports Collection

Martin Matuska <mm@FreeBSD.org >

Revision: 44683

Copyright © 2009 The FreeBSD Documentation Project

FreeBSD is a registered trademark of the FreeBSD Foundation.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this document, and the FreeBSD Project was aware of the trademark claim, the designations have been followed by the "™" or the "®" symbol.

2014-04-28 by wblock.

Abstract

This article describes how to use newer versions of the GCC compilers and binutils from the FreeBSD ports tree. Custom GCC configurations are also discussed.

Table of Contents

1. Introduction 1
2. Prerequisites 2
3. Configuring ports for custom version of GCC 2
4. Impact on the binary performance 4

1. Introduction

The default system compiler as of FreeBSD 8.0 is GCC version 4.2.1. In addition, the base system of FreeBSD includes binutils version 2.15. These versions are several years old and lack, among other things, support for recent CPU instructions like SSSE3, SSE4.1, SSE4.2, etc. Due to licensing issues, new versions of these applications will not be integrated into the base system. Luckily, it is possible to use a newer version of the GCC compiler (e.g. version 4.4) with the help of the FreeBSD ports tree.

2. Prerequisites

2.1. Installing binutils from ports

To make use of all of the new features in the latest GCC versions, the latest version of binutils needs to be installed. Installation of the newer version of binutils is optional; but without it, there will be no support for new CPU instructions.

To install the latest available version of binutils using the FreeBSD ports tree, issue the following command:

```
# cd /usr/ports/devel/binutils && make install
```

2.2. Installing GCC from ports

The FreeBSD ports tree offers several new versions of GCC. The following example is for the stable version 4.4. However, it is possible to install previous or later development versions (e.g. lang/gcc43 or lang/gcc45).

To install one of the mentioned GCC ports, run the following command:

```
# cd /usr/ports/lang/ gcc44 && make install
```

3. Configuring ports for custom version of GCC

Additional system configuration is required in order to use custom version of GCC installed from the FreeBSD ports tree.

3.1. Adjusting `make.conf`

Add the following lines to the `/etc/make.conf` file (or modify appropriately):

```
.if !empty(.CURDIR:M/usr/ports/*) && exists(/usr/local/bin/gcc44)
CC=gcc44
CXX=g++44
CPP=cpp44
.endif
```

Alternatively, it is possible to specify the `${CC}` and `${CPP}` variables manually.

Note

The examples above are for GCC version 4.4. To use `gcc43`, replace `"gcc44"` with `"gcc43"` and `"4.4"` with `"4.3"` and so on.

3.2. Adjusting `libmap.conf`

Many of the ports' binaries and libraries link to libgcc_s or libstdc++. The base system already includes these libraries, but from an earlier version of GCC (version 4.2.1). To supply rtld (and ldd) with correct versions, add the following lines to the `/etc/libmap.conf` file (or modify appropriately):

```
libgcc_s.so.1   gcc44/libgcc_s.so.1
libgomp.so.1    gcc44/libgomp.so.1
libobjc.so.3    gcc44/libobjc.so.2
libssp.so.0     gcc44/libssp.so.0
libstdc++.so.6  gcc44/libstdc++.so.6
```

Note

The examples above are for GCC version 4.4. To use `gcc43`, replace `"gcc44"` with `"gcc43"` and so on. Note also that all of these libraries are fully backwards compatible with base system libraries.

Warning

Some C++ programs may refuse to work if these libraries are not mapped correctly. If it is not feasible to map them all, it is recommended to map at least libstdc++.so.

3.3. Custom `CFLAGS` for the ports tree

To add custom `CFLAGS` for the ports tree which are unsupported by the base system, adjust the `/etc/make.conf` according to the following example:

```
.if !empty(.CURDIR:M/usr/ports/*) && exists(/usr/local/bin/gcc44)
CC=gcc44
CXX=g++44
CPP=cpp44
CFLAGS+=-mssse3
.endif
```

It is possible to completely replace `CFLAGS` and/or define custom `CPUTYPE` as well. We recommend setting `CPUTYPE` because many ports decide their optimizations flags based on this variable.

3.4. Excluding ports that do not build with new version of GCC

To exclude ports that have problems with custom version of GCC, adjust the `/etc/make.conf` according to the following example:

```
.if !empty(.CURDIR:M/usr/ports/*) && exists(/usr/local/bin/gcc44)
.if empty(.CURDIR:M/usr/ports/net/openldap*)
CC=gcc44
CXX=g++44
CPP=cpp44
.endif
.endif
```

The example above excludes the forced use of `gcc` 4.4 for the net/openldap* ports. It is also possible to specify more ports on a single line:

```
.if empty(.CURDIR:M/usr/ports/net/openldap*) && empty(.CURDIR:M/usr/
ports/xxx/yyy) && ...
```

4. Impact on the binary performance

Using GCC version 4.4 with SSSE3 instruction set enabled (if supported by the CPU) may yield up to 10% average increase in binary performance. In certain tests, the results show more than a 20% performance boost (e.g. in multimedia processing).

The table located at `http://people.freebsd.org/~mm/benchmarks/perlbench/` shows a comparison of GCC versions currently available in base FreeBSD system, GCC version 4.3 and GCC version 4.4 with various combinations of `CFLAGS` using the perlbench benchmark suite.