Sega Dreamcast

Dreamcast GNUPro™ Toolkit Getting Started

Introduction

Installation

Important Information

This documentation has been provided courtesy of CYGNUS. The contents are applicable to GNUPro™ Toolkit development, however, all references to development support offered by CYGNUS should be ignored.

Technical support for this product as it applies to the Sega Dreamcast™ development environment should be directed to Sega Third Party Developer Technical Support at 415/701-4060. Future updates and/or additional information may also be found at Sega's DTS Website at, **`http//:www.dts.sega.com/NextGen`**

This documentation has been prepared by Cygnus Technical Publications; contact the Cygnus Technical Publications staff: `doc@cygnus.com`.

Part #: 300-400-101000041

GNUPro Warranty

The GNUPro Toolkit is free software, covered by the GNU General Public License, and you are welcome to change it and/or distribute copies of it under certain conditions. This version of GNUPro Toolkit is supported for customers of Cygnus.

For non-customers, GNUPro Toolkit software has NO WARRANTY.

Because this software is licensed free of charge, there are no warranties for it, to the extent permitted by applicable law. Except when otherwise stated in writing, the copyright holders and/or other parties provide the software “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 software is with you. Should the software prove defective, you assume the cost of all necessary servicing, repair or correction.

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.

Year 2000 compliance

This and all subsequent releases of the GNUPro Toolkit products are *Year 2000 Compliant*.

Cygnus Solutions defines a product to be Year 2000 Compliant (Y2K) if it does not produce errors in recording, storing, processing and presenting calendar dates as a result of the transition from December 31, 1999 to January 1, 2000.

A Y2K product will recognize the Year 2000 as a leap year. This compliance is contingent upon third party products that exchange date data with the Cygnus product doing so properly and accurately, in a form and format compatible with the Cygnus product.

GNUPro Toolkit processes dates only to the extent of using the date data provided by the host or target operating system for date representation used in internal processes, such as file modifications. Any Y2K issues resulting from the operation of the Cygnus products, therefore, are necessarily dependent upon the Y2K compliance of relevant host and/or target operating systems. Cygnus has not tested all operating systems and, as such, cannot assure that every system and/or environment will manage and manipulate data involving dates before and after December 31, 1999, without any time or date related system defects or abnormalities, and without any decreases in functionality or performance. Cygnus cannot assure that applications which you modify using Cygnus products will be Year 2000 compliant.

How to contact Cygnus

Use the following means to contact Cygnus.

Cygnus Headquarters

1325 Chesapeake Terrace
Sunnyvale, CA 94089 USA
Telephone (toll free): +1 800 CYGNUS-1
Telephone (main line): +1 408 542 9600
Telephone (hotline): +1 408 542 9601
FAX: +1-408 542 9699
(Faxes are answered 8 a.m.–5 p.m., Monday through Friday.)
email: `info@cygnus.com`
Website: `www.cygnus.com`.

Cygnus United Kingdom

36 Cambridge Place
Cambridge CB2 1NS
United Kingdom
Telephone: +44 1223 728728
FAX: +44 1223 728728
email: `info@cygnus.co.uk/`

Cygnus Japan

Nihon Cygnus Solutions
Madre Matsuda Building
4-13 Kioi-cho Chiyoda-ku
Tokyo 102-0094
Telephone: +81 3 3234 3896
FAX: +81 3 3239 3300
email: `info@cygnus.co.jp`
Website: `http://www.cygnus.co.jp/`

Use the hotline (+1 408 542 9601) to get help, although the most reliable way to resolve problems with GNUPro Toolkit is by using email:

`bugs@cygnus.com`.

Contents

Installation

GNUPRO™ TOOLKIT

Introduction

Cygnus

This documentation has been prepared by Cygnus Technical Publications; contact the Cygnus Technical Publications staff: **doc@cygnus.com**.

1

Overview of GNUPro Toolkit

Cygnus is the leading provider of single-source, UNIX, and NT desktop and cross-platform development tools for 32- and 64-bit microcontrollers. Cygnus provides GNUPro Toolkit™ as a set of powerful, multi-platform software development tools for advanced desktop and embedded systems. For more details, see “What’s in this package” on page 7.

- To install for the UNIX platforms, see “Installing on Unix Systems from CD” on page 84.
- To install for the Windows platforms, see “Installing on Win 95/NT systems from CD” on page 93.

For information on what’s supported, see the following documentation.

- “Native configurations support” on page 13
- “Embedded cross-configuration support” on page 14
- “Version numbers for programs” on page 16
- “Naming hosts and targets” on page 17

Cygnus sells the GNUPro toolkit together with mission-critical support services for rapid response and resolution of technical questions or problems, as well as regular software upgrades with enhancements. See also “Red flag alerts & enhancements” on page 21 and, to report problems or to find out about problems, see “How to report problems” on page 95.

About Cygnus

The company was founded in 1989 to provide commercial support for open Internet technologies. Cygnus established a support model for the GNU standard, leveraging the contributions of the net community and the requirements of the semiconductor industry to produce a powerful multi-platform software development toolkit. The changes and improvements to the GNU tools are returned to the net community via the Free Software Foundation, while the Cygnus implementation of the these tools is sold and supported by Cygnus directly. Driven by our unique business model, the company's sales have grown at a compound annual rate of over 65% since 1992. Our consistent profitability and solid financial position have enabled the company to rapidly expand operations to support the growing needs of our customer base.

Cygnus is a privately held company with venture capital investment from August Capital and Greylock Management. We are global in both our operations and our customer set. Our headquarters are in Sunnyvale, California, with additional offices in Cambridge, Massachusetts, Atlanta, Georgia; and international offices in Canada, Japan, the United Kingdom, and Germany. Our historical roots and current strong ties with the Internet community also allow us to leverage the global power of the Internet to drive our toolchain innovation and recruit new engineers for our growing company.

Cygnus also benefits from advances in all of the related technologies that make end-user solutions more cost-effective and drive end-user demand. These technologies include storage (disk, flash, DRAM), display (LCD), Internet and communication protocols, and wireless connectivity. Collectively, advances in these technologies allow ever-greater embedding of intelligence in traditional embedded systems (telecommunications equipment, automobiles, office equipment), and enable completely new classes of computing devices such as Personal Digital Assistants (PDAs).

One thing is certain: software tools and software content for them must keep pace in order to realize the promise of powerful, distributed, and densely interconnected computing power. The Cygnus mission is to enable this vision of “pervasive computing” through software solutions.

Cygnus doesn’t deliver technology for technology’s sake. Our clients turn to us to create flexibility (with single-source solutions providing true multi-platform capabilities and easy retargeting), to shorten time-to-market (starting software development before hardware is ready), and to reduce costs (providing cost-effective support services and redistributable solutions). These are the reasons that the best names in the business choose Cygnus.

The company strategy is to continue to extend the functionality and performance of

the development tools, as well as to deliver innovative software component technologies for use in embedded systems. Customers include the world's top microcontroller companies as well as leading telecommunications software, game console, consumer electronics, and office equipment companies.

The world's leading product manufacturers choose the GNUPro toolkit for these benefits:

- Multi-platform capabilities from a single source base to quickly port code
- Knowledgeable support for keeping design teams productive and on-schedule
- Superior software engineering, incorporating leading-edge features and code optimizations
- Early availability of tools for a wide range of processors
- Stringent testing to insure solid, stable tools for fast development
- Custom enhancements to decrease time to market

About the tools

GNUPro Toolkit includes a C/C++ compiler, a powerful source-level debugger with a graphical user interface, an assembler, a linker/loader, binary file utilities, and libraries. Complete source code, tested binaries, and product documentation accompany the tools. Not all tools are available for all platforms and operating systems. See “Embedded cross-configuration support” on page 14 for specific information on your system. See “What’s in this package” on page 7 for information about the tools.

To locate specific documentation, see “GNUPro Toolkit documentation” on page 9 and, also, see the following topics.

What's in this package

GNUPro software delivers productivity, flexibility, performance and portability with its ISO-conforming C and ISO-tracking C++ compiler, macro-assembler, GUI debugger, binary utilities and libraries.

See "GNUPro Toolkit documentation" on page 9 for the discussion to which the following documentation refers for GNUPro Toolkit components.

See the following documentation for more about what GNUPro Toolkit contains.

- "Compilers and development tools" on page 7
- "Libraries" on page 7
- "General utilities" on page 8
- "Binary utilities" on page 8

Compilers and development tools

See ***GNUPro Compiler Tools***, ***GNUPro Debugging Tools*** and ***GNUPro Utilities*** for documentation regarding these tools.

`gcc`	C compiler
`g++`	C++ compiler
`gdb`	Debugger
`gas`	Assembler
`cpp`	C Preprocessor
`ld`	Linker
`gdbtk`	Debugger graphical user interface (evoked by using the GNU debugger)

Libraries

See ***GNUPro Libraries*** for documentation regarding these libraries.

`libstdc++`	C++ class library
`libio`	C++ iostreams library
`libc`	ANSI C runtime library (*only available for cross-development*)
`libm`	C math subroutine library (*only available for cross-development*)

Binary utilities

See ***GNUPro Utilities*** for documentation.

`c++filt`	C++ symbol name deciphering utility
`nm`	Lists object file symbol tables
`objdump`	Displays object file information
`size`	Lists section and total sizes
`ar`	Manages object code archives
`ranlib`	Generates archive index
`strip`	Discards symbols
`objcopy`	Copies and translates object files

General utilities

See ***GNUPro Utilities***, ***GNUPro Advanced Topics*** and ***GNUPro Compiler Tools*** and the online documentation for clarification regarding these utilities.

`flex`	Fast lexical analyzer generator
`make`	Compilation control program
`diff`, `diff3`, `sdiff`	Compare text files
`patch`	Installs source fixes
`cmp`	Compares files byte-by-byte
`send-pr`	Sends structured problem reports to Cygnus
`install-sid`	Customizes `send-pr` for your site
`gcov`	Coverage analyzer

GNUPro Toolkit documentation

GNUPro Toolkit includes documentation in three forms.

- *Printed*
  Available by request.
- *HTML*
  To locate the documentation on the web, see:
  `http://www.cygnus.com/pubs/gnupro`
- *Online*
  See "Using online documentation" on page 11, "Reading online documentation" on page 11 and *GNU Online Documentation* in ***GNUPro Advanced Topics***.

Documentation conventions

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

- `Typewriter-text for input`
  Indicates onscreen text as an example of input for a program, such as `PATH=/usr/cygnus`. It will also indicate other literal bits of text from a program, such as filenames or source code.
- **`Typewriter-text in bold for output`**
  Indicates text that is an example of a program's onscreen output such as '`%`' or '**`(gdb)`**' as command prompt.

IMPORTANT! '`%`' indicates a command prompt in most documentation; a command prompt in some rare examples, depending on the system in use and the program's tool, may actually be '`#`' or '**`$`**' and, as with `gdbtk`, '**`(gdb)`**'.

- *`Typewriter-text in italics for variables`*
  Indicates text that stands for variable input such as *`filename`* where the actual input might be a file with the name, `foobar.c`. For instance, the documentation may read "To delete the file named *`filename`*, type `rm` *`filename`*." *`filename`* stands for the file you want to substitute, according to its name.
- Keys or keystroke combinations to use
  This font indicates keys on your keyboard, such as Ctrl, Del, or even Spacebar.

  The Ctrl key and the Meta key are often indicated in the text by C- and M-, respectively. In addition, Meta (or M-) may indicate, for example, the Alt key on a Windows keyboard, or a '◆' key on Unix keyboards.

  The Enter (carriage return) key on your keyboard may appear as ENTER or RET (for *return*) in text.

What's in the documentation

GNUPro Toolkit documentation includes the following documentation. To locate the documentation, see `http://www.cygnus.com/pubs/gnupro/`.

- ***Getting Started with GNUPro Toolkit***
  - *Introduction*
  - *Installation*
- ***GNUPro Compiler Tools***
  - *Using GNU CC*
  - *The C Preprocessor*
- ***GNUPro Debugging Tools***
  - *Debugging with GDB*
  - *GDBTk*
- ***GNUPro Libraries***
  - *GNUPro C Library*
  - *GNUPro Math Library*
  - *The GNU C++ Iostream Library*
- ***GNUPro Utilities***
  - *Using AS (the GNU Assembler)*
  - *Using LD (the GNU Linker)*
  - *The GNU Binary Utilities*
  - *GNU Make*
- ***GNUPro Advanced Topics***
  - *Rebuilding from Source*
  - *GNU Online Documentation*
  - *Reporting Problems*
  - *Legal Notices*
- ***GNUPro Tools for Embedded Systems***

See also "Using website documentation" on page 12.

Using online documentation

For online use, the accompanying software includes online versions of the following documentation. This is not to be confused with the HTML versions.

`man` *pages*
: For all the tools and programs in this release.

FLEX: A Fast Lexical Analyzer Generator
: Generates lexical analyzers suitable for GNU `gcc` and other compilers.

Using an Porting GNU CC
: Detailed information about what's needed to put **`gcc`** on different platforms, or to modify **`gcc`**. Also includes the information from the printed manual, *Using GNU CC*.

BYacc
: Discussion of the Berkeley Yacc parser generator.

Texinfo: The GNU Documentation Format
: How you can use TEX to print these manuals, and how to write your own manuals in this style.

Cygnus configuration program
: Details on the configuration program used in GNUPro Toolkit.

GNU Coding Standards
: A complete discussion of the coding standards used by the GNU project.

GNU gprof
: Details on the GNU performance analyzer, only for the Sun-3 and Sun-4 (SunOS 4.1 or Solaris 2) platforms.

You have the freedom to copy the documentation using its accompanying copyright statements, which include the necessary permissions.

See *GNU Online Documentation* in ***GNUPro Advanced Topics*** for documentation regarding these tools.

Texinfo†, `texindex`, `texi2dvi`	Documentation formatting tools
`makeinfo`, `info`	Online documentation tools

† Requires TEX, the technical documentation formatting tool

See also "Reading online documentation" on page 11.

Reading online documentation

You can browse through the online documentation using either Emacs or the **`info`** documentation browser program, included in the GNUPro Toolkit. Online, the

information is organized into *nodes*, corresponding to the sections of a printed book.

You can follow them in sequence, as in the printed 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, you can have **`info`** take you immediately to that other section—and you can get back again easily to take up your reading where you had been. Naturally, you can also search for particular words or phrases.

The best way to get started with the online documentation system is to run the browser, **`info`**. After installing GNUPro Toolkit, you can get into **`info`** by just typing its name at your shell prompt (shown as '`%`' by the following example)—no options or arguments are necessary.

```
% info
```

You may need to check that **`info`** is in your shell path after you install GNUPro Toolkit. If you have problems running **`info`**, contact your system administrator.

To learn how to use **`info`**, type `h` for a programmed instruction sequence, or Ctrl-`h` for a short summary of commands.

If at any time you are ready to stop using `info`, type `q`.

See "Reading GNU Online Documentation" in *GNU Online Documentation* in ***GNUPro Advanced Topics*** for detailed discussion of the **`info`** program.

Using website documentation

As with all GNU software, the HTML source for documentation is available (or you can convert it yourself using publicly available utilities) if you wish to put them into an internal Web server.

Documentation is also available in HTML format(see "GNUPro Toolkit documentation" on page 9) at the following location.

```
http://www.cygnus.com/pubs/gnupro/
```

Contact Cygnus to report any problems to the documentation department: `doc@cygnus.com`.

Native configurations support

GNUPro Toolkit software supports the following native configurations.

Sun SPARC Solaris 2.5.1/2.6
Sun SPARC SunOS 4.1.4
HP 9000/700 HP-UX 10.01/10.10/10.20
IBM RS/6000 AIX 3.2.5/4.1.4
IBM PowerPC AIX 4.1.4/4.2
DEC Alpha Digital UNIX 3.2C/4.0
SGI Irix 5.3/6.2
386+ for Unixware
386+ for Windows
Windows NT4.0-sp3/95-osr2
Linux RedHat 5.0

Embedded cross-configuration support

GNUPro Toolkit software supports the embedded cross-configurations as detailed in Table 1 and Table 2.

NOTE: In the following tables, *x* denotes a version number range; for instance, with the HP-UX 10.*x* , the *x* denotes support for all the 10.01, 10.10 or 10.20 versions.

