The GNUPro Toolkit

redhat.

The GNUPro Toolkit

Table of Contents

Chapter 1. GNUPro® Toolkit: Overview

Finding Documentation

This manual helps you to get started with the GNUPro Toolkit. After you read this GNUPro Toolkit Getting Started Guide refer to the following additional documentation files for more details about the GNUPro Toolkit:

Using the GNU Compiler Collection (GCC) in the gcc.info files

Which contains sections on:

- *Using GCC*
- *Using G++*
- *The C Preprocessor*

Debugging with GDB in the gdb.info files

Which contains sections on:

- *Debugging with GDB*
- *Insight, the GNUPro Debugger GUI*

Using ld, the GNU Linker in the ld.info files

Which contains sections on the linker.

Using as, the GNU Assembler in the gas.info files

Which contains sections on the assembler.

Using binutils, the GNU Binary Utilities in the binutils.info files

Which contains sections on:

- *Using objcopy*
- *Using objdump*

The documentation is in DocBook format on your CD. Additional documentation is available at the Red Hat site:

http://www.redhat.com/support/manuals/gnupro.html

For more information on the tools, their history and initiatives, see the following web sites for documentation:

- http://sources.redhat.com/cygwin/
- http://sources.redhat.com/newlib/
- http://sources.redhat.com/binutils/
- http://gcc.gnu.org/libstdc++/
- http://www.linuxdoc.org
- http://www.opensource.org/products.html
- http://www.gnu.org/software/gdb/
- http://gcc.gnu.org

- http://www.fsf.org/manual/

This list of URLs is only an introduction; other sites are available.

Using the Documentation

This documentation uses the following conventions.

> **Warning!**
>
> These boxes help you avoid problems such as damaged files or system or programming errors.

> **Note:** Note text gives you helpful information. For example:
>
> For input or output which is too long to fit on a line, the documentation uses a backslash (\) to signify the UNIX convention for a *linefeed* that would not interrupt a process. The backslash is not part of the actual source code.

This documentation uses the following conventions for commands, filenames, and other program-specific subjects.

- *Emphasised text* indicates content with emphasis, such as for the *first use* of terminology. It is also used to indicate keystrokes and keys on your keyboard (such as Ctrl, Del, or Enter) or **menu choices**. Ctrl keys and Meta keys are often indicated in the text by C- and M-, respectively; Meta (or M-) indicates the diamond-shaped key or the Escape key, depending upon the keyboard.
- Symbols are usually self-explanatory or in common use, such as math characters. With menus, for instance, the arrow symbol indicates sub-menus to use, such as 'Tools' arrow 'Apply Patch' arrow 'To File'.
- `This text` indicates content that is an example of a program's output, such as a command line prompt (see the `%` prompt and the output after the **gcc -print-search-dirs** command in example Example 1-1).

  Example 1-1. Example User Input

  ```
  % gcc -print-search-dirs
  installation problem, cannot exec cpp:
  No such file or directory
  ```

- *`Text like this`* indicates a placeholder that you substitute from a list of options. ***`Text like this`*** indicates content that is an example of input that you substitute when typing in a command. For instance, to remove *`filename`* from a directory, type **rm *`filename`***, replacing *`filename`* with an actual file name.
- **This kind of text** indicates content that is an example of on-screen input. See example Example 1-1.

Host Configurations

GNUPro Toolkit software is configurable [11] for the following host operating systems:

Table 1-1. Naming hosts with their canonical names

Generic Host Name	Canonical Host Name
Microsoft Windows Vista	i686-pc-cygwin
Microsoft Windows XP	i686-pc-cygwin
Microsoft Windows 2000	i686-pc-cygwin
Red Hat Linux 7.2, 7.3, AS2.1, WS2.1 (x86) [a]	i686-pc-linux-glibc2.2
RHEL3/4 AS/ES/WS (x86_64).	x86_64-pc-linux-glibc2.3
Red Hat Linux 8.0, 9, RHEL3/4 AS/ES/WS (x86).	i686-pc-linux-glibc2.3
Sun Sparc Solaris 7.0	sparc-sun-solaris2.7
Sun Sparc Solaris 8.0	sparc-sun-solaris2.8
Sun Sparc Solaris 9.0	sparc-sun-solaris2.9
Sun Sparc Solaris 10	sparc-sun-solaris2.10
Sun Sparc Solaris 7.0 64 Bit	sparc64-sun-solaris2.7
Sun Sparc Solaris 8.0 64 Bit	sparc64-sun-solaris2.8
Sun Sparc Solaris 9.0 64 Bit	sparc64-sun-solaris2.9
Sun Sparc Solaris 10 64 Bit	sparc64-sun-solaris10
Notes: a. x86 denotes 486, 586, and 686 as the supported versions.	

If you have questions about compatibility with different configurations, contact Red Hat.

When configuring, setting environment variables, etc. you will need to use the canonical *architecture-vendor-operating system* naming convention; table Table 1-1 shows the usage of *canonical names* (a *triplet* referring to a host platform that Red Hat supports).

In table Table 1-1, for a Sun Sparc running the Solaris 2.6 operating system, the full name is *sparc-sun-solaris2.6* (*sparc* being the architecture, *sun* being the vendor, and *solaris2.6* being the operating system version). There are exceptions to this pattern, such as *i686-pc-linux-glibc2.2* for Red Hat Linux 7.2, 7.3, and AS/WS2.1 operating systems.

For configurations with embedded development when using the whole environment's names, see the Section called *Embedded Cross-configurations* for the target processors with which the tools work. For more information on configuring your system and your environment before developing, see Chapter 5 and *Using the GNU Compiler Collection (GCC)* in the gcc.info files.

Embedded Cross-configurations

To develop with an embedded cross-configuration, GNUPro Toolkit software works with the following host operating systems for the GNUPro microarchitecture target processors.

- Microsoft Windows
- Red Hat Linux
- Sun Solaris

The following object file output formats are supported:

- ELF

To use specific tools their name must be prefixed with the name of their target. For instance, to use the compiler for a *TOOLKIT*-based board from within a shell, type the following canonical name to invoke the GNU compiler:

***TOOLPREFIX*-gcc**

Note that *TOOLPREFIX* is the name of board's processor and file format. So for example if that board contained an ARM and the file format was ELF then TOOLPREFIX would be "arm-elf" and the command run would be:

arm-elf-gcc

When developing with an embedded cross-configuration, GNUPro Toolkit software works with target processors. You work with the tools, which are on the host, using a *toolchain*, calling a specific tool with the specific embedded target's name.

To use a specific tool, add the tool's name to its *canonical name*. For host names see table Table 1-1 and for the name of the specific tools see Chapter 4.

For more information, See Chapter 4 and Chapter 5 for the complete naming conventions for hosts, targets and tools, and how to begin with embedded development.

Notes

1. http://www.redhat.com/support/manuals/gnupro.html
2. http://sources.redhat.com/cygwin/
3. http://sources.redhat.com/newlib/
4. http://sources.redhat.com/binutils/
5. http://gcc.gnu.org/libstdc++/
6. http://www.linuxdoc.org
7. http://www.opensource.org/products.html
8. http://www.gnu.org/software/gdb/
9. http://gcc.gnu.org
10. http://www.fsf.org/manual/
11. *Important*: For cross-configurations and rebuilding information, see Chapter 8.

Chapter 2. GNUPro® Toolkit: Installing the GNUPro Toolkit

This chapter describes how to install GNUPro Toolkit. The GNUPro Toolkit provides a complete, supported toolset for use in software development.

Cygwin users additionally receive an independent Cygwin release including a supported dll and many components of the Cygwin Net release to create a complete Unix-like development environment.

GNUPro tools are made available in one of two ways, either via Anonymous FTP, or via CD install media. By default, the GNUPro tools are installed into `/opt/redhat/release_name` for UNIX like operating systems and into `C:\RedHat\release_name` for Windows operating systems.

`release_name` is a unique identifier which means the same sources were used to build the files contained within. All GNUPro releases employ a uniform directory structure, allowing multiple host architectures to coexist in the same installation prefix.

For detailed installation information, see either: the Section called *Install from CD* or the Section called *Install from CD*.

Install from CD

This section describes how to install GNUPro Toolkit from CD on both UNIX and Windows systems. Be sure that you are familiar with the naming conventions for the tools as described in Table 8-1, and the Section called *Embedded Cross-configurations* in Chapter 1.

Please note that Windows users must first install the Red Hat Cygwin release before they can install GNUPro releases. The Cygwin release contains the cygwin1.dll which GNUPro require.

In the following installation input examples, for `/mnt`, substitute your directory or `/cdrom/gnupro-07r1-1` (the default directory in which you will mount the tools); `host` designates what you need to specify as your host type in using the binaries for the particular architecture, vendor, and operating system.

1. Insert CD into CD-ROM drive.
2. Mount the CD.

   The procedures for mounting a CD depend on your system type. The following examples of **mount** commands use default configurations; consult your system administrator if you need assistance.

   Microsoft Windows XP or Vista

   Once inserted, the CD will be immediately available via the *My Computer* desktop icon.

   Red Hat Linux (and others)

   Modern Linux desktops will automatically mount newly inserted CDs in `/mnt/cdrom`. If your Linux system does not do this, perform the following action:

   As root, enter the following command (the command assumes that the CD-ROM drive is available as `/dev/cdrom`):

   ```
   mount -t iso9660 /dev/cdrom /mnt/cdrom
   ```

Sun Sparc Solaris

If you are running the volume manager, the CD will automatically mount as `/cdrom/gnupro-07r1-1` and you will not require root access.

If you are not running the volume manager, as root mount the CD manually with the following command:

```
mount -F hsfs -o ro /dev/dsk/c0t6d0s0 /
    mnt
```

3. Start the installer.

   How the graphical installer starts depends upon the host operating system:

   Microsoft Windows XP

   Upon CD insertion, the Installation program should start up at once. If the *autorun* feature has been disabled and the Installer does not start up at once, do the following:

   a. Open the *My Computer* icon on the Windows Desktop.
   b. Double click the CD-ROM Icon.
   c. Double click the *install-gui* Red Hat Shadowman logo to start the installer.

   On Unix Systems (Linux, Solaris, etc)

   First, set the **DISPLAY** environment variable. This variable may already be set if you're using the X11 Windowing System. If you are not certain if or how to set this variable, consult your system administrator.

   Start the installation program (substitute the path prefix according to where the CD media has been mounted):

   ```
   /mnt/cdrom/install-gui
   ```

4. Once loaded, the Installer will provide the option to *Browse* available packages, or just *Install* the release. Select *Install*.

   The next screen will provide several different installation options. These options may include *Typical* Installation, *Typical+SysRoot*, *Full Installation*, or *Customized* Installation.

   Typical Installation

   Selecting a Typical Installation will install all the packages required for a fully functional GNUPro development environment. The following packages are installed with a Typical Installation:

   - GNUPro Compiler
   - GNUPro Debugger
   - GNUPro DejaGNU
   - GNUPro Misc
   - GNUPro Utils
   - GNUPro ManPages
   - GNUPro Tex Info help files

   A Cygwin installation also installs some extra packages:

- Cygwin
- Cygwin Winsup
- Cygwin NetRel (Optional but recommended)
- Windows Mount
- Windows PATH

Typical Installation is recommended for all new users.

Typical Installation Plus Sys-Root

This option is only available on Red Hat Linux systems when installing embedded cross compiler tools. In addition to the typical installation, this option will install an Embedded Linux System Root.

A System Root is a copy of the standard headers and libraries found on a native Operating System. Installing a System Root allows developers to use different version of Linux than the target environment for their software. This allows developers to use workstation hardware to develop software for embedded environments. This feature can also be used with native Linux and Solaris native compilers to target older or newer versions of Linux and Solaris, respectively.

Full Installation

Selecting Full Installation will install all of the GNUPro packages; these include all the packages listed in the Typical Installation plus the following:

- GNUPro Native Sources (where applicable)
- GNUPro Cross Sources (where applicable)
- GNUPro System Root Sources (where applicable)
- Cygwin Net Release Sources (where applicable)

Customized Installation

Selected Customized Installation will enable you to select which packages you would like installed.

On Windows Systems, both Typical Installation and Full Installation will add the GNUPro or Cygwin binary directories to the PATH environment variable, as well as rewrite the Cygwin mount table. If you do not wish the installer to modify your Windows 9x autoexec.bat file or the Windows NT registry, or do not want the Cygwin mount table updated, then select Customized Installation and deselect the Update Windows PATH and Configure Mount Table (respectively) packages before installing.

5. After selecting what packages to install, the Installer will ask what directory prefix to use for the installation.

   By default, Windows systems use `C:\RedHat` and Unix systems use `/opt/redhat`. Windows systems will also create a Cygwin mount table such that the installation directory maps to `/opt/redhat` for greater compatibility (unless the Configure Mount Table option was specifically deselected during the previous step).

   Accept the default or enter a different directory name to use. The installer will create new directories for you, if necessary. Please consult with your local System Administrator if you require assistance with where to install.

6. Once begun, the Installer will unpack each archive and give a continuous update on overall and per-package progress. The installation process can take between 5-30 minutes, depending on the speed of the system.
7. Upon successful installation, the following message appears: `Installation Complete`
8. Once installed, these tools can be made read-and-execute only, to conform to local security policies. No write permissions on the tools are necessary to use the tools.

Once installed, the GNUPro Toolkit will be organized into architecture dependent (such as executables and libraries) and architecture independent (such as manuals and sources) subdirectories.

To accommodate binaries for multiple hosts in a single directory structure, the binary files for your particular host type are in the `H-`*`host`*`/bin` subdirectory (see Table 8-1 for specific host triplet names).

After completing the installation process, see the Section called *Setting the **PATH** Environment Variable*.

Installing via FTP

Make sure when getting files from the FTP server that you download everything to ensure no instructions or important files are missing. Releases made available via FTP are typically available for at least 30 following the availability announcement.

1. All FTP releases contain a file named *README* with installation instructions. Please follow these instructions, as they will be customized particularly for the release being downloaded and installed.
2. If the instructions provided in the README do not mention setting the PATH environment variable, please consult this manual's section on See the Section called *Setting the **PATH** Environment Variable*.
3. Some FTP releases come with unbundled manuals, typically in pdf format. Once the release is installed, consult the documentation to decide what to do next.

After completing the installation process, see the Section called *Setting the **PATH** Environment Variable*.

Setting the PATH Environment Variable

On Windows systems, the default installation will automatically set the **PATH** environment variable and the tools will be available at the next login.

On Unix systems, the **PATH** environment variable needs to be set to point to the executables just installed. PATHs can be updated on a system-wide basis by your System Administrator, or on an individual basis by editing your login shell scripts. Please consult the man page for your login shell for information on how to permanently update your **PATH**

The following examples demonstrate how to update your **path** as a one-time update. If you installed into a different directory, substitute *`installdir`* for the actual directory. For *`host`*, see Table 1-1 for naming conventions. The following examples show how to create the final linking to the *`installdir`*`/H-`*`host`*`/bin` directory.

For SH compatible shells (sh, bash, Korn etc)

```
    PATH=installdir/H-host/bin:$PATH;
export PATH
```

For CSH compatible shells (csh, tsch, zsh etc)

```
    set path=(installdir/H-host/bin $path)
rehash
```

The path should not include any colons for the C shell environment variable settings. The directory path should be separated only by whitespace.

For more information on using other environment variables, see *Using the GNU Compiler Collection (GCC)* in the gcc.info files.

Starting a command interpreter

After installing the software and before using the tools, you must start a program called a*command interpreter* or a *shell*. This program provides a text based interface for driving the GNUPro tools.

- For Windows operating systems, start a shell with the **bash** command either from an MS-DOS window or the **Run**box.

  Use the following examples, replacing *installdir* with your installation directory, *release_name* with the name you received with an announcement of the availability of the tools. Replace H-*host* with **H-i686-pc-cygwin** (where **i686-pc-cygwin** is a triplet name).

  - **SET PROOT=C:*installdir**release_name***
  - **SET INFOPATH=%PROOT%\info**
  - **REM Set TMPDIR to point to a ramdisk if you have one**
  - **SET TMPDIR=%PROOT%**
  - **REM Below only necessary if the path hasn't been previously set**
  - **REM SET PATH=%PROOT%\H-*host*\BIN;%PATH%**

  See also Chapter 8, for getting the tools when compiling on other machines.

- For Sun Solaris or Red Hat Linux operating systems, use the following examples, replacing *installdir* with your installation directory, *release_name* with the name you will have received with an announcement of the availability of the tools. Replace H-*host* (where *host* signifies a triplet name) with **H-sparc-sun-solaris2.7** for Sun Solaris and **H-i686-pc-linux-gnulibc2.3** for Red Hat Enterprise Linux 3 WS, ES, AS versions.
  - For Bourne-compatible shells (sh, bash, ksh), use the following example's input:
    - **PROOT=*installdir*/*release_name***
    - **INFOPATH=$PROOT/info**
    - **SID_EXEC_PREFIX=$PROOT**
    - **PATH=$PROOT/H-*host*/bin:$PATH**
    - **export PATH SID_EXEC_PREFIX INFOPATH**

  - For C shells, use the following example's input:
    - **set PROOT=*installdir*/*release_name***
    - **set path=($PROOT/H-*host*/bin $path)**
    - **setenv SID_EXEC_PREFIX $PROOT**
    - **setenv INFOPATH $PROOT/info**

Next Steps

Please see Chapter 3, then move on to Chapter 4.

Chapter 3. GNUPro® Toolkit: How to Report Problems

The GNUPro Toolkit is meant to be as trouble-free as possible. However, if you have a problem and you purchased a GNUPro or an Engineering Services support contract, connect to *IssueTracker*, our web support site. Our support customers can log into IssueTracker to submit problems and to track a problem's case from its first report through its analysis and resolution.

To use your IssueTracker account:

1. Enter the following URL in your web browser: https://enterprise.redhat.com/gnupro/
2. Log in with your account information.
3. Under the **Main** menu, click on **Issues** to view your current issues.
4. You will have a group for each contract you have with Red Hat. If you have multiple groups, please select the appropriate one.
5. Click on **Create Issues** to create a new issue.
6. Type in a summary of th problem, a concise description, and choose a product and severity for the problem.
7. Click on **Create Issue** to submit the issue to IssueTracker.
8. You will now be at the *Event* screen. Type in a more detailed description here and attach any test files, if required. Update the Status to *Waiting on Tech* and then click **Commit Event**.
9. You can now return to the *Main* screen (by clicking on **Main** on the left toolbar) page where you should see all of the issues you have filed listed under "Issues".

   Click on one of you issues to add additional data to an existing case in the data to an existing case in the database. At this point, you can view a problem case's details, check its status, and add notes to a case.
10. When an issue has been resolved, remember to mark the issues as "Closed by Client".

Notes

1. https://enterprise.redhat.com/gnupro/

Chapter 4. GNUPro® Toolkit: Basic Principles of the Tools

The GNUPro Toolkit's C and C++ compilers, macro-assembler, debugger, binary utilities, libraries, and other development tools provide productivity, flexibility, performance, and portability. This chapter begins with a summary of the tools, then describes each tool in more detail.

To use a tool, open your system's terminal shell window and enter the tool's name as a command (**gcc**, for instance, invokes the compiler, and **insight** invokes the debugger). For a summary of the tools in the GNUPro Toolkit see the Section called *The Compiler and Development Tools*, and the Section called *Auxiliary Development Tools*.

To learn about working with the tools, see Chapter 5, and Chapter 8.

The Compiler and Development Tools

The following tools are the main programs for developing projects with the GNUPro Toolkit.

cpp

The C preprocessor. (See the Section called *cpp the GNU Preprocessor* and *Using the GNU Compiler Collection (GCC)* in the gcc.info files).

diff
diff3
sdiff

Comparison tools for text files. (See *Using the GNU Compiler Collection (GCC)* in the gcc.info files).

gcc

ISO-conforming compiler collection. (See the Section called *gcc, the GNU Compiler Collection* and *Using the GNU Compiler Collection (GCC)* in the gcc.info files).

g++

A front end to **gcc** that defaults to interpreting files as being C++ source files even if they look like C source files.

gcov

Coverage analyzer, for testing code for efficiency and performance, and for use as a profiling tool. (See the Section called *gcov, the Test Coverage Tool*).

gdb

Debugger for making applications work better. (See the Section called *GDB, the Debugging Tool* and *Debugging with GDB* in the gdb.info files).

insight

Debugger using a graphical user interface, a visual debugger, known as *Insight*, also conceptually known as **gdbtk**. (See the Section called *Insight, a GUI Debugger*, Chapter 6 and *Debugging with GDB* in the gdb.info files).

ld

The linker. (See the Section called *ld the GNU Linker* and *Using ld, the GNU Linker* in the ld.info files).

make

Compilation control program. (See the Section called *make, the GNU Recompiling Tool* and *Using the GNU Compiler Collection (GCC)* in the gcc.info files).

patch

Installation tool for source fixes. See *Using the GNU Compiler Collection (GCC)* in the gcc.info files).

The GNUPro Toolkit Libraries

The GNUPro Toolkit includes several libraries. See the Section called *newlib and libstdc++-v3, the C and C++ Libraries*, and *GNUPro Libraries* for documentation regarding the following libraries:

libc

An ANSI C runtime library (*only available for cross-development*)

libm

A C math subroutine library (*only available for cross-development*)

libstdc++-v3

The C++ class library, implementing the ISO 14882 Standard C++ library. See http://gcc.gnu.org/onlinedocs/libstdc++/documentation.html[1]

Auxiliary Development Tools

The GNUPro Toolkit also provides the following components for general development.

as

An assembler. (See the Section called *as the GNU Assembler* and *Using as, the GNU Assembler* in the gas.info files).

Cygwin

Porting layer for making UNIX applications run on Windows systems. (See the Section called *Cygwin, for Porting UNIX Applications for Working on Windows Systems* and also http://sources.redhat.com/cygwin/).

info

Online documentation tools. (See the Section called *Using info, the Documentation Tools* and *Using the GNU Compiler Collection (GCC)* in the gcc.info files).

man

The **man** pages, the standard UNIX online documentation

The GNU binary utilities provide functionality beyond the main development tools (See the Section called *binutils, the GNU Binary Utilities* and *Using binutils, the GNU Binary Utilities* in the binutils.info files).

ar

Creates, modifies and extracts from object code archives.

Porting layer for making UNIX applications run on Windows systems. (See the Section called *Cygwin, for Porting UNIX Applications for Working on Windows Systems* and also http://sources.redhat.com/cygwin/

info

Online documentation tools. (See the Section called *Using info, the Documentation Tools* and *Using the GNU Compiler Collection (GCC)* in the gcc.info files).

man

The *man pages*, which are the documentation files used by the standard UNIX online documentation program **man**.

The GNU binary utilities provide functionality beyond the main development tools (See the Section called *binutils, the GNU Binary Utilities* and *Using binutils, the GNU Binary Utilities* in the binutils.info files). They include the following programs:

addr2line

Converts addresses into file names and line numbers.

c++filt

Demangles and deciphers encoded C++ symbol names.

dlltool

Creates dynamic link libraries (DLLs).

nm

Lists symbols found in object and executable files.

objcopy

Copies and translates object files.

objdump

Displays information from object files.

ranlib

Generates index for library files.

readelf

Displays information about ELF format object files.

size

Lists file section sizes and total sizes.

strings

Lists printable strings found inside binary files.

strip

Removes symbols and sections from executable files.

windres

Manipulates Win32 resources. To use the GNUPro Toolkit on Windows systems, see http://sources.redhat.com/cygwin/.

gcc, the GNU Compiler Collection

The GNU Compiler Collection (also known as *GCC*), is a complete set of tools for compiling programs written in C, C++, or languages for which you have installed front-ends, invoking the GNU compiler passes with the following utilities:

cpp

The GNU preprocessor that processes all the header files and macros which your target requires (see the Section called *cpp the GNU Preprocessor*).

as

The GNU assembler that produces binary code from assembly language code and puts it in an object file (see the Section called *as the GNU Assembler*).

ld

The GNU linker that binds code to addresses, linking a startup file and libraries to an object file, for producing an executable binary image (see the Section called *ld the GNU Linker*).

