MKS Toolkit

Using GCC

MKS Inc.

MKS Toolkit: Using GCC

MKS Inc.
12450 Fair Lakes Circle
Suite 400
Fairfax, Virginia 22033
Phone: +1 703 803-3343
Fax: +1 703 803-3344
E-mail: tk_info@mkssoftware.com

8.7-0694

Technical Support

To request customer support, please contact us by one of the means listed below and in your request include the name and version number of the product, your serial number, and the operating system and version/patch level that you are using. Contact MKS customer support at:

Web:	`http://www.mkssoftware.com/support`
E-mail:	`tk_support@mkssoftware.com`
Telephone:	+1-703-803-7660 (9:00am to 7:00pm Eastern, Mon-Fri)
Fax:	+1-703-803-3344

When reporting problems, please provide a test case and test procedure, if possible. If you are following up on a previously reported problem, please include the problem tracking number in your correspondence.

Finally, tell us how we can contact you. Please give us your e-mail address and telephone number.

Table of Contents

Introduction

1

Traditionally, MKS Toolkit for Professional Developers and MKS Toolkit for Enterprise Developers have featured support for the Microsoft C+/C++ compilers and the Absoft FORTRAN compilers. But not everyone uses these compilers, at least not for every task. To help widen the selection of compilers available, these two MKS Toolkit products now also feature the GCC for N*u*TCRACKER Add-On, which lets you use the popular GNU C and C++ compilers for software porting or development on the N*u*TCRACKER Platform.

The GCC for N*u*TCRACKER Add-On is provided on the Resource Kit CD. At the heart of this add-on is the installer which not only installs a pre-built version of a standard GCC distribution (including compilers, linker, debugger, and support tools), but also sets up Windows registry entries and environment variables that let you seamlessly use the GNU C and C++ compilers with the N*u*TCRACKER developer environment. As is the case with all compilers, it is usually best to install the GCC add-on before you install MKS Toolkit. However, if you have an existing MKS Toolkit installation, you can install the GCC add-on and then use the Repair feature of the MKS Toolkit installer to change your compiler selection.

Currently, the GCC for N*u*TCRACKER Add-On only supports 32-bit compilation and linking, because, at the time of writing, the GNU linker could not generate either Itanium or Extended Architecture 64-bit executables. If this capability becomes available in the future, MKS may rebuild the add-on to include it.

For more information, see the online *MKS Toolkit UNIX to Windows Porting Guide* available under **Documentation** on the MKS Toolkit **Start** menu.

This document is intended as a supplement to the *MKS Toolkit UNIX to Windows Porting Guide* and provides GCC-specific information on many of the topics covered there.

Chapter 2: “GCC and the Porting Process” on page 3 describes differences between the GCC porting process and that described in “The Porting Process” chapter of the *UNIX to Windows Porting Guide.*

Chapter 3: “Working with GCC” on page 7 describes how to compile, link, and debug applications using GCC.

Chapter 4: “GCC Documentation” on page 11 describes the documentation provided with the GCC Add-On for N*u*TCRACKER.

GCC and the Porting Process

2

This chapter provides an overview of some of the major differences between the process of porting applications to the N*u*TCRACKER Platform using Microsoft Visual Studio, as described in the *UNIX to Windows Porting Guide*, and the same process using the GCC environment.

For details on compiling, linking, and debugging with GCC, see "Working with GCC" on page 7.

The key thing to remember is that the N*u*TCRACKER Platform is the same regardless of what compiler is being used. The compiler just converts your code into objects based on the N*u*TCRACKER Platform headers. That being said, there are some issues that should be discussed.

Selecting a Work Environment

So why use GCC to port your application? One reason might be that you are simply more familiar with the GNU compiler and how it works. This familiarity can allow you to complete porting projects more efficiently, as you have effectively removed the overhead of dealing with a less familiar environment. Another reason for choosing GCC might be that the code being ported was originally designed to be compiled using the GCC environment. In such a case, using the GCC for N*u*TCRACKER Add-On to handle the port is definitely the way to go.

When working with the GCC environment, it is recommended that you use the **`gmake`** utility (provided with the GCC for N*u*TCRACKER Add-On). This is especially true when porting applications designed for a GCC environment as the makefiles for such applications are likely to have been designed to take advantage of **`gmake`** features.

Header File Issues

The GCC for N*u*TCRACKER Add-On provides all of the same headers for the GCC compilation environment as are normally available for Microsoft Visual Studio, with the exception of `G++ Sdc++`.

Note The GNU C compiler (**`gcc`**) does not use the GNU `libc` library. As a result, the code generated is not subject to GPL (GNU General Public License). The GNU C++ compiler (**`g++`**) does however use the GNU G++ runtime library which is covered by a version of the GPL that contains a special exception: if you link the G++ library with files compiled with a GNU compiler to produce an executable, this does not cause the resulting executable to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the executable file might be covered by the GNU General Public License.