Table 1: Embedded cross-configurations supported

Host	*Target*	*Output Format*
HP-UX 10.*x*	PowerPC	EABI
SGI Irix 5.3/6.2	MIPS R3*xx*0	ELF, ECOFF
	MIPS R4*xx*0	ELF
	PowerPC	EABI
	SH	COFF
SPARC Solaris 2.5.1-2.6	H8/300	COFF
	i960	COFF
	M68K	a.out, COFF, ELF
	MIPS R4*xx*0	ELF
	MIPS R3*xx*0	ELF, ECOFF
	PowerPC	EABI
	SH	COFF
	*x*86	ELF
SPARC SunOS 4.1.4	H8/300	COFF
	i960	COFF
	M68K	a.out, COFF
	MIPS R3*xx*0	ELF
	MIPS R4*xx*0	ELF
	PowerPC	EABI
	SH	COFF
	SPARClet	a.out
	SPARClite	a.out, COFF
	*x*86	a.out, ELF

Table 2: Embedded cross-configurations supported (*cont'd*)

Host	***Target***	***Output Format***
PowerPC AIX 4.2.1	PowerPC	EABI
	*x*86	ELF
Windows NT4.0/95	H8/300	COFF
	i960	COFF
	M68K	COFF
	MIPS V_R4100	ELF
	MIPS V_R4300	ELF
	PowerPC	EABI
	SH	COFF
	SPARC	a.out
	SunOS 2.*x*	COFF
	SunOS 4.1.4	COFF
	*x*86	a.out, ELF

Version numbers for programs

Table 3 shows the current version numbers for individual programs in GNUPro Toolkit.

Table 3: Versions of the tools.

Program	*Version Numbers*
bfd	2.8-gnupro-98r1
binutils	2.9-gnupro-98r1
diff	2.7-gnupro-98r1
expect	5.21-gnupro-98r1
flex	2.5-gnupro-98r1
gas	2.9-gnupro-98r1
gcc	2.9-gnupro-98r1
gcov	1.5-gnupro-98r1
gdb	4.17-gnupro-98r1
ld	2.8-gnupro-98r1
libstdc++	2.8-gnupro-98r1
libio	2.8-gnupro-98r1
make	3.75-gnupro-98r1
makeinfo	1.67-gnupro-98r1
newlib/libc	1.8-gnupro-98r1
newlib/libm	1.8-gnupro-98r1
patch	2.5-gnupro-98r1
send-pr	3.104-gnupro-98r1

Naming hosts and targets

Your CD is labeled to indicate the host (and target, if applicable) for which the binaries in GNUPro Toolkit are configured. The specifications used for hosts and targets in the configure script are based on a three-part naming scheme, though the scheme is slightly different between hosts and targets.

The full naming scheme for hosts encodes three pieces of information in the following standard pattern.

`architecture-vendor-os`

For instance, the full name for a Sun SPARCstation running SunOS 4.1.4 is `sparc-sun-sunos4.1.4`.

WARNING! `configure` can represent a very large number of combinations of architecture, vendor, and operating systems. Support is not possible for all combinations.

Host names

Table 4 shows the usage of canonical names for referring to the corresponding host platforms that Cygnus supports. For any questions about compatibility, contact Cygnus (see “How to contact Cygnus” on page v).

Table 4: Naming hosts

Canonical name	*Platform*
`alpha-dec-osf3.2C`	DEC Alpha Digital UNIX v3.2C
`alpha-dec-osf4.0`	DEC Alpha Digital UNIX v4.0
`hppa1.1-hp-hpux10`	HP 9000/700, HP-UX B.10.01
`hppa1.1-hp-hpux10.20`	HP 9000/700, HP-UX B.10.20
`i386-cygwin32`	Windows NT-sp3/95-osr2
`i386-pc-linux-gnu`	Intel PC, Linux RedHat 5.0
`mips-sgi-irix5`	SGI Irix 5.3
`mips-sgi-irix6`	SGI Irix 6.2
`powerpc-ibm-aix4.1`	IBM PowerPC, AIX 4.1.4
`powerpc-ibm-aix4.2`	IBM PowerPC, AIX 4.2.1
`rs6000-ibm-aix4.1`	IBM PowerPC, AIX4.1.4
`sparc-sun-solaris2.5`	SPARCstation, Solaris 2.5.1
`sparc-sun-solaris2.6`	SPARCstation, Solaris 2.6
`sparc-sun-sunos4.1`	SPARCstation, SunOS 4.1.4

Target names

The following tables (Table 5-Table 12) list some of the more common targets supported by Cygnus. Not all targets have support on every host. See also “Native configurations support” on page 13 and “Embedded cross-configuration support” on page 14 for the matrices of the host/target combinations supported by Cygnus. Also, for more informaton on using particular tools and their targets, see ***GNUPro Tools for Embedded Systems***.

WARNING! `configure` can represent a very large number of target name combinations of architecture, vendor, and object formats. Support is not possible for all combinations.

Table 5: H8/300 processor name and output format

`h8300-hms-coff`	COFF object code format

Table 6: i960 processor names and output format

`i960-coff`	MON960 monitor (COFF format)

Table 7: M68K processor names and output formats

`m68k-aout`	a.out object code format
`m68k-coff`	COFF object code format
`m68k-elf`	ELF object code format

Table 8: MIPS processor names and output formats

`mips-elf`	ELF object code format
`mips-sgi-irix6`	ELF object code format

Table 9: PowerPC processor names and output formats

`powerpc-eabi`	ELF object code format (EABI)

Table 10: SH processor name and output format

`sh-hms-coff`	COFF object code format

Table 11: SPARC processor names and output formats

`sparc-aout`	a.out object code format
`sparc-coff`	COFF object code format
`sparc-elf`	ELF object code format

Table 12: *x*86 processor names and output formats

`i386-aout`	a.out object code format
`i386-elf`	ELF object code format

Using `config.guess`

`config.guess` is a shell script that attempts to deduce the host type from which it is called, using system commands like `uname` if they are available.

`config.guess` is remarkably adept at deciphering the proper configuration for your host; if you are building a tree to run on the same host on which you're building, we recommend not specifying the *`hosttype`* argument.

`config.guess` is called by `configure`; you need never run it by hand, unless you're curious about the output.

2

Red flag alerts & enhancements

The following documentation discusses issues that are new with this release. These issues may provide clues that may help if you run into problems with the GNUPro tools. They may also be suggested or requested enhancements.

See also "Issues from previous releases" on page 27 for issues that may be pertinent from versions of GNUPro Toolkit prior to this release.

General issues

The following documentation discusses new features and alerts for this release of GNUPro Toolkit.

- The `malloc` routines in the embedded C library have been updated. The library includes a new header file, `<malloc.h>`, and some useful new functions, including `memalign` and `mallinfo`. The `malloc` routines now always call `__malloc_lock` and `__malloc_unlock` to lock and unlock access to the memory pool. The library provides default versions of these functions which do nothing; if you invoke `malloc` from multiple threads, or reentrantly, you should provide your own versions of these functions. See *GNUPro C Library* in ***GNUPro Libraries*** for more details.
- This release contains GNUPro for the Windows environment, specifically, the Windows NT 4.0 (WinNT4.0-sp3) and the Windows 95 environments (Win95-osr2), using the GNUPro `cygwin32` tools.

  Unix by default does not have spaces in any of the system paths where as the NT and Win95 products love to use them in (such as, "My Documents"). Make sure when using the tools that this alerts you to the file naming conventions.
- As for the engineering of `cygwin32`, some general parameters need further discussion.

  `@file` is a general command option that will work with all programs that are dependent on the `cywin32.dll`. Simply, the file named after the `@` symbol is inserted into the command line.

  The command, `gcc @`*`foo`*, will use the contents of the file, *`foo`*, as a command line option for GCC. If for some reason you wish to use text such as `@`*`foo`* as command line display, you can use the command, `gcc \@`*`foo`*. The `\` acts as an escape character.

  This option is mainly to get around the 127 character limit on the DOS prompt in Win95. In a UNIX shell, you use `cat` *`foo`*. However, one advantage of the `@file` command is with using a file that, having another `@file` command in its contents, that file will also be inserted into the command line.

  `cat` *`foo`* and `cat @`*`foo`* should not produce identical output. Expect the latter to compress unquoted strings of spaces, tabs, and newlines into a single space. In particular, barring quoting, expect it to always print only a single newline. Command line parsing/dequoting is done after the file is read. So you can put a long quoted string into a file by putting the quotes in the file.
- For all the new supported hosts, see "Native configurations support" on page 13 and "Embedded cross-configuration support" on page 14.

Compiler issues

The following documentation concerns enhancement issues that may affect the API or ABI when working with the GNUPro compiler tools (`gcc` and `g++`).

- GNUPro Toolkit includes the compiler, `egcs 1.0`, which is similar to `gcc 2.8.x`.
- There have been further enhancements to the Global Common Sub-Elimination (`gcse`) routines in the libraries.
- There have been improvements to the general induction variable identification and elimination, along with improvements to the loop invariant code motion.
- There are now accurate warnings, especially when using exception handling (such as when using uninitialized variables, for example)
- There is improved global constant and copy propogation.
- There are several new switches: `-Os` (for optimizing space), `-fmove-all-movables` (for forcing all invariant computations in loops to be moved outside the loop), `-freduce-all-givs` (for forcing all general induction variables in loops to be strength reduced), and `-frerun-loop-opt` (for rerunning loop optimizations twice).
- There is now partial redundancy elimination with lazy code motion and critical edge splitting.
- There is now global code hoisting and unification.
- GNUPro Toolkit fully supports the following functionality for the GNU C++ programming.
  - Thread-safe exception handling, including nested exceptions and placement delete. `operator new` now throws `bad_alloc`.
  - Member class templates.
  - Local classes in templates.
  - Template parameters.
  - Template friends.
  - Protected virtual inheritance.
  - Loop optimization improvements (with the test at the end in more cases); for class D derived from B, which has a member, `int i`, `&D::i` is now of type `int B::*` instead of `int D::*`.
  - Compact name mangling with `-fsquangle`. Future versions of the compiler may use the new mangling to indicate a different, incompatible ABI. To use this, rebuild the libraries (including `libgcc`) with this option.
  - Member function and template support.

- Cygnus fully supports the `m68k-elf` toolchain. An earlier version of `m68k-elf` that was previously available is obsolete. Migrating to the new `m68k-elf` toolchain will require rebuilding existing object files and modifying any existing source code written in assembly language. Source code that was written in C will not need to be changed.

  Any object files that were created with the old toolchain will not be link-compatible with object files created with the new toolchain. Even if old object files are successfully linked with new object files, they will not be compatible at run-time, because the calling convention has changed. To circumvent this problem, recompile all object files from their source code with the new `m68k-elf` toolchain.

  The assembly language format expected by the new assembler has changed slightly to make it more similar to the format of `m68k-coff` and `m68k-aout` assembly language. Programs that are written in C do not need modification. Programs written in assembly language will need to be modified in the following ways:

  - The `fp` register must be referred to instead as the `a6` register. `link %fp,&0` must be replaced with `link %a6,#0` as a modification.
  - Incidences of the `&` symbol must be replaced with `#`. `link %fp,&0` must be replaced with `link %a6,#0` as a modification.

  There have been some ABI changes:

  - Pointers are now returned in register `d0`, not `a0`.
  - Long doubles are now returned in `fp0` when an FPU is present, not `d0`.
- Two functions resembling `__attribute__` functionality, `-ffunction-sections` and `-fdata-sections`, move elements that don't have section attributes set. `-ffunction-sections` moves `function` elements and `-fdata-sections` moves `data` elements.

  GCC uses `.text.`*`objectname`* if transforming a function, and uses `.data.`*`objectname`* or `.rodata.`*`objectname`* if transforming constant data elements (which are read-only elements). `.data.`*`objectname`* is the default.

Debugger issues

The following documentation concerns the enhancement issues for the GNUPro debugger tools.

- With this release of GNUPro Toolkit, there is a graphical user interface for the GNU debugger, gdb. Cygnus refers to it as GDBtk, since it uses the Tcl/Tk programming system, and it is the same source for both Unix and Windows environments.

  Tcl/Tk is a programming system developed by John Ousterhout. Easy to use with very useful graphical interface facilities, 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.
- When using a non-ANSI compiler, its code for `--disable-gdbtk` avoids building the `gdb` libraries for the GUI, `libgui`, and automatically turns on `--disable-gdbtk`. This allows use of such non-ANSI compliant compilers.
- There is now support for DWARF2 object file format.
- Some improvements were made for Solaris thread debugging.

Utilities issues

The following documentation concerns enhancement issues for GNUPro utilities.

- There are two new binary utilities.
  - `addr2line`
    Converts addresses into file names and line numbers; see `addr2line` in *The GNU Binary Utilities* in ***GNUPro Utilities***.
  - `windres`
    Manipulates Windows resources; see `windres` in *The GNU Binary Utilities* in ***GNUPro Utilities***.
- Of the hosts we currently support, we do not support the assembler (`gas`) for the `alpha-dec-osf` or `mips-sgi-irix6` configurations. These are not normally configured, but the sources are included with every release.
- Of the hosts we currently support, we do not support the linker (`ld`) for the `alpha-dec-osf`, `hppa-hp-hpux` or `mips-sgi-irix` configurations. These are not normally configured, but the sources are included with every release.
- DejaGnu is a framework for testing other programs. Its purpose is to provide a single front end for all tests and several of the following advantages for testing:
  - The flexibility and consistency of writing tests for any program.
  - Providing a layer of abstraction in order to write tests that port to any host or target where a program must test. For instance, a test for `gdb` can run (from any Unix based host) on any target that DejaGnu supports. DejaGnu runs tests on several single board computers, whose operating software ranges from a boot monitor to a full Unix-like realtime operating system.
  - All tests have the same output format, making it easy to integrate testing into other development, to parse by other filtering scripts, and to be readable.

  DejaGnu is written in `expect`, which in turn uses the Tcl command language.

  Running tests requires two things: the testing framework, and the testsuites themselves. Tests are usually written in `expect`, using Tcl, but you can also use a Tcl script to run a test suite not based on `expect`. `expect` script filenames conventionally use `.exp` as a suffix; for example, the main implementation of the DejaGnu test driver is in the file, `runtest.exp`.

3

Issues from previous releases

The following documentation discusses issues from previous releases. These issues can provide clues that may help if you run into problems with the current release.

General issues with the tools

The following discussions address some general issues that apply to the GNUPro tools for previous releases, affecting many different environments and usage requirements.

- No tapes or floppy disks shipped by default. The installation media is now CD format.
- None of the libraries were thread-safe.
- Some tools were not available for the following platform environments.
  - GNUPro Toolkit did not include a linker (`ld`) for the DEC ***Alpha*** running ***Digital Unix (***formerly OSF/1). The native linker was the default.
  - GNUPro Toolkit required the SGI ***Irix*** operating system's C library and include files in a native configuration. The SGI Irix operating system does not contain these files by default, but they are included in a separate developer's package. You cannot use the GNUPro Toolkit without this package.
  - GNU `ld` was not included for the HP***9000/700 native*** in the native configuration.
- If you ran **`dejagnu`** in an Emacs shell buffer, `expect` generated incorrect results for `pass-fail`.
- Finding a shared library at run-time was a problem with the (default) C++ library, `libstdc++`, as well as with `libg++`.

  Fixes used the following attributes.
  - Added '`/usr/progressive/lib`' to `LD_LIBRARY_PATH` environment variable.
  - Used the following appropriate option when linking:

    For Solaris:
    `...-R/usr/progressive/lib...`

    For DEC Unix/OSF1 and Irix 5:
    `...-Wl, -rpath,/usr/progressive/lib...`
  - Removed or renamed completely the shared libraries under the installation directory ('`/usr/progressive/lib/libg++.s*`' and '`/usr/progressive/lib/libstdc++.s*`').
  - Linked with `-static` to avoid using any shared libraries.
  - The library directory, '`/usr/progressive/lib`', was different if GNUPro Toolkit was installed in a non-standard location. This directory was thought of as '`$GCC_EXEC_PREFIX/../`' if using '`GCC_EXEC_PREFIX`'.
- After rebuilding from source, in order to report problems, a user reran `install-sid` in order to reset `customer-id` in `send-pr`. Previously, the default

value was the value in the `send-pr` program in a path at the time of configuration.

- The following issues were pertinent to the PowerPC environment.
  - `-mno-fused-madd` was a new option, preventing the compiler from combining a floating point multiply with an addition or subtraction. Generally, the compiler would do this optimization, if it was important to get the exact rounding as specified by the IEEE 754 floating point standard (on the Power and PowerPC machines, the combined multiply and add/subtract instruction does not do a round operation between the multiply and the addition/subtraction).

    The `-mcpu=403` switch did a `-mstrict-align` operation, since the 403 had no support for unaligned memory operations.