To invoke the compiler, type:

gcc *`options`*

`options` signifies input that allows you to control how the program compiles, i.e. interrupt a compile process at intermediate stages. Use commas to separate options. There are many options available that provide specific types of compiled output, some for preprocessing, others controlling assembly, linking, optimization, debugging, and still others for target-specific functions. For instance, call the compiler with a **'-v'** option to see precisely which options are in use for each compilation pass.

gcc implicitly recognizes the following file extensions:

.c

For C source code which must be preprocessed.

.i

For C source code which does not need to be preprocessed.

.C

For C++ source code.

.s

For assembler source code.

.S

For assembler source code which must be preprocessed.

Compilation involves up to four stages, always in the following order:

1. preprocessing
2. compiling
3. assembling

4. linking

The first three stages apply to individual source files. *Preprocessing* interprets special directives in the source file, performing text-based substitutions. *Compiling* converts the preprocessed high level language source code into assembler source code. *Assembling* then converts this into binary machine code and places it into an object file. Finally, the last stage, *linking*, combines multiple object files (either standalone or in library archives) into a single executable file.

For more information on working with the GNU compiler and using its options, see *Using the GNU Compiler Collection (GCC)* in the gcc.info files.

cpp the GNU Preprocessor

cpp is a C-compatible macro preprocessor which works with the GCC to direct the parsing of preprocessor directives. Preprocessing directives are the lines in your program that start with a **#** directive name (a **#** sign followed by an identifier). For instance, **cpp** merges **#include** files, expands macro definitions, and processes **#ifdef** sections. Another example is **#define**, a directive that defines a macro (**#define** must be followed by a macro name and the macro's intended expansion).

To see the output of **cpp**, invoke **gcc** with the **'-E'** option; the preprocessed file prints on **stdout**.

The C preprocessor provides the following separate facilities:

Inclusion of header files

Declarations that can be substituted into your program.

Macro expansion

When defining macros, which are abbreviations for arbitrary fragments of C code, the C preprocessor replaces them with their definitions throughout a program.

Conditional compilation

Using special preprocessing directives, include or exclude parts of a program according to various conditions.

Line control

Using a program to combine or rearrange source files into an intermediate file that is then compiled, use line control to provide a source line s origin.

There are two convenient options to assemble handwritten files that require C-style preprocessing. Both options depend on using the compiler driver program, **gcc**, instead of directly calling the assembler.

- Name the source file using the extension, **.S** (capitalized), rather than **.s** (this indicates that the file is in assembly language requiring C-style preprocessing).
- Specify a source language explicitly for a situation, using the **'-xassembler-with-cpp '** option

For more information on **cpp**, see *Using the GNU Compiler Collection (GCC)* in the gcc.info files.

as the GNU Assembler

as, the GNU assembler, translates text-based assembly language source files into binary-based object files. Normally it operates without you being aware of it, as the

compiler driver program (**gcc**) will invoke it automatically. If, however, you are creating your own assembler source files then you will need to invoke it directly.

If, while using **gcc** you want to pass special command line options to the assembler to control its' behavior, you will need to use the **-Wa,[text]** command line option. This option passes **[text]** directly on to the assembler's command line. For example:

Example 4-1. Passing options to the assembler

gcc -c -g -O -Wa,-alh,-L *file*.c

Passes the '`-alh`' command line option on to the assembler. (This will cause the assembler to emit a listing file which shows the assembler source generated by the compiler for each line of the C source file *file* .c).

For more information, see *Using as, the GNU Assembler* in the gas.info files.

Object Files

The assembler creates object files, which, by convention, have the extension **.o**. These are binary files that contain the assembled source code, information to help the linker integrate the object file into an executable program, debugging information and tables of all of the symbols used in the source code.

Special programs exist to manipulate object files. For example **objdump**

Can disassemble an object file back into assembler source code, and **ar** can group together multiple object files into an archive or library file.

Assembler Directives

Assembler directives are commands inside the assembler source files that control how the object file is generated. They are also known as *pseudo-ops* or *pseudo-operations* because they can look like commands in the assembler programming language.

Assembler directives always start with a period (**.**). The rest of their name is letters, usually in lower case. They have a wide range of different uses, such as specifying alignments, inserting constants into the output, and selecting in which sections of the output file the assembled machine instructions will be placed.

ld the GNU Linker

The GNU linker, **ld** combines multiple object files together and creates an executable program from them. It does this by resolving references between the different object files, grouping together similar sections in the object files into one place, arranging for these sections to be loaded at the correct addresses in memory, and generating the necessary header information at the start of a file that allows it to be run.

The linker moves blocks of bytes of your program to their load-time addresses. These blocks slide to their addresses as rigid units; their length does not change and neither does the order of the bytes within them. Such a rigid unit is called a *section*. Assigning run-time addresses to sections is called *relocation*. It includes the task of adjusting mentions of object-file addresses so they refer to the proper run-time addresses.

Each section in an object file has a name and a size. Most sections also have an associated block of data, known as the *section contents*. A section may be marked as

allocatable, meaning that space should be reserved for it in memory when the executable starts running. A section may also be marked as *loadable*, meaning that its contents should be loaded into memory when the executable starts. A section which is allocatable but not loadable will have a zero-filled area of memory created for it.

A section, which is neither loadable nor allocatable, typically contains some sort of debugging information. Every loadable or allocatable output section has two addresses associated with it. The first is the *virtual memory address* or *VMA* which is the address the section will have when the executable is running. The second is the *load memory address* or *LMA* which is the address in memory where the section will loaded. In most cases the two addresses will be the same. An example of when they might be different is when a data section is loaded into ROM, and then copied into RAM when the program starts. This technique is often used to initialize global variables in a ROM-based system. In this case, the ROM address would be the LMA, and the RAM address would be the VMA. To see the sections in an object file, use the **objdump** binary utility with the **-h** option.

Every object file also has a list of *symbols*, known as the *symbol table*. A symbol may be defined or undefined. Each symbol has a name, and each defined symbol has an address. If you compile a C or C++ program into an object file, you will get a defined symbol for every defined function and global or static variable. Every undefined function or global variable, which is referenced in the input file, will become an undefined symbol. You can see the symbols in an object file by using the **nm** binary utility, or by using the **objdump** binary utility with the **-t** option.

The linker is controlled by a *linker script* which is a text file containing commands in a simple language. The main purpose of the script is to describe how *sections* in the input files should be mapped into sections in the output file and to control memory layout of the output file. When necessary, the linker script also directs the linker to perform other operations. For an example of a linker script, see Example 5-3).

The linker will use a default script that compiles into the linker executable, if you do not supply one via the `-T` command line option. Use the `--verbose` option to display the default linker script. Certain options (such as `-r` or `-N`) will affect the default linker script. Linker scripts are written in a superset of AT&T's *Link Editor Command Language* syntax.

For more information, see *Using ld, the GNU Linker* in the ld.info files.

gcov, the Test Coverage Tool

The **gcov** program is a code coverage/basic block profile display tool which allows you to analyze the basic block profile of your program by recording how often each basic block is executed. This information allows you to determine the sections of code on critical paths and the code blocks that are not executed at all.

GDB, the Debugging Tool

GDB, the GNU debugger, is a tool designed to help locate the cause of bugs in a computer program. It can be run as a text based tool or using a graphical interface. To use it the program's source files must be compiled with the `-g` option enabled. The instructs the compiler to generate extra information which is used by the debugger. See *Debugging with GDB* in the gdb.info files for more information about the steps involved in debugging.

GDB allows you to set breakpoints in the program being debugged. These will cause the program's execution to stop as soon as one of them is reached. It is then possible to examine the state of the program before resuming execution. It is also possible to step through the program a single line at a time, watching what happens as it does so.

The debugger can handle threads and signals. It also knows about the *execution stack*, so it is possible to display backtraces, examine local variables and so on.

For more information, see *Debugging with GDB* in the gdb.info files and also the Section called *Insight, a GUI Debugger* and *Using the GNU Compiler Collection (GCC)* in the gcc.info files.

Insight, a GUI Debugger

Besides the standard command line based debugger the GNUPro Toolkit includes *Insight* a graphical user interface to GDB. Insight works on a range of host systems and target microprocessors, allowing development with complete access to the program state, for source and assembly level, with the ability to manage breakpoints, variables, registers, memory, threads and other functionality. Adding a series of intuitive views into the debugging process, Insight provides you with a wide range of system information.

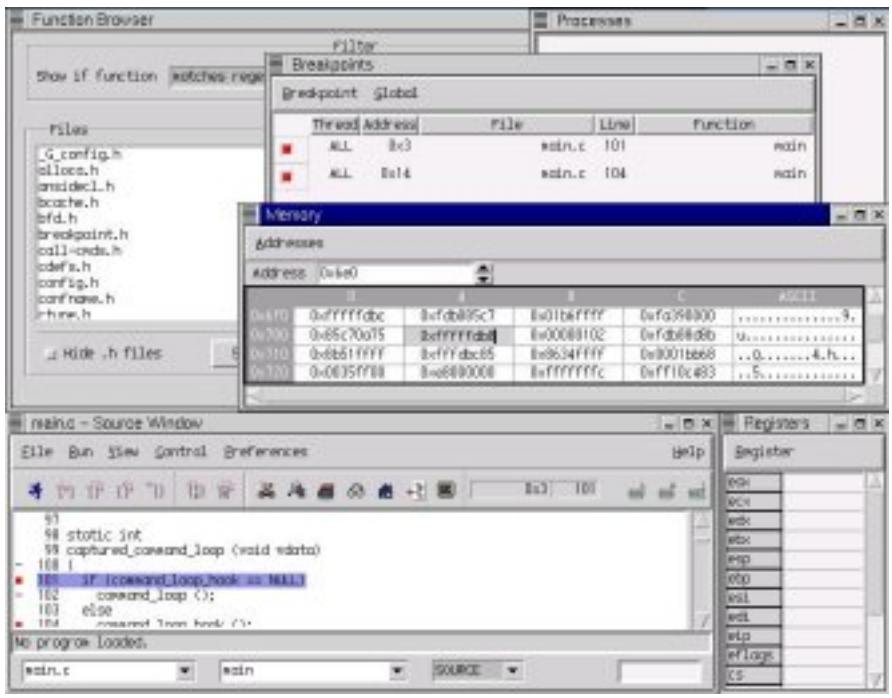

Figure 4-1. A Composite View of Working With Insight

For developing with Insight, see Chapter 6.

make, the GNU Recompiling Tool

make, the GNU recompiling tool, determines automatically which pieces of a large program you need to recompile and then issues commands to recompile them. **make** conforms to *IEEE Standard 1003.2-1992* (POSIX.2) and is compatible with any programming language whose compiler can run with command line input from a shell. **make** is not limited only to building programs; it can be used for any task in which some files must update automatically whenever associated files change.

To use **make**, you must write a file (a *makefile*) that describes the relationships among files in your program and provides commands for updating each file. In a program,

typically, the executable file is updated from object files, which are in turn made by compiling source files.

When using **make** to recompile an executable, the result may change source files in a directory. If you changed a header file, to be safe, you must recompile each source file that includes that header file. Each compilation produces an object file corresponding to the source file. If any source file has been recompiled, all the object files, whether newly made or saved from previous compilations, must be linked together to produce the new executable.

make uses the makefile database and the last modified files to decide which of the other files needs updating. For each of those files, **make** implements the commands recorded in the data base of the makefile. The makefile has *rules* which explain how and when to remake certain files that are the targets of a particular rule. A simple makefile rule has the following form:

***target*... : *dependency*... command**

target is usually the name of a file that a program generates; examples of targets are executable or object files. A target can also be the name of an action to carry out, such as with the **clean** command (a command that, deletes all files from a build directory before building). *dependency* is a file that is used as input to create the target. A target often depends on several files. *command* is activated by **make** when *dependency* has changed. A rule may have more than one command, with each command on its own line. Usually a command is in a rule with dependencies and serves to create a target file if any dependencies change. However, the rule that specifies commands for the target does not require dependencies; for instance, the rule containing the **delete** command that associates with the target, **clean**, does not have dependencies. **make** activates commands on the dependencies to create or to update the target. A rule can also explain how and when to activate a command. A makefile may contain other text besides rules; a simple makefile requires only rules. Rules generally follow the same pattern.

TABs

Every command line in a makefile must have a TAB as the first character on the line. Any line that is not a command line must not have a TAB as its first character.

The example Example 4-2, shows a simplified makefile that describes the way an executable file called *edit* depends on eight object files which, in turn, depend on eight C source and three header files. In the example, all of the C files include `defs.h`, but only those defining editing commands include `command.h`, and only low level files that change the editor buffer include `buffer.h`.

Example 4-2. Example Makefile

```
edit: main.o kbd.o command.o display.o insert.o search.o files.o utils.o
 cc -o edit $<

main.o: main.c defs.h
 cc -c main.c

kbd.o: kbd.c defs.h command.h
 cc -c kbd.c

command.o: command.c defs.h command.h
 cc -c command.c

display.o: display.c defs.h buffer.h
 cc -c display.c
```

```
insert.o: insert.c defs.h buffer.h
 cc -c insert.c

search.o: search.c defs.h buffer.h
 cc -c search.c

files.o : files.c defs.h buffer.h command.h
 cc -c files.c

utils.o : utils.c defs.h
 cc -c utils.c

clean :
 rm edit main.o kbd.o command.o display.o insert.o \
 search.o files.o utils.o
```

The example makefile's targets include the executable file, `edit` and the `main.o` and `kbd.o` object files. `main.c` and `defs.h` are the dependency files. Each *.o* file is both a target and a dependency. When a target is a file, it needs to be recompiled or relinked if any of its dependencies change. In Example 4-2 *edit* depends on eight object files; the object file, `main.o`, depends on the source file, `main.c` *and* on the `defs.h` header file. A shell command follows each line that contains a target and dependencies, saying how to update the target file; a TAB character must come at the beginning of every command line to distinguish command lines from other lines in the makefile. **make** does not know anything about how the commands work; it is up to you to supply commands that will update the target file properly. All **make** does is execute the commands in the rule you have specified when the target file needs updating.

For more information, see *Using the GNU Compiler Collection (GCC)* in the gcc.info files.

newlib and libstdc++-v3, the C and C++ Libraries

newlib and **libstdc++-v3** are the C and C++ libraries. They serve as a collection of functions in compiled form, which link with a program to form a complete executable, linking either statically or, with some systems, dynamically.

newlib includes the GNU C library, **libc**, and the GNU C math library, **libm**.

libstdc++-v3 is based on the ISO 14882 Standard C++ Library, with all its compliant classes and functions. See http://sources.redhat.com/libstdc++/links.html for the **libstdc++** library documentation.

binutils, the GNU Binary Utilities

binutils, the GNU binary utilities, are available on all hosts. They include **ar**, **nm**, **objcopy**, **objdump**, **ranlib**, **size**, **strings**, and **strip**. For targets which use the ELF file format there is also a tool called **readelf**. There are three binary utilities, **addr2line**, **windres**, and **dlltool**, which are for use with Cygwin, the porting layer application for Win32 development. The most important of the binary utilities are **objcopy** and **objdump**.

- **objcopy** is a tool to convert object and executable files. It can add and remove sections and symbols, but the most commonly used feature is it ability to change the file format. For example, it can convert an ELF or COFF format executable into an S-record or Intel I-Hex format file. These two formats are often used in building ROM images for standalone and embedded systems. For more information, see *Using binutils, the GNU Binary Utilities* in the binutils.info files

- **objdump** is a tool to display information about the contents of an object or executable file. It can display symbol tables and section headers and it can also act as a disassembler. **Objdump** also knows about archives and libraries and can be used to display information on all of the object files inside them. For more information, see *Using binutils, the GNU Binary Utilities* in the binutils.info files.

  A few of the more useful options for **objdump** are: `-h` (to display section headers), `-t` (to display symbols), `-p` (to display private header information) and `-d` for assembler (to display a disassembly). `-d` will normally only disassemble sections that are expected to contain instructions. To disassemble all sections use `-D` instead. The option `--prefix-addresses` can be used to print a complete address on each line of the disassembler's output.

Cygwin, for Porting UNIX Applications for Working on Windows Systems

Cygwin a full-featured Win32 porting layer for UNIX applications, is compatible with all Win32 hosts (currently, these are Microsoft Windows XP systems). Cygwin makes all directories have similar behavior to a UNIX system, with all the UNIX default tools in their familiar place. Available scripting languages include **bash**, **tsh**, and **tcsh**. Tools such as Perl, Tcl/Tk, **sed**, **awk**, **vim**, Emacs, xemacs, **telnetd** and **ftpd** work in Cygwin.

In order to emulate a UNIX kernel and access all processes that can run with it, use the Cygwin DLL (dynamically linked library). The Cygwin DLL will create shared memory areas so that other processes using separate instances of the DLL can access the kernel. Using DLLs with Cygwin, you can link into your program at run time instead of build time. The following documentation describes the three parts of a DLL and their usage.

- *exports*, a list of functions and variables that the `.dll` file makes available to other programs as a list of *global* symbols, with the rest of the contents hidden. Create this list with a text editor; it's also possible to do it automatically from the list of functions in your code. The **dlltool** utility creates the exports section of the `.dll` file from your text file of exported symbols.
- *code and data*, the parts you write, the functions, variables, and so forth, merged together, building one object file and linking it to a `.dll` file; they are not put into a `.exe` file.
- *import library*, a regular UNIX-like `.a` library, containing the vital information for the operating system and the program to interacts as it or *imports* the `.dll` as data, linking the data into an `.exe` file, all generated by the **dlltool** utility.

The following example shows the use of the compile command with **gcc** demonstrating how to build a DLL library using a single source file *`mydll.c`* and then creating an executable *`myprog.exe`* from another source files *`myprog.c`* which links against the DLL.

Example 4-3. Creating a DLL

```
gcc –shared mydll.c gcc myprog.c –lmydll
```

For more information, find the `~/cygwin/doc` directory to locate documentation discussing use of the GNU development tools with a Win32 host and exploring the architecture of the Cygwin library; see also http://sources.redhat.com/cygwin/.

Using info, the Documentation Tools

info provides the sources for documentation for the GNU tools; it requires the following tools:

- **Texinfo** [7], **texindex** and **texi2dvi**, the standard GNU documentation formatting tools.
- **makeinfo** and **info** the GNU online documentation tools.
- *man pages*, the GNU documentation on all the tools and programs in this release.
- **flex**: A Fast Lexical Analyzer Generator [8] , which generates lexical analyzers suitable for GCC and other compilers.
- *Using and Porting GCC*, information about requirements for putting GCC on different platforms, or for modifying GCC; includes documentation from *Using the GNU Compiler Collection (GCC)* in the gcc.info files.
- **byacc** [9] , a compiler parser generator.
- *Texinfo: The GNU Documentation Format*, the documentation that details TeX and the printing and generating of documentation, as well as how to write manuals in the TeX style.
- *Configuration program*, descriptions of the configuration program that GNUPro Toolkit uses.
- *GNU Coding Standards*, the more elaborate details on the coding standards with which the GNU projects are developed.
- *GNU* **gprof**, details of the GNU performance analyzer (only for some systems).

You have the freedom to copy the documentation using its accompanying copyright statements, which include necessary permissions. To get the documentation in HTML or printable form, see http://www.fsf.org/doc/doc.html and http://www.fsf.org/doc/other-free-books.html.

See *Using the GNU Compiler Collection (GCC)* in the gcc.info files for documentation regarding these tools.

Reading info Documentation

Browse through the documentation using either **emacs** or the **info** documentation browser program. The information is in *nodes*, corresponding to the sections of a printed book. Follow them in sequence, as in books, or using the hyperlinks, find the node that has the information you need. **info** has *hot* references (if one section refers to another section, **info** takes you directly to that other section while allowing you to return easily to where you had been). You can also search for particular words or phrases. After installing the GNUPro Toolkit you can use **info** by typing its name at a shell prompt; no options or arguments are necessary. If you have problems running **info**, contact your system administrator.

To get help with using **info**, type **h** for a programmed instruction sequence, or **Ctrl + h** for a short summary of commands. To stop using **info**, type **q**.

See *Using the GNU Compiler Collection (GCC)* in the gcc.info files for detailed references for the **info** tools.

Learn More About the Tools

See http://www.redhat.com/docs/manuals/gnupro/ for more information about the tools. For help with using the tools, see Chapter 5.

To learn about rebuilding sources for the tools, see Chapter 8.

Notes

1. http://gcc.gnu.org/onlinedocs/libstdc++/documentation.html
2. http://sources.redhat.com/cygwin/
3. http://sources.redhat.com/cygwin/
4. http://sources.redhat.com/cygwin/
5. http://sources.redhat.com/libstdc++/links.html
6. http://sources.redhat.com/cygwin/
7. Requires TeX, the free technical documentation formatting tool written by Donald Knuth. See Texinfo: The GNU Documentation Format (ISBN: 1-882114 67 1).
8. See Flex: The Lexical Scanner Generator (ISBN: 1-882114 21 3).
9. See Bison Manual: Using the YACC-compatible Parser Generator (ISBN: 1-882114 44 2).
10. http://www.fsf.org/doc/doc.html
11. http://www.fsf.org/doc/other-free-books.html
12. http://www.redhat.com/docs/manuals/gnupro/

Chapter 5. GNUPro® Toolkit: Developing with GNUPro Toolkit

Use the following tutorials for standard *native* development with the tools:

- the Section called *Create Source Code*
- the Section called *Compile, Assemble, and Link from Source Code*
- the Section called *Run the Executable*
- the Section called *Debug the Executable*

Use the following tutorials for *cross-compilation* development where the programs being produced are intended to run on a different type of machine from the host machine.

- Chapter 6
- the Section called *Assembler Listings*

See http://www.redhat.com/docs/manuals/gnupro/ for more details about the tools.

Create Source Code

To start, create the following sample source code file and save it as `hello.c`. The following sections will show you how to compile this file to form an executable, and how to run it. This will also help to verify that the tools have been installed correctly.

Example 5-1. Example Source code to save as hello.c

```
#include <stdio.h>

int a, c;

static void
foo (int b)
{
  c = a + b;
  printf ("%d + %d = %d\n", a, b, c);
}

int
main (void)
{
  int b;

  a = 3;
  b = 4;

  printf ("Hello, world!\n");

  foo (b);

  return 0;
}
```

Compile, Assemble, and Link from Source Code

To compile the code to run on a stand-alone simulator, use the following command:

***TOOLPREFIX*-gcc -g hello.c -o hello**

Note that *TOOLPREFIX* is the name of compiler's target and file format. So for example if that target were an ARM and the file format was ELF then TOOLPREFIX would be "arm-elf" and the command run would be:

arm-elf-gcc -g hello.c -o hello

The `-g` option generates debugging information and `-o` specifies the name of the executable to be produced. Both of these can be omitted if you wish. Other useful options include `-O` to enable standard optimizations, and `-O2` for extensive optimizations. If no `-O` is specified, GCC will not optimize.

See *Using the GNU Compiler Collection (GCC)* in the gcc.info files for more basic compiler information.