As discussed in the *UNIX to Windows Porting Guide*, various defines are set by the compiler and can be used to help identify the precise environment in which the code is building. For example, the GCC compiler sets the `__NUTC__`, `__Win32__` and `__GNUC__`

This program:

```
#include <stdio.h>
void main(void)
{
#if defined(_MSC_VER)
   printf("Microsoft compiler\n");
#endif
#if defined(__GNUC__)
   printf("GNU compiler\n");
#endif
#if defined(__WIN32__)
   printf("__WIN32__ defined\n");
#endif
#if defined(_WIN32)
   printf("_WIN32 defined\n");
#endif
#if defined(__NUTC__)
   printf("NuTCRACKER defined\n");
#endif
}
```

when compiled with GCC produces:

```
GNU compiler
__WIN32__ defined
_WIN32 defined
NuTCRACKER defined
```

and when compiled with VC++ produces:

```
Microsoft compiler
_WIN32 defined
NuTCRACKER defined
```

As you can see from this example, `__NUTC__` is the important conditional for the N*u*TCRACKER Platform and that there are other defines for VC++, GCC and Win32 that should be used when a piece of code is compiler or platform specific (such as a call out to native Win32).

Working with GCC 3

This chapter discusses how to compile, link, and debug using the GCC environment. It assumes that you have properly set up a shell window for N*u*TCRACKER Platform development using a GCC environment.

There are two ways to do this.First, installing the GCC for N*u*TCRACKER Add-On and then installing, MKS Toolkit for Professional Developers or MKS Toolkit for Enterprise Developers and selecting GCC as the default compiler automatically configures the shells available from the MKS Toolkit **Start** menu to use the GCC development environment. Second, you can open any command window and use the command:

```
ncenv gcc
```

to configure the window for GCC development. Any shell that you then launch from this window will also be configured for GCC development.

For more information about the **ncenv** utility, see the **ncenv** reference page in the online *MKS Toolkit Utilities Reference* available from the MKS Toolkit **Start** menu.

Compiling with GCC

Whether you are using Microsoft Visual Studio or a GCC development environment, the basic philosophy of N*u*TCRACKER Platform development remains the same. That philosophy is that you should be able change as little about your build environment as possible and still be able to do the same things on Windows that you did on UNIX. To this end, you may also want to build and install GNU Make, as it is probable that the compilation environment for the application being ported uses this utility along with the GNU compiler suite (GCC).

For full details on the **gcc** utility, see the *GNU C and C++ Reference* available under **Documentation > GCC for NuTCRACKER Add-On** on the MKS Toolkit **Start** menu. A UNIX-style **gcc** man page is also available as described in "GCC Documentation" on page 11.

The **gcc** utility launches the GCC compiler in a similar manner to the **cc** utility. There are two ways to invoke **gcc**.

The first method is to use the **cc** utility just as it is described in the *MKS Toolkit UNIX to Windows Porting Guide*. While this approach is simple and straightforward, it does have its drawbacks. **cc** simply passes all of its command options directly to **gcc**. This includes options specific to Microsoft Visual Studio (such as **-Wv**), which **gcc** treats as invalid compiler options. This method is launched by specifying `CC=cc` on the **make** (or **gmake**) command line.

The second method to call **gcc** directly by setting `CC=gcc` on the **make** (or **gmake**) command line.

Compiling with G++

Because the N*u*TCRACKER Platform does not provide a C++ runtime library and instead uses the Microsoft library, to compile with G++ (the GNU C++ compiler), you must use the GNU G++ runtime library which is covered by a version of the GPL that contains a special exception: if you link this G++ library with files compiled with a GNU compiler to produce an executable, this does not cause the resulting executable to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the executable file might be covered by the GNU General Public License.

Documentation for the G++ runtime library can be found online in the *GNU Standard C++ Library Reference*, which is available under **Documentation > GCC for NuTCRACKER Add-On** on the MKS Toolkit **Start** menu.

Linking with GCC

When compiling and linking for the NuTCRACKER Platform, it is important to note that the same headers and link libraries are used regardless of the compiler or linker being used. That is, whether you are using the GCC add-on or Microsoft Visual Studio, it is always the NuTCRACKER Platform's `libc` library that is used. No GNU `libc` is ever used. Effectively, the only difference between using the GCC add-on and Microsoft Visual Studio is the actual compiler and linker used.

Note As mentioned in the previous section, the G++ compiler included in the GCC add-on is an exception to this.

For full details on **ld** (installed as **gld**), see the *GNU Linker Reference* available under **Documentation > GCC for NuTCRACKER Add-On** on the MKS Toolkit **Start** menu. A UNIX-style **ld** man page is also available as described in "GCC Documentation" on page 11.

As a result, when you link, the object files created by the assembler are joined together with the libraries from the N*u*TCRACKER Platform and, perhaps, any native Win32 libraries you might need to produce Windows executables. When working with the GCC development environment, the linker is GNU **ld** (installed as **gld**).