    With the previous release, better code was generated when adding or doing logical operations with large constants.

    Two new switches , `-mads` and `–myellowknife`, were added to the `powerpc-eabi` toolchain.

    - The `-mads` switch links in the appropriate libraries for the Motorola ADS target boards (860/821/823).
    - `-myellowknife`, the other switch, links in the appropriate libraries for the Motorola Yellowknife target boards (603e/604e).

    Jump tables, for switch statements, were in the `.rodata` section, and not the `.text` section, so it wasn't necessary to include the text section in the data address mapping.

    `wchar_t` became 4 bytes instead of 2 (and **`"a"`** `L`-strings consisted of 4 byte characters instead of 2).

    The `-mregnames` switch was only passed to the assembler for assembling `.s` and `.S` files (i.e., it was not passed to the assembler when assembling the output of the compiler). This is so that global symbols like "`r0`" in C code wouldn't get confused with register names.
  - PowerPC configuration included a simulator contributed by Andrew Cagney.
  - The following support enhancements were made to both the **`powerpc-aix`** and **`powerpc-eabi`** configurations.
    - Support was added for `-mcpu=505`, `-mcpu=602`, `-mcpu=860`, `-mcpu=821`, and `-mcpu=power2`.
    - The default PowerPC model is now `604`, not `601`.
    - `-mtune=`*`xxx`* was added to select scheduling parameters such as `-mcpu=`*`xxx`*, but not to select use of `cpu` specific instructions.

- If configuring and building the compiler, the switch, `--with-cpu=`*`xxx`*, will allow for selecting the default processor.
- Instruction timings had been improved.
- `__attribute__((longcall))` was added to function attributes so that the function is now always called through a pointer. This allows the function to be located anywhere in program memory.
- The compiler was set to generate correct code for a `nor` instruction combined with a `compare` instruction.

- The following enhancements were made *only* to the **`powerpc-eabi`** support.
  - A new switch, `-msdata` was added to put small static and global items in small data regions. This allows them to be referenced with 1 instruction instead of 2.
  - `long long` was set to pass according to the System V/EABI specifications, (in other words, they always pass in odd/even register pairs, never even/odd).
  - Code was changed for the functions `prolog` and `epilog` so that references beyond the end of the current stack pointer never generate.
  - The assembler added support for generating negative address, `GOT`, PLT, and small data, relative to section start relocations.
  - The linker added support for negative address, `GOT`, and small data, relative to section start relocations. PLT relocations were not handled.
- An enhancement was made to the `powerpc-aix` support. The default for AIX 4.1 was changed to `-mcpu=common`. This replaced the default of the machine on which the compiler was configured (either Power or PowerPC)
- The PowerPC simulator must be built with the `gcc` compiler. It uses several `gcc` extensions, as well as requiring an ISO standard compiler. If you configure your `powerpc-eabi` build without using `gcc`, it will not build the simulator.

- The following issues are pertinent to the `m68k` environments.
  - The `m68k-aout` and `m68k-coff` toolchains now include support for converting executables files into IEEE-695 format. The IEEE-695 format is used by some emulators.

    To convert a file, first link your program into a fully linked executable. Then run the following script.

    ```
    m68k-coff-objcopy -O ieee --debugging file file.x
    ```

    This converts a `m68k-coff` file '*`file`*' into the IEEE-695 file, '*`file.x`*'.

    To use the `m68k-aout` toolchain, `m68k-aout-objcopy` is the corresponding

tool.

- The `--debugging` option directs `objcopy` to translate debugging information into the IEEE-695 format. This option only works if you use the `stabs` debugging format. The `stabs` debugging format is the default for the `m68k-aout` toolchain.

  For the `m68k-coff` toolchain, you must compile your files using the option, `-gstabs+`, rather than a simple `-g`, in order to get `stabs` debugging information.

  A conversion facility is available for other toolchains, but the binaries built by Cygnus do not provide it. To use it, you must rebuild the programs from source, as described in *Rebuilding From Source* in ***GNUPro Advanced Topics***.

  Running the configure script, add '`--enable-targets=CPU-ieee`', where `CPU` is the processor type (such as i960). Cygnus had not tested the convertor for processors other than the `m68k`.

- The Motorola CPU32 and CPU32+ targets are part of the family of 68000 chips, which Cygnus supports. There are a few options to help you compile code for these targets.

  - `gcc` has an option, '`-m68332`', to be used specifically when compiling for the Motorola 68332 board. (`gcc` also has an updated option, `-m68302`. The 68302 technically isn't a CPU32 chip.)
  - It is also possible to configure `gcc` for a target of '`m68332-aout`' or '`m68332-coff`' when rebuilding from source, in which case '`-m68332`' is the default.
  - GNU `as` accepts the following board-specific options: `-mcpu32`, `-m68331`, `-m68332`, `-m68333`, `-m68340`, and `-m68302`.

Contact Cygnus for more information on our support for CPU32 and CPU32+ targets.

C and C++ compiler issues

There were several new warnings for the GNU compiler tools, `gcc` (C compiler) and `g++` (C++ compiler).

- *Member function templates have support.*
  This uses scripts, for instance, like the following example.

```
struct S {
  template <class T> operator T();
  /* ... */
};

template <class T>
S::operator T()
{
  /* ... */
}

main ()
{
  S s;
  int i = s;
  void *p = s;
}
```

- ***Explicit qualification of function templates has support.***
  This uses statments, for instance, like the following example.

```
template <class T, class U> T implicit_cast (U u) { return u;
}
int i = implicit_cast<int>(1.5);
```

  As a result, guiding declarations no longer have support. Function declarations, including friend declarations, do not refer to template instantiations. To restore the old behavior, use `-fguiding-decls` until you fix your code. Broken code looks like the following example's script.

```
template <class T> struct A {
  friend ostream& operator<< (ostream &, const A &);
  /* ... */
};

template <class T> ostream&
operator<< (ostream &o, const A<T>& a)
{
  /* ... */
}

main ()
```

```
{
  A<int> a;
  cout << a;
}
```

- ***Two exception handling mechanisms are used.***

  One, which is supported on all platforms, uses `setjmp` and `longjmp` to unwind stack frames and slow down your code.

  The second mechanism uses `DWARF2`-format data to unwind stack frames, and has no time impact on code that doesn't throw exceptions, but does require that the `unwind info` be written out. The `DWARF2` unwinder is the default on systems where it is supported; `setjmp/longjmp` is the default everywhere else.

  To select between the two, use `-fsjlj-exceptions`, except, where the `DWARF2` unwinder is not supported, the old mechanism, `EH`, will be used. Turn `EH` off with the flag, `-fno-exceptions`.

- Standard usage syntax for the `std` namespace is supported; `std` is treated as an alias for global scope. General namespaces are still not supported.
- The flag, `-Wno-pmf-conversion`, tells the compiler not to warn about converting from a bound member function pointer to function pointer.
- A flag, `-Weffc++`, is used for violations of some of the style guidelines in the "Effective C++" books by Scott Meyers.
- `__FUNCTION__` and `__PRETTY_FUNCTION__` are treated as variables by the parser; previously they were treated as string constants.

  Code like `printf (__FUNCTION__ ": foo")` must be rewritten to be:

  ```
  printf ("%s: foo", __FUNCTION__)
  ```

  This is necessary for templates.

- Local static variables in extern inline functions will be shared between translation units.
- `-fvtable-thunks` is supported for all targets, and is the default for Linux with `glibc 2.x` (also called `libc 6.x`).
- A new flag, `-Wold-style-cast`, can be used to warn if an old-style (C-style) cast is used within a C++ program.
- **`hppa*-*-proelf`** (or a HP PRO target) no longer includes floating point support by default. Therefore, it is no longer necessary to include `-msoft-float` on either the compilation or link line for these targets.
- A public review copy of the December 1996 *Draft* of the *ANSI C++ Standard* is available. For PostScript and PDF (using Adobe Acrobat) versions, see the archives: `ftp://research.att.com/dist/c++std/WP`. For HTML and ASCII versions, use `ftp://ftp.cygnus.com/pub/g++` or

`http://www.cygnus.com/misc//wp`.

- The `overload` resolution code was on by default. The old code could still be selected with `-fno-ansi-overloading`, although this was not supported.

WARNING: By not invoking this option, installation will fail.

The following issues may also be useful to those users who are rebuilding from previous releases of the GNU tools, mostly for the `progressive-96q4` and the `progressive-97r1` releases.

- The overload resolution code has been rewritten to conform to the latest C++ Working Paper.

  Built-in operators are now considered as candidates in operator overload resolution. Function template overloading chooses the more specialized template, and properly handles base classes in type deduction and guiding declarations. For those customers having the `progressive-96q4` release, this code was not on by default and, so, they can use the `-fansi-overloading` flag to turn on the overload resolution code. It is, starting with the `98r1` release, now on by default.
- The GNU C++ driver (`g++`) no longer links with `libg++` by default; it is now functionally identical to the C++ driver.
- RTTI support has been rewritten to work properly and is now on by default.

  This means code that uses virtual functions will have a modest space overhead, and will also depend on the RTTI support library code in `libstdc++`. This dependency is removed. Use the `-fno-rtti` flag to disable RTTI support.
- On ELF systems, duplicate copies of symbols with initialized common linkage like template instantiations, vtables, and extern inlines will now be discarded by the GNU linker, so `–frepo` isn't necessary
- Partial specialization of class templates is now supported.
- Synthesized destructors are no longer made virtual just because the class already has virtual functions; it is only if they override a virtual destructor in a base class. The compiler will warn if this affects your code.
- '`(void *)0`' is no longer considered a null pointer constant; `NULL` in `<stddef.h>` is now defined as `__null`**.** This is a magic constant of type, `(void *)`, normally, or `(size_t)` with `-ansi`.
- The new '`template <>`' specialization syntax is now accepted and ignored.
- The name of a class is now implicitly declared in its own scope; for example, `A::A` refers to `A`.
- The STL code is based on the free SGI version, which is more efficient and conformant than the older HP distribution.

- Default function arguments in templates will not be evaluated (or checked for semantic validity) unless they are needed.
- The `-ftemplate-depth-NN` flag can be used to increase the maximum recursive template instantiation depth, defaulting to 17. If you need to use this flag, the compiler will tell you.
- The internal interface between RTTI-using code and the RTTI support library has changed, so code that uses `dynamic_cast` should be recompiled. The RTTI support library has moved from **`libstdc++`** to **`libgcc`**, so you no longer need to link against **`libstdc++`** for a program that doesn't use the "hosted" library.
- `bool` is now always the same size as another built-in type. Previously, a 64-bit RISC target using a 32-bit ABI would have 32-bit pointers and a 64-bit `bool`. This should not affect any supported platforms.
- `new (nothrow)` is supported.
- Some warnings were added for violation of style guidelines.
- **`g++`** uses an implementation of templates that are minimally parsed when seen and then later expanded. This allows conformant early name binding and instantiation controls, since instantiations no longer go through the parser.

  What you get:

  - Inlining of template functions works without any modifications.
  - Instantiations of class templates and methods defined in the class body are deferred until required (unless `-fexternal-templates` is specified).
  - Nested types in class templates work.
  - Static data member templates work.

  Possible problems:

  - Types and class templates used in templates must be declared first, or the compiler will assume they are not types, and fail.
  - Similarly, tag nested types of template type parameters with `typename`.
  - Syntax errors in templates that are never instantiated will now be diagnosed.
- Synthesized methods are emitted in any translation units that need an out-of-line copy. They are no longer affected by `#pragma interface` or `#pragma implementation`.
- Local classes are supported.
- The warning flag, `-Wsign-compare`, included in `-Wall`, warns about dangerous comparisons of signed and unsigned values; it was previously part of `-W`.
- The flag, `-fno-weak`, disables the use of weak symbols.
- The type directive, `__attribute__`, is supported.

- `-Woverloaded-virtual` now warns if a virtual function in a base class is hidden in a derived class, rather than warning about virtual functions being overloaded (even if all of the inherited signatures are overridden).
- The compiler no longer emits a warning if an ellipsis is used as a function's argument list.
- Exception handling support has been significantly improved, though optimization is still not supported.
- Definition of nested types outside of their containing class is supported. Use the following source code, as an example.

```
struct A {
       struct B;
       B* bp;
};

struct A::B {
       int member;
};
```

- Explicit instantiation of template constructors and destructors is now supported. For example: `template A<int>::A(const A&);`
- All HPPA targets support the `-mspace` option, which is experimental code aimed at reducing the size of a program at the expense of increasing execution time.
- On the HPPA, some classes that do not define a copy constructor will be passed and returned in memory again so that functions returning those types can be inlined.

Debugger issues

The following issues pertain to the GNU **gdb** debugger.

- The `d10v-elf` configuration includes commands to collect and display trace data. Use '`trace`' to enable tracing, '`untrace`' to disable, '`info trace`' to describe the information collected in the trace buffer, and '`tdisassemble`' to display the list of instructions that were executed.
- `gdb` now supports the debugging of overlays. Do '`help overlay`' for more information. Example code may be found in `examples/overlay`.
- Embedded MIPS configurations (such as `mips-elf` and `mips-ecoff`) may use hardware breakpoints and watchpoints when communicating with boards using PMON (using the '`target pmon`' command).
- You may now use `gdb` with the Macraigor Systems "wiggler" and "OCD serial box" devices. The target commands are '`target ocd wiggler lpt1`' (for PC hosts, for the wiggler device, only) and '`target ocd port`' (normal serial transport; `/dev/ttya`… being the pathname for addressing the variable argument for `port`).
- The `powerpc-elf` configuration includes two new target protocols; '`target sds`' is an SDS-compatible protocol that is useful with Motorola's ADS821/860 boards, and '`target dink32`' works with Motorola systems running the DINK32 ROM monitor (such as the embedded Yellowknife).
- The '`set architecture`' command allows for an explicit choice of the processor type being debugged, while '`info architecture`' displays the current target architecture. Most configurations presently support only a single architecture.
- The debugger was modified to properly debug executables that were compiled with the Cfront C++ version 2 compiler.
- The debugger includes support for the Apple Macintosh, as a host only. GDB can be run as either an MPW tool or as a standalone application, and it can debug through the serial port. All of the usual debugger commands are available, but you must supply '`serial`' as the device type to the target command, instead of '`/dev/ttyXX`'. Use `target serial` for input.

  See '`mpw-README`' in the main directory for more information on how to build. The MPW configuration scripts '`*/mpw-config.in`' support only a few targets, and only the `mips-idt-ecoff` target has been completely tested. Both `m68k` and PowerPC Macs are supported.
- Use `target ppcbug` for support for the PowerPC PPCBUG monitor.
- Use `target sh3` for support for the Hitachi SH3 monitor ROM.

- Use the `auto-solib-add` variable to read in symbols from all shared libraries. If the value of `auto-solib-add` is `1`, then symbols from all shared libraries will be read in when the program starts up. This is convenient if you want to reference a symbol in a shared library without having to stop in that library first, such as in setting a breakpoint. The default value is `0`, which improves startup time.

NOTE: The command shared library is always available to load shared library symbols manually.

- Use the command, `dont-repeat`, in user-defined commands to defeat the `auto-repeat` of GDB when an empty command is entered.
- The symbol reader for AIX `gdb` now uses partial symbol tables. This can greatly improve startup time, especially for large executables.
- When printing the type of a variable declared with a typedef, `gdb` uses the `typedef` name if possible instead of the `typedef` definition.
- Performance is improved in MIPS IDT debugging (MIPS targets), both for stepping and for downloads.
- The `remotedelay` option is set by default to `1`. Loading executables can be considerably slower with `remotedelay` set to `1`, but it gets around a loading bug on certain H8/300 boards.

  To see a noticeable speed-up in loading when you're not using an H8/300 board, set `remotedelay` to `0`. Use `set remotedelay 0` as input.

  A common hurdle in cross development is to get the communications set up properly between the target board and the development platform.

  The debugger's '`set remotedebug`' command can help. It was designed to help develop new remote targets; it displays the packets transmitted back and forth between the debugger and the target environment. This command can be helpful in diagnosing communications problems, for example, allowing you to observe packets not getting through or picking up noise on the line.