Run the Executable

To run the program on the stand-alone simulator, use the following example:

***TOOLPREFIX*-run hello**

The simulator generates:

```
hello world! 3 + 4 = 7
```

The simulator executes the program, and returns when the program exits.

Debug the Executable

To start GDB, use the following commands:

***TOOLPREFIX*-gdb -nw hello**

The `-nw` option is used to select the command line interface to GDB (the Insight interface is the default; see Chapter 6). The `-nw` option is useful when you wish to report a problem you have with GDB, since a sequence of commands is simpler to reproduce. After the initial copyright and configuration information, GDB returns its own prompt, `(gdb)`. The following is a sample debugging session using the `target sim` command to specify the simulator as the target.

1. To specify the target on which to debug the executable (in this case the `sim` simulator), type:

   target sim

   GDB will then display:

   ```
   Connected to the simulator.
   ```

2. To load the simulator into memory, type:

   load

   This will produce some output describing the sections of the program as they are loaded into (simulated) memory. eg:

   ```
   Loading section .init, size 0x10 lma 0x0 Loading section .text,
   size 0xad6e lma 0x10 Loading section .fini, size 0x8 lma 0xad7e
   Loading section .rodata, size 0x372 lma 0xad88 Loading section
   .data, size 0x3d6 lma 0xb0fc Loading section .ctors, size 0x4
   lma 0xb4d2 Loading section .dtors, size 0x4 lma 0xb4d6 Loading
   ```

```
section .eh_frame, size 0x1054 lma 0xff04 Start address 0x10
Transfer rate: 403792 bits in <1 sec.
```

3. To set a breakpoint, type:

   break main

   GDB will then display:

   ```
   Breakpoint 1 at 0x132: file hello.c, line 15.
   ```

   Note the exact address and line number may vary, depending upon the target architecture being simulated and the exact layout of the C code in the `hello.c` file.

4. To run the program, type:

   run

   GDB will then display (when the program stops at a breakpoint):

   ```
   Starting program: hello Breakpoint 1, main () at hello.c:15 15
   a = 3;
   ```

   Again note that the exact line number and instruction displayed will be dependent upon the target architecture and source code layout.

5. To print the value of the variable *a* type:

   print a

   GDB displays:

   ```
   $1 = 0
   ```

6. To execute the next (source) line type:

   next :

   and GDB will display:

   ```
   16 b = 4;
   ```

7. To display the value of *a* again, type:

   print a

   This time GDB displays:

   ```
   $2 = 3
   ```

8. To display the program being debugged, type:

   list

   Which produces this output:

   ```
   12 int 13 main (void) 14 { 15 int b; 16 17 a = 3; 18 b = 4; 19
   20 printf ("Hello, world!\n"); 21 22 foo (b); 23 24 return 0;
   25 }
   ```

9. To list a specific function code, use the list command with the name of the function to be display. For example, type:

   list foo

   To display this:

   ```
   1 #include <stdio.h> 2 3 int a, c; 4 5 static void 6 foo (int
   b) 7 { 8 c = a + b; 9 printf ("%d + %d = %d\n", a, b, c); 10 }
   ```

10. To set a breakpoint at line seven, enter the following input (set a breakpoint at any line by entering break linenumber, where linenumber is the specific line number to break):

    break 8

    To which GDB responds with:

```
Breakpoint 2 at 0xf4: file hello.c, line 8.
```

11. To resume normal execution of the program until the next breakpoint, type:

    continue

    GDB displays:

    ```
    Continuing. Hello, world! Breakpoint 2, foo (b=4) at hello.c:8
    8 c = a + b;
    ```

12. To step to the next instruction and execute it, type:

    step

    GDB displays:

    ```
    9 printf ("%d + %d = %d\n", a, b, c);
    ```

13. To display the value of c, type:

    print c

    GDB displays:

    ```
    $3 = 7
    ```

14. To see how you got to where you are, type:

    backtrace

    GDB displays:

    ```
    #0 foo (b=4) at hello.c:9 #1 0x15c in main () at hello.c:18
    ```

15. To exit the program and quit the debugger, type:

    quit

Assembler Listings

The compiler normally turns a text based source file into a binary object file. It is possible however to instruct it to just convert the source code into assembler and stop there. The `-S` option does this. It also possible to instruct the compiler to produce an *assembler listing* as well as an object file. That can be done as follows:

***TOOLPREFIX*-gcc -c -O2 -Wa,-al hello.c**

The `-c` tells GCC to compile or assemble source files, but not to link them. `-O2` produces more fully optimized code. These are both optional. The `-Wa` tells the compiler to pass the comma-separated list of options which follows it on to the assembler. Hence the `-al` option is an assembler option to request an assembler listing.

This example shows a partial excerpt of an assembler listing for a MIPS based target.

Example 5-2. Example Assembly listing excerpt

```
38                              .align 2
39                              .globl main
40                              .type   main,@function
41                      main:
42 004c 9421FFF8                stwu 1,-8(1)
43 0050 7C0802A6                mflr 0
44 0054 9001000C                stw 0,12(1)
45 0058 48000001                bl __eabi
46 005c 3D200000                lis 9,a@ha
47 0060 3C600000                lis 3,.LC1@ha
48 0064 38000003                li 0,3
49 0068 38630010                la 3,.LC1@l(3)
50 006c 90090000                stw 0,a@l(9)
```

```
51 0070 48000001             bl puts
```

For some targets (but not all) it also possible to produce an assembler listing which intermixes the original input source code with the assembler instructions produced by the compiler. This can help track down bugs, discover how the compiler handles certain language constructs (such as function calls) and to learn more about assembly language. To do this, just add an `h` to the assembler option like this:

***TOOLPREFIX*-gcc -c -O2 -Wa,-alh hello.c**

Disassembling

The **objdump** tool can be used to to produce an disassembly of an object or executable file. It is used like this:

***TOOLPREFIX*-objdump -d hello.o**

You could also use the `-D` option to produce a disassembly of all of the sections in an object file, not just those that have been marked as containing instructions, and `-z` to tell objdump to disassemble instructions whose value is zero. (Normally such instructions are just skipped).

Creating a Startup File

In order to execute C and C++ programs it is necessary to run some special code at the very beginning of execution, *before* main is called. This code initializes system hardware, sets up any structures needed by the libraries, eg to handle calls to atexit(), and also calls any C++ global constructors.

The file containing this startup code is often called `crt0.o`. It is usually written in assembler as `crt0.s` and then turned into an object file for linking. The startup code defines a special *start* symbol which is the default startup address for your application.

Although the tools do come with an already assembled `crt0.o` file, for embedded targets it is often necessary to write your own `crt0.s`. Especially if the hardware you are using does not match that hardware that the crt0.s was written for. To write your own `crt0.s` module, you need the following information about your target.

- The memory map, showing the size of available memory and memory location
- The direction the stack grows (up or down)
- The output format in use (COFF, ELF, etc)

There are some examples available in the sources of GNUPro Toolkit for `crt0.s` code (along with examples of system calls with subroutines); look in the *installdir*`/src/libgloss/`*TARGET* directory (*installdir* refers to your installation directory).

Your `crt0.s` module must at least do the following in order for it to be useful:

- Define the **start** symbol. This is often called **_start**. Execution begins with it.
- Set up the stack pointer.

  Choose where to store your stack within the constraints of your target s memory map, the simplest choice being a fixed-size area somewhere in the uninitialized data section (often **bss**). Whether you choose a low address or a high address in this area depends on the direction your stack grows.

- Initialize all memory in the uninitialized-data **bss** section to zero.

  Use a linker script (see the Section called *Linker Scripts* and also *Using ld, the GNU Linker* in the ld.info files) to define symbols such as *bss_start* and *bss_end* to record the boundaries of this section; then you can use a **for** loop to initialize all memory between them in the `crt0.s` module.

- Call **main**. Nothing else will.

A more complete `crt0.s` module might also do the following:

- Provide code to be run after **main** returns. The exact behavior depends upon your system. There might be functions registered with **atexit** to be run. There might be a system monitor that can be called, or the code could just enter an infinite loop.
- If your target has no monitor for you to interpret output from the debugger, set up the hardware exception handler in the **crt0.s** module. See The GDB Remote Serial Protocol in *Debugging with GDB* in the gdb.info files for details for using the generic remote-target facilities for this purpose.
- Perform other hardware-dependent initialization (for example, initializing an **MMU** or an auxiliary floating-point chip).
- Define low-level input and output subroutines (for example, the **crt0.s** module is a convenient place to define the minimal assembly-level routines; see *System Calls in GNUPro Libraries*).

Linker Scripts

In the *`install-prefix`*`/`*`host`*`/`*`target`*`/lib/ldscripts/` directory you can find the default scripts used by the linker. (*`install-prefix`* signifies the top level directory where the tools are installed. *`host`* signifies your host configuration and *`target`* signifies your target configuration). In that directory there will be files with the extensions `.x`, `.xbn`, `.xn`, `.xr`, `.xs`, and `.xu`. The basic script is the one with the `.x` extension. The others are used when certain linker command line options are used. `.xr` is used when linking without relocating (`-r`), `.xu` is used when when linking without relocating, but when constructors are built (`-Ur`), `.xn` is used when page boundaries are not being used (`-n`), `.xbn` is used when page boundaries are not used and the .text section is writable (`-N`) and `.xs` is used when generating a shared library (`--shared`).

The linker script does the following:

- Sets up the memory map for the application.

  Different processors and different operating systems make different areas of memory available to an application. The linker script describes the layout of this memory, so that the linker can choose where to place the application's instructions and data.

- Sets up the constructor and destructor tables for C++ compiling. These are often built from several different components which need to be assembled in the correct order in order to work properly.
- Sets the default values for variables in use by the operating system. For example it can set up the initial value for the stack pointer, and the memory allocator's heap pointer and so on. It also specifies the name of the symbol that will contain the execution start address for the application.
- Describes how to map sections in the input files into sections in the output file. Sections can be grouped together, thrown away, overlapped and so on, depending upon how the script is written.

The linker script's format and syntax are described in the linker documentation which is normally in the form of an info file `ld.info` in the installed toolchain's

filespace. The following is small selections of fragments taken from a fictional linker script. They serve to demonstrate the general flavor of linker scripts.

A memory map is specified using the **MEMORY** command like this:

```
MEMORY { ram : ORIGIN = 0x10000, LENGTH = 2M rom (rx): ORIGIN =
0xff0000, LENGTH = 1M }
```

This map describes a region called *ram* which is 2 megabytes long starting at address 0x10000 and a region called *rom* which is only one megabyte long and which starts at address 0xff0000. This second region also has the attributes *read-only* and *execute* whereas the ram region does not have any attributes.

Memory region attributes are used when the linker decides where to place input sections which have not been explicitly assigned to a memory region. The linker will attempt to match the attributes of the input section to the attributes of each memory region in turn, going for the best match.

Input sections are mapped to output sections using the **SECTIONS** command:

Example 5-3. Linker Script

```
SECTIONS { .text : { *(.text) } > rom .data { *(.data) *(.sdata)
__edata = .; } > ram .bss { *(.bss) } /DISCARD/ { *(.dwarf*) } }
```

This example shows that any section *.text* in any input file will be mapped to a section called *.text* in the output file. The sections will be appended one after another in the order in which they occur on the linker's command line. The resulting combined input sections will be placed in the output file at address that lie within the *rom* memory section.

Any input section called *.data* or *.sdata* will be mapped to an output section called *.data*, which will be placed within the *ram* memory region. The linker will also create a variable called *__edata* whose value will be the address just after the end of the .data section in the output file.

Any input section called *.bss* will be mapped to a section also called *.bss* in the output file, and the memory region chosen will depend upon the attributes of the input .bss sections.

Any input section whose name starts with the string *.dwarf* will be thrown away. Any other input section, not explicitly matched by any of the above rules, will be mapped to sections matching their own names, and placed after all other sections in the output file.

For more information on linking, see *Using ld, the GNU Linker* in the ld.info files.

Toolchains and Usage

When you use the GNUPro Toolkit, you typically use a standard convention for calling each tool with a toolchain name. For cross-compilers, where the code produced by the compiler does not run on the same machine as the compiler, but is instead copied over to a different machine with a different architecture, the convention is to use the host name followed by the target name as a prefix to the tool.

The tables that follow provide the specific host and target names that you require. If you are not already familiar with the names of specific tools, review See the Section called *The Compiler and Development Tools* in Chapter 4.

Table 5-1. Toolchain names for Cygwin cross compilers

Target	Name
ARM	i686-pc-cygwin-x-arm-elf
MIPS	i686-pc-cygwin-x-mips-elf
MIPS64	i686-pc-cygwin-x-mips64-elf
MIPS ISA64	i686-pc-cygwin-x-mipsisa64-elf
PowerPC	i686-pc-cygwin-x-powerpc-eabi
PowerPC	i686-pc-cygwin-x-powerpc-eabispe

Table 5-2. Toolchain names for Red Hat Enterprise Linux cross compilers

Target	Name
ARM	i686-pc-gnulibc2.3-x-arm-elf
MIPS	i686-pc-gnulibc2.3-x-mips-elf
MIPS64	i686-pc-gnulibc2.3-x-mips64-elf
MIPS ISA64	i686-pc-gnulibc2.3-x-mipsisa64-elf
PowerPC	i686-pc-gnulibc2.3-x-powerpc-eabi
PowerPC	i686-pc-gnulibc2.3-x-powerpc-eabispe
XStormy 16	i686-pc-gnulibc2.3-x-xstormy16-elf
PowerPC Linux	i686-pc-gnulibc2.3-x-ppc-redhat-linux
ARM v5l Linux	i686-pc-gnulibc2.3-x-armv5l-linux-gnu
ARM v5 Soft Virtual Floating Point Linux	i686-pc-gnulibc2.3-x-armv5-softvfp-linux-gnu
ARM v5b Linux	i686-pc-gnulibc2.3-x-armv5b-linux-gnu
MIPS 64 Linux	i686-pc-gnulibc2.3-x-mips64-linux-gnu
MIPS 64 EL Linux	i686-pc-gnulibc2.3-x-mips64el-linux-gnu
MIPS Linux	i686-pc-gnulibc2.3-x-mips-linux-gnu
MIPS EL Linux	i686-pc-gnulibc2.3-x-mipsel-linux-gnu

Table 5-3. Toolchain names for Solaris cross compilers

Target	Name
StrongARM/XScale	sparc-sun-solaris2.7-x-arm-elf
MIPS	sparc-sun-solaris2.7-x-mips-elf
MIPS64	sparc-sun-solaris2.7-x-mips64-elf
MIPS ISA64	sparc-sun-solaris2.7-x-mipsisa64-elf
PowerPC	sparc-sun-solaris2.7-x-powerpc-eabi
PowerPC	sparc-sun-solaris2.7-x-powerpc-eabispe

Notes

1. http://www.redhat.com/docs/manuals/gnupro/

Chapter 6. GNUPro® Toolkit: Debugging with Insight

The following documentation serves as a general reference for debugging with GNUPro Toolkit's graphical user interface, Insight. In addition to this information, see Insight's **Help** menu for discussion of general functionality and use of menus, buttons, or other features. (See also *Debugging with GDB* in the gdb.info files and http://www.redhat.com/docs/manuals/gnupro/).

Starting Insight

Insight works as the default means for debugging; to disable the GUI, use the **gdb -nw** command for *non-windowing* command line work.

1. From a shell window, enter the following command:

   insight

   Insight launches, displaying the *Source Window*. (See figure Figure 6-1).

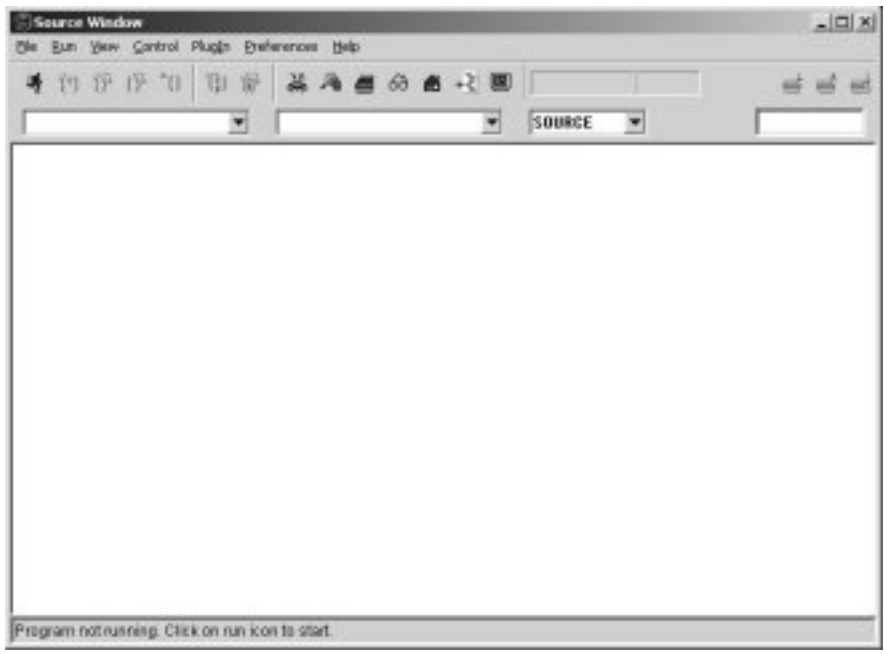

Figure 6-1. Source Window: the main window interface for Insight

The menu selections in the *Source Window* are **File**, **Run**, **View**, **Control**, **Plugin**, **Preferences**, and **Help**. To work with the other windows for debugging purposes specific to your project, use the **View** menu or the buttons in the toolbar.

1. To open a specific file as a project for debugging, select **File** then **Open** in the *Source Window*. The file's contents will then pass to the GDB interpreter.
2. To start debugging, click the **Run** button (See figure Figure 6-2) from the *Source Window*.

Figure 6-2. The Run Button

> **Warning**
>
> When debugging a target, do not click on the **Run** during an active debugging process, since using the **Run** button will effectively restart the session with all work unrecoverable.

When the debugger runs, the button turns into the **Stop** button. (See figure Figure 6-3).

Figure 6-3. The Stop Button

The **Stop** button interrupts the debugging process for a project,provided that the underlying hardware and protocols support such interruptions. Generally, machines that are connected to boards cannot interrupt programs on those boards. In such cases, a dialog box appears as a prompt asking if you want to abandon the session and if the debugger should detach from the target. For an embedded project, click **Run**; then click the **Continue** button. (See figure Figure 6-4).

Figure 6-4. The Continue Button

For more information on Insight, see its **Help** menu. For examples of debugging session procedures for using Insight, see the following documentation (the content assumes familiarity with debugging procedures):

- the Section called *Select and Examine a Source File*
- the Section called *Set Breakpoints and View Local Variables*
- the Section called *Set Breakpoints on Multiple Threads*

To specify how source code appears and to change debugging settings, from the **Preferences** menu, select **Source**.

When debugging remotely, you can find the processor name and identification coded of the target board by selecting **Plugin** then `target` from the *Source Window*.

CPU Information, information which is obtained from the processor's response to the CPUID instruction (`target` changes with every target architecture for Insight; environment variables that you set help Insight automatically to determine this functionality). To add identification codes to the debugger's table of processors, see the *GDB Internals* documentation, distributed with the source code.

Select and Examine a Source File

To select a source file, or to specify what to display when examining a source file when debugging, use the following processes.

1. Select a source file from the file drop-down list with the *Source Window* (`hello.c` in figure Figure 6-5).

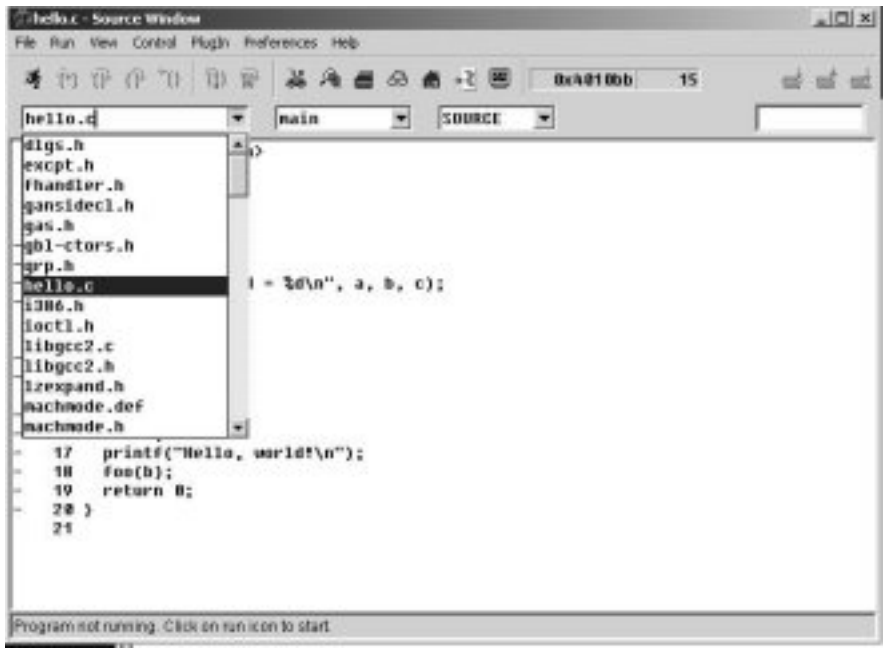

Figure 6-5. Source File Selection

1. Select a function from the function drop-down list to the right of the file drop-down list, or type its name in the text field above the list to locate the function (in figure Figure 6-6, see the executable line 11, where the *main* function displays).

Figure 6-6. Searching For Functions

1. Use the *Enter* key to repeat a previous search. Use the *Shift* and *Enter* keys simultaneously to search backward.
2. Type @ with a number in the search text box in the top right of the *Source Window*. Press *Enter*. Figure Figure 6-7, shows a jump to line 8 in the `hello.c` source file.

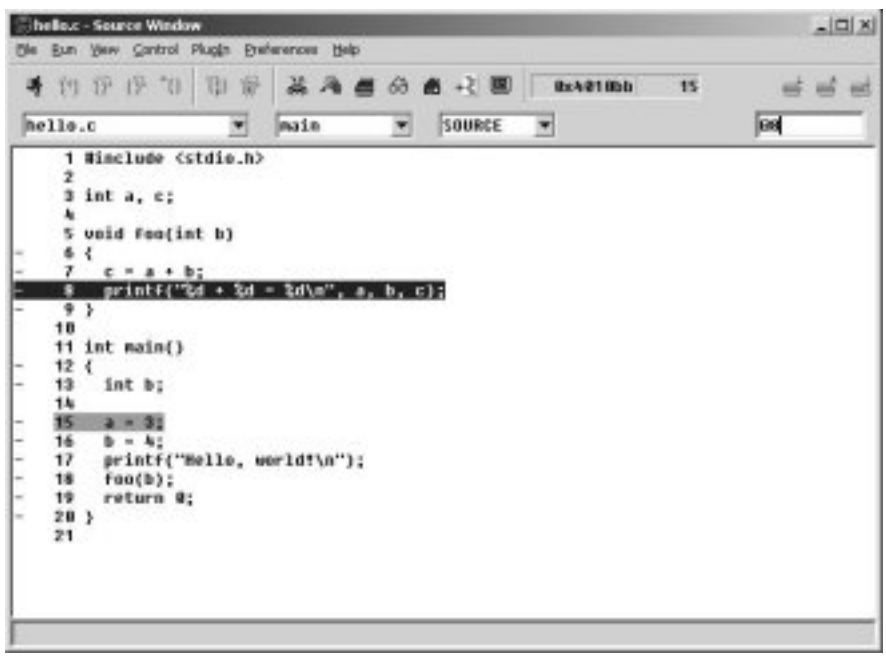

Figure 6-7. Searching For a Specific Line in the Source Code

Set Breakpoints and View Local Variables

A breakpoint can be set at any executable line in a source file.