Note At present, the GCC for N*u*TCRACKER Add-On generates several linker warnings due to the fact that GNU **ld** (**gld**) does not understand the `defaultlib` linker directives in the object files within the N*u*TCRACKER Platform libraries.

As is common with many compilers and linkers from the UNIX world, you should include the **-g** option on both the compiler and linker command lines to generate the proper output for use with a symbolic debugger.

Debugging with GCC

For full details on the **gdb** utility, see the *GNU Debugger Reference* available under **Documentation > GCC for NuTCRACKER Add-On** on the MKS Toolkit **Start** menu. A UNIX-style **gcc** man page is also available as described in "GCC Documentation" on page 11.

The symbols created by the compiler and linker included in the GCC add-on are not compatible with the debugger embedded in Microsoft Visual Studio. As a result, Microsoft Visual Studio is not a practical choice for debugging applications built using the GCC development environment—unless you enjoy debugging compiled C code at the assembly language level and without symbolic name. Fortunately, the GCC add-on includes the gdb debugger which is designed to work with the symbols created by the GNU compiler and linker.

Other GCC Utilities

For full details on these utilities, see the *GNU BinUtils Reference* available under **Documentation > GCC for NuTCRACKER Add-On** on the MKS Toolkit **Start** menu. UNIX-style man pages are also available as described in "GCC Documentation" on page 11.

The GCC for N*u*TCRACKER Add-On includes a preprocessor (**cpp**), a C compiler (**gcc**), a C++ compiler (**g++**), a linker (**gld**), an assembler (**as**), a debugger (**gdb**), and a profiler (**gprof**). In addition, the add-on provides a number of useful GNU utilities (such as **strings**, **ar**, and **size**). While some of these utilities duplicate utilities found in the MKS Toolkit, the GNU versions are included in the add-on for ease of use and to ensure compatibility with GNU build environments on UNIX and Linux systems.

GCC Documentation 4

This chapter discusses how to build and display the documentation included with the GCC for N*u*TCRACKER Add-On.Primarily, the documentation is provided in two forms: HTML Help files (with a `.chm` extension) and the original UNIX-style man pages.

HTML Help Files

Under **Documentation > GCC for NuTCRACKER Add-On** on the MKS Toolkit **Start** menu, are the following HTML Help files:

- *GNU Assembler Reference*
- *GNU C and C++ Reference*
- *GNU BinUtils Reference*
- *GNU C and C++ Internals Reference*
- *GNU C and C++ Reference*
- *GNU C Preprocessor Internals Reference*
- *GNU C Preprocessor Reference*
- *GNU Debugger Internals Reference*
- *GNU Debugger Reference*
- *GNU Linker Internals Reference*
- *GNU Linker Reference*
- *GNU Make Reference*
- *GNU Standard C++ Library Reference*

UNIX-Style Man Pages

The GCC for N*u*TCRACKER Add-On also includes UNIX-style man pages for each utility included in the distribution. These man pages are provided in GROFF (GNU Roff) format. To view these pages, you must first build the GROFF package provided in the GNU samples directory on the MKS Toolkit CD. But before you can build GROFF, you must first build GNU Make.

Compiling GNU Make

To compile GNU Make for use not only in compiling GROFF but also as a part of the GCC development environment. Follow these steps to compile the utility:

1. Copy the GNU samples directory from the MKS Toolkit CD to a location on your hard disk (for example, `$ROOTDIR/samples/GNU`).
2. Change to the `make-3.79.1` directory within the GNU samples directory and build the GNU Make package with the following commands:

```
sh configure --prefix=d:/gnu
make
```

Note The `readme.nutc` file in the `make-3.79.1` directory contains the up-to-date instructions for building GNU Make.

3. Copy the newly built **`make.exe`** from the `make-3.79.1` directory to either `Program Files/GNU C For Windows/ncbin` or `$ROOTDIR/bin`. You may also wish to consider renaming this file to **`gmake.exe`** to avoid confusion with the MKS Toolkit **`make`** utility.

Compiling GROFF

To build the GROFF package, you using the same copy of the GNU samples directory created when you built GNU Make. Within that directory, change to the `groff-1.15` directory and issue the command:

```
make install
```

You may find that the `INSTALL` file needs to be renamed to `INSTALL.txt` so that **`make install`** can function correctly.

Note The `readme.nutc` file in the `groff-1.15` directory contains the up-to-date instructions for building GROFF.

Accessing the Man Pages

Once the GROFF package has been built, you can use the **gman** utility that it provides to view the GCC man pages.To do so, you must specify the directory to search for the particular man page. The `GNU C For Windows` directory has `man1`, `man2`, etc. subdirectories which contain the actual pages.

For example, to view the **as** man page, use:

```
gman –M C:/PROGA~1/GNUCFO~1/man 1 as
```

Or to view the same man page in an HTML browser, use:

```
gman –h –M C:/PROGA~1/GNUCFO~1/man 1 as
```

Note The UNIX-style man pages are provided as a convenience for users who prefer viewing reference information in this form. The HTML Help files are more up-to-date and are the recommended source for GCC related information.