  The `set remotedebug` command is now consistent among the MIPS remote target; remote targets using the `gdb`-specific protocol; UDI (the AMD debug protocol for the 29k); the 88k `bug` monitor; and Hitachi ROM monitors. You can set it to an integer specifying a protocol-debug level (normally `0` or `1`, but `2` means more protocol information for the MIPS target). See "GDB and remote MIPS boards" in *Debugging with GDB* in ***GNUPro Debugging Tools*** for details.
- If you use the `gcc` option, '`-gstabs+`', `gcc` embeds extended debugging information in COFF object files. The extended debug information is based on the *stabs* debugging format, which was originally used only with the `a.out` object file format; see *The stabs debug format*, in your sources as '`src/gdb/doc/stabs.texinfo`', or contact Cygnus for more information. With

this additional debugging information, you can debug C++ programs with GDB, even on systems that use COFF. You can get better C++ debugging by compiling with '`-gstabs+`' for the following targets: `a29k-amd-udi`, `h8300-hms`, `m68k-coff`, `m88k-coff`, `sh-hms`, and `z8k-coff`.

Assembler issues

The following issues pertain to the GNU assembler, **`gas`**.

- The assembler now has a `-alm` option, which can be used to list macro expansions. The `-alc` option can be used to skip false conditionals in listings.
- The default `objdump` disassembly format has changed. You can get the old format by using `--prefix-addresses`.
- The assembler now supports macros without requiring the assembler's preprocessor, `gasp`.
- `gas` now supports the `-M` or `--mri` option, permitting the assembly of MRI-format assembler files.
- The SunOS assembler is now able to assemble PIC.

Linker issues

The following issues pertain to the GNU linker, `ld`.

- The linker now accepts the `--no-whole-archive` flag, to force it to not include the entire contents of an archive file.
- The `-rpath-link` option has been added for SunOS and ELF systems.
- The COFF linker now automatically combines `struct`, `union`, and `enum` debugging information, so that the information only appears once in the output file. This only applies when using COFF debugging information, as opposed to `stabs`.
- The SunOS linker is now able to create shared libraries.

Rebuilding issues

Details on issues with past releases about rebuilding specific platforms and features are shown in the following discussions. See *Rebuilding From Source* in ***GNUPro Advanced Topics*** for more detailed instructions.

- When rebuilding from source on an AIX platform, on some versions, the `tr` utility in '`/usr/ucb`' has a bug. Make sure the `PATH` for the build will make '`/bin/tr`' or '`/usr/bin/tr`' available ahead of '`/usr/ucb/tr`'. This bug is fixed in AIX 4.1.4.
- There is a reported problem in rebuilding GNUPro Toolkit using AIX 3.2.*x* native tools. (This problem does not crop up if you use `gcc` to rebuild the tools.) On the RS/6000, XLC version 1.3.0.0 miscompiles '`jump.c`'. XLC version 1.3.0.1 or later fixes this problem. You can obtain XLC version 1.3.0.2 by requesting PTF 421749 from IBM. This is not relevant for AIX 4.1.4.
- Use `--with-gnu-as` when configuring MIPS, if you rebuild the entire GNUPro Toolkit from source.

  Top-level configuration files handle the configuration for you automatically. But if you rebuild the compiler *alone* for a MIPS target, we highly recommend that you specify '`--with-gnu-as`' on the command line for `configure`. This avoids an incompatibility between the assembler, **`as`**, and the MIPS assembler. The MIPS assembler does not support debugging directives, and `gcc` uses a special program, `mips-tfile`, to generate them. **`as`** parses the debugging directives directly without `mips-tfile`.

  You should also specify '`--with-stabs`' on the command line to `configure`. This provides better debugging symbols, in particular for C++. If you plan to use the linker, be sure to specify '`--with-gnu-ld`' when you rebuild on any platform for which the linker is available.
- To rebuild the tools from source on a SPARC system running Solaris 2, use either the original Solaris 2 native-development binaries from GNUPro Toolkit or the unbundled compiler sold separately by Sun.

WARNING! There is a program called '`/usr/ucb/cc`' that *you should not use* since it is incompatible with the real compiler which is in '`/opt/SUNWspro/bin/cc`'.

- As in previous releases, you can reconfigure GNUPro Toolkit to support more than one object format. For detailed instructions, see *Rebuilding From Source* in ***GNUPro Advanced Topics***.

  To add support for more object file formats (besides the format appropriate for the configured target), list the additional targets as arguments to the `configure` option, `--enable-targets`, separated by commas.

Use the following input as an example.

```
./configure --enable-targets=m68k-coff,i386-elf,decstation
```

To find out what targets are available, look in the file '`bfd/config.bfd`' in the source distribution. To configure the tools to support all available object formats, use '`--enable-targets=all`' rather than listing individual targets.

- Starting with the 97r1 release, `make` had been altered to support the Win32 host environment better. It now has two major modes of operation:
  - The default mode is to support native Win32 makefiles. In this mode, Makefile rules may use either backslashes or forward slashes as filename directory separators. Makefile rules may invoke "del", "copy", or other Win32 shell builtins since `make` will now invoke a native Win32 subshell when necessary. Note that some programs like del do not support filenames containing forward slash directory separators so you probably want to use backslashes if you intend to use those types of programs. `make` does not automatically convert forward slashes to backslashes because doing so would break the use of the forward slash as the option specifier for some Win32 programs.

    For this mode to work correctly, the correct Win32 subshell must be in your path (cmd.exe under Windows NT or command.com under Windows 95). When writing your makefiles, avoid the use of backslashes as end of line continuation characters because Win32 shell commands like "del" will interpret that backslash as a reference to the root partition of the current drive! Finally, command line lengths are limited to what's supported by the Win32 subshell: 255 characters under cmd.exe, 127 characters under command.com.
  - An optional Unix compatibility mode to support Unix-style Makefiles.

    There are two ways to tell `make` to operate in this mode: you can either set the environment variable, `MAKE_MODE`, to "`unix`" before running `make` or you can invoke `make` as "`make --unix`" (in this mode, backslashes retain the usual Unix semantics and cannot be used in filenames as directory separators; however, Win32 paths without backslashes are still supported). Use of Bourne shell built-ins is permitted but Win32 shell built-ins like del and copy are unavailable. You must provide a working Bourne shell for this mode to work correctly. It should be named "sh.exe" and needs to be in your path so `make` can find it.
- After rebuilding from source, it is now necessary to rerun `install-sid` in order to reset your `customer-id` in `send-pr`. Previously, the default value was the value in the `send-pr` program in your path at the time of configuration.

4

Using GNU tools on embedded systems

The following seven GNUPro Toolkit tools can be run on embedded targets.

- `gcc`, the GNUPro Toolkit compiler (see “gcc, the GNU compiler” on page 46)
- `cpp`, the GNU C preprocessor (see “cpp, the GNU preprocessor” on page 47)
- `gas`, the GNUPro Toolkit assembler (see “gas, the GNU assembler” on page 47)
- `ld`, the GNUPro Toolkit linker (see “ld, the GNU linker” on page 47)
- `binutils`, the GNUPro Toolkit directory of utilities (see “binutils, the GNU binary utilities” on page 48)
- `gdb`, the GNUPro Toolkit debugger (see “gdb, the debugging tool” on page 49)
- `libgloss`, the support library for embedded targets and `newlib`, the C library developed by Cygnus (see “libgloss, newlib and libstd++, the GNU libraries” on page 50)

See the following documentation for more discussion on using the GNU tools.

- “Invoking the GNU Tools” on page 46
- “crt0, the main startup file” on page 51
- “The linker script” on page 55
- “I/O support code” on page 58
- “Memory support” on page 59
- “Miscellaneous support routines” on page 60

Invoking the GNU Tools

`gcc` invokes all the required GNU passes for you with the following utilities.

- `cpp`
  The preprocessor which processes all the header files and macros that your target requires.
- `gcc`
  The compiler which produces assembly language code from the processed C files. For more information, see *Using GNU CC* in ***GNUPro Compiler Tools*** (use the following Web site location for links to documentation).
  `http://www.cygnus.com/pubs/gnupro/`
- `gas`
  The assembler which produces binary code from the assembly language code and puts it in an object file.
- `ld`
  The linker which binds the code to addresses, links the startup file and libraries to the object file, and produces the executable binary image.

There are several machine-independent compiler switches, among which are, notably, `-fno-exceptions` (for C++), `-fritti` (for C++) and `-T` (for linking).

You have four implicit file extensions: `.c`, `.C`, `.s`, and `.S`. For more information, see *Using GNU CC* in ***GNUPro Compiler Tools*** (use the following Web site location for links to documentation).

`http://www.cygnus.com/pubs/gnupro/`

gcc, the GNU compiler

When you compile C or C++ programs with gnu C, the compiler quietly inserts a call at the beginning of `main` to a `gcc` support subroutine called `__main`. Normally this is invisible—you may run into it if you want to avoid linking to the standard libraries, by specifying the compiler option, `-nostdlib`. Include `-lgcc` at the end of your compiler command line to resolve this reference. This links with the compiler support library `libgcc.a`. Putting it at the end of your command line ensures that you have a chance to link first with any of your own special libraries.

`__main` is the initialization routine for C++ constructors. Because GNU C is designed to interoperate with GNU C++, even C programs must have this call: otherwise C++ object files linked with a C main might fail. For more information on `gcc`, see *Using GNU CC* in ***GNUPro Compiler Tools***(use the following Web site location for links to documentation).

`http://www.cygnus.com/pubs/gnupro/`

cpp, the GNU preprocessor

cpp merges in the #include files, expands all macros definitions, and processes the #ifdef sections. To see the output of cpp, invoke gcc with the -E option, and the preprocessed file will be printed on stdout.

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 calling the assembler directly.

- Name the source file using the extension .S (capitalized) rather than .s. gcc recognizes files with this extension as assembly language requiring C-style preprocessing.
- Specify the "source language" explicitly for this situation, using the gcc option, -xassembler-with-cpp.

For more information on cpp, see *The C Preprocessor* in ***GNUPro Compiler Tools***(use the following Web site location for links to documentation).

http://www.cygnus.com/pubs/gnupro/

gas, the GNU assembler

gas can be used as either a compiler pass or a source-level assembler.

When used as a source-level assembler, it has a companion assembly language preprocessor called gasp. gasp has a syntax similar to most other assembly language macro packages.

gas emits a relocatable object file from the assembly language source code. The object file contains the binary code and the debug symbols.

For more information on gas, see *Using AS* in ***GNUPro Utilities*** (use the following Web site location for links to documentation).

http://www.cygnus.com/pubs/gnupro/

ld, the GNU linker

ld resolves the code addresses and debug symbols, links the startup code and additional libraries to the binary code, and produces an executable binary image.

For more information on ld, see *Using LD* in ***GNUPro Utilities*** (use the following Web site location for links to documentation).

http://www.cygnus.com/pubs/gnupro/

.coff for object file formats

.coff is the main object file format when using the tools on embedded target systems. For more information on object files and object file formats, see *The GNU Binary Utilities* in ***GNUPro Utilities***(use the following Web site location for links to documentation).

```
http://www.cygnus.com/pubs/gnupro/
```

binutils, the GNU binary utilities

The following are the binary utilities, although they are not included on all hosts: ar, nm, objcopy, objdump, ranlib, size, strings, and strip.

For more information on binutils, see *The GNU Binary Utilities* in ***GNUPro Utilities*** (use the following Web site location for links to documentation).

```
http://www.cygnus.com/pubs/gnupro/
```

The most important of these utilities are objcopy and objdump.

objcopy

A few ROM monitors, such as a.out, load executable binary images, and, consequently, most load an S-record. An S-record is a printable ASCII representation of an executable binary image.

S-records are suitable both for building ROM images for standalone boards and for downloading images to embedded systems. Use the following example's input for this process.

```
objcopy -O srec infile outfile
```

infile in the previous example's input is the executable binary filename, and *outfile* is the filename for the S-record.

Most PROM burners also read S-records or some similar format. Use the following example's input to get a list of supported object file types for your architecture.

```
objdump -i
```

For more information on S-records, see the discussions for FORMAT *output-format* in the documentation for "MRI Comaptible Files" and the discussion for "BFD" in *Using LD* in ***GNUPro Utilities***. For more discussion of making an executable binary image, see "objcopy" in *The GNU Binary Utilities* in ***GNUPro Utilities*** (use the following Web site location for links to documentation).

```
http://www.cygnus.com/pubs/gnupro/
```

`objdump`
: `objdump` displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

  When specifying archives, `objdump` shows information on each of the member object files. `objfile...` designates the object files to be examined.

A few of the more useful options for commands are: `-d`, `--disassemble` and `--prefix-addresses`.

`-d`
`--disassemble`
: Displays the assembler mnemonics for the machine instructions from *`objfile`*. This option only disassembles those sections that are expected to contain instructions.

`--prefix-addresses`
: For disassembling, prints the complete address on each line, starting each output line with the address it's disassembling. This is the older disassembly format. Otherwise, you only get raw opcodes.

gdb, the debugging tool

To run `gdb` on an embedded execution target, use a `gdb` backend with the `gdb` standard remote protocol or a similar protocol. The most common are the following two types of `gdb` backend.

- A `gdb` stub
  This is an exception handler for breakpoints, and it must be linked to your application. `gdb` stubs use the `gdb` standard remote protocol.
- An existing ROM monitor used as a `gdb` backend
  The most common approach means using the following processes.
  - With a similar protocol to the `gdb` standard remote protocol.
  - With an interface that uses the ROM monitor directly. With such an interface, `gdb` only formats and parses commands.

For more information on debugging tools, see *Debugging with GDB* in ***GNUPro Debugging Tools***(use the following Web site location for links to documentation).

`http://www.cygnus.com/pubs/gnupro/`

Useful debugging routines

The following routines are always useful for debugging a project in progress.

- `print()`
  Runs standalone in `libgloss` with no `newlib` support. Many times `print()` works when there are problems that make `printf()` cause an exception.
- `outbyte()`
  Used for low-level debugging.
- `putnum()`
  Prints out values in hex so they are easier to read.

`libgloss`, `newlib` and `libstd++`, the GNU libraries

GNUPro Toolkit has three libraries: `libgloss`, `newlib` and `libstd++` (use the following Web site location for links to documentation).

`http://www.cygnus.com/pubs/gnupro/`

`libgloss`

`libgloss`, the library for GNU Low-level OS Support, contains the startup code, the I/O support for `gcc` and `newlib` (the C library), and the target board support packages that you need to port the GNU tools to an embedded execution target.

The C library used throughout this manual is `newlib`, however `libgloss` could easily be made to support other C libraries. Because `libgloss` resides in its own tree, it's able to run standalone, allowing it to support GDB's remote debugging and to be included in other GNU tools.

Several functions that are essential to `gcc` reside in `libgloss`. These include the following functions.

- `crt0`, the main startup script (see "crt0, the main startup file" on page 51)
- ld, the linker script (see "The linker script" on page 55)
- I/O support code (see "I/O support code" on page 58)

`newlib`

The Cygnus libraries, including the C library, `libc`, and the C math library, `libm`.

`libstd++`

The C++ library in development by Cygnus.

crt0, the main startup file

The crt0 (C RunTime 0) file contains the initial startup code.

Cygnus provides a crt0 file, although you may want to write your own crt0 file for each target. The crt0 file is usually written in assembler as 'crt0.s', and its object gets linked in first and bootstraps the rest of your application. The crt0 file defines a special symbol like _start, which is both the default base address for the application and the first symbol in the executable binary image.

If you plan to use any routines from the standard C library, you'll also need to implement the functions on which libgloss depends. The crt0 file accomplishes the following results. See "I/O support code" on page 58.

- **crt0 *initializes everything in your program that needs it.***
  This initialization section varies. If you are developing an application that gets downloaded to a ROM monitor, there is usually no need for special initialization because the ROM monitor handles it for you. If you plan to burn your code in a ROM, the crt0 file typically does all of the hardware initialization required to run an application. This can include things like initializing serial ports and running a memory check; however, results vary depending on your hardware.

  The following is a typical basic initialization of crt0.s.

  1. Set up concatenation macros.

     ```
     #define CONCAT1(a, b) CONCAT2(a, b)
     #define CONCAT2(a, b) a ## b
     ```

     Later, you'll use these macros.

  2. Set up label macros, using the following example's input.

     ```
     #ifndef __USER_LABEL_PREFIX__
     #define __USER_LABEL_PREFIX__ _
     #endif
     #define SYM(x) CONCAT1 (__USER_LABEL_PREFIX__, x)
     ```

     These macros make the code portable between coff and a.out. coff always has an __ (underline) prepended to the front of its global symbol names. a.out has none.

