PRQA
Programming Research

PRLGCC 4.1

CONFIGURATION OF CROSS PLATFORM GNU COMPILER PERSONALITIES

October 2012

This document describes the configuration of the cross platform GNU compiler personalities and the changes history for the PRLGCC.

PRQA Programming Research

IMPORTANT NOTICE

DISCLAIMER OF WARRANTY

The staff of Programming Research Ltd have taken due care in preparing this document which is believed to be accurate at the time of printing. However, no liability can be accepted for errors or omissions nor should this document be considered as an expressed or implied warranty that the products described perform as specified within.

COPYRIGHT NOTICE

TRADEMARKS

CONTACTING PROGRAMMING RESEARCH LTD

For technical support, contact your nearest Programming Research Ltd authorized distributor or you can contact Programming Research's head office:

by telephone on	+44 (0) 1 932 888 080
by fax on	+44 (0) 1 932 888 081
or by e-mail on	support@programmingresearch.com
	www.programmingresearch.com

PRQA Programming Research

Table of Contents

1. THE PROBLEM

The compiler personality files for QA·C and QA·C++ can define implicit types, parsing extensions, extension keywords which allow the parser to analyze source code or include files which are compliant with the compiler.

The compiler personality is also the place to specify search paths for system include files, implicitly defined macros and substitute header files which may be necessary if the header files supplied with a compiler are too non-compliant to allow parsing.

This last group of attributes prevents Programming Research from providing a GNU C / C++ compiler personality out of the box. The system include paths and implicitly defined macros are system dependent and it may be necessary to provide a set of substitute headers to allow for nonstandard compiler constructs. Further still it may be necessary to supply a file which is always included to define implicit types or macros which are too complex for a simple personality file.

1.1 System Include Paths and Implicit System Defines

GNU C/C++ (gcc and g++ from now on) can provide some of the information we need while compiling a source file. The configuration script will temporarily create a dummy source file in its installation directory and extract this information.

1.2 Substitute Header Files

Programming Research may need to supply substitute header files, which are used for analysis. The configuration script allows you to specify where these may go although a directory structure within the specified destination directory may have to be enforced.

2. UNPACKING YOUR INSTALLATION

The received configuration package will contain a file named prlgcc_4.1.zip. This file can be unpackaged as follows on UNIX (or in a UNIX like environment on Windows):