Executable lines are marked by a minus sign in the left margin of the *Source Window*. When the cursor is over a minus sign for an executable line, the cursor changes to a circle. When the cursor is in this state, a breakpoint can be set. The *Breakpoints* window is for managing the breakpoints: disabling them, enabling them, or erasing them; an enabled breakpoint is one for which the debugging session will stop, a disabled breakpoint is one which the debugging session ignores.

The following exercise steps you through setting four breakpoints in a function, as well as running the program and viewing changed values in local variables.

For more information about breakpoints, see *Debugging with GDB* in the gdb.info files and http://www.redhat.com/docs/manuals/gnupro/.

1. To set a breakpoint, have an active source file open in the *Source Window*, and, with the cursor over a minus sign on a line, click the left mouse button. When you click on the minus sign, a red square appears for the line, signifying a set breakpoint (see the highlighted line 15 in figure Figure 6-8 , for a set breakpoint).

Clicking the line again will remove the breakpoint.

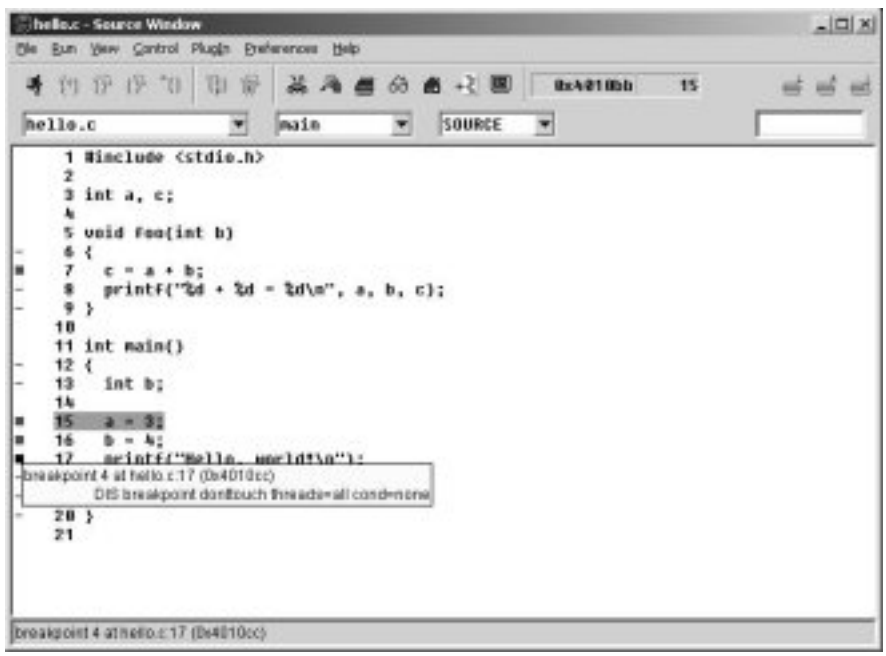

Figure 6-8. Results of Setting a Breakpoint on Line 17

1. Open the *Breakpoints* window (figure Figure 6-9) using the **Breakpoints** button from the *Source Window*. See a line with a check box in the window appears showing that you set a breakpoint for a corresponding line in the *Source Window* frame. With the cursor over a breakpoint, a *breakpoint information balloon* displays in the *Source Window* (the information details the breakpoint, its address, its associated source file and line, its state, whether enabled, temporary, or erased, and the association to all threads for which the breakpoint will cause a stop; see also the Section called *Set Breakpoints on Multiple Threads* , for details about threads).

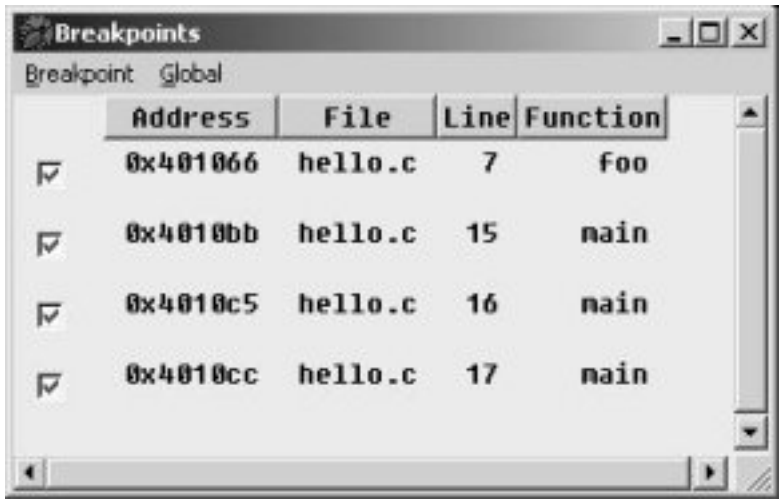

Figure 6-9. The Breakpoints Window

1. The debugger ignores disabled breakpoints, lines indicated having a black square over them in the *Source Window* frame (see line 17 in figure Figure 6-8). Click on a breakpoint to disable it. figure Figure 6-10, shows the results in the *Breakpoints* window of disabling a breakpoint. Re-enable a breakpoint at a line by clicking on the check box in the *Breakpoints* window. Once a breakpoint is enabled for a line, it will again have a red square in the *Source Window* frame.

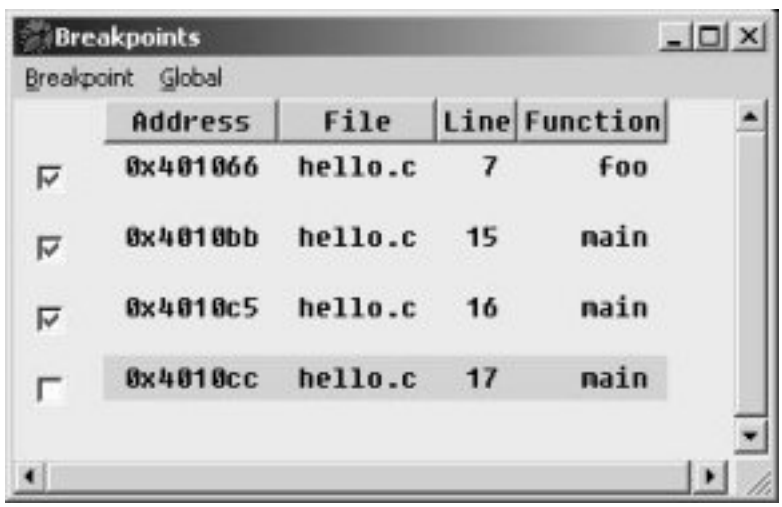

Figure 6-10. Results of Disabling a Breakpoint at Line 17

1. Repeat the process to set breakpoints at specific lines.
2. Click **Run** in the *Source Window* to start the executable. The debugger runs until it finds a breakpoint. When the target stops at a breakpoint, the debugger highlights a line (see highlighted line 17 in figure Figure 6-13, where the debugging stopped).

The following exercises illustrates how to view local variables, and use this information while using breakpoints:

1. Open the *Local Variables* window by clicking its button in the tool bar for the *Source Window*; the *Local Variables* window displays the values of the variables (see figure Figure 6-11, for the *b* variable in `hello.c`).

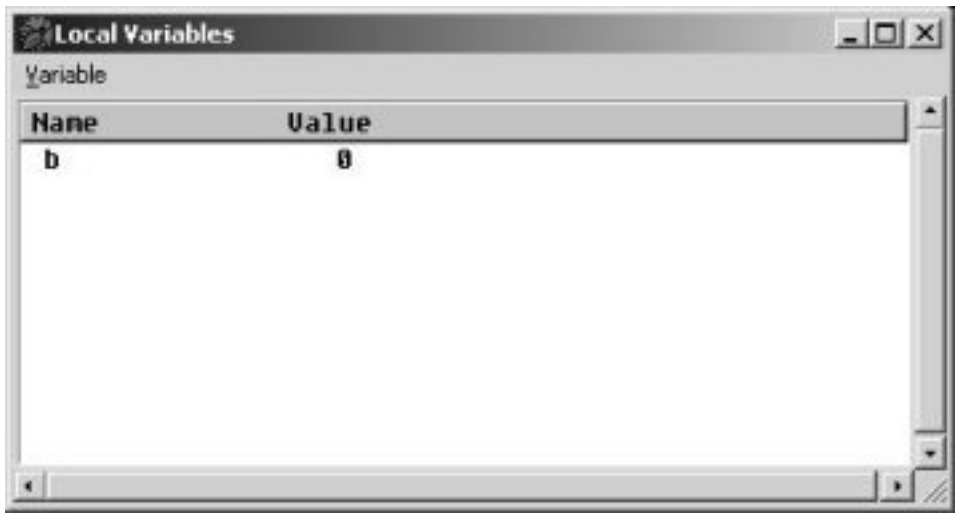

Figure 6-11. The Local Variables Window

1. Click the **Continue** button in the *Source Window* tool bar to move to the next breakpoint. The variables that changed value turn color in the *Local Variables* window (see results in figure Figure 6-12 for the *b* variable in `hello.c`).

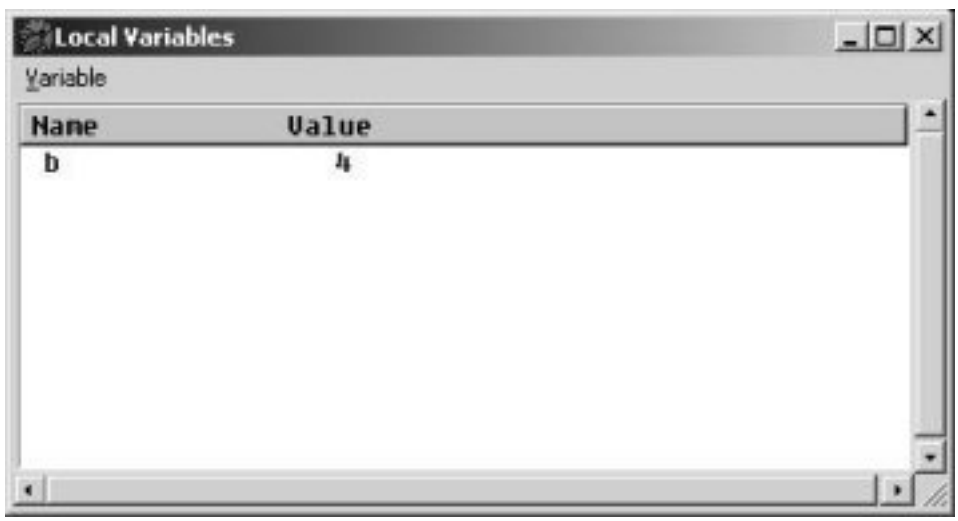

Figure 6-12. The Local Variables Window After Setting Breakpoints

1. Click the **Continue** button two more times to step through the next two breakpoints (until execution stops at line 17) and see the values of the local variables change (compare results from `hello.c` in figure Figure 6-8 and the results in figure Figure 6-13).

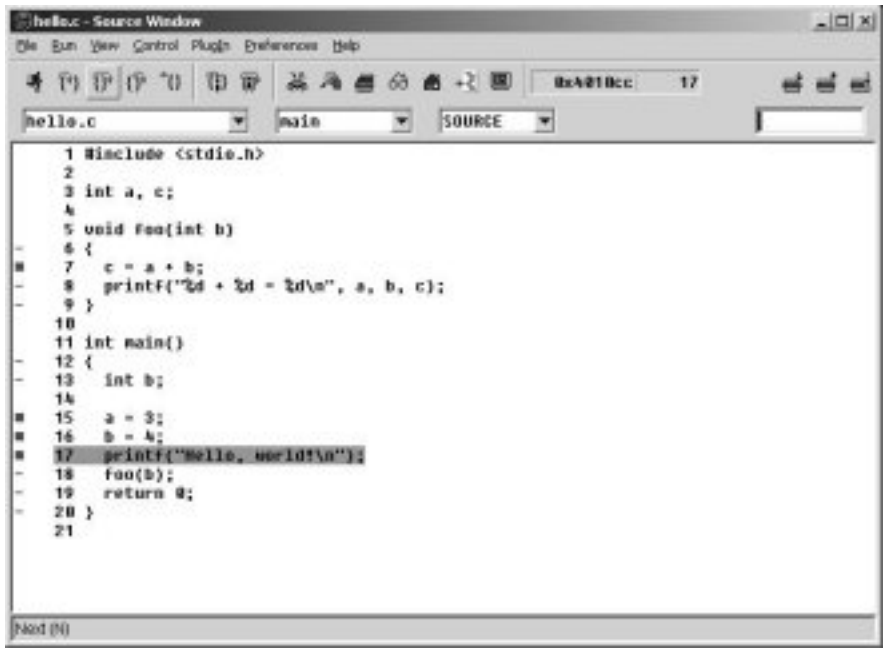

Figure 6-13. The Executable After Changing a Local Variable

Set Breakpoints on Multiple Threads

You can select threads and set breakpoints on one or more threads when debugging a multi-threaded application with Insight.

> **Warning**
>
> Working with multiple threads does not function similarly on all embedded targets. When debugging C++ code, for instance, breakpoints and exceptions may not work on multiple threads.

A process can have multiple threads running concurrently, each performing a different task, such as waiting for events or something time-consuming that a program does not need to complete before resuming. The thread debugging facility allows you to observe all threads while your program runs. However, whenever the debugging process is active, one thread in particular is always the focus of debugging. This thread is called the *current thread*. The precise semantics of threads and the use of threads differs depending on operating systems. In general, the threads of a single program are like multiple processes, except that they share one address space (that is, they can all examine and modify the same variables). Additionally, each thread has its own registers and execution stack and, perhaps, private memory.

1. In the *Source Window*, right click on an executable line without a breakpoint to open the breakpoint pop-up menu (figure Figure 6-14).

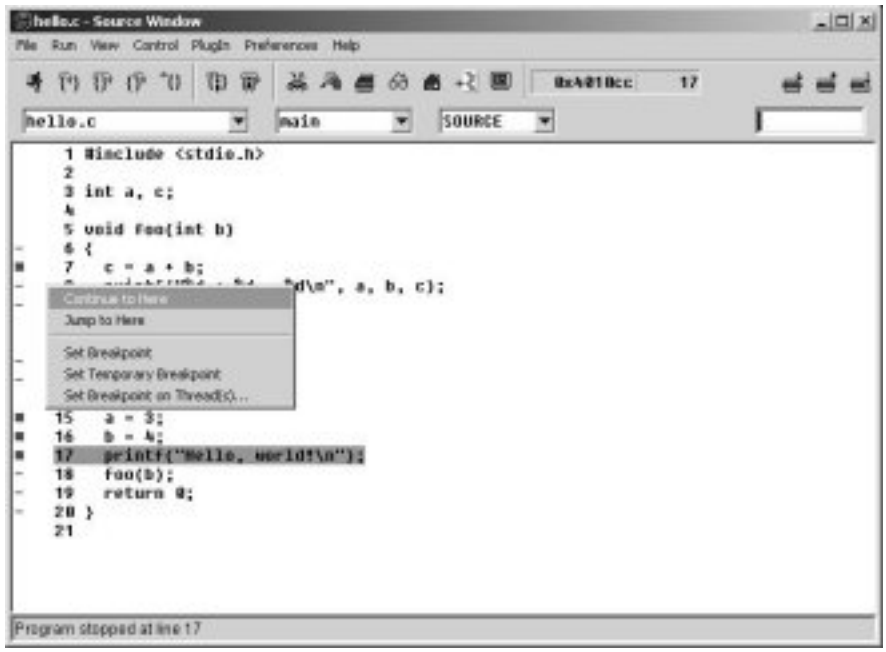

Figure 6-14. The Breakpoint pop-up Menu in the Source Window

1. Select *Set Breakpoint on Thread(s)* to display a window allowing you to choose the threads with which you set breakpoints. The *Processes* window (see figure Figure 6-15), available from the *Source Window*'s **View** then **Threads List** menu, displays all the available threads in the system and allows you to change the current thread.

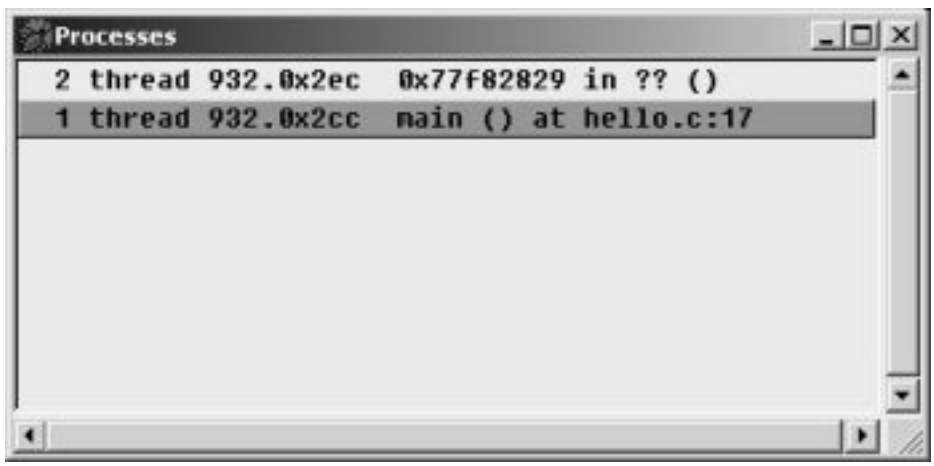

Figure 6-15. The Process Window

For more general information about threads, see *Debugging with GDB* in the gdb.info files and http://www.redhat.com/docs/manuals/gnupro/)

Notes

1. http://www.redhat.com/docs/manuals/gnupro/
2. http://www.redhat.com/docs/manuals/gnupro/

3. http://www.redhat.com/docs/manuals/gnupro/

Chapter 7. GNUPro® Toolkit: Using Cygwin

This section discusses the architecture underlying the Cygwin tools for porting UNIX applications to Windows 32-bit environments. For more information, see:

http://sources.redhat.com/cygwin/

GNUPro Toolkit provides the following packages for Cygwin:

GNUPro development tools:

binutils, **bison**, **byacc**, **dejagnu**, **diff**, **expect**, **flex**, **as**, **gcc**, **gdb**, **itcl**, **ld**, **libstdc++**, **make**, **patch**, **tcl**, **tix**, **tk**.

Cygwin Net Release tools

ash, **bash**, **bzip2**, **diff**, **fileutils**, **findutils**, **gawk**, **grep**, **gzip**, **m4**, **sed**, **shellutils**, **tar**, **textutils**, **ssh**, **cvs**, **rsync**, **perl**, **psql**, **python**, **ssl**, **vim**, **time**, and more...

Cygwin emulates a standard UNIX directory structure. To ensure the structure is the same, always have `/tmp` both with and without the mount table translations. To emulate the `/etc` directory (so that the **ls -l** UNIX directory contents listing command works), use the following example's declarations as a guide:

```
mkdir /etc/ cd /etc mkpasswd > /etc/passwd mkgroup > /etc/group
```

Cygwin comes with both **bash.exe** and **sh.exe** shells. **sh.exe** is based on **ash**. In case of trouble with **ash**, make **sh.exe** point to the **bash.exe** shell.

When executing a binary linked against the library, the Cygwin dynamically linked library (DLL) loads into the application's text segment. To emulate a UNIX kernel for allowing access to all processes that can run with it, use the Cygwin DLL. The Cygwin DLL will create shared memory areas so that other processes using separate instances of the DLL can access the kernel. The DLL keeps track of open file descriptors and, among other purposes, assists **fork** and **exec** calls. In addition to the shared memory regions, every process also has a per-process structure that contains information such as process ID, user ID, signal masks, and other similar process-specific information.

The DLL uses the Win32 API. Because processes run under the standard Win32 subsystem, they can access both the UNIX compatibility calls provided by Cygwin as well as any of the Win32 API calls. This gives the programmer complete flexibility in designing the structure of their program in terms of the API in use; for example, a project might require a Win32-specific GUI using Win32 API calls on top of a UNIX back-end that uses Cygwin.

The **CYGWIN** environment variable helps to override the Win32 default behavior and force POSIX [2] standards compliance.

UNIX applications that have to switch the user context have the **setuid** and **seteuid** calls, which are not part of the Microsoft Windows API. Nevertheless these calls are supported under Microsoft Windows 2000, Microsoft Windows XP and Microsoft Windows Vista with Cygwin.

Before Using Cygwin

Before using Cygwin tools, read these sections: the Section called *The Mount Table and the mount Utility*, the Section called *Text and Binary Modes with Cygwin*, the Section called *File Permissions With Cygwin*, the Section called *Special Cygwin File Names*, the Section called *Building and Using DLLs with Cygwin* and the Section called *Defining Windows Resources for Cygwin*.

The Mount Table and the mount Utility

The **mount** utility controls a mount table for emulating a POSIX view of a Windows file system space. To change a Windows path that uses `/` and mount arbitrary Win32 paths into the POSIX file system space, use **mount** to mount each drive letter under a slash partition (such as `C:\` to `/c` or `D:\` to `/d`, and so forth).

Executing the **mount** command without any arguments prints the current mount table to a screen. Alternatively, provide the intended Win32 path to **mount** as a first argument and the POSIX path as the second argument.

In executing the **mount** command without any arguments, you may note that there are already some mount points created. By default, the graphical installer for the Cygwin DLL Release will update the mount table to use the Cygwin Net Release Binaries as `/`. If an old mount table was already on the system, it can be restored by running the restoreoldmounts.bat file in the installation directory.

If desired, one may issue the following command to remove all mount points.

mount --remove-all-mounts

The example Example 7-1 demonstrates mounting `C:/cygwin/H-i686-pc-cygwin/bin` to `/bin` (assuming `/bin` exists), as well as mounting `\\pollux\home\joe\data` to a `/data` network directory, making `/bin/sh` a valid shell, to satisfy **make**.

> **Note:** With a UNIX derived shell like bash or tcsh, the backward slash will not work; substitute / or use surround the path with single quotes.

Example 7-1. Using mount

```
$ ls /bin /data ls: /data: No such file or directory $ mount
C:\cygwin\H-i686-pc-cygwin\bin /bin $ mount \\pollux\home\joe\data
/data Warning: /data does not exist! $ mount Device Directory
Type Flags \\pollux\home\joe\data /data native text!=binary
C:\cygwin\H-i686-pc-cygwin\bin /bin native text!=binary D: /d native
text!=binary C: / native text!=binary $ ls /bin/sh /bin/sh
```

Text and Binary Modes with Cygwin

The following documentation discusses some of the main distinction with text and binary modes with UNIX and Microsoft Windows interoperability, and how Cygwin solves the problems. See the Section called *File Permissions With Cygwin* and the Section called *Special Cygwin File Names*. On a UNIX system, when an application reads from a file it gets exactly what's in the file on disk and the same is true for writing to the file. The situation is different in the DOS and Microsoft Windows world where a file can be opened in one of two modes, either binary or text. In the binary mode, the system behaves exactly as in UNIX. However in text mode there are major differences:

- On writing in text mode, a new line, NL (**\n**, **^J**), is transformed into a carriage return/new line sequence, or CR (**\r**, **^M**) NL.
- On reading in text mode, a carriage return followed by a new line is deleted and a **^Z** character signals the end of file.