  3. Set up register names (with the right prefix), using the following example's input.

     ```
     #ifndef __REGISTER_PREFIX__
     #define __REGISTER_PREFIX__
     #endif
     /* Use the right prefix for registers. */
     #define REG(x) CONCAT1 (__REGISTER_PREFIX__, x)
     ```

```
#define d0 REG (d0)
#define d1 REG (d1)
#define d2 REG (d2)
#define d3 REG (d3)
#define d4 REG (d4)
#define d5 REG (d5)
#define d6 REG (d6)
#define d7 REG (d7)
#define a0 REG (a0)
#define a1 REG (a1)
#define a2 REG (a2)
#define a3 REG (a3)
#define a4 REG (a4)
#define a5 REG (a5)
#define a6 REG (a6)
#define fp REG (fp)
#define sp REG (sp)
```

Register names are for portability between assemblers. Some register names have a `%` or `$` prepended to them.

4. Set up space for the stack and grab a chunk of memory.

```
.set stack_size, 0x2000 .
comm SYM (stack), stack_size
```

This can also be done in the linker script, although it typically gets done at this point.

5. Define an empty space for the environment, using the following example's input.

```
      .data
      .align 2
SYM (environ):
      .long 0
```

This is bogus on almost any ROM monitor, although it's best generally set up as a valid address, then passing the address to `main()`. This way, if an application checks for an empty environment, it finds one.

6. Set up a few global symbols that get used elsewhere.

```
.align 2
.text
.global SYM (stack)
.global SYM (main)
.global SYM (exit)
.global __bss_start
```

This really should be `__bss_start`, not `SYM (__bss_start`.

`__bss_start` needs to be declared this way because its value is set in the linker script.

7. Set up the global symbol, `start`, for the linker to use as the default address for the `.text` section. This helps your program run.

```
SYM (start):
link a6, #-8
moveal #SYM (stack) + stack_size, sp
```

- **`crt0` *zeroes the* `.bss` *section***

  Make sure the `.bss` section is cleared for uninitialized data, using the following example's input. All of the addresses in the `.bss` section need to be initialized to zero so programs that forget to check new variables' default values will get predictable results.

```
moveal #__bss_start, a0
moveal #SYM (end), a1
1:
movel #0, (a0)
leal 4(a0), a0
cmpal a0, a1
bne 1b
```

  Applications can get wild side effects from the `.bss` section being left uncleared, and it can cause particular problems with some implementations of `malloc()`.

- **`crt0` *calls* `main()`**

  If your ROM monitor supports it, set up `argc` and `argv` for command line arguments and an environment pointer before the call to `main()`, using the following example's input.

  For `g++`, the code generator inserts a branch to `__main` at the top of your `main()` routine. `g++` uses `__main` to initialize its internal tables and then returns control to your `main()` routine.

  For `crt0` to call your `main()` routine, use the following example's input. First, set up the environment pointer and jump to `main()`. Call the main routine from the application to get it going, using the following example's input with `main (argc, argv, environ)`, using `argv` as a pointer to NULL.

```
            pea 0
            pea SYM (environ)
            pea sp@(4)
            pea 0
jsr SYM (main)
movel d0, sp@-4
```

- **`crt0` *calls* `(exit)`**

  After `main()` has run, the `crt0` file cleans things up and returns control of the hardware from the application. On some hardware there is nothing to return to—especially if your program is in ROM— and if that's the case, you need to do a hardware reset or branch back to the original start address.

If you're using a ROM monitor, you can usually call a user trap to make the ROM take over. Pick a safe vector with no sides effects. Some ROM's have a built-in trap handler just for this case.

Implementing `(exit)` here is easy.. First, with `_exit`, exit from the application. Normally, this causes a user trap to return to the ROM monitor for another run. Then, using the following example's input, you proceed with the call.

```
SYM (exit):
trap #0
```

Both `rom68k` and `bug` can handle a user-caused exception of `0` with no side effects. Although the `bug` monitor has a user-caused trap that returns control to the ROM monitor, the `bug` monitor is more portable.

The linker script

The linker script accomplishes the following processes to result.

- Sets up the memory map for the application.

  When your application is loaded into memory, it allocates some RAM, some disk space for I/O, and some registers. The linker script makes a memory map of this memory allocation which is important to embedded systems because, having no OS, you have the ability then to manage the behavior of the chip.
- For `g++`, sets up the constructor and destructor tables.

  The actual section names vary depending on your object file format. For `a.out` and `coff`, the three main sections are `.text`, `.data` and `.bss`.
- Sets the default values for variables used elsewhere.

  These default variables are used by `sbrk()` and the `crt0` file, typically called by `_bss_start` and `_end`.

There are two ways to ensure the memory map is correct.

- By having the linker create the memory map by using the option, `-Map`.
- By, after linking, using the `nm` utility to check critical addresses like `start`, `bss_end` and `_etext`.

The following is an example of a linker script for an `m68k`-based target board.

1. Use the `STARTUP` command, which loads the file so that it executes first.

```
STARTUP(crt0.o)
```

The `m68k-coff` configuration default does not link in `crt0.o` because it assumes that a developer has `crt0`. This behavior is controlled in the `config` file for each architecture in a macro called `STARTFILE_SPEC`. If `STARTFILE_SPEC` is set to NULL, `gcc` formats its command line and doesn't add `crt0.o`. Any filename can be specified with `STARTUP`, although the default is always `crt0.o`.

If you use only `ld` to link, you control whether or not to link in `crt0.o` on the command line.

If you have multiple `crt0` files, you can leave `STARTUP` out, and link in `crt0.o` in the makefile or use different linker scripts. Sometimes this option is used to initialize floating point values or to add device support.

2. Using `GROUP`, load the specified file.

```
GROUP(-lgcc-liop-lc)
```

In this case, the file is a relocated library that contains the definitions for the low-level functions needed by `libc.a`. The file to load could have also been specified on the command line, but as it's always needed, it might as well be here

as a default.

3. SEARCH_DIR specifies the path in which to look for files.
   ```
   SEARCH_DIR(.)
   ```
4. Using _DYNAMIC, specify whether or not there are shared dynamic libraries. In the following example's case, there are no shared libraries.
   ```
   __DYNAMIC = 0;
   ```
5. Set _stack, the variable for specifying RAM for the ROM monitor.
6. Specify a name for a section that can be referred to later in the script. In the following example's case, it's only a pointer to the beginning of free RAM space with an upper limit at 2M. If the output file exceeds the upper limit, MEMORY produces an error message. First, in this case, we'll set up the memory map of the board's stack for high memory for both the rom68k and mon68k monitors.
   ```
   MEMORY
   {
       ram     :     ORIGIN = 0x10000, LENGTH = 2M
   }
   ```

***Setting up constructor and destructor tables for* g++**

1. Set up the .text section, using the following example's input.
   ```
   SECTIONS
   {
      .text :
      {
         CREATE_OBJECT_SYMBOLS
         *(.text)
         etext = .;
         __CTOR_LIST__ = .;
         LONG((__CTOR_END__ - __CTOR_LIST__) / 4 - 2)
         *(.ctors)
         LONG(0)
         __CTOR_END__ = .;
         __DTOR_LIST__ = .;

         LONG((__DTOR_END__ - __DTOR_LIST__) / 4 - 2)
         *(.dtors)
         LONG(0)
         __DTOR_END__ = .;
         *(.lit)
         *(.shdata) }
      > ram
      .shbss SIZEOF(.text) + ADDR(.text) : {
         *(.shbss)
      }
   ```
   In a coff file, all the actual instructions reside in .text for also setting up the

constructor and destructor tables for g++. Notice that the section description redirects itself to the RAM variable that was set up in Step 5 of the earlier process for the crt0 file, “Set _stack, the variable for specifying RAM for the ROM monitor.” on page 56.

2. Set up the .data section.

```
        .talias : { } > ram
        .data : {
        *(.data)
        CONSTRUCTORS
        _edata = .;
    } > ram
```

In a coff file, this is where all of the initialized data goes. CONSTRUCTORS is a special command used by ld.

Setting default values for variables, _bss_start and _end

Set up the .bss section:

```
        .bss SIZEOF(.data) + ADDR(.data) :
        {
        __bss_start = ALIGN(0x8);
        *(.bss)
        *(COMMON)
           end = ALIGN(0x8);
           _end = ALIGN(0x8);
           __end = ALIGN(0x8);
        }
        .mstack : { } > ram
        .rstack : { } > ram
        .stab . (NOLOAD) :
        {
           [ .stab ]
        }
        .stabstr . (NOLOAD) :
        {
           [ .stabstr ]
        }
    }
```

In a coff file, this is where uninitialized data goes. The default values for _bss_start and _end are set here for use by the crt0 file when it zeros the .bss section.

I/O support code

Most applications use calls to the standard C library. However, when you initially link `libc.a`, several I/O functions are undefined. If you don't plan on doing any I/O, you're OK; otherwise, you need to create two I/O functions: `open()` and `close()`. These don't need to be fully supported unless you have a file system, so they are normally stubbed out, using `kill()`.

`sbrk()` is also a stub, since you can't do process control on an embedded system, only needed by applications that do dynamic memory allocation. It uses the variable, `_end`, which is set in the linker script.

The following routines are also used for optimization.

`-inbyte`
: Returns a single byte from the console.

`-outbyte`
: Used for low-level debugging, takes an argument for `print()` and prints a byte out to the console (typically used for ASCII text).

Memory support

The following routines are for dynamic memory allocation.

`sbrk()`
: The functions, `malloc()`, `calloc()`, and `realloc()` all call `sbrk()` at their lowest levels. `sbrk()` returns a pointer to the last memory address your application used before more memory was allocated.

`caddr_t`
: Defined elsewhere as `char *`.

`RAMSIZE`
: A compile-time option that moves a pointer to heap memory and checks for the upper limit.

Miscellaneous support routines

The following support routines are called by `newlib`, although they don't apply to the embedded environment.

`isatty()`
: Checks for a terminal device.

`kill()`
: Simply exits.

`getpd()`
: Can safely return any value greater than 1, although the value doesn't effect anything in `newlib`.

5

Cross-development environment

Using GNUPro Toolkit in one of the cross-development configurations usually requires some attention to setting up the target environment.

A cross-development configuration can develop software for a different target machine than the development tools themselves (which run on the host)—for example, a SPARCstation can generate and debug code for a Motorola Power PC-based board.

For our tools to work with a target environment (except for real-time operating systems, which provide full operating system support), set up the tools by using the following documentation.

- To set up the C run-time environment, see “The C run-time environment (crt0)” on page 62.
- To create ***stubs***, or minimal versions of operating system subroutines for the C subroutine library, see “System Calls” in *GNUPro C Library* in ***GNUPro Libraries*** (use the following Web site location for links to documentation).
  `http://www.cygnus.com/pubs/gnupro/`
- To understand the connection to the debugger, see “Remote debugging” in *Debugging with GDB* in ***GNUPro Debugging Tools*** (use the following Web site location for links to documentation).
  `http://www.cygnus.com/pubs/gnupro/`

The C run-time environment (crt0)

To link and run C or C++ programs, you need to define a small module (usually written in assembler as 'crt0.s') that makes sure the hardware is initialized for C conventions before calling main.

There are some examples of 'crt0.s' code (along with examples of system call stub code) available in the source code for GNUPro Toolkit.

Look in the following path.

installdir/gnupro-98r1/src/newlib/libc/sys

installdir refers to your installation directory, by default '/usr/cygnus'.

For example, look in '.../sys/h8300hms' for Hitachi H8/300 bare boards, or in '.../sys/sparclite' for the Fujitsu SPARClite board.

More examples are in the following directory.

installdir/gnupro-98r1/src/newlib/stub

To write your own crt0.s module, you need the following information about your target.

- A memory map. What memory is available, and where?
- Which way does the stack grow?
- What output format do you use?

At a minimum, your crt0.s modulemust do the following processes.

- Define the symbol, start (_start in assembler code). Execution begins at this symbol.
- Set up the stack pointer, sp. It is largely up to you to choose where to store your stack within the constraints of your target's memory map. Perhaps the simplest choice is to choose a fixed-size area somewhere in the uninitialized data section (often called 'bss'). Remember that whether you choose the low address or the high address in this area depends on the direction your stack grows.
- Initialize all memory in the uninitialized-data ('bss') section to zero.

  The easiest way to do this is with the help of a linker script (see "Command Language" in *Using LD* in ***GNUPro Utilities***). Use a linker script 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 processes.

- Define an '`_exit`' subroutine. This is the C name; in your assembler code. Use the label, `__exit`, with two leading underbars. Its precise behavior depends on the details of your system, and on your choice. Possibilities include trapping back to the boot monitor, if there is one; or to the loader, if there is no monitor; or even back to the symbol, `start`.
- If your target has no monitor to mediate communications with the debugger, you must set up the hardware exception handler in the '`crt0.s`' module. See "The gdb remote serial protocol" in *Debugging with GDB* in ***GNUPro Debugging Tools*** for details on how to use the gdb 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 described in "System Calls" in *GNUPro C Library* in ***GNUPro Libraries***.

6

Cygnus glossary

The following glossary documentation lists some terms that either often or sometimes require definition. Many may have common usage in the technical community, while some have a lineage with Cygnus engineers who have needed to create names for designating common tools, platforms or processes.

A

a29k

AMD 29000 family of RISC processors.

ABI

Application Binary Interface. The ABI defines how programs should interface with the operating system, and may include specifications of executable format, calling conventions, and so forth.

ADP

Angel Debug Protocol, a protocol used with ARM's newer Angel monitor and their Embedded ICE.

AIX

IBM's version of Unix for RS/6000 and PowerPC. Pronounced *aches*.

Alpha

Name for Digital's family of 64-bit RISC processors.

API

Application Programming Interface, defining how programmers write source code that makes use of a library's or operating system's facilities.

ARC

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

Architecture

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

ARM

Acorn RISC Machine's family of RISC processors. Also, ***Annotated Reference Manual for C++***. Often used to describe a name-mangling style.

Assembler

The tool that produces object files from assembly code.

B

BDM

Background Debugging Mode, referring to the ability of the CPU32 sub-family (68302, 68360, etc.) of m68k and Motorola PowerPC chips to be directly controlled through a special set of pins.

BFD

Binary File Descriptor, 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.

Big-endian

A byte-ordering scheme in which the most significant bytes are at lower addresses. The Motorola 68000 is a big-endian microprocessor.

Bison

The GNU parser generator, a workalike for *yacc*.

boot, bootstrap

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

BSD

Berkeley System Distribution, U.C. Berkeley's version of Unix, originally licensed from AT&T, and later upgraded to all-free code. Formed the basis for SunOS.

BSP

Board Support Package. Exact meaning varies. Typically refers to the low-level code or scripts that build programs running on a particular chip on a particular circuit board. Also refers to the ROM that boots an RTOS onto a specific board.

bug

A problem with software that needs a *patch*.

build

The process of configuring, compiling, and linking a set of tools. Also used as a noun, to denote the results of the process.

Byacc

Berkeley yacc, version in BSD Unix. See also *yacc* and *Bison*.

C

Canadian cross

A cross-compilation in which a program being compiled is a cross compiler for some other host/target pair. Example: building a 486 PC with a Motorola 68k cross compiler on a Sun SPARC station.

CHILL

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

CISC

Complex Instruction Set Computer. This class of machines typically has variable-length instructions with a variety of addressing modes. Examples include x86, m68k, and vax.

COFF

Common Object File Format. This format appeared with Unix SVR3, formerly common for Unix, and still used by some embedded systems. The Microsoft PE format for Windows is based on COFF.

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 assembly language.

CVS

Concurrent Version System, a free source version control system currently used for all sources at Cygnus.

CX/UX

A version of Unix produced by Harris Computer Systems.

CygMon

Cygnus' standard ROM monitor.

Cygnus

The leading provider of single-source, Unix, and NT desktop and cross-platform development tools for 32- and 64-bit microcontrollers. The company strategy is to continue to extend the functionality and performance of its development tools, as well as to deliver innovative software component technologies for embedded systems.

With Roman etymology (for *swan*), named for the constellation, Cygnus is within the plane of our Milky Way galaxy and is 2,500-10,000 light-years away, forming a cross.

The brightest star in Cygnus is Deneb.

cygwin32

Cygnus' Unix emulation library for Windows 95 and NT.

D

D10V

A small VLIW processor developed by Mitsubishi, featuring 32-bit instructions each, with two 15-bit subinstructions.

D30V

A VLIW processor developed by Mitsubishi, and intended for use in video processing applications (camcorders and DVD players). It has 64-bit instructions each, with two 30-bit subinstructions.

dbx

The standard debugger on many Unix systems.

Debug format

The layout of debugging information within an object file format. Debug formats include stabs, COFF, DWARF, and DWARF 2.

Debug protocol

The mechanism by which a debugger examines and controls the program being debugged.

Debugger

A tool that allows programmers to examine and control a program, typically for the purpose of finding errors in the program.