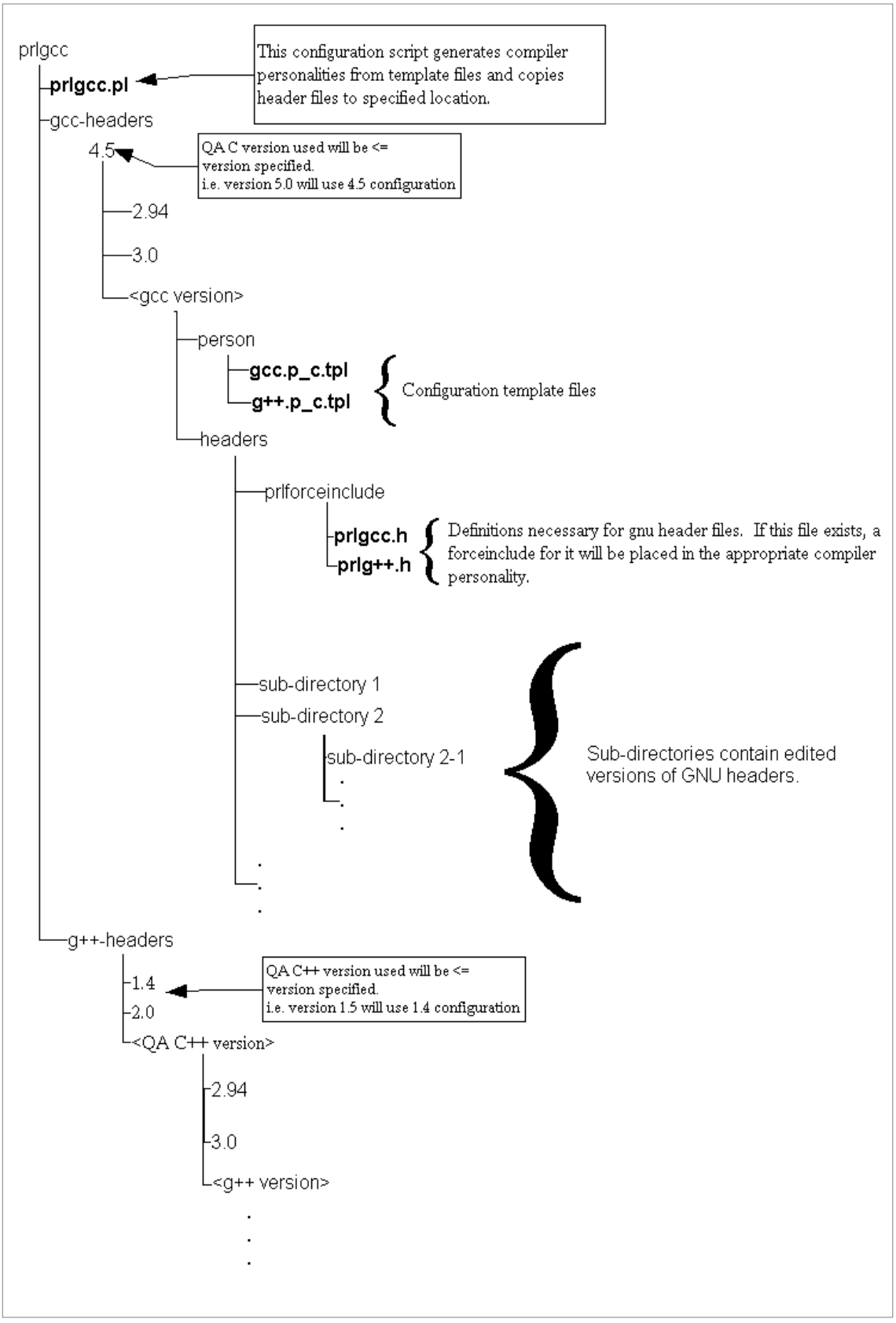
```
$ unzip prlgcc_4.1.zip
```

This will create a prlgcc directory containing the script, User Guide and template directories.

On Windows, different commercial unzipping programs can be used or files may be extracted directly by using Windows Explorer.

2.1 Directory Structure

The directory structure is illustrated in the diagram below:



The top-level gcc directory contains the configuration script prlgcc.pl. The Windows package also contains a compiled version of prlgcc.exe in the same location. The subdirectories gcc-headers and g++-headers contain a series of directories that correspond to the version numbers of supported Programming Research products. These directories contain sub-directories which correspond to the version numbers of supported GNU compilers.

For example, to find a matching configuration for QA·C 4.5.2, only the “major.minor” part of the version number (in this case 4.5) is required. In this case we have a match. However, if version 5.0 is specified, then the highest numbered configuration less than 5.0 will be used as a match. If version 4.4.2 is specified, then the lowest numbered configuration will be used as the closest match.

Each of the version directories contain the same structure. A person directory will contain the templates for gcc and/or g++ compiler personalities. These templates follow a fixed naming convention. If the corresponding template does not exist, it is assumed that the compiler is not yet supported.

The header directory contains optional QA·C and QA·C++ forceinclude header files. Any other files / directories are assumed to be substitute headers. The directory structures are copied to the location specified by the user when prlgcc.pl is run.

3. USING THE CONFIGURATION SCRIPT PRLGCC

3.1 prlgcc.pl or prlgcc.exe on Windows?

The Windows package contains two versions of the configuration script: a PERL version (prlgcc.pl) and a compiled executable version (prlgcc.exe). The choice of which one to use depends on the environment in which your GCC compiler runs in. If your compiler is a native Windows version (that is, it was compiled for Windows and runs using only the standard Windows dlls - for example MINGW GCC), then you must use prlgcc.exe. If your compiler is a Cygwin compiler, then you must use prlgcc.pl.

Both versions take the same parameters; in the following sections prlgcc.pl can be substituted with prlgcc.exe as required.

3.1.1 Usage on Cygwin

The PERL script detects if it is being run on Cygwin and will substitute Cygwin mount points for include paths in the compiler personality where necessary. It will also detect the Cygwin root directory and replace paths such as

```
/usr/include
```

with

```
c:\cygwin\usr\include
```

Assuming that your c:\cygwin is the Cygwin root directory.

Note:

Some directories available within Cygwin may not be visible directly from DOS / Windows - and hence QA·C/QA·C++ will not be able to see them. Prlgcc will convert these paths to full paths accessible from Windows.

3.2 Usage

The prlgcc.pl script has the following usage:

```
prlgcc.pl <qac|qacpp|qac++> -personalities <personality_file_dir> [-version <product
version no>] [-compiler compiler_exe] [-headers <substitute_header_dir>] [-pcpaths] [-
manual] [-display]
```

The options can be specified with the minimum letters that make the option unique, for example:

```
prlgcc.pl <qac|qacpp|qac++> -v <product version no> -h <substitute_header_dir> -pe
<personality_file_dir> [-c compiler_exe] [-pc]
```

Typical usage for gcc on UNIX would be:

```
./prlgcc.pl QAC -pe /opt/PRQA/QAC-8.0-R/personalities
```

For Cygwin typical usage would be:

```
./prlgcc.pl QAC -pe /cygdrive/C/Program\ Files/PRQA/QAC-8.0-R/personalities
```

3.3 Parameters

This section describes the PRLGCC parameters in more detail.

3.3.1 Product Option

The first option to prlgcc.pl must be the product. It can be qac, qacpp or qac++ (or upper case equivalent). This tells the script to generate a personality with attributes for QA·C or QA·C++ and integrate the appropriate GNU C or GNU C++ compiler.

This parameter is mandatory.

3.3.2 -personalities, -pe

This parameter is the location to store the generated personality file. The directory specified must be an absolute path: relative paths are not accepted.

This can be (but doesn't have to be) the same directory as the -headers parameter.