The mode can be specified explicitly. In an ideal DOS and Microsoft Windows world, all programs using lines as records (such as **bash**, **make**, or **sed**) would open files

(changing the mode of their standard input and output) as text. All other programs (such as **cat**, **cmp**, or **tr**) would use binary mode. In practice with Cygwin, programs that deal explicitly with object files specify binary mode (as is the case of **od**, which is helpful to diagnose CR problems). Most other programs (such as **cat**, **cmp**, **tr**) use the default mode. The Cygwin system gives us some flexibility in deciding how files are to open when the mode is not specified explicitly:

- If the file appears to reside on a file system that is mounted (that is, if its pathname starts with a directory displayed by **mount**), then the default is specified by the **mount** flag.

  The **CYGWIN** environment variable will affect a disk file when you are using **stdio** redirection from the Microsoft Windows prompt. Otherwise, the mode of the file is based on the mode specified by a `--change-cygdrive-prefix` call, as the following example shows (which makes the default mode be binary).

  mount -b --change-cygdrive-prefix /cygdrive

  If the file is a symbolic link, the mode of the target file system applies.
- Pipes and non-file devices are always opened in binary mode.
- When a Cygwin program is launched by a non-Cygwin shell, its standard input, output, and error are in text mode by default. This can be overridden by setting the CYGWIN environment variable to 'binmode'.

  When redirecting, the Cygwin shells uses the first three rules. For these shells, the relevant value of **CYGWIN** is that at the time the shell was launched and not that at the time the program is executed.

  Non-Cygwin shells always pipe and redirect with binary mode. With non-Cygwin shells, the **cat** *`filename`* **|** *`program`* and *`program`* **<** *`filename`* commands are not equivalent when *`filename`* is on a text-mounted partition. To illustrate the various rules, the following example's script deletes CRs from files by using the **tr** program, which can only write to standard output.

  ```
  #!/bin/sh # Remove \r from the files given as arguments for file
  in "$@" do CYGWIN=binmode sh -c "tr -d \\\"\\\r\\\" < '$file' >
  c:tmpfile.tmp" if [ "$?" = "0" ] then rm "$file" mv c:tmpfile.tmp
  "$file" fi done
  ```

  The script works irrespective of the mount because the second rule applies for the path, `c:tmpfile.tmp`. According to the fourth rule, **CYGWIN** must be set before invoking the shell. These precautions are necessary because **tr** does not set its standard output to binary mode. It would thus reintroduce **\r** when writing to a file on a text mounted partition. The desired behavior can also be obtained by using **tr -d \r** in a **.bat** file.

UNIX programs that have been written for maximum portability will know the difference between text and binary files and act appropriately under Cygwin. For those programs, the text mode default is a good choice. Programs included in official distributions should work well in the default mode.

Text mode makes it much easier to mix files between Cygwin and Microsoft Windows programs, since Microsoft Windows programs will usually use the carriage return/line feed (CR/LF) format. Unfortunately you may still have some problems with text mode. First, some of the utilities included with Cygwin do not yet specify binary mode when they should. Second, you will introduce CRs in text files you write, causing problems when moving them back to a UNIX system.

If you are mounting a remote file system from a UNIX machine, or moving files back and forth to a UNIX machine, you can access them in binary mode since text files found there will normally be NL format anyway, and you would want any files put there by Cygwin programs to be stored in a format that the UNIX machine will understand. Remove CRs from all Makefiles and shell scripts and make sure that you only edit the files with DOS/Windows editors that can cope with binary mode files.

Note: You can decide this on a disk by disk basis (for example, mounting local disks in text mode and network disks in binary mode). You can also partition a disk, for example by mounting `c:` in text mode, and `c:\home` in binary mode.

File Permissions With Cygwin

On Windows 95 and Windows 98 systems, files are always readable, and Cygwin uses the native read-only mode to determine if they are writable. Files are executable if the filename ends with `.bat` or `.com` or `.exe` file extensions, or if **#!** starts. Consequently, **chmod** can only affect the **w** mode,whereas it silently ignores actions involving the other modes. Under NT, file permissions default to the same behavior as Windows 95 and Windows 98 systems. However, there is optional functionality in Cygwin that can make file systems behave more like files for UNIX systems; for instance, use the **ntea** option to the **CYGWIN** environment variable (see the Section called *Environment Variables for Cygwin*). When using **ntea**, Cygwin will start with basic permissions, storing POSIX file permissions in NT *Extended Attributes*. On NTFS partitions, the attributes can be stored sensibly inside the normal NTFS filesystem structure. However, on a FAT partition, NT stores extended attributes in a flat file at the root of the **EA DATA. SF.** file partition, which can grow to extremely large sizes if you have a large number of files on the partition, slowing the system processes. In addition, the **EA DATA. SF.** file can only be deleted outside of Windows because of its *in use* status. For these reasons, the use of NT *Extended Attributes* is off by default in Cygwin. Specifying **ntea** in **CYGWIN** has no effect for Windows 95 and Windows 98 systems. Under NT, the **[-w** *filename* **]** test is only true if *filename* is writable across the board, such as with a **chmod +w** *filename* call. However, **ntea** only simulates the permissions inside of Cygwin. On NT systems (2000, XP, Vista) you can use real file permissions when using the NTFS filesystem. In this case it might makes sense to set **ntsec** in the **CYGWIN** environment variable. Then Cygwin uses the NT file permission access control lists to set POSIX like permissions.

Note: In contrast to FAT, FAT32 does not support storing extended attributes.

Special Cygwin File Names

The following documentation discusses some special file naming usage by Cygwin.

- DOS devices: Windows filenames invalid under Windows are also invalid under Cygwin. This means that base filenames such as `AUX`, `COM1`, `LPT1` or `PRN` cannot be used in a regular Cygwin Windows or POSIX path, even with an extension (`prn.txt`). However, the special names can be used as filename extensions (`file.aux`). You can use the special names as you would under DOS; for example, you can print on your default printer with the **cat** *filename* **> PRN** command (making sure to end with a Form Feed).
- POSIX devices: do not create a POSIX `/dev` directory as it is automatically simulated within Cygwin. It supports the following devices: `/dev/null`, `/dev/tty` and`/dev/comX` (the serial ports). These devices cannot be seen with the **ls /dev** command, although commands such as **ls /dev/tty** work fine.
- The `.exe` extension for executable program filenames end with `.exe` although the `.exe` extension is not necessary in a command, so that traditional UNIX names can be used. On the contrary, you cannot omit `.bat` and `.com` extensions. As a side effect, the **ls** *filename* declaration gives information about *filename* `.exe`

if *filename* .exe exists and *filename* does not. In the same situation, call **stat("*filename*"...)** to give information about the *filename* .exe file.

Differentiate any two files by examining their inodes.

- The GCC compiler produces a *filename* .exe executable when asked to produce *filename*, allowing many makefiles written for UNIX systems to work well under Cygwin. Unfortunately the **install** and **strip** commands do distinguish between *filename* and a *filename* .exe file. They fail when working on a non-existing *filename* even if *filename* .exe exists, thus breaking some makefiles. To solve this problem, write **install** and **strip** shell scripts to provide the .exe extension.
- @pathname circumvents the limitations on shell line length in the native Windows command shells, allowing Cygwin programs to expand their arguments by starting with @ in a special way. If a file pathname exists, the **@pathname** argument expands recursively to the content of the program's pathname. Use double quotes in the file's content to delimit strings containing blank space.

Defining Windows Resources for Cygwin

windres reads a Windows resource file (*.rc) and converts it to a .res COFF file. The syntax and semantics of the input file are the same as for any other resource compiler; see any publication describing the Windows resource format for details.

windres compiles a .res file to include all the bitmaps, icons, and other resources you need, into one object file. Omitting the -O coff declaration would create a Windows .res format file without linkable COFF objects. Instead, **windres** produces a COFF object, for compatibility with how a linker can handle Windows resource files directly, maintaining the .res naming convention.

For more information on **windres**, see *Using binutils, the GNU Binary Utilities* in the binutils.info files.

Building and Using DLLs with Cygwin

The following example shows how to build a DLL library, using a single file *mydll.c* for its contents.

Example 7-2. Create a DLL

gcc -shared mydll.c -o *mydll.dll* -e *_mydll_init@12*

The following example shows how to build an executable which links against a DLL and which uses a single file *myprog.c* as its source.

Example 7-3. Linking against a DLL

gcc *myprog.c* *mydll.dll* -o *myprog.exe*

Using GCC with Cygwin

The following documentation discusses using the GNUPro compiler with Cygwin. For example compiling a simple hello world type program can be done by typing this command into a shell:

Example 7-4. Compiling Hello World

gcc hello.c -o hello.exe

Cygwin allows you to build programs with full access to the standard Windows 32-bit API, including the GUI functions (as defined in Microsoft publications); however, the process of building those applications is slightly different using the GNU tools instead of the Microsoft tools. Your sources will not need to change; just remove all `__export` attributes from functions and replace them, as the following example shows.

Example 7-5. Using the __dllexport__ attribute

```
int foo (int) __attribute__((__dllexport__)); int foo (int i) {
```

The Makefile is similar to any other UNIX-like or Cygwin Makefile. The only difference is that you use a **gcc -mwindows** declaration to link your program (*myapp*.exe in the following example's script) into a GUI application instead of into a command line application.

Example 7-6. Makefile using -mwindows

```
myapp.exe : myapp.o myapp.res gcc –mwindows myapp.o myapp.res –o $@
myapp.res : myapp.rc resource.h windres $lt; –O coff –o $@
```

Debugging Cygwin Programs

Before you can debug your program, you need to prepare your program for debugging. Add a `-g` declaration to all the other flags you use when compiling your sources to objects, in order to add extra information to the objects (making them much bigger), and to provide critical information to the debugger regarding line numbers, variable names, and other useful things; these extra symbols and debugging data give your program enough information about the original sources so that the debugger can resolve the problems. Use declarations like the following example's commands.

Example 7-7. Compiling with -g

```
gcc –g –O2 –c myapp.c gcc –g myapp.c –o myapp
```

To invoke **gdb**, use the **gdb** *myapp***.exe** declaration (substituting the executable file's name for *myapp*). The copyright text displays followed by the **(gdb)** prompt, waiting for you to enter commands like **run** or **help**.

If your program crashes and you re trying to determine why it crashed, the best thing to do is type **run** and let your program run. After it crashes, you can use the **where** command to determine where it crashed, or the **info locals** call to see the values of all the local variables. The **print** declaration lets you examine individual variables or a line to which pointers point. If your program is doing something unexpected, use the **break** command to tell the debugger to stop your program when the debugging process gets to a specific function or line number.

Using the **run** command, stop your program at a breakpoint, and use other **gdb** commands to look at the state of your program at that point, to modify variables, and to step through your program's statements one at a time.

Use the **help** command to get a list of all the commands to use, or see *Debugging with GDB* in the gdb.info files.

Environment Variables for Cygwin

Before starting **bash**, you must set some environment variables, some of which can also be set or modified inside **bash**. You have a `.bat` file where the most important ones are set before initially invoking bash. The fully editable `.bat` file installs by default in `\..\cygwin\cygnus.bat` and the **Start** menu points to it.

The most important environment variable is the **CYGWIN** variable. The **CYGWIN** variable is used to configure many global settings for the Cygwin runtime system. Initially you can leave **CYGWIN** unset or set it to **tty** using input like the following example's syntax in a DOS shell, before launching **bash**.

Example 7-8. Setting the CYGWIN environment variable

```
set CYGWIN=tty notitle
```

The **PATH** environment variable is used by Cygwin applications as a list of directories to search for executable files to run. Convert this environment variable, when a Cygwin process first starts, from a Microsoft Windows format (`C:\WinNT\system32;C:\WinNT`) to UNIX format (`/WinNT/system32:/WinNT`).

Set the **PATH** environment variable so that, before launching **bash**, it contains at least a `bin` directory: `C:\..\cygwin\H-i686-pc-cygwin\bin`.

make uses an environment variable, **MAKE_MODE** lines. If you get errors from **make** with the **/c not found** message, set **MAKE_MODE** to **UNIX** with the following example's form.

Example 7-9. Setting the MAKE_MODE environment variable

```
set MAKE_MODE=UNIX
```

The **HOME** environment variable is used by UNIX shells to determine the location of your home directory. This environment variable is converted from the Microsoft Windows format (that is, `C:\home\bob`) to UNIX format (that is, `/home/bob`) when a Cygwin process first starts. To prevent confusion, ensure that **HOME** and `/etc/passwd` agree on your home directory.

The **TERM** environment variable specifies your terminal type. It is set it to **cygwin** by default.

The **LD_LIBRARY_PATH** environment variable is used by the Cygwin function, dlopen (), as a list of directories to search for **.dll** files to load. This environment variable is converted from the Microsoft Windows format (that is, `C:\WinNT\system32;C:\WinNT`) to UNIX format (that is, `/WinNT/system32:/WinNT`) when a Cygwin process first starts.

The **CYGWIN** environment variable is used to configure many global settings for the Cygwin runtime system, using the following options.

> **Note:** Each option is separated by others with a space. Many options can be turned off by prefixing with **no** (such as **nobar** or **bar** options).

(no)binmode

If set, unspecified file opens by default to binary mode (no **CR**/**LF** translations) instead of text mode. This option must be set before starting a Cygwin shell to have an effect on redirection. On by default.

(no)envcache

If set, environment variable conversions (between Win32 and POSIX) are cached. Note that this is may cause problems if the mount table changes, as the cache is not invalidated and may contain values that depend on the previous mount table contents. Defaults to set.

(no)export

If set, the final values of these settings are re-exported to the environment as **$CYGWIN** again.

(no)title

If set, the title bar reflects the currently running program's name. Off by default.

(no)glob

If set, command line arguments containing UNIX-style file wildcard characters (brackets, question mark, asterisk) are expanded into lists of files that match those wildcards. This is applicable only to programs running from a windows command line prompt. Set by default.

(no)tty

If set, Cygwin enables extra support (such as **termios**) for UNIX-like **tty** calls. Off by default.

(no)ntea

If set, use the full Microsoft Windows NT Extended Attributes to store UNIX-like inode information. Off by default.

Warning!

ntea only operates under Microsoft Windows NT.

(no)ntsec

If **ntsec** is set, use the Microsoft Windows NT security model to set UNIX-like permissions on files and processes. The file permissions can only be set on NTFS partitions. FAT and FAT32 don t support the Microsoft Windows NT file security (for more information, see http://sources.redhat.com/cygwin/). Defaults to be not set.

(no)smbntsec

If **smbntsec** is set, use **ntsec** on remote drives as well (this is the default). If you encounter problems with Microsoft Windows NT shares or Samba drives, setting this to **nosmbntsec** could help. In that case the permission and owner/group information is faked as on FAT partitions. A reason for a non working **ntsec** on remote drives could be insufficient permissions of the users. Since the needed user rights are somewhat dangerous, it's not always an option to grant those

rights to users. However, this shouldn't be a problem in Microsoft Windows NT domain environments. Default is **smbntsec**.

(no)reset_com

If **reset_com** is set, serial ports are reset to 9600-8-N-1 with no flow control when used. This is done at open time and when handles are inherited. Default is **reset_com**.

(no)strip_title

If **strip_title** is set, strips directory part off the window title, if any. Defaults to **strip_title**.

(no)title

If **title** is set, the title bar reflects the name of the program currently running. Under Microsoft Windows 95 and Microsoft Windows 98, the title bar is always enabled and it is stripped by default, but this is because of the way Microsoft Windows 95 and Microsoft Windows 98 work. In order not to strip, specify **title** or **title nostrip_title**. Defaults to **nostrip_title**.

Special Options with the CYGWIN Environment Variable

The **CYGWIN** environment variable uses the following options. Each option is separated by others with a space. To turn off any option, prefix it with **no** (such as the **nobinmode** for the **binmode** option).

binmode

Opens an unspecified file by default in binary mode, with no **CR /LF** translations, instead of text mode. **binmode** must be set before starting a Cygwin shell to have an effect on redirection.

envcache

Caches environment variable conversions (between Win32 and POSIX); this may cause problems if the mount table changes, as the cache is not invalidated and may contain values that depend on the previous mount table contents.

export

Re-exports the final values of these settings to the environment as **$CYGWIN** again.

title

Makes the title bar provide the currently running program's name.

strip_title

Strips the directory part off the window title.

glob

Expands command line arguments containing UNIX-style file wildcard characters (brackets, question mark, asterisk) into lists of files that match those wildcards. Applicable only to programs running from a Windows command line prompt.

tty

Enables extra support (such as **termios**) for UNIX-like **tty** calls. Off by default.

strace=*n*[:*cache*][,*filename*]

Configures system tracing. Off by default, setting various bits in *n* (a bit flag) enables various types of system messages. Setting *n* to **1** enables most messages. Other values can be found in the **sys/strace.h** directory. **:** *cache* lets you specify how many lines to cache before flushing the output; for example, type **strace=1:20** to cache 20 lines. *filename* lets you send messages to a file instead of to the shell's buffer.

ntea

Allows use of full NT *Extended Attributes* to store UNIX-like inode information.

Warning!

The **ntea** setting can create additional large files on non-NTFS partitions.

reset_com

If set, serial ports are reset to 9600-8-N-1 with no flow control when used. This is done at open time and when handles are inherited.

Notes

1. http://sources.redhat.com/cygwin/
2. ISO/IEC 9945-1:1996 (ANSI/IEEE Std 1003.1, 1996 Edition); POSIX Part 1: System Application Program Interface (API) [C Language].
3. http://sources.redhat.com/cygwin/

Chapter 8. GNUPro® Toolkit: Rebuilding the Tools

RedHat may provide updates to this release in the form of source code patches. Once these patches are applied, you will need to rebuild the binaries before the update can be used. The following documentation details the instructions for rebuilding this release using the GNUPro 07r1-1 native compiler.

The following is an outline of the process for rebuilding:

- Install the source code for this release and apply relevant patches. To install the source code, see the Section called *Configuring, Building, and Installing the Tools*. To apply the patches, follow the instructions provided with each patch that you receive.
- Install and set up the native toolchain environment supported for this release; see the Section called *Setting up the Native Environment*.
- Configure, build, and install binaries from the patched sources; see the Section called *Configuring, Building, and Installing the Tools*.
- Perform a check to ensure the binaries are installed properly; see the Section called *Ensuring Completion of Rebuild Process*.

The amount of disk space required for rebuilding varies depending on the filesystem used. Red Hat recommends at least 2GB of free disk space for the source code, the build directory, and the new installation directory.

Setting up the Native Environment

This release was built using the GNUPro 07r1-1 release. You should have already received a copy of GNUPro 07r1-1 for all of your supported hosts. If you do not have a copy of GNUPro 07r1-1 please contact your account representative to receive a copy.

Rebuilding Requirements

If you are installing GNUPro 07r1-1 from the CD, be sure to install the compiler, misc, and utils packages at a minimum. Cygwin users will also need the RedHat Cygwin NetRel package. By default your installation will be in `/opt/redhat/cygwin-1.6.6`. The remainder of these instructions will be based upon this path so adjust it as needed.

Once GNUPro is installed, you'll need to add the path to the compiler to your shell. GNUPro supports a variety of operating systems, so the path used depends on what operating system is used. Follow the general instructions in the Section called *Setting the **PATH** Environment Variable* in Chapter 2. The path to add would be:

For Solaris 7, the path would be:

```
/opt/redhat/gnupro-07r1-1/H-sparc-sun-solaris2.7/bin
```

For RedHat Enterprise Linux the path would be:

```
/opt/redhat/gnupro-07r1-1/H-i686-pc-linux-gnulibc2.3/bin
```

You are now ready to rebuild GNUPro 07r1-1.

> **Warning!**
>
> Do not use an environment from another Cygwin, UNIX or GNU/Linux release; doing so will cause problems rebuilding and subsequently using the tools.

In the following instructions, substitute the actual name for the release instead of *RELEASE_NAME*.

Configuring, Building, and Installing the Tools

Use the following steps for configuring, building and installing. Do not continue to the next step unless the previous steps are successful. If a step fails for any reason, save a copy of the exact error message as a file (cut and paste, screen dump, etc.) along with any relevant log files and submit them in a bug report.

Set up the build area

Keeping projects tidy will keep everything to a minimum of confusion. By making a tree of directories for your different builds it s easy to keep track of what you re working on.

1. Start a bash shell with the **bash** command from a shell window (or from a MS-DOS console if using Cygwin).
2. From the **bash-2.04$** prompt, create a project directory for yourself. As an example, make a directory called *project* in your home directory:

   mkdir ~/project
3. Then, make a build directory under that directory to hold object files:

   mkdir ~/project/build
4. Then change directories into ~/project:

   cd ~/project
5. Lastly, install the source code via the graphical installer or according to the README from an FTP shipment. GNUPro 07r1-1 releases have two sets of sources in directories named `src-cross` and `src-native` for cross tools and native tools respectively. (Host-x-host configurations use src-native).

   You are now ready to configure and build the toolchain.

Running configure

1. Change directories to `~/project/build`:

   cd ~/project/build
2. Execute **PREFIX/src-TYPE/configure** with all the necessary flags where *PREFIX* is the complete path to the sources and *TYPE* is either cross or native. Necessary flags include a minimum:

   ```
   --host=host-triplet
   --target=target-triplet
   --prefix=~/project/install
   --exec-prefix=~/project/install/H-host-triplet
   ```

   The switches should all be on one line. It is possible to put a backlash character (\) at the end of a line to indicate that the *logical* line actually continues onto the next *physical* line.

When configuring a native compiler, the target option may be omitted. In this case the target will default to being the same as the host.

When configuring Linux and Solaris native compilers, one may optionally enable the sys-root option. A sys-rooted compiler allows a developer to use one version of Linux or Solaris but create executables for a different version. This option can be enabled by adding `--with-sysroot` to the command line arguments passed to configure.

The target and host triplets vary depending on the release; please see below for a list of supported triplets. The example prefix is where the new toolchain will be installed; the exec prefix is where architecture-dependent files will be installed. You may want to redirect *stdout* and *stderr* to a file. A complete configure line for the sparc-sun-solaris2.7 hosted MIPS ELF toolchain might appear thus:

```
~/project/src/configure \ --host=sparc-sun-solaris2.7
\ --target=mips-elf \ --prefix=~/project/install \
--exec-prefix=~/project/install/H-sparc-sun-solaris2.7 \ >&
../configure.log
```

This will configure the system to install in `~/project/install` with the executables sitting in `~/project/install/H-sparc-sun-solaris2.7/bin`.

The example Example 8-1 shows what you will see after completing the input in the build directory.

/*build_dir*/redhat/*RELEASE_NAME*/ is an example for a directory from which to execute the **configure** command.

Warning!

Never run the **configure** command in the source directory! Always create a separate build directory and run the configure command there.

Warning!

Never rerun the **configure** command in the build directory if you want to create a different toolchain. Always create a new, separate build directory and use that.

Example 8-1. Running the configure command from the build directory

```
/opt/redhat/RELEASE_NAME/src-TYPE/configure -v
\ --prefix=/build_dir/redhat/RELEASE_NAME \
--exec-prefix=/build_dir/myredhat/RELEASE_NAME/H-HOST \ --host=HOST
\ --target=TARGET \ > configure.log 2>&1 &
```

You can watch the output of the **configure** command as it executes using:

tail -f configure.log

Use the *Ctrl-C* key sequence to exit the **tail** process.

The following tasks are the fundamentals of the process of ensuring that your configuration works:

- **configure** generates a Makefile in each relevant source directory, for use by the GNU **make** tool to use when you build applications.
- **configure** establishes and populates build directories for your application's compiled code to reside.

- **configure** creates an initialization file in the build directory in order to exchange information with the GNU debugger when you debug your applications.
- **configure** provides a shell script, **config.status**, for reconfiguration use.