DejaGNU

The regression testing framework used at Cygnus, based on `tcl` and `expect`.

DJGPP

DJ Delorie's DOS port of GNU, using the GO32 DOS extender. It includes all the tools, and runs under DOS, Win95, etc.

DWARF

A debugging format. Versions include DWARF 1, 1.1, 2, and extensions to 2.

E

E7000

An ICE produced by Hitachi for its SH and H8/300 processors.

EABI

Generic term for an ABI adapted for embedded use.

ECOFF

Extended COFF, a format used with MIPS and Alpha processors, both for workstations and embedded uses.

ELF

Extended Linker Format. Appeared with Unix SVR4 and used on many systems, including Solaris/SunOS, Irix, and Linux. Many embedded systems also use ELF.

Executable file

A binary-format file containing machine instructions in a ready-to-run form.

expect

A program that allows scripted control over another program.

F

flex

The GNU lexical analyzer generator.

Foundry

Cygnus IDE, with a Project Manager, Source Code Editor and a Debugger. These are a GUI recompiling/reconfiguring tool (known as `vmake`), a simple editor (known as `jedit`), and GDBtk (from which you can perform extensive analysis while debugging), with a message-passing backplane currently based on ILU.

FreeBSD

A free Unix operating system for PCs.

G

gas

Acronym for the GNU assembler.

gcc

Acronym for the GNU C compiler.

gdb

Acronym for the GNU debugger.

GNU

Recursive acronym for *GNU's Not Unix*. 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, a debugger, and many other programming tools.

GO32

Freeware 32-bit DOS extender. Also the Cygnus name for GNU tools ported to DOS using GO32. See DJGPP.

GUI

Graphical User Interface.

H

h8300

Cygnus name for Hitachi's H8/300 family of microprocessors, including H8/300, H8/300H.

h8500

Cygnus name for Hitachi's H8/500 family of microprocessors.

Host

The computer on which the compiler runs.

hppa

Cygnus name for HP's PA architecture.

HP/UX

HP's version of Unix for m68k and PA architectures. Versions include 7, 8, 9, 10, and 11.

I

i370

Cygnus name for the IBM 370 mainframe computer.

i386

Cygnus name for the 32-bit members of the Intel *x*86 family. Members include 386, 486, Pentium (“i586”), and Pentium Pro (“i686”).

i860

Cygnus name for an obsolete family of Intel RISC processors.

i960

Cygnus name for Intel’s 80960 family of RISC processors.

ICE

In-Circuit Emulator, 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

Integrated Development Environment, a GUI program.

ILU

Inter-Language Unification, a partial CORBA implementation from Xerox PARC, allowing programs to associate.

ISA

Instruction Set Architecture.

Irix

SGI’s version of Unix for MIPS architectures. Versions include 4, 5, and 6.

J

jedit

Foundry’s text editor.

JTAG

Joint Test Advisory Group, 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.

L

Linker

The tool that merges object files and library archives into a single executable file.

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 supplies the linker script explicitly.

Linux

A free Unix operating system for PCs and other kinds of computers. Currently runs on i386, m68k, Alpha, PowerPC, MIPS, and SPARC architectures.

Little-endian

A byte-ordering scheme in which the most significant bytes are at higher addresses. The Intel *x*86 family is all little-endian.

LynxOS

A Unix-like realtime operating system developed by Lynx Real-Time Systems.

M

m68k

Cygnus name for Motorola's 68000 family of microprocessors. Depending on context, the abbreviation may include the CPU32 and ColdFire families as well. Members include 68000, 68020, 68030, 68040, 68060, 68302, 68332, 5200.

m88k

Cygnus name for Motorola's 88000 family of RISC microprocessors, now discontinued.

Mach

An operating system developed at Carnegie-Mellon.

mangling, name mangling

The process by which C++ types and classes are turned into symbols in object files that are compatible with other languages.

mingw32

Minimal gnu-win32, a configuration of the gnu-win32 tools that avoids the Unix emulation of cygwin32.

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*.

mips

Cygnus name for the MIPS family of RISC processors. Members include R2000, R3000, R4000, R5000, R8000, R10000, and TinyRISC. There are many vendors of MIPS parts, each using a distinct naming scheme, such as VR4*xxx* for NEC, and TX39*xx* for Toshiba parts.

MON960

Intel's ROM monitor for their i960 processor.

multilib

A collection of libraries built with different GNU compiler options. This ensures that a program using `-msoft-float` (using software floating point), will link with libraries built using the same option.

N

NetBSD

A free Unix operating system for PCs and other kinds of computers. Currently runs on i386, m68k, Alpha, PowerPC, MIPS, and SPARC processors.

NINDY

An obsolete ROM monitor for the i960.

NRE

Non-Recurring Engineering, typically used to refer to one-time-only development, such as retargeting to a new architecture or adding a feature.

ns32k

Cygnus name for the National Semiconductor 32000 family of processors.

O

Object file

A binary-format file containing machine instructions and possibly symbolic relocation information. Typically produced by an assembler.

Object file format

The layout of object files and executable files. Common formats include a.out, COFF, and ELF.

OS/9, OS/9000

A realtime operating system from Microware.

OSF/1

The Open Software Foundation's version of Unix, used in Digital's Alpha machines.

P

PA

Name for Hewlett-Packard's family of *Precision Architecture* processors.

patch

A change in source code to correct or enhance processes.

PE

Portable Executable. This is Microsoft's object file format for Windows 95 and NT. It is basically COFF with additional header information.

PPC

PowerPC family of RISC processors, designed jointly by IBM and Motorola. Members include 601, 604, 401, 403, 801, 860.

PPC Bug

ROM monitor from Motorola.

pSOS

A realtime operating system from ISI.

ptrace

The Unix system call, traditionally used by debuggers to control other Unix processes. `ptrace` arguments may include commands to read/write registers, single-step, etc.

R

RDI

Remote debugging library, used by ARM.

RDP

Remote Debugging Protocol, a protocol used with ARM's Demon monitor.

remote target

See *target*.

RISC

Reduced Instruction Set Computer. Machines typically have fixed-length instructions, limited addressing modes, many registers, and visible pipelines.

Examples include MIPS, ARM, SH, PowerPC.

RS6000

IBM's RS/6000 family of RISC processors. Depending on context, this term may also include PowerPC systems.

RTEMS

A real-time operating system.

RTOS

Real-Time Operating System.

S

SCO

The *Santa Cruz Operation*, a vendor of *SVR3* Unix and more recently Unixware for PCs.

SDS

A company that makes embedded tools, primarily debuggers for Motorola chips.

sh

Cygnus name for the Hitachi Super-H family of RISC microprocessors. ISAs include SH-1, SH-2, SH-3, SH-3e, SH-DSP, and SH-4; within each ISA, parts have numbers like SH7032 or SH7780.

Solaris

Sun's current version of Unix, superseding SunOS. Based on *SVR4* Unix. Sun officially calls it *SunOS 5.x*, with versions including 2.0-2.6 (or, as Sun refers to them, 5.0-5.6).

sparc

Cygnus name for the family of RISC processors based on Sun's SPARC architecture. Members include SPARClite, SPARClet, UltraSPARC, v7, v8, v9.

SPARClet

An embedded SPARC processor from TSqware (formerly Matra).

SPARClite

An embedded SPARC processor family from Fujitsu. Members include 86930, 931, 932, 933, 934, and 936.

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.

stabs

A debug format originally 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.

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.

sun4

Informal name for a SPARC workstation running SunOS 4.*x*.

SunOS

Sun's former version of Unix, derived from BSD 4.3. Versions include 2, 3, 4.0, 4.1, 4.1.3, 4.1.4. Sun currently refers to Solaris 2.x versions as SunOS 5.*x*.

SVID

System V Interface Definition, for defining interfaces and partitioning components within System V environments.

SVR3

System V Release 3, the third version of Unix for the AT&T 3B2.

SVR4

Acronym for *System V Release 4*, the fourth version of Unix for the AT&T 3B2. Currently owned by *SCO*, after being owned by Novell.

System V

A Unix system for porting source code to build binary operating system products.

T

Target

The computer for which the compiler generates code. Used both to refer to an actual physical device, and to the class of devices.

Toolchain

Informal term for the collection of programs that make up a complete set of cross-compilation tools. Typically consists of the following example's sequence:

```
compiler->assembler->archive     r->linker->debugger.
```

Three-way cross

See *Canadian cross*.

thumb

Cygnus name for the 16-bit instruction frontend available with some ARM processors.

Triple cross

See *Canadian cross*.

U

UDI

Universal Debug Interface, a debugging protocol used only by AMD, and only for the `a29k` architecture.

Unix

Unix operating system.

Unixware

Name for the version of Unix based on *SVR4*, produced by Novell.

V

vax

Digital's popular CISC minicomputer of the 1980s.

VFS

Virtual File System architecture.

VxWorks

A real-time operating system from Wind River Systems.

X

x86

Cygnus name for the Intel 8086 architecture family.

XCOFF

eXtended COFF, IBM's object file format for RS/6000 and PowerPC systems.

XENIX

Microsoft version of *SVR4*.

xor-endian

A horrible way of implementing a big-endian ISA. See *MIPS* and *PowerPC* for example members.

Y

yacc

GNU parser generator.

Z

z8k

Cygnus name for the Z8000, a long-obsolete 16-bit descendent of the Z80 8-bit microprocessor.

GNUPRO™ TOOLKIT

Installation

CYGNUS

This documentation has been prepared by Cygnus Technical Publications; contact the Cygnus Technical Publications staff: **doc@cygnus.com**.

1

Installing GNUPro Toolkit

The following documentation describes how to install your GNUPro Toolkit software on Unix and Win32 systems.

- To install for Unix platforms, see “Installing on Unix Systems from CD” on page 84.
- To install for Windows platforms, see “Installing on Win 95/NT systems from CD” on page 93.

For information about rebuilding the tools for other target environments, see “Rebuilding GNUPro Toolkit” on page 94.

Installing on Unix Systems from CD

Use the following procedures for full installation from CD for Unix systems. In the following procedure examples, the system prompt appears as ‘ `%`’.

1. ***Mount the CD.***

   The procedures for mounting a CD depend on your system type. The following discussion details some examples of mount commands for each host.

   The device used will depend on your system configuration (defaults where appropriate are used in the examples).

   Consult your system administrator if you need assistance.

NOTE: In the following installation examples, substitute `/cdrom/gnupro_98r1` for the directory in which you’ll mount the tools, */mnt*.

- ***SPARC Solaris 2.x***

  If you are running the volume manager, the CD should automatically be mounted as `/cdrom/gnupro_98r1` and will not require root access.

  If you are not running the volume manager, you will need to mount the CD manually with the following command.

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

- ***SPARC SunOS 4.1.x***

  ```
  % mount -t hsfs -o ro /dev/sr0 /mnt
  ```

- ***HP-UX 9.x/10.x***

  ```
  % mount -t cdfs -o ro /dev/dsk/c201d2s0 /mnt
  ```

- ***SGI IRIX 5.x/6.x***

  ```
  % mount -t iso9660 -o ro /dev/scsi/sc0d7l0 /mnt
  ```

- ***AIX 3.2.5/4.1.x***

  ```
  % mount -t cdrfs -o ro /dev/cd0 /mnt
  ```

- ***Digital Unix 3.2X/4.0***

  ```
  % mount -t cdfs -o ro /dev/rz4c /mnt
  ```

2. ***Install the tools in a directory that has writable access permissions.***

   Make sure you can write in ‘`/usr/cygnus`’ using the following input.

   ```
   % su root
   ```
   (enter root password)
   ```
   % mkdir /usr/cygnus
   ```
   (ignore “`File exists`*” error if any)*
   ```
   % chmod 777 /usr/cygnus
   % exit
   ```
   (root access not needed beyond this)

3. ***Run the*** `Install` ***script to set up a target machine.***

   Using the following command, substitute the target's name for *`target`* and substitute `/cdrom/gnupro_98r1` for *`/mnt`*.

   ```
   % /mnt/target/Install --tape=/mnt/target/target.tar.Z binaries
   ```

   `Install` displays messages about its activity, ending with the following output.

   ```
   Done.
   ```

4. ***Install the source code.***

   ```
   % cd /usr/cygnus/gnupro-98r1
   % uncompress < /mnt/src.tar.Z | tar xpf -
   ```

5. ***Install the HTML documentation.***

   ```
   % cd /usr/cygnus/gnupro-98r1
   % uncompress < /mnt/doc.tar.Z | tar xpf -
   ```

6. ***Build symbolic links to make executable paths easy to negotiate.***

   You may need root access to put the link in '`/usr`' folder. '`host`' in the fourth line's instruction stands for the host machine you're using.

   ```
   % cd /usr/cygnus
   % ln -s gnupro-98r1 gnupro
   % su root
   % ln -s /usr/cygnus/gnupro-98r1/H-host /usr/gnupro
   ```

7. ***Enable the problem reporting utility.***

   Using your Cygnus customer ID (see cover letter), enable the electronic mail tool for reporting problems. With the following commands, enable the `send-pr` tool..

   ```
   % /usr/gnupro-98r1/bin/install-sid customer-ID
   ```

8. ***Remove public write access from*** '`/usr/cygnus`'.

   See your system administrator for the correct permissions at your site.
   You're done! Now, anyone with '`/usr/gnupro-98r1/bin`' in their `PATH` can use this GNUPro Toolkit package.

Installation information for Unix

The following documentation describes host-specific information and installation standards for Unix systems.

Platform names

Your package contains a label that indicates the host (and target, if applicable) with which to configure the binaries. The specifications used for hosts and targets in the `configure` script are based on a three-part naming scheme, though the scheme is slightly different between hosts and targets.

```
architecture-vendor-operating system
```

Host names

Table 13 shows the usage of canonical names for referring to the corresponding host platforms that Cygnus supports. For any questions about compatibility, contact Cygnus (see “How to contact Cygnus” on page v).

Table 13: Naming hosts

Canonical name	***Platform***
`alpha-dec-osf3.2C`	DEC Alpha Digital UNIX v3.2C
`alpha-dec-osf4.0`	DEC Alpha Digital UNIX v4.0
`hppa1.1-hp-hpux10`	HP 9000/700, HP-UX B.10.01
`hppa1.1-hp-hpux10.20`	HP 9000/700, HP-UX B.10.20
`i386-cygwin32`	Windows NT-sp3/95-osr2
`i386-pc-linux-gnu`	Intel PC, Linux RedHat 5.0
`mips-sgi-irix5`	SGI Irix 5.3
`mips-sgi-irix6`	SGI Irix 6.2
`powerpc-ibm-aix4.1`	IBM PowerPC, AIX 4.1.4
`powerpc-ibm-aix4.2`	IBM PowerPC, AIX 4.2.1
`rs6000-ibm-aix4.1`	IBM PowerPC, AIX4.1.4
`sparc-sun-solaris2.5`	SPARCstation, Solaris 2.5.1
`sparc-sun-solaris2.6`	SPARCstation, Solaris 2.6
`sparc-sun-sunos4.1`	SPARCstation, SunOS 4.1.4

Target names

The following tables list some of the more common targets supported by Cygnus. Not all targets have support on every host. See also “Native configurations support” on page 13 and “Embedded cross-configuration support” on page 14 for the matrices of

the host/target combinations supported by Cygnus. Also, for more informaton on using particular tools and their targets, see “Using GNU tools on embedded systems” on page 45, “Cross-development environment” on page 61, and ***GNUPro Tools for Embedded Systems***.

WARNING! `configure` can represent a very large number of target name combinations of architecture, vendor, and object formats. Support is not possible for all combinations.

Table 14: H8/300 processor name and output format

`h8300-hms-coff`	COFF object code format

Table 15: i960 processor names and output format

`i960-coff`	MON960 monitor (COFF format)

Table 16: M68K processor names and output formats

`m68k-aout`	a.out object code format
`m68k-coff`	COFF object code format
`m68k-elf`	ELF object code format

Table 17: MIPS processor names and output formats

`mips-elf`	ELF object code format
`mips-sgi-irix6`	ELF object code format

Table 18: PowerPC processor names and output formats

`powerpc-eabi`	ELF object code format (EABI)

Table 19: SH processor name and output format

`sh-hms-coff`	COFF object code format

Table 20: SPARC processor names and output formats

`sparc-aout`	a.out object code format
`sparc-coff`	COFF object code format
`sparc-elf`	ELF object code format

Table 21: *x*86 processor names and output formats

`i386-aout`	a.out object code format
`i386-elf`	ELF object code format

`config.guess` is a shell script that attempts to deduce the host type from which it is called, using system commands like `uname` if they are available.