Table 8-1. Supported host-triplet target-triplet combinations

Host Triplet	Target Triplet
i686-pc-cygwin	i686-pc-cygwin
i686-pc-cygwin	arm-elf
i686-pc-cygwin	armv5-elf
i686-pc-cygwin	armv6-elf
i686-pc-cygwin	arm-eabi
i686-pc-cygwin	i386-elf
i686-pc-cygwin	sparc-elf
i686-pc-cygwin	h8300-elf
i686-pc-cygwin	iq2000-elf
i686-pc-cygwin	m68k-elf
i686-pc-cygwin	mips-elf
i686-pc-cygwin	mips64-elf
i686-pc-cygwin	mipsisa64-elf
i686-pc-cygwin	sb1-elf
i686-pc-cygwin	mn10300-elf
i686-pc-cygwin	v850-elf
i686-pc-cygwin	powerpc-eabi
i686-pc-cygwin	powerpc-eabispe
i686-pc-cygwin	xstormy16-elf
i686-pc-linux-gnulibc2.3	arm-elf
i686-pc-linux-gnulibc2.3	armv5-elf
i686-pc-linux-gnulibc2.3	armv6-elf
i686-pc-linux-gnulibc2.3	arm-eabi
i686-pc-linux-gnulibc2.3	sparc-elf
i686-pc-linux-gnulibc2.3	h8300-elf
i686-pc-linux-gnulibc2.3	iq2000-elf
i686-pc-linux-gnulibc2.3	m68k-elf
i686-pc-linux-gnulibc2.3	i386-elf
i686-pc-linux-gnulibc2.3	armv5l-linux-gnu
i686-pc-linux-gnulibc2.3	armv5b-linux-gnu
i686-pc-linux-gnulibc2.3	armv4l-linux-gnu
i686-pc-linux-gnulibc2.3	armv5-softvfp-linux-gnu
i686-pc-linux-gnulibc2.2	i686-pc-linux-gnulibc2.2
i686-pc-linux-gnulibc2.3	i686-pc-linux-gnulibc2.3
i686-pc-linux-gnulibc2.3	mips-elf

Host Triplet	Target Triplet
i686-pc-linux-gnulibc2.3	am33_2.0-linux-gnu
i686-pc-linux-gnulibc2.3	mips-linux-gnu
i686-pc-linux-gnulibc2.3	mipsel-linux-gnu
i686-pc-linux-gnulibc2.3	mipsisa64-linux-gnu
i686-pc-linux-gnulibc2.3	mipsisa64el-linux-gnu
i686-pc-linux-gnulibc2.3	mips64-linux-gnu
i686-pc-linux-gnulibc2.3	mips64el-linux-gnu
i686-pc-linux-gnulibc2.3	mips64-elf
i686-pc-linux-gnulibc2.3	mipsisa64-elf
i686-pc-linux-gnulibc2.3	sb1-elf
i686-pc-linux-gnulibc2.3	mn10300-elf
i686-pc-linux-gnulibc2.3	v850-elf
i686-pc-linux-gnulibc2.3	powerpc-eabi
i686-pc-linux-gnulibc2.3	powerpc-eabispe
i686-pc-linux-gnulibc2.3	xstormy16-elf
sparc-sun-solaris2.7	arm-elf
sparc-sun-solaris2.7	armv5-elf
sparc-sun-solaris2.7	armv6-elf
sparc-sun-solaris2.7	arm-eabi
sparc-sun-solaris2.7	i386-elf
sparc-sun-solaris2.7	h8300-elf
sparc-sun-solaris2.7	iq2000-elf
sparc-sun-solaris2.7	m68k-elf
sparc-sun-solaris2.7	sparc-elf
sparc-sun-solaris2.7	mips-elf
sparc-sun-solaris2.7	mips64-elf
sparc-sun-solaris2.7	mipsisa64-elf
sparc-sun-solaris2.7	sb1-elf
sparc-sun-solaris2.7	mn10300-elf
sparc-sun-solaris2.7	v850-elf
sparc-sun-solaris2.7	powerpc-eabi
sparc-sun-solaris2.7	powerpc-eabispe
sparc-sun-solaris2.7	sparc-sun-solaris2.6
sparc-sun-solaris2.7	sparc-sun-solaris2.7
sparc-sun-solaris2.8	sparc-sun-solaris2.8
sparc-sun-solaris2.9	sparc-sun-solaris2.9
sparc-sun-solaris2.10	sparc-sun-solaris2.10

Sys-roots and Rebuilding

A sys-root is a directory containing portions of a Unix system's libraries and

headers, maintaining the original directory layout. Where Unix systems have a `/lib`, `/usr/lib`, and `/usr/include`, a system root has a file for file identical `sys-root/lib`, `sys-root/usr/lib`, and `sys-root/usr/include` directory.

A typical system root contains all the headers and libraries available in `/lib`, `/usr/lib`, `/usr/include`, `/usr/X11R6/lib`, and `/usr/X11R6/include`. Copying all this can take up a large amount of disk space, so copying just the files one needs is often a good idea. For example, if no X11 software is going to be built, the header and libraries from /usr/X11R6 can completely excluded.

When creating a toolchain with the `--with-sysroot` option, the sys-root must be created before building the tools themselves. The sys-root is located in a directory named `sys-root`. This directory is located under the named target triplet directory which resides directly below the exec-prefix path.

As an example, the GNUPro 07r1-1 i686-pc-linux-gnulibc2.3 exec prefix is `/opt/redhat/gnupro-07r1-1/H-i686-pc-linux-gnulibc2.3`. The target triplet is i686-pc-linux-gnulibc2.3. Thus, put together, its sys-root is in /opt/redhat/gnupro-07r1-1/H-i686-pc-linux-gnulibc2.3/i686-pc-linux-gnulibc2.3/sys-root.

One can either create the sys-root directory in this deep path or use a symlink to point to the sys-root you'd like to use. Using a symlink has the advantage that it's easy to build multiple versions of a toolchain, but use the same sys-root for all of them.

Running make

Use the **make** tool to build the binaries and **info** files.

> **Warning!**
>
> If building a native compiler using the `--with-sysroot` option, be sure to populate the sys-root first! See the Section called *Sys-roots and Rebuilding* for more information.

make -w all info > make.log 2>&1 &

Installing the Built Toolchain

Use the **make** tool to install the binaries and **info** files.

make -w install install-info > install.log 2>&1 &

Ensuring Completion of Rebuild Process

Test that the newly rebuilt tools work with the following instructions (which by no means show a comprehensive test).

1. Start a bash shell with the **bash** command from a shell window (or from an MS-DOS console for Cygwin); typing **bash** after the prompt, you get a **bash-2.04$** prompt.
2. Add the new installation binaries to the **PATH** environment variable information as the following example shows. Replace *host* (where *host* signifies the toolchain's triplet name) with the host triplet for your release.

   export PATH=/build_dir/redhat/*RELEASE_NAME*/H-*host*/bin:$PATH
3. Create a **"Hello World"** program with the following input:

   ```
   cat > hello.c int main (void) { return printf ("Hello
   World!\n"); }
   ```

Use the *Ctrl-D* key sequence to exit the process.

4. Compile the **"Hello World"** program.

   ***TOOLPREFIX*-gcc -Wall hello.c -o hello**

5. Execute the **"Hello World"** program.

   ***TOOLPREFIX*-run hello**

   At the **bash-2.04$** prompt you should expect to see the following output.

   ```
   Hello World!
   ```

Rebuilding is complete.

Patching

After submitting a support request, your solution may come in the form of a *patch* (a solution that *mends* the problem). To apply a patch to your source code, you need to navigate to the source directory (use the **cd** command to change to the `/src` directory). `/opt/redhat/`*`instdir`*`/src` is the full default pathname for the source directory. Save the patch as a `/tmp/patch` file, and run the **patch** program like the following example's declaration shows.

patch -p < /tmp/patch

You do not need to edit out all the non-patch text from the **/tmp/patch** file; the **patch** program will recognize where the real patch begins.

If the patch is rejected, there will be a filename ending in **.rej** in the source directory; for instance, if the patch was for a file in the **src/gcc/reload.c** path, and the patch was rejected, **src/gcc/reload.c.rej** is the rejection found there. Although it will take a while to run, you can search all files for a rejected patch with a command like the following example.

find . -name *.rej -print

> **Note:** Do not cut-and-paste the patch with a windowing system like X-Windows; tab characters are important and they are usually not preserved correctly when using cut-and-paste methods.

See also *Using the GNU Compiler Collection (GCC)* in the gcc.info files.

Troubleshooting the Rebuilding Process

The following documentation discusses warnings or error messages that may display during your build process. Each message has a troubleshooting approach accompanying it for resolution of the problem that you re addressing. If the problem's cause is still unclear, send in a problem report; See Chapter 3. The following warnings or error messages may display.

Make: Fatal error: Don t know how to make target foo.c

The most likely problem is that you are not using GNU **make**. Use the `--version` option for telling which version you are running; if you have this error message, run the **make --version** command. If you are not using GNU **make**, the program will not recognize the `--version` option.

Incorrect compiler used

When **configure** runs, it looks for an appropriate compiler, first GCC, then another C compiler, **cc**. If neither of these is correct, specify the name of the compiler at configuration time. We recommend that you always build with the GNU compiler, using the **gcc** command. Specify the compiler by setting the **$CC** environment variable before running **configure**.

- With a C shell, use a command similar to the following example's input (where */INSTALLDIR*/bin/ is the path to the compiler, *NOT-YOUR-USUAL-CC*).

  ```
  setenv CC /INSTALLDIR/bin/NOT-YOUR-USUAL-CC configure ... make
  ```

- With a Bourne shell (**/bin/sh**) or a Korn shell, use the following example's input (where */INSTALLDIR*/bin/ is the path to the compiler, *NOT-YOUR-USUAL-CC*).

  ```
  CC=/INSTALLDIR/bin/NOT-YOUR-USUAL-CC export CC configure ...
  make
  ```

If you still experience configuration problems, first try to rerun **configure** by adding the `--verbose` command line option. It's best to redirect the output to a log file while running this process.

- With a C shell, use a command similar to the following example's input.

  configure --verbose ... >& configure.out

- With a Bourne shell, use the following example's input.

  configure --verbose ... >configure.out 2>&1

Some seemingly unrelated problems arise after applying patches or making other changes. For instance, sometimes file dependencies get confused. With any trouble you have building, an easy step to take is to remove your build directory completely and then rebuild in an empty build directory, using input like the following example.

```
rm –rf /opt/redhat/INSTDIR/build mkdir /opt/redhat/INSTDIR/build
```

Then install the GNUPro Toolkit. See Chapter 2.

If it's not obvious which part of the toolkit is failing, check the last line in the log that begins `Configuring ...` (such as with the debugging tool, GDB, where you d see the `Configuring gdb...` status message) before the processing for the **configure** command stops.

configure also creates a `config.log` file in each sub-directory in which it runs tests. Check the end of the `config.log` that failed for specific information about what went wrong. The last 25-30 lines of this file should be adequate to diagnose the failure.

configure Problem Reporting

If the problem's cause is still unclear, submit a support request; see Chapter 3 If your configuration problems are still unsolved, send in the **configure** file's top-level `config.status` file. This file shows which command line options were used to configure the toolkit.

`/opt/redhat/`*instdir*`/build/config.status` should be the path to `config.status` (assuming the example pathnames here). The `--verbose` option for **configure** produces the entire output from the last directory. For instance, if **configure** fails in the GNU assembler directory, when creating a problem's case, provide everything after the line which reads `Configuring gas...` and, if in doubt, with the case report, provide a copy of all the output that generated from the **configure --verbose** input, so that technical support staff can more easily determine the problem and quickly resolve it.

build Problem Reporting

If your build problems are still confusing, report the **build** problem; Chapter 3. Use **build** for your category when creating a new case when submitting a support request. First, verify that you re using GNU **make**, as outlined in the the Section called *Troubleshooting the Rebuilding Process* section.

The following information is vital when you report the problem.

- Provide the top-level `config.status` file, which will indicate the command line options that you used to configure. Find the file in the `/opt/redhat/`*`instdir`*`/build/config.status` directory.
- Provide the output of the failing **make** command. Only the output from the last directory is probably useful, as with a **configure** case report. If in doubt, send the entire output from **make** and technical support staff will determine the problem and resolve it.

Appendix A. GNUPro® Toolkit: Alerts, Warnings, and Improvements to the Tools

The following documentation discusses issues that serve as important red flag alerts, warnings, or new enhancements with this distribution.

These issues may provide clues that may help if you are a new user of the GNUPro Toolkit or if you run into problems with the GNUPro Toolkit.

See Appendix B for issues that may still be pertinent from earlier versions of the GNUPro Toolkit.

For help with terms that are unfamiliar, see *GNUPro® Toolkit: Glossary*.

General Issues

There are no known issues with this release.

Appendix B. GNUPro® Toolkit: Enhancements and Features from Previous Releases

This appendix describes enhancements and features to previous releases of the GNUPro Toolkit As well as providing a historical background it also shows where development work was previously concentrated and indicates the kind of improvements that might be expected in future releases.

Compiler Features

The following list concerns enhancements and features that were added to earlier releases of the GNUPro compiler.

- A new scheme for accurately describing processor pipelines, the DFA scheduler, was added.
- A new superblock formation pass was added. Enabled using -ftracer, the pass simplifies the control flow of functions allowing other optimizations to do better job.
- Type based alias analysis was implemented for C++ aggregate types.
- Improved hard register allocation pass.
- Improved code flow analysis both before and after hard register allocation.
- An new optimisation pass to simplify comparison operations was added.
- Improved support for C++ exception handling.
- Improved support for the PowerPC 750 architecture.

Debugger Features

The following list discusses features and enhancements to previous releases of GDB, the GNUPro debugger..

- C++ overload resolution works properly in almost all cases.
- On SVR4 native platforms (such as Solaris), when attaching to a process without first loading a symbol file, GDB will attempt to locate and load symbols from the running process's executable file.
- Better support for debugging floating-point programs on all x86 targets. In particular, "info float" displays the FP registers in the same format on all x86 targets, with a greater level of detail.
- Better support for hardware watch-points. Array elements, struct members, and bitfields can all be watched.
- A new machine oriented interface was added to GDB. This interface was designed for debug environments which run GDB as a separate process.

Utilities

The following list concerns enhancements to the GNUPro utilities.

- GCC has been revised to the latest 4.0.x stable revision.
- GDB has been revised to the latest 6.2.x stable revision.
- Binutils has been revised to the latest 2.16.x stable revision.

Appendix C. GNUPro® Toolkit: General Licenses and Terms for Using GNUPro Toolkit

You can use, copy, modify, and redistribute GNUPro Toolkit according to the agreements in the *GNU General Public License*. Some terms and conditions vary for specific components; see the following documentation for the licenses of the different source components.

- the Section called *GNU General Public License*
- the Section called *GNU Library General Public License*
- the Section called *GNU Free Documentation License*
- the Section called *Tcl/Tk Tool Command Language and Windowing Toolkit License*

Licensing Terms

This documentation contains legal licensing terms and conditions applying to the main components of GNUPro Toolkit. It is not exhaustive. It provides the terms applying to the main components as well as the notices applying to the lesser components which we must include.

For complete copyright status and licensing terms of each file, see the `'Copying'` and `'Copying.lib'` files in the top level of the binary distribution, and see the `'src/Copying'`, `'src/Copying.lib'`, and `'src/COPYING.NEWLIB'` files in the source distribution.

For the Tcl/Tk tools, see the `license.terms` file in the `src/tcl` and the `src/tk` directories of both the binary and source distributions.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Boston, MA 02111-1307, USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that

they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and conditions for copying, distribution and modification

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in

the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

    a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

    b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

    c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

    The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

    If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgement or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that ver-

sion or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

    No Warranty

12. *Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.*

13. *In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.*

End of terms and conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and an idea of what it does.>
Copyright (C) 20xx <name of author> This program is free software;
you can redistribute it and/or modify it under the terms of the GNU
General Public License as published by the Free Software Foundation;
either version 2 of the License, or (at your option) any later
version. This program is distributed in the hope that it will be
useful, but WITHOUT ANY WARRANTY; without even the implied warranty
of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details. You should have received a
copy of the GNU General Public License along with this program; if
not, write to the Free Software Foundation, Inc., 59 Temple Place -
Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 20xx <name of author>
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type
```

```
'show w'. This is free software, and you are welcome to redistribute
it under certain conditions; type 'show c' for details.
```

The hypothetical commands **'show w'** and **'show c'** should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than **'show w'** and **'show c'**; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
program 'Gnomovision' (which makes passes at compilers) written
by James Hacker. <signature of Ty Coon>, 1 April 2003 Ty Coon,
President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

GNU Library General Public License

Version 2, June 1991

```
Copyright (C) 1991 Free Software Foundation, Inc. 59 Temple Place
- Suite 330, Boston, MA 02111-1307, USA Everyone is permitted to
copy and distribute verbatim copies of this license document, but
changing it is not allowed. [This is the first released version of
the library GPL. It is numbered 2 because it goes with version 2 of
the ordinary GPL.]
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

Terms and conditions for copying, distribution and modification

1. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

2. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. The modified work must itself be a software library.

   b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

   c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

   d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

      (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control

the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

   Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

   This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

5. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

   If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

6. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

   However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

   When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

   If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

   Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

7. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided

that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

c. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

d. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

8. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

9. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who

have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

11. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

12. If, as a consequence of a court judgement or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

    If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

    It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practises. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

    This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

13. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

14. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

15. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.
16. *No warranty*
17. *Because the library is licensed free of charge, there is no warranty for the library, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the library "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the library is with you. Should the library prove defective, you assume the cost of all necessary servicing, repair or correction.*
18. *In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the library as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the library (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the library to operate with any other software), even if such holder or other party has been advised of the possibility of such damages.*

End of terms and conditions

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and an idea of what it does.>
Copyright (C) <year> <name of author> This library is free software;
you can redistribute it and/or modify it under the terms of the GNU
Library General Public License as published by the Free Software
Foundation; either version 2 of the License, or (at your option)
any later version. This library is distributed in the hope that it
will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
the GNU Library General Public License for more details. You should
have received a copy of the GNU Library General Public License along
with this library; if not, write to the Free Software Foundation,
Inc., 59 Temple Place - Suite 330, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library 'Frob' (a library for tweaking knobs) written by James
Random Hacker. <signature of Ty Coon>, 1 April 2003 Ty Coon,
President of Vice
```

That's all there is to it!

GNU Free Documentation License

Version 1.1, March 2000

1. *Preamble*

   The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

2. *Applicability and definitions*

   This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you."

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

   The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

   A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for

drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque."

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

3. *Verbatim copying*

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

4. *Copying in quantity*

   If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

5. *Modifications*

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   - Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
   - List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
   - State on the Title page the name of the publisher of the Modified Version, as the publisher.
   - Preserve all the copyright notices of the Document.
   - Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
   - Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
   - Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
   - Include an unaltered copy of this License.
   - Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
   - Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
   - In any section entitled "Acknowledgments" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments and/or dedications given therein.
   - Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
   - Delete any section entitled "Endorsements." Such a section may not be included in the Modified Version.
   - Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

6. *Combining documents*

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgments", and any sections entitled "Dedications." You must delete all sections entitled "Endorsements."

7. *Collections of documents*

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

8. *Aggregation with independent works*

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other

self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

9. *Translation*

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

10. *Termination*

    You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

11. *Future revisions of this license*

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Addendum: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C) <year> <your name>. Permission is granted to copy,
distribute and/or modify this document under the terms of the
GNU Free Documentation License, Version 1.1 or any later version
published by the Free Software Foundation; with the Invariant
Sections being <list their titles>, with the Front-Cover Texts being
<list>, and with the Back-Cover Texts being <list>. A copy of the
license is included in the section entitled "GNU Free Documentation
License."
```

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being *`list`*"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Tcl/Tk Tool Command Language and Windowing Toolkit License

Certain portions of this product have the following copyrights:

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., and other parties.

The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

In no event shall the authors or distributors be liable to any party for direct, indirect, special, incidental, or consequential damages arising out of the use of this software, its documentation, or any derivatives thereof, even if the authors have been advised of the possibility of such damage.

The authors and distributors specifically disclaim any warranties, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, and noninfringement. This software is provided on an as is basis, and the authors and distributors have no obligation to provide maintenance, support, updates, enhancements, or modifications.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only Restricted Rights in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as Commercial Computer Software and the Government shall have only Restricted Rights as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

RESTRICTED RIGHTS: Use, duplication or disclosure by the government is subject to the restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software Clause as DFARS 252.227-7013 and FAR 52.227-19.

Notes

1. http://www.gnu.org/copyleft/

GNUPro® Toolkit: Glossary

For more etymology and history of the technical jargon in open source and embedded communities, see the following HTML documentation:

- A Web of Online Dictionaries: http://www.ora.com/reference/dictionary/

  This is the location for terms for all systems and nearly every technology.
- Webopedia: Online Computer Dictionary: http://pcwebopedia.com/

  This has become a very popular online dictionary.
- Technology Buzzwords for Students: http://www.tekmom.com/buzzwords/

  This is a site for all ages with some basics.
- Web Dictionary of Cybernetics and Systems: http://pespmc1.vub.ac.be/ASC/INDEXASC.html

  This provides one of the most popular online dictionaries for UNIX systems.
- CNET glossary: http://coverage.cnet.com/Resources/Info/Glossary/

  This is the main online dictionary for Microsoft Windows systems.

Here are the terms

A

a.out

The original UNIX object file format. Supports only three sections; see *sections*.

ABI

*A*pplication *B*inary *I*nterface, which an operating system uses to define as a standard for how programs should interface with that operating system, including specifications such as executable format, calling conventions, and chip-specific requirements.

accumulator

A register being used for arithmetic or logic (as opposed to addressing or a loop index), especially one being used to accumulate a sum or count of many items.

address

A number identifying a location in the computer's memory that informs the computer where to find information such as a file name or data for processing.

See Also: bus.

aliases

Links or symbolic links to files or devices on a network.

API

*A*pplication *P*rogramming *I*nterface, which defines how programmers write source code that makes use of a library's or an operating system's facilities by accessing the behavior and state of classes and objects.

See Also: objects.

architecture

A term for a family of processors, generally used in reference to features common to all members.

ARC

Argonaut RISC Chip, a simple RISC processor designed into custom chips.

ARM

Annotated Reference Manual for C++, often used to describe a *name-mangling*.

Also, *A*dvanced *R*ISC *M*achine's family of RISC processors

See Also: name-mangling.

array

A collection of data items, all of the same type, in which integers designate each item's position.

ASCII

*A*merican *S*tandard *C*ode for *I*nformation *I*nterchange, the predominant character set encoding of present-day computers. The modern version uses seven bits for each character, whereas most earlier codes (including an early version of ASCII) had fewer assignments.

ash

An *a*ssembly *sh*ell.

See Also: shell.

ASIC

*A*pplication-*S*pecific *I*ntegrated *C*ircuit.

ASM

Assembler programming.

assembler

A tool that produces object files from assembly code like the GNU assembler, GAS.

B

backend

1) Software performing either the final stage in a process, or a task not apparent to the user, such as with a compiler with its backend generating machine language and performing optimizations specific to a machine's architecture.

2) A set of functions which appear in a particular target vector.

3) The implementation within BFD of a specific object file format.

backtrace

A summary of how a program got where it is with a debugging process.

bar

Used very generally as a sample name for absolutely anything, especially programs and files.

See Also: foo.

bash

A *ba* sic command *sh* ell (*B* ourne *A* gain *Sh* ell), typically for UNIX or GNU/Linux operating systems, based on the Bourne shell.

See Also: shell.

BDM

B ackground *D* ebugging *M* ode, referring to Motorola chips controlled through a special set of pins.

BFD

*B*inary *F*ile *D*escriptor, the library used by GNU tools to read and write object files.

bi-endian

Refers to a processor or toolchain that supports both big-endian and little-endian code.

See Also: big-endian, little-endian.

big-endian

A byte-ordering scheme in which the most significant bytes are at lower addresses (big-end-first), such as the Motorola 68000 family of microprocessors and most of the various RISC designs.

bignum

A multiple-precision computer representation for very large integers. Most computer languages provide a kind of *integer-based* data, but such computer integers are usually very limited in size; to work with larger numbers, use floating-point

numbers, which are usually accurate to only six or seven decimal places. Computer languages that provide bignums can perform exact calculations on very large numbers, such as 1000! (the factorial of 1000, which is 1000 x 999 x 998 x ... x 2 x 1).

binary

Base-two number system, of which the only digits are 0 and 1. Used as a signal for processing as either off (0) or on (1); from right to left, the digits have a binary value of 1, 2, 4, 8, 16, and so on in multiples of 2, exponentially (the binary number of 101, for instance, is equivalent to the decimal number, 5, and the binary number of 1011 is equivalent to the decimal number, 13).

See Also: unary, ternary, floating-point notation.

binary operator

An operator that has two arguments.

BIOS

*B*asic *I*nput *O*utput *S*ystem, which loads and executes operation functionality usually stored on the computer's hard disk (in ROM mode); also may be accessible from CD-ROM or floppy disk at install time.

Bison

The GNU parser generator, a workalike for yacc.

See Also: yacc.

bit

A computational quantity that can take on one of two values, such as true and false or 0 and 1. A bit is said to be set if its value is true or 1, and reset or clear if its value is false or 0. One speaks of setting and clearing bits. To toggle or invert a bit is to change it, either from 0 to 1 or from 1 to 0.

bogoMIPS

Reference to the measure of the time elapsed for a GNU/Linux system to perform processes.

See Also: MIPS.

boot

See: bootstrap

bootstrap

The action for a machine to run through its opening processes, for instance, for a compiler's initialization. Known by having to put on boots by pulling on the sidestraps before going out in the world.

See Also: network boot.

BOOTP

*Boot p*rotocol, which lets a network user be automatically configured (for instance, to receive an IP address) and have an operating system booted or initiated without user involvement. A BOOTP server, managed by a network administrator, automatically assigns the IP address from a pool of addresses for a certain duration of time.

BOOTP is the basis for a more advanced network manager protocol, DHCP.

See Also: DHCP.

breakpoint

What makes a program stop whenever a certain point is reached in a debugging process.

See Also: catchpoint, watchpoint, tracepoint.

BSD

*B*erkeley *S*oftware *D*istribution, a family of UNIX software tools from U.C. Berkeley, originally licensed from AT&T, and later upgraded to all-free code. Formed the basis for SunOS, incorporating paged virtual memory, TCP/IP networking enhancements, among other features.

See Also: FreeBSD, NetBSD.

BSP

*B*oard *S*upport *P*ackage, typically referring to the low-level code or scripts that build programs running on a particular chip on a particular circuit board. Provides memory sizing, testing, interrupt/DMA control, and other features. Also refers to the ROM that boots an RTOS onto a specific board. Exact meaning varies.

buffer overflow

What happens when you try to stuff more data into a buffer than the buffer has been constructed to hold.

buffer

A holding area in a program's memory (like an Emacs buffer holding text waiting to be edited.

build

The process of configuring, compiling, and linking a set of tools, source files, libraries, executable files to produce binary resources. As a noun, denoting the results of the process, the build process has four steps: editing code, compiling the code into object files, linking the object files to make an executable, and debugging the executable; the build process repeats until the executable is a working program.

See Also: Makefile, target.

bus

1) Collection of wires through which data transmits; buses consist of two parts: an address bus and a data bus, with the data bus transferring actual data and the address bus transferring information about where the data should go.

2) In networking, a bus is a central cable that connects all devices on a local-area network.

See Also: LAN.

Byacc

*B*erkeley *yacc*, a version in BSD UNIX.

See Also: yacc, Bison.

byte

A sequence of eight bits.

See Also: bit, bytecode.

bytecode

Machine-independent code generated by a compiler and executed by an interpreter.

C

cache

See: cache file system

cache file system

Recently accessed data in storage on disk on a computer or network, using a process for filing the boot process, as well as archives, kernel information and other data.

See Also: disk caching.

Canadian cross

A compilation of a program with a compiler for some other host and target pair; for example, to build a Motorola 68000 cross compiler that runs on a 486 PC using a Sun SPARC station.

canonical

The standard state or manner or usage in technical terminology, derived from computation theory and mathematical logic, when, for instance, two formulas such as 1 + x and x + 1 are said to be equivalent because they mean the same thing, but the second one is in canonical form because it is written in the usual way, with the highest power of x first. Also refers to the naming of host operating system combinations with target boards, such as the i686-elix-linux-gnu canonical name.

cat

UNIX command, from concatenate, meaning to lay pieces of data end to end.

catchpoint

A special breakpoint when debugging that stops your program when a certain kind of event occurs, such as the throwing of a C++ exception or the loading of a library.

CGEN

A framework for generating CPU-related tools such as assemblers, disassemblers and simulators.

CHILL

A high-level language popular in Europe for telecommunications programming.

CISC

C omplex *I* nstruction *S* et *C* omputer, a class of machines typically having variable-length instructions with a variety of addressing modes.

classes

In object-oriented programming, objects that define instance, variables, and methods, whose combined action is to specify the interfaces that a class implements and the immediate superclass or hierarchy of the class and all its properties. See also *objects*.

class file

The Java binary file format for the Java Virtual Machine (JVM) for the `.class` file that, after compiling, contains bytecode and symbols (like an object file).

See Also: bytecode, JVM.

COFF

*C*ommon *O*bject *F*ile *F*ormat, a format formerly common for UNIX, and still used by some embedded systems.

COFF debugging

The debug format that is defined as part of the COFF specification.

compiler

A tool that translates high-level source code in a language such as C or Pascal into machine-executable programs. The term may also refer specifically to the tool that translates from source to machine code.

configure

A shell script command that sets up an environment in which programs will compile correctly for a machine and operating system, installing the programs in proper places.

constants

In object-oriented programming, an object that retains a consistent value throughout the exceution of a program.

See Also: variables.

copyleft

The copyright notice for GNU Emacs and other GNU software, the GNU General Public License, which grants reuse and reproduction rights to all users.

CORBA

*C*ommon *O*bject *R*equest *B*roker *A*rchitecture, from Xerox PARC, a system to define and document interfaces between the modules of non-distributed programs.

critical section

A segment of code in which a thread uses resources (such as instance variables) that can be used by other threads, but that must not be used by them simultaneously.

CRLF

A *c*arriage *r*eturn (CR; for the ASCII, 0001101...for the octal, \015... for the hex, 0x0f) followed by a *l*ine *f*eed (LF; for the ASCII, 0001010... for the octal, \012... for the hex format, 0x0c).

cross-configuration

A different target machine than the development tools themselves, which run on the host--for example, when working on a software application with a SPARC station that generates and debugs code for a Motorola Power PC-based board.

cruft

Writing assembler code for an application or project when it would be more expediently performed by a compiler.

CSE

Common Sub-expression Elimination, for compiler optimization.

See Also: GCSE.

csh

*C sh*ell, a command shell for users to type commands, interacting with the operating system. Uses a C-like command syntax, typically for UNIX or Linux developers.

CVS

*C*oncurrent *V*ersion *S*ystem, a free source version control system.

Cygnus

A company known for pioneering GNU development; now a Red Hat company.

Cygwin

A UNIX emulation library for Windows operating systems. Cygwin is a UNIX or GNU/Linux shell environment and portability layer enabling delivery of open source projects to Windows. Cygwin provides corporate IT and software developers a solution for integrating a heterogeneous environment of Windows and UNIX-based systems. In addition, developers can use Cygwin to quickly migrate applications from UNIX or GNU/Linux to Windows. For more information detailing Cygwin, see http://sources.redhat.com/cygwin/ for documentation.

D

daemon

A continuously running server process, waiting for some condition(s) to occur. The perpetrator of the condition need not be aware that a daemon is lurking.

See Also: HTTPD.

data type

In programming, classification in a file of a particular type of information for a computer to use. For specific types of files that computers can store for use,

See Also: filetype.

dbx

A standard debugger on many UNIX systems instead of the GNU debugger, GDB.

debug format

The layout of debugging information within an object file format.

debug protocol

A mechanism by which a debugger examines and controls a program being debugged.

debugger

A tool that allows programmers to examine and control a program, typically for the purpose of stopping the program while it runs and finding errors in the program.

declaration

Binding of an identifier to the information to which it relates in a program's source code; declaration happens in source code where actual binding happens at compile time or runtime.

DejaGNU

A regression testing framework for use on nearly any program, based on the embeddable scripting language, tcl input.

delta

A quantitative change, especially a small or incremental one, for a program.

DevKit

See: EDK

diff utilities

GNU file-comparison utilities (diff, diff3, sdiff, and cmp), which generate a listing of changes, especially giving differences between (and additions to) lines of source code or between different formats.

See Also: patch.

disk caching

Process by which a disk on a hard drive or LAN caches its recently stored data.

diskless

Means of operating systems to use high-speed network cards in order to have cheaper and faster access with networks than with local disk access or LAN.

DHCP

*D*ynamic *H*ost *C*onfiguration *P*rotocol, a protocol for getting an IP address, specifically, an Ethernet address. DHCP assigns IP addresses, delivering TCP/IP stack configuration parameters, while also providing other configuration information such as addresses for routers, printers, time and news servers. A demonstration program, dhcpd, comes with the Embedded DevKit software to show this protocol's usage.

See Also: BOOTP.

dynamic link

A program's link against a shared library that, when run, is locatable by the appropriate shared library, arranging to have inclusion in the program that is running.

dynamic object

Another name for an ELF shared library.

DWARF

A debugging format based on attribute records. Versions include DWARF 1, 1.1, 2, and extensions to 2.

E

EABI

Generic term for an ABI adapted for embedded use.

EBCDIC

*E*xtended *B*inary *C*oded *D*ecimal *I*nterchange *C*ode, a character set used by IBM before its open-systems policy, allegedly adapted from punched card code.

ECOFF

*E*xtended *COFF*, a format used with MIPS and Alpha processors, both for workstations and embedded uses.

eCos

*E*mbedded *C*onfigurable *O*perating *S*ystem, a complete, open-source run-time environment, allowing embedded system developers to focus on differentiating their products instead of developing, maintaining, or configuring proprietary, real-time kernels.

ed

The line-oriented text editor.

EDK

Embedded Development Kit

See Also: Embedded DevKit.

EEPROM

*E*lectrically *E*rasable *P*rogrammable *R*ead-*O*nly *M*emory, a special type of EPROM that can be erased when exposed to an electrical charge and reprogrammed. EEPROM retains its contents even when the system powers down.

EGCS

Historical name for the GNUPro compiler tools, EGCS is pronounced *eggs* with the C silent.

See Also: GCC.

ELF

*E*xtended *L*inker *F*ormat, for many UNIX and embedded systems.

EL/IX

An API, a layered subset of the POSIX API, that is scalable, configurable and meant for development of embedded and real-time applications, using GNU/Linux or other compliant embedded operating systems.

See Also: EDK.

Emacs

GNU programmable text editor (derived from *E*diting *MAC* ro*S*), including facilities to run compilation subprocesses and send and receive mail.

embedded development

Development for machines that do not run a desktop environment, such as a printer, an engine controller card, or a cellular phone. Development typically uses cross-compilation; for example, a SPARC host operating system generating and debugging code for a Motorola Power PC-based target processor board.

Embedded DevKit

Also referred to as *EDK* or *DevKit*, it is a set of open-source tools for embedded development (an infrastructure of compilers, debuggers, utilities and libraries); see http://www.redhat.com/products/edk/ for more information.

emulation

Ability of a program or device to imitate another program or device. For instance, the Cygwin tools allow devlopers to use UNIX tools while working with Windows systems.

See Also: Cygwin.

encryption

Use of algorithms to alter data, making it incomprehensible to unauthorized viewers.

end point

Device at which a virtual circuit or virtual path begins or ends.

enterprise network

Large and diverse network connecting an organization of systems.

enum

See: enumerated data type

enumerated data type

A data type restricted to sequencing of named values given in a particular order.

See Also: data type.

enumerated values

Values named in an enumerated data type.

environment

Usually, a host operating system. In the case of cross-compiler a host operating system and a target (usually, a processor board) working together with an executable application.

See Also: target environment.

environment variable

Assignments for an operating system server or client, dependent upon discretion of the user or developer and their requirements.

EPROM

*E*rasable *P*rogrammable *R*ead-*O*nly *M*emory, non-volatile memory chips that are programmed after they are manufactured, and, if necessary, made erasable and reprogrammed. A special type of ROM that retains its contents until it is exposed to ultraviolet light.

See Also: diskless, EEPROM, PROM.

Etherboot

Software for booting x86 PCs over a network, useful for booting PCs diskless, for maintaining software for a cluster of equally configured workstations centrally, for an X-terminal, for platforms where remote partitions are mounted by NFS and there are no problems with the slowness of data transfers that results from NFS (compared to a local disk, for various kinds of remote servers, such as a tape drive server that can be accessed with the RMT protocol), for routers, and for machines doing tasks in environments unfriendly to disks.

See Also: bootstrap, BOOTP, ethernet, network boot, RAM Disk.

ethernet

A LAN protocol for connecting to a network and the internet.

exception

An event during program execution that prevents the program from continuing normally; generally, an error.

exception handling

Event that occurs when a block of code reacts to a specific type of exception. If the exception is for an error from which the tool, the debugger, for instance, can recover, the debugger resumes its process.

executable file

A binary-format file bound by a linker, containing machine instructions, object files and libraries, in a ready-to-run application.

expansion

The process of running a compressed data set through an algorithm that restores the data set to its original size.

expressions

A specification for an address or numeric value. An *empty expression* has no value (being either whitespace or null). An *integer expression* is one or more arguments delimited by operators. Arguments can use symbols, numbers or sub-expressions. *Sub-expressions* have a parenthetical usage containing an integer expression, or they are a prefix operator followed by an argument. See *Using as, the GNU Assembler* in the gas.info files for more discussion of these subjects.

expect

A program that allows scripted control over another program.

F

FIFO

A *f*irst-*i*n-*f*irst-*o*ut interprocess communications method.

filetype

There are several types of files: batch or command files (contains operating system commands), binary files (contains data or instructions in binary format), data files (contains data), directory files (contains bookkeeping information about files in sub-directories), executable files (contains bookkeeping information about files), library files (contains functions in object format), map files (contains a map of a program's contents), object files (contains compiled code), and text files (human-readable text in ASCII format).

filesystem

An hierarchical directory structure where files may exist at any level of the directory hierarchy.

See Also: cache file system.

flash memory

A variation of EEPROM that can be erased and programmed as units of memory called blocks; EEPROM is erasable and is rewritten at the byte level, which is slower than flash memory updating. Flash memory is for digital cellular phones, digital cameras, LAN switches, embedded controllers, and other embedded devices.

flex

The GNU lexical analyzer generator.

floating-point

A number representing a mantissa (usually a value between 0 and 1) and an exponent according to a given base (usually a value of 2).

floating-point notation

Numeric system used to represent very large and very small real numbers. With two parts, a mantissa and an exponent; for example, 153,000,000 and 0.0009375 have the floating point notation, respectively, of 153E4 and 9375E-7.

See Also: FLOPS.

FLOPS

*Fl*oating-point *o*perations *p*er *s*econd, a measure of computer's speed of performing floating-point operations.

See Also: MIPS.

foo

Used very generally as a sample name for absolutely anything, especially programs and files. Etymology from Army slang acronym, FUBAR (bowdlerized to mean *F*ouled *U*p *B*eyond *A*ll *R*epair).

See Also: bar.

foobar

See: foo

foo.bar

See: foo

frame

When a program stops, you can use debugging commands to have access to and information about the location of a function call, arguments about the call, called local variables, and the block of data called the stack frame (using these commands is referred to as examining the stack).

See Also: stack, stack frame.

FreeBSD

A free UNIX operating system for PCs.

See Also: BSD.

friends

A non-member function or class allowed access to a member function or class.

ftp

See: FTP

FTP

*F*ile *T*ransfer *P*rotocol, based on TCP/IP which enables getting and storing files between hosts on the web. Use the ftp command to direct files to new location.

See Also: TCP/IP.

function

Segments of C or C++ programming languages that provide a means for a program to transfer and to generate data in a modular way. A function consists of parameters and a return value to define a compiler's requirements for acion with a program. A program can have many functions but only one main function to which all other functions address.

See Also: library, subroutine.

G

g++

The GNU C++ compiler tool.

garbage collection

The automatic detection and freeing of memory that is no longer in use. A runtime system performs garbage collection so that programmers never explicitly free objects.

gas

Acronym for the GNU assembler.

GCC

GNU Compiler Collection, consisting of gcc, g++ and other compiler development tools. For more information, see *Using the GNU Compiler Collection (GCC)* in the gcc.info files.

gcc

Acronym for the GNU C compiler.

gcj

Front end to GCC that is able to read JavaTM `.class` files, generating assembly code.

gcjh

A program to generate C++ header files corresponding to Java `.class` files.

GCSE

Global Common Sub-expression Elimination, for compiler optimization.

See Also: CSE.

gdb

Acronym for the GNU debugger.

gdbserver

A small special-purpose debugging application running on a target. gdb connects to the gdbserver using either a serial or an ethernet (TCP/IP) connection.

gdbtk

Obsolete name for the GUI debugger.

See Also: Insight.

glibc

GNU ANSI C library, a fully POSIX and ANSI compliant C library, more suited to native UNIX systems and heavily used for Linux.

See Also: newlib.

global variable

Any program variable whose values affect a whole system and are external to a function.

See Also: local variable, variables.

GNOME

*G*NU *N*etwork *O*bject *M*odel *E*nvironment, the configurable GNU/Linux interface.

GNU

Recursive acronym for *G*NU's *N*ot *U*NIX. A project to build a free operating system, started by Richard Stallman in 1985, with many useful spinoffs, such as the Emacs text editor, a C compiler (gcc or egcs), a debugger (gdb), and many other programming tools.

GO32

Freeware 32-bit DOS extender. Also the name for GNU tools ported to DOS using GO32. See http://www.delorie.com/djgpp/.

gprof

Program for determining which parts of a program are taking most of the execution time; see also http://sources.redhat.com/binutils/docs-2.10/gprof.html for more profiling information. Profiling allows you to learn where your program spent its time and which functions called which other functions while it was executing, so that you can know which pieces of your program are slower than you expected, which might be candidates for rewriting to make your program execute faster, and which functions are being called more or less often than you expected.

grep

UNIX functionality for rapidly scanning a file or directory of files, searching for a specific pattern or specific data.

H

hashing

A means computing systems use to sort.

heterogeneous

Characteristic of components containing dissimilar constituents or the ability of a component to contain or to be part of a network, which consists of different components that can interoperate. In updating or accessing the GNUPro tools, this term refers to configuring them on networks with strategical symbolic links; see also the Section called *Toolchains and Usage* in Chapter 5 about using `--exec` and `--exec-prefix=` options for a using the tools in a heterogeneous environment.

hex

See: hexadecimal

hexadecimal

The numbering system that uses 16 as its base. The code, 0-9, a-f, represents the digits, 0 through 15.

host

A computer on which tools, such as the compiler, run.

HTTPD

*H*yper *T*ext *T*ransfer *P*rotocol *d*aemon, a Web server program providing information to a client using the HTTP protocol (the standard for communicating information on the World Wide Web). See also http://www.apache.org/.

I

ICE

I n-*C* ircuit *E* mulator, a hardware device that gives an engineer control over the execution of a processor while it's connected to the rest of a system's circuitry. Emulators are powerful hardware debugging tools that can connect to debuggers.

IDE

*I*ntegrated *D*evelopment *E*nvironment, a GUI tool or a set of tools that uses GUI functionality.

IEEE

*I*nstitute of *E*lectrical and *E*lectronics *E*ngineers, a non-profit, international, technical professional association, authorities in technical areas who set the standards for areas ranging from computer engineering, biomedical technology and telecommunications to electric power, aerospace and consumer electronics, among others. See http://www.ieee.org/ and http://standards.ieee.org/ for more information.

implementation

Execution of source code for a process whose result is a program that builds and runs.

include files

Often designated as #include file, sometimes leading to excessive multi-leveled inclusion.

inetrd

Command to initialize a RAM Disk with a GNU/Linux operating system.

See Also: RAM Disk.

inheritance

In object-oriented programming, the ability of one class of objects to derive properties from another class.

init

A command responsible for starting *initial* processes on a GNU/Linux system.

Insight

The GNUPro visual debugger, formerly known as GDBTk. Default debugger for the *GNUPro Toolkit*. See Chapter 6, and for more detailed information and *Debugging with GDB* in the gdb.info files.

instance variable

An object (not a reference) is defined as an instance of exactly one class (in classical object oriented programming), called its most *derived class*. An object not directly contained in any other class is called the complete object.

interpreter

A *module* that alternately decodes and executes every statement in some body of code.

IP

*I*nternet *P*rotocol, the basic protocol of the Internet, enabling the delivery of individual packets from one computer to another across the web. A packet may not be instantly delivered, or if multiple packets are sent simultaneously, they may not simultaneously arrive in the order they were sent. Protocols built on top of this add the notions of connection and reliability.

See Also: TCP/IP.

IRQ

*I*nternet *Req*uest.

ISA

*I*nstruction *S*et *A*rchitecture bus, a means to carry signals for transferring data between the processor and any peripheral devices.

jcf-dump

Reads a .class Java-type file and prints out all sorts of useful information.

JIT

*J*ust-*i*n-*t*ime compiler that converts all of the bytecode into native machine code just as a Java program is run, resulting in run-time speed improvements over code interpreted by a Java Virtual Machine.

See Also: JVM.

JTAG

*J*oint *T*est *A*dvisory *G*roup, referring to a type of hardware interface that allows the testing of chips and boards within a complete system; programs running on processors with JTAG support may be controlled through the processor's JTAG port.

jvgenmain

A small program to generate an appropriate main for a Java class.

JVM

*J*ava *V*irtual *M*achine, part of the Java Runtime Environment responsible for interpreting Java bytecodes.

jv-scan

Reads a .java file and prints some useful information, such as for instance, which classes are defined in that file.

K

K&R

From Brian Kernighan and Dennis Ritchie's book, *The C Programming Language* (Prentice-Hall, 1978; ISBN 0-113-110163-3), generally signifying the C standards.

kermit

A program used to perform file transfers.

kernel

The core of an operating system, where all the drivers are that a system needs for associating with hardware such as Etherboot and other devices.

ksh

A Korn shell.

L

LAN

*L*ocal *A*rea *N*etworking that connects systems in order to share information and expensive computing resources.

leaf functions

Functions that do not call any other functions.

ld

The GNU linker. Interchangeably used with capitalization (as *LD*).

See Also: linker.

lexical

How characters in source code are translated into tokens that the compiler interprets.

lexical scoping

Nested functions that can access all the variables of the containing function that are visible at the point of its definition.

LGPL

GNU *L*ibrary *G*eneral *P*ublic *L*icense. See Appendix C and the Section called *GNU Library General Public License* in Appendix C

libopcodes

GNU library for manipulating machine opcodes. Its main use is disassembling binary files. Distributed under the GPL, used by the GNU binary utilities, assembler and debugger.

library

A collection of precompiled routines that a program can use. The routines, sometimes called *modules*, are stored in object format. Libraries are particularly useful for storing frequently used routines because you do not need to link them explicitly to every program that uses them. However, libraries cannot execute by their own routine. Multiple executables can include the library and other libraries can include the library. The linker automatically looks in libraries for routines not found elsewhere. Depending on the operating system, a library's routine is the product of special tools that linkers or other library tools merge to activate.