`config.guess` is remarkably adept at deciphering the proper configuration for your host; if you are building a tree to run on the same host on which you’re building, we recommend not specifying the *`hosttype`* argument.

`config.guess` is called by `configure`; you need never run it by hand, unless you're curious about the output.

Links for easy access and updating

Once you extract the tools from the CD, they are installed into a directory named ‘*installdir*/gnupro-98r1’ where ‘*installdir*’ refers to the full directory pathname within which you’re locating the ‘gnupro-98r1’ files.

For example, if you’ve installed in the default location under /usr/cygnus, use the following input.

```
ln -s /usr/cygnus/gnupro-98r1 /usr/cygnus/gnupro
```

The release number is in the directory name so that you can keep several releases installed at the same time, if you wish. In order to simplify administrative procedures (such as upgrades to future Cygnus releases), we recommend that you establish a symbolic link of ‘/usr/cygnus/gnupro’ to this directory with the following input.

```
ln -s installdir/gnupro-98r1 installdir/gnupro
```

NOTE: The input for the last two examples of input match; *installdir* means the same location as the default that you assign for usr/cygnus.

Directories of ***machine-independent*** files (source code and documentation) are installed directly under ‘gnupro-98r1’. However, to accommodate binaries for multiple hosts in a single directory structure, the binary files for your particular host type are in a subdirectory ‘H-*hosttype*’. (*hosttype* indicates a particular architecture, vendor and operating system; see “Host names” on page 86 for more specific details on names for input.)

This means that one more level of symbolic links is helpful, to allow your users to keep the same execution path defined even if they sometimes use binaries for one machine and sometimes for another. Even if this doesn’t apply now, you might want it in the future; establishing these links now can save your users the trouble of changing all their paths later.

The idea is to build ‘/usr/gnupro/bin’ on each machine so that it points to the appropriate binary subdirectory for each machine—for instance, ‘/usr/cygnus/gnupro-98r1/H-*hosttype*’ where ‘*hosttype*’ corresponds to the host machine.

You may need super-user access again briefly to establish the following link.

```
ln -s /usr/cygnus/gnupro-98r1/H-hosttype /usr/gnupro
```

We recommend building these links as the last step in the installation process.

Running the programs

In order to run the tools in GNUPro Toolkit after you install them, you must first set a few environment variables so your shell can find them.

- At the very least, set your `PATH` variable. See “Setting PATH” on page 91.
- If you install the tools in a non-default location and don’t set the standard symbolic links (see “Links for easy access and updating” on page 89), you must also set the `GCC_EXEC_PREFIX` environment variable. Otherwise, the compiler can’t find its resources. This should not be necessary. See “gcc paths” on page 92.

Setting PATH

To run the tools in this distribution, make sure the PATH environment variable can find the tools. Whether you install in the default location, as in the following example for the input, or in an alternate location, you need to alter your PATH environment variable to point toward the newly installed tools.

If you create the symbolic links we recommend (see “Links for easy access and updating” on page 89), users who want to run the GNUPro Toolkit—regardless of whether they need binaries for your particular host, or for some other platform—can use initialization files settings. The following shows examples with the final linked installation directory as `/usr/gnupro/bin`. If you installed into a different directory, substitute ‘`/usr/gnupro/bin`’ for the actual directory.

- For Bourne-compatible shells (`/bin/sh`, bash, or Korn shell):

```
% PATH=/usr/gnupro/bin:$PATH
% export PATH
```

- For C shell:

```
% set path=(/usr/gnupro/bin $path)
```

gcc paths

You can run the compiler gcc without recompiling, even if you install the distribution in an alternate location, by first setting the environment variable, `GCC_EXEC_PREFIX`. This variable specifies where to find the executables, libraries, and data files used by the compiler. Its value will be different depending on which set of binaries you need to run. For example, if you install the CD distribution under '`/local`' (instead of the default '`/usr/cygnus`'), and you wish to run **`gcc`** as a native compiler, you could set `GCC_EXEC_PREFIX` as follows.

- ***For shells compatible with Bourne shell*** (`/bin/sh`, `bash`, or Korn shell):

```
% GCC_EXEC_PREFIX=/local/gnupro-98r1\
                                /H-hosttype/lib/gcc-lib/
% export GCC_EXEC_PREFIX
```

- ***For C shell***:

```
% setenv GCC_EXEC_PREFIX /local/gnupro-98r1\
                                /H-hosttype/lib/gcc-lib/
```

NOTE: The trailing slash '`/`' is important. The **`gcc`** program uses `GCC_EXEC_PREFIX` as a prefix. If you omit the slash (or make any other mistakes in specifying the prefix), **`gcc`** fails with a message: `installation problem, cannot execute ...`.

Installing on Win 95/NT systems from CD

All releases of GNUPro Toolkit for Win95/NT systems use the following CD installation procedure.

1. ***Mount the CD.***

   Insert CD into CD-ROM drive. The Cygnus installation will start; if it doesn't, open the CD in the Windows Explorer and open setup.exe.

2. ***Determine where to install the tools.***

   Setup prompts for an installation directory; the default is '**`C:\CYGNUS`**'. The tools may be installed in any directory or drive. Make sure the installation directory is where you want to install the files. Check the box for installing souce code files, if that is an option you require. Click Next.

3. ***Ensure adequate disk space for the installation.***

   Setup first checks the installation location to make sure it has enough space before unpacking the tools. Installed GNUPro Toolkit disk usage varies from about 10 to about 50 megabytes, depending on the target (generally, the requirement is 19 megabytes).

4. ***Proceed with the automatic installation process.***

   Setup reads the CD, builds the directories and expands the files.

5. ***Run the programs to use the tools.***

   Click on the Start button on the lower left hand corner of your screen, and choose Programs, where Cygnus programs are found as a shortcut. Choose Cygnus. A bash shell opens.

6. ***Test the installation by building the tools with the '`make`' command.***

   To test the installation, change your working directory to the `demo` subdirectory of the **`CYGNUS`** directory (as in the following example), and type `make`.

   ```
   C:\CYGNUS\> cd C:\CYGNUS\gnupr0-98r1\demo
   C:\CYGNUS\gnupro-98r1\demo> make
   ...
   ```

7. ***Set up the environment for which you'll use the tools.***

   For the Win95 environment, set your working environment to the Autoexec.bat initialization file.

   For the NT environment, as long as you use the Start button in the lower left hand corner of your screen, you can always start the programs as in **Step 5**. If not, choose Settings, and click Control Panel. Choose System. Make the appropriate changes according to your requirements.

Rebuilding GNUPro Toolkit

To rebuild, see *Rebuilding from Source* in the ***GNUPro Advanced Topics*** documentation. The following Web site location provides links to the appropriate topics and procedures.

```
http://www.cygnus.com/pubs/gnupro/
```

2

How to report problems

If you find a problem in this release (also known, more familiarly, as a *bug*), please report it to Cygnus. Use the Cygnus bug-report form to ensure that we can respond to your problem as quickly as possible.

- For UNIX environments, see the following documentation as well as the Web site location for links to discussion of the `send-pr` program described in *Reporting Problems* in ***GNUPro Advanced Topics***.

  `http://www.cygnus.com/pubs/gnupro/`

- Win95/NT users can use a blank copy of the `SENDPR.TXT` file in the installation directory. See "Reporting problems for Win95/NT systems" on page 101. To save time, customize the form beforehand with your Cygnus customer ID (see "Some things that might go wrong" on page 96). The easiest and the most reliable means to report a bug is to copy this form to your email program and email to **`bugs@cygnus.com`**.

  If email is not possible, print the '`SENDPR.TXT`' file; see "Reporting problems for Win95/NT systems" on page 101. Then, fill in the problem report, and FAX it (**+1 408 542 9699**) to Cygnus, after alerting the technical support hotline (**+1 408 542 9601**) to apprise the Technical Support staff of the transmission.

  Email is the most reliable way to submit problem reports and to get speedy resolution to any problems with GNUPro Toolkit.

For issues pertaining to this release, see "Red flag alerts & enhancements" on page 21.

Some things that might go wrong

The following discussion shows some warning messages for some installation problems and solutions to manage the problems.

`No access to /usr/cygnus`

If you can't sign on to an account with access to write in '`/usr`' or '`/usr/cygnus`', use the '`-installdir=`*`directory`*' option to `Install` to specify a different installation directory to which you can write. For example, if all the other installation defaults are right, you can execute something like '`./Install --tape=/dev/`*`tape`*`--installdir=`*`mydir`*'. You'll need either to override default paths for the pre-compiled tools or else to recompile the software. See "Running the programs" on page 90 and "Links for easy access and updating" on page 89 for details.

WARNING! If you can't install in '`/usr/cygnus`' (or link your installation directory to that name), some of the defaults configured into this release of GNUPro Toolkit won't work. See "Running the programs" on page 90 for information on overriding or reconfiguring these defaults.

`No customer ID for send-pr`

Make sure the `send-pr` program for the Problem Reporting Managment System (PRMS) knows your customer ID. Contact Cygnus at **+1 408 542-9601** if you do not know your customer ID. Install your customer ID with `install-sid`, as in the following command (*`customer-ID`* is your assigned customer number).

```
install-sid customer-ID
```

If you install the GNUPro Toolkit into a location other than the default, and don't set up symbolic links pointing to the real installation location, use the '`--install-dir`' option to `install-sid` as in the following directive.

```
install-sid --install-dir=install-dir-prefix customer-ID
```

`install-dir-prefix` points to the top level of the directory into which you install.

After rebuilding from source, rerun `install-sid` in order to reset your `customer-id` in `send-pr`. Previously, the default value was the value in the `send-pr` program in your path at the time of configuration. For information on rebuilding from source, see *Rebuilding from Source* in the ***GNUPro Advanced Topics*** documentation. Go to the following Website location for links to the appropriate information.

```
http://www.cygnus.com/pubs/gnupro/
```

Some error messages from `Install`

The `Install` script checks for errors and inconsistencies in the way its arguments are used. If you get one of these messages and can't use the recommended solution, call the Cygnus support hotline, +1 408 542 9601, or send email to: `bugs@cygnus.com`.

`gcc: cannot exec cpp`

If you've installed the binary distribution of GNUPro Toolkit software in a non-standard location, set the environment variable, `GCC_EXEC_PREFIX`, accordingly. See"Running the programs" on page 90.

`...This is a problem.`
`Cannot cd to installdir`
`I do not know why I cannot create installdir`
`hello.c fails to run`
`test-ioctl.c fails to run`

These errors (the first covers anything that ends in '`This is a problem`') are from paranoia checks; they are issued for situations that other checks should cover, or for unlikely situations that require further diagnosis. If you get one of these messages, contact Cygnus.

Reporting problems for Unix systems

Unix users use a script, `send-pr`, to send Cygnus problem reports (PRs). `send-pr` is configured by the Cygnus Problem Report Management System (PRMS) to update reports of problems to Cygnus technical support staff.

`send-pr` invokes an editor on a problem report form (after filling in some fields with reasonable default values). When you exit the editor, `send-pr` sends the filled out form to the PRMS at Cygnus. You can use the environment variable, `EDITOR`, to specify which editor to use (the default is `vi`). Emacs users will find PRMS especially easy to use. For more information on `send-pr`, see "Introduction to **`send-pr`**" in *Reporting Problems* in ***GNUPro Advanced Topics***. Use **`bugs@cygnus.com`** or **`support@cygnus.com`** if you have any problems. See "How to contact Cygnus" on page v for other means to reach Cygnus.

Filling out a problem report for Unix users

Use `send-pr` for both questions and bug reports. `send-pr` ensures that questions and bug reports are tracked and routed directly to the appropriate person. (`Cc:` the assigned Cygnus engineer, if you know who that is. Do not email the engineer directly; the most reliable means to get a response is to send the report to **`bugs@cygnus.com`**.) Include the category and PR number in the `Subject` line of your email, such as `gcc/13844`.

Problem reports have a structure so that a database program can manage them. When you fill out the form, remember the following guidelines:

- Each PR needs a valid **`customer-id`** and **`category`**.
- Describe only one problem per problem.
- For follow-up mail, use the same subject line as the one in the automatic acknowledgment. It shows the category, the PR number and the original synopsis line. This causes your mail to automatically be filed with the original bug report. Your follow-up comments will be sent to all the people working on the bug.
- Try to make the subject or synopsis line as informative as possible. For instance, you might use a sentence of the form '`Encrypted rlogin hangs if you send interrupt`' or '`g++: calling wrong overloaded function.`'
- You don't need to delete the comment lines while editing the PR form; this is done by `send-pr`. Put your information before or after the comments. See "An Example" in *Reporting Problems* in ***GNUPro Advanced Topics*** for more discussion on this topic.

- See “Valid categories for problems” on page 106 for a list by category of bugs and their descriptions.
- See “Confidential information in reports for Unix users” on page 99 for a discussion regarding how to deal with confidential topics in the reports.

To see a synopsis of the possible commands for PRMS, send a query message like the following on the **`To:`** line: `query-pr@cygnus.com`. Then, on the **`Subject:`** line, use the command, `--help`. A list of possible commands will display.

You can interrogate non-confidential bug reports in the Cygnus Problem Report Management System (PRMS) by electronic mail. Send mail to ‘`query-pr@cygnus.com`’ with query parameters in the ‘**`Subject:`**’ line of your mail header. (The message body is ignored.) For example, to inquire about problem reports numbered `4020` and `5004`, send mail including the following input lines in the `To:` header.

```
query-pr@cygnus.com Subject: 4020 5004
```

You can also include many command line options to request information on bugs in a particular state, or a particular category; for instance, the following header’s example input requests a list of all open `g++` bugs that are not confidential.

`To:` query-pr@cygnus.com
`Subject:` --state=open --category=g++

If you do not include a ‘`--state=`’ specification in your subject line, the mail server uses the following statement (showing all the various input for **`State:`**).

```
--state="open|analyzed|feedback|suspended"
```

WARNING! Since the default state specification for electronic mail queries does not include `closed`, a closed bug yields a response with no message body.

Confidential bug reports are not available using the mail query server. You can request the status of your confidential PRs by contacting Cygnus technical support by phone (408-542-9601).

Confidential information in reports for Unix users

There has been some confusion about where to put confidential information in problem reports sent with `send-pr`. If you submit a problem report to Cygnus, and you want its detailed contents to remain confidential, set the ‘`>Confidential:`’ field to ‘`yes`’.

However, the ‘`Subject:`’ line in the mail header and the ‘`>Synopsis:`’ field in the body of the PR are *not* treated as confidential information. They are used when Cygnus compiles reports of ***fixed*** problems (see “Valid categories for problems” on page 106 for a list by category of the bugs and their descriptions). *Do not* put confidential information in these fields.

Any code samples, machine descriptions, problem details, and so on, remain, of course, strictly confidential in any problem report marked as such in the report.

The mail query server for problem reports never reports any information from confidential bug reports.