See Also: function, subroutine.

libreadline

GNU command line editing program, distributed under the GPL. This library implements full Emacs or vi key bindings with history support. Used by the GNU debugger.

libbfd

GNU object file manipulation library, distributed under the GPL. Used by the GNU binary utilities, assembler, linker, and debugger. It can read and write files in any supported object file format, and presents a somewhat loosely defined standard for manipulating object files.

See Also: BFD, FreeBSD, NetBSD.

libelix

Library comprising the precompiled routines for embedded environment development.

See Also: library.

LILO

*Li*nux *Lo*ader, a program that starts GNU/Linux on a system's hard disk.

linker

A tool that merges object files and library archives (such as compiled classes), building an executable, a complete program or a single executable file. For GNU, ld is the linker tool.

linker script

A set of programmer-supplied instructions that tell the linker how to handle object file sections, how to lay out memory, and so forth. For native linking, the contents of the linker script are normally determined by the needs of the operating system; for embedded targets, the programmer explicitly supplies the linker script.

Linux

A UNIX like operating system for many kinds of computers, created by Linus Torvalds and friends starting about 1990.

LISP

*LIS*t *P*rocessing language, based on the ideas of variable-length lists and/or trees as fundamental data types, and the interpretation of code as data (and vice-versa).

literal

The basic representation of any integer, floating point, or character value. For example, 3.0 is a single-precision floating point literal, and a is a character literal.

little-endian

A byte-ordering scheme, such as used by the Intel x86 family, which is all little-endian. Describes a computer architecture in which, within a given 16- or 32-bit word, bytes at lower addresses have lower significance (the word is stored little-end-first).

See Also: big-endian.

local variable

A data item known within a block, but inaccessible to code outside the block. For example, any variable defined within a Tcl method is a local variable and can not be used outside the method.

M

m68k

Motorola's 68000 family of microprocessors.

machine registers

Used for *general* register (for intermediary and temporary results), *index* register (for arithmetic operation and addressing memory), *pointer* register (for arithmetic operation and addressing memory), and *segment* register (for built-in memory management) purposes.

macro

A set of instructions for a program that a compiler substitutes whenever the macro's name appears in the source code.

See Also: function, build, library.

main

A program's fundamental system call for processing memory allocation and initialization resources for the program.

make

The GNU utility for automating recompilation, linking, and other program processes that take account of the interdependencies of modules and their modification times. make reads instructions from a file, Makefile, which specifies a set of targets to build, a set of files on which the targets depend and the commands to execute in order to produce them. If make is run with no arguments, it looks for a makefile, *Makefile*. For historical purposes, see also *Make: A Program for Maintaining Computer Programs* by A.I. Feldman (a Bell Labs publication).

Makefile

A specific script which tells a UNIX program **make** how to build a particular computer program or set of programs. A `makefile` contains variable assignments and rules that which, with input, say if any files have been modified more

recently than a target file (or if the target does not exist), then execution of specific commands will normally build the target from the inputs. For an example of a makefile, see Example 7-6.

See Also: build, target.

mangling

The process by which C++ types and classes are turned into symbols in object files that are compatible with other languages.

member function

Alternate term for method

See Also: method.

metacharacter

A character in a program conveying information rather than representing the character. The backslash charcater (/) is an example in C programming, indicating that the letter following the backslash character is an escape sequence enabling C to display an unprintable character such as a tab or a return action.

method

A procedure, function, or routine in programming languages, usually associated with a class.

Minix

A tutorial version of UNIX, written by Andy Tanenbaum and described in his textbook. *Minix* is said to have been the inspiration for Linux.

See Also: Linux.

MIPS

*M*illions of *I*nstructions *P*er *S*econd, a measurement of processing speed.

MMU

M emory *M* anagement *U* nit, part of a CPU optimized for the control of memory that the CPU uses.

modules

A self-contained hardware or software component that interacts with a larger system. For instance, modular programming has evolved into object-oriented programming, which provides formal rules for developing self-contained software modules.

MON960

Intel's ROM monitor for their i960 processor.

mount

Process or command for remote systems to use to move files from one location to another system.

multilibs

A collection of libraries built with different GNU compiler options, for target processors having several variants that require changes in code generation, one for each variant. This ensures that a program using -msoft-float (enabling software floating point functionality) will link with libraries built using the same option.

multiple inheritance

In object-oriented programming, relationship between objects, where an object derives its behavior from more than one other object.

See Also: inheritance.

multithreading

Functionality of a program that is designed to have parts of its code execute concurrently.

See Also: thread.

munge

To make changes to a file, irrevocably, such as a comprehensive rewrite of a routine, data structure or a whole program. Often an unintentional process of corrupting a file or memory block so that it is unusable, unreadable, unviewable or unrecoverable.

mutex

Object to provide *mutually exclusive* resource locking for a program using threads.

See Also: pthreads.

N

name-mangling

The process by which C++ types and classes are turned into symbols in object files that are compatible with other languages.

namespaces

An ANSI/ISO standard in development for the C++ libraries, allowing users to identify the scope for selecting specific functions by class or type from separate libraries, while still being able to let libraries work together and separately.

nested function

A function defined inside another function.

See Also: lexical scoping.

NetBSD

A free UNIX operating system for PCs and other kinds of computers.

See Also: BSD.

network boot

A process for getting system files over a network link, allowing any updating of files to happen at that single, robust server.

See Also: bootstrap, BOOTP.

newlib

Two C libraries, libc (C library), and libm (C math library) for embedded systems work.

newline

The ASCII LF character (for 0001010), used under UNIX as a text line terminator. See also CRLF.

NFS

*N*etwork *F*ile *S*ystem, a way to share files between machines as if shared on a local hard drive. With NFS, export files systems to other systems, and mount file systems exported from other machines.

See Also: BOOTP, filesystem, network boot, RAM Disk.

NOP

A machine instruction that does nothing (sometimes used in assembler-level programming as filler or for overwriting code to be removed in binaries).

NRE

*N*on-*R*ecurring *E*ngineering, typically used to refer to one-time-only development, such as re-targeting to a new architecture or adding a feature.

O

objects

The principal building blocks of object-oriented programs. Each object is a programming unit consisting of data variables and functionality.

See Also: classes.

object file

A binary-format file containing machine instructions and possibly symbolic relocation information. Typically produced by an assembler for a linker, for compiling with a program into an executable file.

object file format

The layout of object files and executable files.

opcodes

Assembler macros that provide commands instructing a specific device to perform a specific task.

overflow bit

A flag on some processors indicating an attempt to calculate a result too large for a register to hold.

overload

In programming languages, a feature that allows an object to have different meanings depending on its context. Used most often in reference to operators that can behave differently depending on the data type, or class, of the operands. For example, x+y can mean different things depending on whether x and y are simple integers or complex data structures.

P

partition

The swap space on a system's disk for storing idle portions of the memory used by running programs, and the filesystem space to store directories and files.

patch

A change in source code to correct or to enhance processes. Also a file that contains changes to source code; specifically, the results of a differentiation and comparison between the new file(s) and the old file(s).

See Also: diff utilities.

path

A filename, fully specified relative to the root directory (as opposed to relative to the current directory; the latter is sometimes called a relative path). Also called a pathname or full path, with UNIX and MS-DOS, the search path uses an environment variable, specifying the directories in which the shell (or Command.com, under MS-DOS) should look for commands. Other similar constructs exist with UNIX (for example, the C preprocessor has a search path it uses in looking for #include files).

PCI

*P*eripheral *C*omponent *I*nterconnect bus, a high performance means to carry signals for transferring data between the processor and any peripheral devices.

Perl

*P*ractical *E*xtraction *R*eport *L*anguage, a scripting language that Larry Wall designed which interprets data from text files and converts the data into an intermediate form before executing the data as a program. See http://www.perl.com/, the Perl Home Page.

PPC

*P*ower *PC* family of RISC processors, designed jointly by IBM and Motorola.

pragma

An ANSI C command comment that conveys non-essential yet pragmatic information, often to help with optimization when compiling.

profiling

See gprof.

project

A collection of references to source code files, the location of the files and how to operate on them.

project database

The result, a combination of which is a file that represents: a program's structures, the location of function declarations, the components of function declarations such as symbols and other references, and the relationship between a program's components.

PROM

*P*rogrammable *R*ead-*O*nly *M*emory, ROM that can be programmed using special equipment. PROMs can be programmed only once.

See Also: ROM, EPROM.

ps

Cygwin utility for displaying system status for all Cygwin processes.

pseudo-ops

Assembler directives.

See Also: opcodes.

pseudo registers

Pseudo registers can only contain scalar variables that cannot be aliased. This means that global variables, local variables that have their addresses taken, and aggregates (such as structures and unions) cannot be stored in pseudo registers, and thus they must be accessed with separate load and store instructions. Because of the guarantee that pseudo registers are not aliased, they are ideal targets for optimization.

pthreads

POSIX threads, which allow multiple tasks to run concurrently within the same program.

ptrace

The UNIX system call, traditionally used by debuggers to control other UNIX processes. ptrace arguments may include commands to read/write registers, to single-step, and to perform other debugging functions.

R

RAM

*R*andom-*A*ccess *M*emory, the volatile memory of an operating system, from which a microprocessor can read data or to which a microprocessor can write data.

RAM Disk

A disk drive that resides in memory, taking up very little space, for storing temporary work files or to help decrease the I/O load on a system's disks. Can use RAM from the buffer cache to dynamically grow in size.

See Also: RAM.

rc

A shell, for interactive use or for use with scripts, providing C-syntax features with better quoting rules then C or Bourne shells.

rc file

runcom files, using a startup script file that contains startup instructions for an application program (or an entire operating system), usually a text file containing commands of the sort that might have been invoked manually once the system was running but are to be executed automatically each time the system starts.

RCS

*R*evision *C*ontrol *S*ystem, the tools for controlling software revisions.

See Also: CVS.

RedBoot

A ROM monitor. RedBoot is a standard bootstrap and debugging environment for embedded systems from Red Hat.

reentrancy

A characteristic of library functions for multiple processes to use the same address space with assurance that the values stored in those spaces will remain constant between calls. libm library functions ensure that, when possible, they are reentrant.

registers

Registers are settings representing values that serve as temporary storage devices in a processor, allowing for faster access to data. Registers are divided into several classes:

See Also: pseudo registers, temporary registers, machine registers.

relocation

When doing assembly, assigning run-time addresses to sections, which includes the task of adjusting the mention of object-file addresses so that they refer to the proper run-time addresses.

See Also: section.

remote target

See: target

RFC

*R*equest *F*or *C*omment(s), one of a long-established series of numbered Internet informational documents and standards widely followed by commercial software and freeware in the Internet and UNIX communities.

RISC

*R*educed *I*nstruction *S*et *C*omputer, machines typically having fixed-length instructions, limited addressing modes, many registers, and visible pipelines.

ROM

*R*ead-*O*nly *M*emory, non-volatile memory that can be read, but not written, by the microprocessor, pre-recorded, often holding critical programs, such as boot data.

root

In a hierarchy of items or of separate files in a directory, the one item or directory from which all other items or directory paths descend. Also the name of the primary administrative account on GNU/Linux and other UNIX machines.

router

Network device that determines the optimal path along which network traffic should be forwarded, using packets from one network to another based on network layer data.

RPM

*R*ed Hat *P*ackage *M*anager, a means for having all the necessary components of the GNU/Linux software in a single file, from which an install process unpackages and then moves to appropriate directories.

To perform tasks with RPM, use the rpm --help | more command line input.

rsh

*R*emote *sh*ell protocol that allows a user to execute commands on a remote system without having to log in to the system.

RTEMS

A real-time operating system for telephones, automobile control systems, kitchen appliances, complex air traffic control systems, military weapon systems, production line control including robotics and automation and other rapidly changing technology.

RTI

The mnemonic for the return from interrupt instruction on computers.

RTOS

*R*eal-*T*ime *O*perating *S*ystem.

See Also: RTEMS.

runtime memory

Memory accessed while a program runs.

runtime system

The software system or environment in which compiled programs can run. The runtime system includes all the code necessary to load programs, dynamically link native methods, manage memory, handle exceptions, and an implementation of what may be an interpreter.

S

scope

Rules determining where a name is usable, such as a function or variable name (the general rule being that a name can be used from the point of declaration to its innermost enclosing compound statement).

screaming tty

A terminal line that disgorges an infinite number of random characters at an operating system.

See Also: tty.

SDK

Software Development Kit.

section

Object files and executables that are composed of a range of addresses, having optional data and optional relocation information. An object file written by the GNU assembler, as, has at least three sections, any of which may be empty; they are the *text*, *data* and *bss* sections.

sed

Stream-oriented version of the ed editor. For running repetitive edits on files or on files in several directories. Also for creating conversion programs.

segmentation fault

An error in which a running program attempts to access memory not allocated to it and core dumps with a segmentation violation error.

selective linking

When configuring libraries, the linker's usage of source in libraries, whereby the linking explicitly includes those libraries in the linking instruction.

semaphore

Object in source code allowing for a program to serialize access to shared memory.

sh

The Bourne shell. Also, the Hitachi Super-H family of RISC microprocessors.

shared libraries

A library of functions used by many executables, linked into each executable only when the particular executable runs.

shell

An interface program for accessing resources and for performing tasks. The original Multics shell ran programs by starting processes that were dynamically linking programs into existing code, calling them as subroutines (see *stub*) and dynamically de-linking them on return.

The main shells in use are ash, bash, csh, ksh, sh, and tcsh.

simulation

A means to interpret, classify and present information about an embedded system's behavior that is being modeled, comprising both hardware and software elements.

simulator

A tool that imitates the behavior of an embedded system so developers can run their code to see how the code will work once the actual hardware is available for testing (for instance, a simulator could show what happens when pressing cellphone buttons or what happens when moving video game 's joystick). A simulator makes simultaneous development of code and hardware then possible.

sleep

State of waiting, especially in debugging a program with loops, where code needs to have an *awakening* for an expected action to occur.

S-record

A binary download format, consisting of a series of records, each beginning with S, with *symbolic* data encoded as hexadecimal digits. Before downloading to a board, for instance, a program must be converted using S-records.

ssh

A *s*tandard C *sh*ell.

See Also: shell.

stabs

Based on symbol tables, a debug format introduced with the Berkeley UNIX system which records debugging information in certain symbols in the object file's symbol table. stabs information may also be encapsulated in COFF or ELF files.

stack

The layers (TCP, IP, and sometimes others) through which all data passes at both client and server ends of a data exchange. A data area or buffer used for storing requests that need to be handled, as in a list of tasks or, specifically, the contiguous parts of the data associated with one call to a specified function in a frame. The frame contains the arguments given to the function, the function's local variables, and the address at which the program is executing. In *The Art of Computer Programming* [2nd edition, vol. I, pg. 236], Donald Knuth writes, Many people who realized the importance of stacks and queues independently have given other names to these structures: stacks have been called push-down lists, reversion storages, cellars, nesting stores, piles, last-in-first-out (*LIFO*) lists, and even yo-yo lists!

stack frame

When debugging, the location of a function call, arguments about the call and called local variables, are within a block of data called the stack frame. Stack frames are allocated in a region of memory called the call stack. When stopping a program or when a program stops, debugging commands for examining the stack allow seeing all this information.

static library

A set of functions with which a program links, particularly for that program.

STL

*S*tandard *T*emplate *L*ibrary, a C++ library. See http://www.stl.org or http://phobos.nsrl.rochester.edu/liu/stl.html for more information.

stub

A small piece of code that executes on the target and communicates with the debugger, acting as its agent, collecting registers, setting memory values, etc. Also, in a native shared library system, the part of the shared library that actually gets linked with a program.

See Also: subroutine.

subroutine

Part of a program which calls another part of a program so that developers can simplify code in their programs and save memory.

See Also: stub.

swapping

Moving blocks of information in units known as pages between memory and disk as necessary during execution of an application; for instance, moving data from fast-access memory to a slow-access memory (swap out) or reverse (swap in). Supported by many operating systems, a process of exchanging two values; for instance, swapping could be exchanging values between two variables. Especially for use as temporary storage during reconfiguration.

See Also: variables.

symbols

Symbols are used to refer to variables, labels, functions, methods, macros, and other procedures or programatic constructs in a program; basically, symbols are names and addresses, representing simple instructions for a program. Typically every global function and variable in a C program will have an associated symbol.

T

target

Usage varies: (1) an actual physical device, such as a target microprocessor motherboard that gets files from a host operating system, or (2) an application or program run on a target board, or (3) a class of devices whose types include executables and libraries, either of which may include other libraries and object files.

See Also: build, Makefile.

target environment

A host operating system and a target (usually, a processor board) when working together with an executable application.

Tcl/Tk

The code language used to develop an IDE. Developed by John Ousterhout, Tcl is the basic programming language while Tk is a toolkit of widgets (graphical objects similar to those of other GUI toolkits, such as Xlib, Xview and Motif). Unlike many of the other toolkits, it is not necessary to use C or C++ in order to manipulate the widgets, and useful applications can be built very rapidly with Tcl/Tk. For more information, see http://www.scriptics.com/ftp/ (distribution sources) and http://www.tclconsortium.org (Tcl/Tk consortium).

See Also: widgets.

TCP/IP

Transmission control protocol (based on IP), an internet protocol that provides for the reliable delivery of streams of data across the web.

See Also: IP.

tcsh

A C shell.

telnet

Standard terminal emulation protocol in the TCP/IP protocol stack, when using a remote terminal connection, enabling remote log in to systems, thereby using resources as if connected to a local system.

temporary registers

Temporary registers are used to hold intermediate results of computations within a basic block. Each temporary register must be defined and used in exactly one place, and never assign a value to a temporary register that is never used, and do not use the value in a temporary register more than once.

ternary

System using three as the base, such as a ternary logarithm.

TFTP

*T*rivial *F*ile *T*ransfer *P*rotocol, used when getting an IP address with tftpboot command line input. A restricted version of TCP with no authentication, using UDP rather than TCP.

See Also: TCP/IP.

thread

The basic unit of instructions for a program to execute. A process can have several threads running concurrently, each performing a different job, such as waiting for events or performing a time-consuming job that the program doesn t need to complete before resuming. For a debugging process, when a thread finishes its job, the debugger suspends or destroys the thread running.

three-way cross

See: Canadian cross

toolchain

A complete set of GNUPro tools for a particular native or host architecture and a target environment, typically using the naming convention combining a host name and a target name as a prefix to the tool name. See the Section called *Host Configurations* in Chapter 1 and the Section called *Toolchains and Usage* in Chapter 5 for more explanation. A program follows a standard of four stages: compiling, assembling, archiving, linking, and debugging.

tracepoint

A hit during a debugging process.

trampolines

On-the-fly generation by a compiler (such as gcc) of small executable code objects that do indirection between code sections, taking the address of a nested function.

triple cross

See: Canadian cross

tty

Any serial port, whether or not the device connected to it is a terminal; so called because under UNIX such devices have names of the form, tty*. Its derivation is from the term, *t* ele*ty* pe.

twiddle

A small and insignificant change to a program. Derived from squiggle, or in ASCII shorthand, tilde (for the ASCII character definition, 1111110, of the character, ~).

type

A classification in a file of particular information for a computer to use. Information can be about the nature of a variable, such as an integer, floating-point number or text character, with the variable defining and and restricting the values that it can hold. For specific types of files that computers can store for use.

See Also: filetype.

typedef

In C programming, a keyword for naming new data types.

See Also: data type.

U

Unicode

A 16-bit character set defined by ISO 10646.

unary

Having, consisting of, or acting on a single component.

union

A structure using different types of variables, such as integers, characters or Boolean values, storing them in the same location, only one at a time.

UNIX

UNIX operating system (lowercase spelling of Unix used interchangeably). Invented in 1969 by Ken Thompson after Bell Labs left the Multics project, UNIX subsequently underwent mutations and expansions under many different hands, resulting in a uniquely flexible and developer-friendly environment, becoming what was once the most accepted multiuser, general-purpose operating system in the world. See also .

uuencode/uudecode

For transmitting binary files over any media that support ASCII data transmission.

V

variables

A variable is a symbol or name that a program uses to represent a value. *Global variables* have one value at a time, and this value is in effect for the whole system. *Constant variables* have values that never change. *Instance variables* are defined as of exactly one class, called its most *derived class*. *Local variables* help create variable values that exist temporarily while within a certain part of the program; these values are called *local*, and the variables so used are called local variables. For example, when a function is called, its argument variables receive new local values that last until the function exits. *Void variables* use symbols that lack values.

VFS

V irtual *F* ile *S* ystem; its functionality, among other things, is to flush the read buffer when it detects a disk change on the floppy disk drive.

vi

*V*isual *I*nterface, a screen editor crafted together by Bill Joy for an early BSD release. Became the de facto standard UNIX editor and a nearly undisputed hacker favorite (arguably) outside of MIT until Emacs after about 1984.

See Also: ed, Emacs.

virtual machine

An abstract specification for a computing device that can be implemented in different ways, in software or hardware. Compiling to the instruction set of a virtual machine is much like compiling to the instruction set of a microprocessor, using a bytecode instruction set, a set of registers, a stack, a garbage-collected heap, and an area for storing methods.

W

watchpoint

A special breakpoint that stops a program's debugging process when the value of an expression changes.

whitespace

In code, newlines, spaces and tab characters which the compiler treats as a contiguous, single character. An instance that is noticeable for the program output, and only as formatting for human readability.

widgets

The components of software by which we get many of the interface features that provide interoperability, such as the scrollbars or the means of selecting text with a cursor or pointer with the functionality of object-oriented programming.

worm

A *virus* program (one that propagates itself over a network, reproducing itself as it goes, the famous *Great Worm of 1988* being one of the best-known of the early examples; it was Robert T. Morris's 1988 virus, a benign one that got out of control and hogged hundreds of Sun and VAX systems across the U.S.).

X

X

A windowing system for GUIs developed at MIT that lets users run applications on other systems on a network and view the output on their computer screen. The X Windows window manager component of the GUI allows multiple resizable, relocatable X windows to be viewed on screen, allowing simultaneous processes to develop.

x86

<FunctionVariant> x </FunctionVariant> 86

Name for Intel's 80x86 processor chip family (the first chip from Intel being the 8086).

XCOFF

e*X* tended *COFF*, IBM's object file format for PowerPC systems.

xor-endian

A way of implementing a big-endian ISA.

See Also: PPC.

xterm

The X Windows system terminal program.

See Also: X.

Y

yacc

GNU parser generator that takes a description of tokens in the form of a grammar and generates a parser in the form of a C program.

Notes

1. http://www.ora.com/reference/dictionary/
2. http://pcwebopedia.com/
3. http://www.tekmom.com/buzzwords/
4. http://pespmc1.vub.ac.be/ASC/INDEXASC.html
5. http://coverage.cnet.com/Resources/Info/Glossary/
6. http://sources.redhat.com/cygwin/
7. http://www.redhat.com/products/edk/
8. http://www.delorie.com/djgpp/
9. http://sources.redhat.com/binutils/docs-2.10/gprof.html
10. http://www.apache.org/
11. http://www.ieee.org/
12. http://standards.ieee.org/
13. http://www.perl.com/
14. http://www.stl.org
15. http://phobos.nsrl.rochester.edu/liu/stl.html
16. http://www.scriptics.com/ftp/
17. http://www.tclconsortium.org