Reporting problems for Win95/NT systems

The following file is available for reporting problems for WIN95/NT users.

```
USING SEND-PR TO REPORT PROBLEMS FOR WINDOWS 95/NT

The following fields appear in the send-pr interface.
Use this example and these descriptions to to inform us
of any problems you encounter.
In a Windows NT/95 system, clip the text from Problem Report Fields
in the following example and paste it into your email window.
Where a line begins with  ">" (greater-than mark), there should
be no more than than one in your email transmission.  If there is
more than one ">", we can't process your report efficiently,
which may delay our response to you.
Then consult the following pages for instructions.

THE FULL PROBLEM REPORT SCREEN

                        E-Mail Header Fields

To:      bugs@cygnus.com
Subject: <problem description>
From:    <yourname@youraddress.com>

                        Problem Report Fields
--------------------------------Cut Here----------------------------

>Submitter-Id:   <identifier assigned by Cygnus salesperson>
>Originator:     <Your full name>
>Organization:   <Your company name>
>Confidential:   <[ yes | no ] (one line)>]
>Synopsis:       <synopsis of the problem (one line)>
>Severity:       <[ non-critical | serious | critical ] (choose one)>
>Priority:       <[ low | medium | high ]  (choose one)>
>Category:       <name of the product
```

```
                  (chose one from “>Category:” field  page 104)>
>Class:          <[ sw-bug | doc-bug | change-
request | support ] (one line)>
>Release:        <the release you are reporting on, such as 98r1>
>Environment:    <machine, os, target, libraries (multiple lines)>
                  Architecture: <machine-architecture>
                  System        <operating system>
>Description:    <precise description of the problem (multiple lines)>
>How-To-Repeat:  <code/input/activities to reproduce the problem
                  (multiple lines)>
>Fix:            <how to correct or work around the problem, if known
                  (multiple lines)>
---------------------------------Cut Here----------------------------
The following text describes each field and the information you need
to include in each one.
Prompts in the Problem Report are surrounded by quotes.  For example,
where a prompt item appears in the following description, it is
surrounded by quotes, as in the first prompt, ">Submitter ID:"
Please Note: Seven critical fields are underlined in the following
text. They begin with ">" and end with ":".  They are:
     ">Submitter-ID:"
     ">Category:"
     ">Release:"
     ">Description:"
      >Environment:    <machine, os, target, libraries (multiple lines
)>
                  Architecture: <machine-architecture>
                  System:        <operating system>

The last two fields in the “>Environment:” field do not use the “>”
character.
If you no not fill out these fields, your problem report may be
delayed.
THE E-MAIL HEADER FIELDS

"To:" bugs@cygnus.com
     Send your problem report to this e-mail address.
"Subject:"
```

```
     Describe your problem here briefly, in one line.
"From:"
     Your email address.
THE PROBLEM REPORT FIELDS

">Submitter-Id:"
     Critical information.
     If you know your Submitter ID enter it here. If you do not
     have a contract with Cygnus, contact Cygnus Sales at
     sales@cygnus.com for information on support contracts.
">Originator:"
     Full name of the submitter.
">Organization:"
     The name of the originator's company or parent organization.
">Confidential:"
     Cygnus changes this to "no" only when information is no
     longer sensitive for the customer, or if the sender does
     not have a support contract.
     yes is the default
">Synopsis:"
     This should reflect the Subject: field and be a single line.
">Severity:"
    Use one of the following values:

    "non-critical"    The product, component or concept is working
                      in general, but lacks features, has irritating
                      behavior, does something wrong, or doesn't
                      match its documentation.
    "serious"        The product, component or concept is not
                      working properly. Problems that would otherwise
                      be considered "critical" are rated "serious"
                      when a workaround is known.
    "critical"        The product, component or concept is completely
                      non-operational or some essential functionality
                      is missing. No workaround is known.

     The default is "serious."
```

```
">Priority:"
     How soon the originator requires a solution. Use one of the
     following values:

    "low"             The problem to be solved in a future release.
    "medium"          The problem to be solved in the next release,
                      if possible.
    "high"            A solution is needed as soon as possible.

The default is "medium."

">Category:"
     Critical information.
     Choose from the table of valid categories.

     bfd          binutils    build
     byacc        config      diff
     doc          dos         expect
     flex         g++         gas
     gcc          gcov        gdb
     glob         gprof       help-request
     id-request   info        info-request
     install      ld          libc
     libg++       libiberty   libm
     make         makeinfo    mgmt
     netware      other       patch
     query-pr     readline    send-pr
     sim          shipping    tcl
     test         testsuites  texindex
     texinfo

">Class:"
     The class of a problem uses one of the following subjects
     as input:

    "sw-bug"           A general product problem.
                       ("sw" stands for software.)
```

```
    "doc-bug"          A problem with the documentation.
    "change-request"   A request for a change in behavior, etc.
    "support"          A support problem or question.

     The default is "sw-bug".

">Release:"
     Critical information.
     Provide release or version number of the product, component
     or concept-for example, "gnupro-98r1" or, for custom
     contracts, "chipname-yymmdd".
">Environment:"
     Description of the environment where the problem occurred:
     machine architecture, operating system, host and target
     types, libraries, pathnames, etc.
"Architecture"
     For example, "solaris 2.5.1"
"System"
     For example, "sun4".
">Description:"
     Critical information.
     Provide a precise description of the problem.

">How-To-Repeat:"
     Example code, input, or activities to reproduce the
     problem. The support organization uses example code both to
     reproduce the problem and to test whether the problem is
     fixed. Include all preconditions, inputs, outputs,
     conditions after the problem, and symptoms. Include any
     additional important information, such as all the details,
     however obvious, that would be necessary for someone else
     to recreate the reported problem. Sometimes seemingly
     arbitrary or obvious information can point the way toward a
     solution.
">Fix:"
     How to work around the problem, if known.
```

Valid categories for problems

The following list describes valid entries for the **`>Category:`** field for reporting problems.

`bfd`
GNU binary file descriptor library.

`binutils`
GNU utilities for binary files (`ar`, `nm`, `objcopy`, `objdump`, `size`...).

`bison`
GNU parser generator.

`build`
Cygnus software re-building.

`byacc`
Free parser generator.

`config`
Cygnus software configuration and installation.

`diff`
GNU `diff` program.

`doc`
Documentation and manuals.

`flex`
GNU lexical analyzer.

`g++`
GNU C++ compiler.

`gas`
GNU assembler.

`gcc`
GNU C compiler.

`gdb`
GNU source code debugger.

`gprof`
GNU profiler.

`info`
GNU `info` hypertext reader (non-critical PRs only).

`ld`
GNU linker.

libc
Cygnus C Library (use `newlib` for more expedient and accurate fixes).

libiberty
GNU ‘`libiberty`’ library.

libm
Cygnus Math Library (use `newlib` for more expedient and accurate fixes).

make
GNU `make` program.

makeinfo
GNU utility to build `info` (online help files) from Texinfo documents (non-critical PRs only).

newlib
Cygnus C and Math Libraries.

other
Anything which is not covered by the previous categories.

patch
GNU bug patch program.

send-pr
GNU Problem Report submitting program.

test
Category to use when testing `send-pr`.

texindex
GNU documentation indexing utility (non-critical PRs only).

texinfo
GNU documentation macros (non-critical PRs only).

Fixed problems

The following documentation shows the bugs that have been fixed by Cygnus since the last release.

They are listed alphabetically by category, such as `config` or `gcc`, and by their assigned number to that category. The first line of the problem's description (from the submitter of the bug) accompanies each listing.

binutils

```
9082
     ar does not archive non .o files
12993
     ARM/Thumb disassembly not consistent
14422
     objcopy generates an exception with option --strip-all
14559
     objcopy incorrectly converts ppc-elf format to aixcoff
14804
     objcopy doesn't compute the tekhex record checksum
14892
     objcopy doesn't relocate sections when converting
15356
     objdump displays wrong address for PC-relative Thumb loads
```

build

```
14420
     Can't build 97r2 from source
14805
     Can't build 97r2 on powerpc-ibm-aix4.1.5.0
```

config

```
14861
     96q1 native won't configure target libs
14951
     97r2 configure doesn't work
```

g++

```
6910
     Template instatiation too strict - not deferred until
```

```
8529
     sparc-sun-sunos X mips-idt-ecoff g++ fails to link
9647
     void * parameter accepts const T *
9652
     passing a const ref may produce bogus error messages
10239
     Template friends do not work
10439
     Trouble recognizing the implicitly instantiated templates
11873
     program which works using HP10.10 cyngus g++ cores on DEC
11925
     Internal compiler error with -O2 (hpux 10.01)
12029
     Overloading virtual functions with default parameters
12043
     Unable to compile bison output on DEC
12194
     -fansi-overloading works differenly on HP than DEC
12211
     Weird thing happening with exception I think?
12379
     Internal compiler error with -O3.
12525
     g++ exceptions defect
12721
     Warnings in gnu header files using -W option (Solaris 2.5)
13437
     assembler fails on bison/g++ generated code
13439
     param.h header automatically included by g++ overides
13532
     won't compile with -O
13723
     Declaring an array of objects within another class fails.
13804
     Compiler defined operator= not copying correctly.
13857
     Compilation of specific class results in "virtual memory
13939
     integer sqrt inefficient
14025
     Can not throw exception through libio.
```

```
14126
     istream Segmentation fault
14127
     gcc/g++ moves big global objects into local scope
14275
     internal compiler error
14294
     A test in the Rogue Wave test suite failed
14328
     Internal compiler error
14384
     __sjthrow calls __terminate for some reason.
14431
     Internal compiler error - static initialisers.
14503
     Internal compiler error when using STL reserve() method
14534
     Internal compiler error when compiling bison code.
14543
     g++ methods with squirrely ($sp) action
14584
     SEGV from executable produced by g++
14596
     g++ bails out on ...
14609
     Internal Compiler Error in STL with -O
14629
     g++ produces internal compiler error
14687
     g++ using directive doesn't work as specified in CD2
14699
     Optimization produces incorrect results
14746
     Optimizing code, don't understand why
14748
     cc1plus croaks on no copy constructor
14788
     non-virtual destructor warnings
14823
     -fignore-overflow causes incorrect optimization
14836
     poor handling of
14837
     poor handling of
```

14856
EH - which platforms use
14858
internal compiler error on template instantiation
14866
can't resolve method when in templates
14915
-O and -Wall cause internal compiler error
15015
shared libraries hard code directory name
15030
purify reports freeing mismatched memory
15045
internal compiler with -O and static member
15058
g++ Problem
15121
undefined reference to __default_alloc_template<0, 0>
15139
String ctor incorrectly computes string length
15165
Problems linking more than one library
15174
basic_string copy (...) method should be declared const
15183
Internal compiler error.
15312
Static structure with members initialized by "new" is left
15360
Stabs debugging information incorrect
15377
contains() broken in g++/String.h
15393
Incorrect macro definition in va-i960.h
15430
cyg134 -- How to turn off Runtime Type Information
15446
breakpoint before global ctors
15453
internal compiler error on
15502
Incorrect code generated coverting compound statement
15526
internal compiler error

15528
internal compiler error
15551
g++ and templates

gas

11103
address in gas warning changes when using -a option
11784
v850-ELF-as generate 16MB object after detected small
13915
The PC relative branch "bra ." returns an error in H8
13916
The H8/300 as does not support local labels
13940
validate_immediate buggy
14354
GAS produces illegal operation code for moveq command
14515
GAS macro prerocessor fails with no reason
14536
GAS can't parse some movem commands in absolute addressing
14689
GAS option requested to suppress generation of absolute
14720
register value vs .set defined constant conflict
14721
non-intuitive error|warning messages
14722
No error|warning message for an illegal statement
14723
Some out of range values are not detected
14724
assembler displays only the first encountered message
14934
problem with .align
14940
.fill as pseudo op won't accept absolute expression
14986
-as reports .set symbols as undefined
15057
why not count 'ones'?
15358
Thumb out-of-range branches missed

gcc

5977
Misses redundant store optimization

6216
mips64-elf-gcc warning nit

8012
compiling a file with -frtti which uses math.h causes a

8704
gcc missed optimization - loop unrolling

10242
profiling problem with AIX version of gcc

11911
Verbose code generated for halfword access

11922
Adjacent stores not merged into store-multiple

11927
Stackframe created even for leaf functions

11968
Redundant instructions generated

11970
No tail-call optimisation

11983
Shift that can be done in following instruction

11984
no use of known address relationships

12022
class methods named 'main' produce incorrect code

12227
Argument stacking best done in the prologue

12231
R10 never used when stack checking not in use

12266
info files not updated

13092
undefined library symbol __call_via_sl

13528
Compilation generates "internal compiler error"

13732
GCC doesn't support TX39 instructions (sync,cache)

13816
"-gdwarf-2" option of as doesn't work well

14106
G++ won't generate DWARF, powerpc-EABI, or IEEE-695

14302
tools don't handle filenames with spaces

14310
__declspec(naked) and "control reaches end of non-void

14475
cyg118 -- 'fopen()' returns an error after repeating

14593
amd29k compiler (95q4) generates incorrect code

14644
Internal compiler error - insn does not satisfy its

14671
-fomit-frame-pointer can fail to restore sp

14700
long long arg slobbers all over the stack

14707
sp-relative access inconsistent with fp-relative access

14749
gcc Mips64 dies in assignment to 'long long' variable.

14790
-fignore-overflow causes incorrect optimization

14801
missing library functions

14840
gcc does not use fabs

14874
can't match function prototype anymore

15161
Difference between 97r1 and 97r2a powerpc-eabi-gcc

15185
Big-endian Thumb floats do not work

15192
How does power-pc compiler utilize registers?

15296
no mips machine description template to match rtl -> crash

15396
powerpc-eabi-gcc finds wrong multilib library under

15445
gcc.97r1.arm-coff internal error--unrecognizable insn

gdb

6919
gdb abort()s on loading executable

7691
Would like clarification on support for CPU32BUG monitor

7965
Wingdb problem
8088
gdb hbreak
8359
Command Path is awkward if drive name is specified
8527
native gdb
8875
bug in c-valprint.c
8926
gdb target loss
8928
gdb representation of breakpoints
8929
gdb display values of variables
8932
80-bit-floating-point feature
8951
gdb debuger hangs
9245
Query about PowerPC target stub support
9496
How to debug Sun's light weight processes (lwp) with gdb
9747
Using Sparc Exec. gdb
10100
gdb symbol-file command doesn't work with Solaris 2.5
10358
linker load data in the wrong location
10745
Help needed understanding SIGCONT in remote stubs
11427
gdb isn't demangling C++ names
12074
GDB does not print FP values correctly
12076
CALL_DUMMY mechanism flawed
12077
push_dummy_frame/pop_frame not APCS-32
12557
Problem init'ing target using gdb.
12719
gdb doesn't reread symbol file in various cases

```
12948
     gdb and threads follow up problem
13206
     Simulator doesn't work well
13246
     GDB will not set hbreak for mips16
13374
     gdb unable to see dynamically loaded modules
13500
     corrupted symbol/debug crashing our debugger
13618
     gdb incorrectly reports thread information
13620
     Control-C to interrupt GDB command only works once.
13681
     gdb loads sections to the vma address instead of the lma
13803
     gdb incorrectly reports number of threads and thread info.
13903
     gdb get segmentation fault loading program
14043
     problem in 'sizemem
14062
     Cannot create suitable debug output file
14079
     Exception during call foo() aborts gdb
14088
     call foo() needs target-specific breakpoint
14129
     Cygnus toolchain doesn't support OMF object file format
14342
     simulator request (Coprocessor 0 support)
14376
     GDB working with MacRaigor OCD (On Chip Debug box) makes
14545
     gdb.96q4.ppc seg faults on large .sun files
14546
     gdb.96q4.ppc seg faults on large .sun files
14548
     gdb-96q4-68k gets errors "line # out of range for source
14549
     gdb.96q4.ppc seg faults on large .sun files
14571
     can't call my functions in gdb
```

14645
can't set breakpoint in template

14787
Program name only 14 chars in gdb (fwd)

14821
shared library cores gdb when loading symbols

14871
Symbol load time very slow

14923
instructions for building little-endian mips64 gdb

15038
need gdb97r1's C source files for callback.h/c, monitor.c

15049
gdb does not start

15126
Problem building TK for GDB 4.16-97r1

15169
gdb shell output does not go to the GUI

15182
Disassembly ignores the endianity

15251
gdb confuses emacs with executable in subdir

help-request

10252
How do i tell configure to ignore xmkmf?

14927
gcc cannot find CYGWIN.DLL

info-request

15436
GNU make errors

15440
Cygnus progressive releases and egcs

info

14809
how to extract the offset in C structur to use it in

install

14513
uncompressing the src.tar.Z results in a directory

ld

10555
loader dies with "bfd assertion fail stabs.c 515"

10685
ld -oformat srec creates bad code for address@ha

11346
fails srec check tests

12714
Mips64-ELF linker has bug when mixing mip2 and mips3

13845
Some question about the sections and where they (should)

13955
wrong .debug information in ELF

14160
Linker leaves holes in object file

14437
The linker doesn't free space taken by unused functions

14444
linker does not preserve alignment info

14703
ld lexer doesn't recognize comments in script file.

14752
Cannot longjmp from Thumb code

14810
ld can't resolve a reference to "virtual table"

14904
Cygnus dose not work on Solaris 2.6

15074
97r1 i960-nindy-coff-ld VERY SLOW

15357
Interworking warnings only show half the problem

15373
Out-of-range branch checks faulty

libc

15063
sbrk is not called by malloc (prob. with _impure_ptr?)

libm

15017
Is there a way to speedup the "pow" math function?

make

15354
GNU make errors

15416
make trivially broken on i386-cygwin32 host

15597
make.exe core dump

newlib

12918
bit 0 of PC not Thumb state in APCS-26

13786
RDI targets not handled

15103
Need source code to certain object files (e.g. crt0.o)

other

14282
essai d'erreur sur frog

14289
Problem with gdb/libraries?

14484
ld's unable to locate libpmon.a

14830
Cygnus C/C++ compilers are not supported by purify

14869
can not Purify Dynamic executable created by Cygnus 97r1

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z