This parameter is mandatory.

3.3.3 -version, -v

The QA·C++ version number e.g. 2.0, 3.0, etc.

This parameter is not required when the Product options is set to 'qac'.

This parameter is required only when the Product Option is set to 'qacpp' or 'qac++' and the version is 2.0 or previous. The default is to generate a personality compatible with QA·C++ 3.0 onwards.

3.3.4 -compiler, -c

Without this parameter prlgcc.pl will run either gcc or g++ depending on the product parameter. However, most cross compiler variants of gcc/g++ have the target name encoded into the compiler executable. Additionally there may be several different gcc/g++ compilers installed for various targets (including the native one). This parameter is used to give the name of the compiler that the personality should be generated for. It can either be just the compiler name if it is on the path or the full path to the compiler executable.

This parameter is optional.

3.3.5 -headers, -h

This parameter specifies the directory in which substitute header files should be stored. The directory specified must be an absolute path: relative paths are not accepted.

When running the script, it is important not to specify a directory above in which the script was unpacked, e.g. the directory one level above the directory containing the script. After the substitute headers are copied, this directory is searched recursively to build the list of substitute header files. This will result in a long list containing substitute headers for every supported version of the compiler.

The best solution is to create a directory for the substitute headers and then to specify that to the configuration script - see the -personalities parameter section.

If this parameter is not specified, the headers will be placed in the directory specified by the -personalities option.

3.3.6 -pcpaths (optional)

This parameter is implied when the script is run under Cygwin.

Compilers under Cygwin or other Unix-like environments will report include directories of the form:

```
/usr/include
```

On Windows, QA·C and QA·C++ require Windows paths. The -pcpaths option converts unix style paths (see above) to Windows equivalent (see below):

```
C:\cygwin\usr\include
```

The script will also detect if any paths reported by the compiler are symbolic links (which QA·C and QA·C++ cannot follow) and convert these to the full Windows paths.

This parameter is optional.

3.3.7 -manual

By default, prlgcc runs in a non-interactive way, choosing sensible default options where there is a choice. The -manual option causes prlgcc to stop and request information from the user where a decision is required.

3.3.8 -display

This option makes prlgcc prints out the full path to the created personality in double square brackets after it is generated. This option is useful when running prlgcc from a script.

3.4 Script Operation

If the -compiler option is not specified, the script will use the first gcc or g++ it finds on the path. If you have more than one version installed, ensure that the version you want to use is the first one in your search path or use the -compiler option.

If the version of the compiler does not match one of the configuration templates and -manual is not set, PRLGCC will automatically choose the nearest version below the version found. However when -manual is set, PRLGCC will provide the list of available versions from which a configuration can be selected.

Note:

The -manual option should only be used if the personality created by the automatic method is invalid

```
Version 3.2 of your compiler was detected.
An exact match for a configuration template was not found
Configurations found for g++ versions:
==========================================
1. 2.96
2. 3.0
3. 3.1
Q. Quit configuration.
Please select the version which is the closest match for your system
Enter a number[1..3] or Q to quit: 3
```

At this point you can enter Q to quit without generating a personality or choose the number to the left of the configuration version, in which case a personality will be generated for that configuration.

A complete session could look something like this.

```
Version 3.2 of your compiler was detected.
An exact match for a configuration template was not found
Configurations found for g++ versions:
==========================================
1. 2.96
2. 3.0
3. 3.1
Q. Quit configuration.
Please select the version which is the closest match for your system
Enter a number[1..3] or Q to quit: 3
Creating configuration for g++(3.2) using g++(3.1) template.
Creating substitute headers.
Substitute headers completed.
Configuring compiler personality.
Compiler personality configuration complete.
```

Upon completion your gcc/g++ compiler personality will be generated in the specified personality directory. The substitute headers will be copied to their new location and this will be registered in the personality file. If you need to change the location of these files in the future, you will need to edit the personality either directly or through the appropriate user interface.

Depending on the configuration, a “forceinclude” file may have been necessary. This is a file that is silently included before any other source files are parsed by QA·C/QA·C++. If this was necessary, it will be located at the prlforceinclude directory within the header file subdirectory.

Additional substitute headers may also be created, possibly in a directory structure. These headers are included instead of the compiler header files.

4. Using Your Compiler Personality

The generated personality can be used like any other compiler personality. This can be in the GUI, on the command line or in an integration. Refer to the QA·C/QA·C++ documentation for more information on how to do this.

4.1 Limitations

Although the script attempts to gather all the settings from the compiler output, in some cases this may be incomplete or inaccurate (especially on cross compilers). The generated personality can be edited either directly in a text editor or by opening it in the QA·C/QA·C++ GUI.

4.2 Moving The Personality

The generated personality will contain absolute paths to the compiler include directories, force include files and substitute header directories. If you decide to relocate the personality then it is strongly advised that you move the force include and substitute header directories with the personality and modify the entries in the personality to reflect the new location.

5. Change History

The following tables summarize the change requests (CRs) that have been implemented in PRLGCC.
CRs have been categorized into 3 types (column “T”):

C - A significant change has been implemented to the existing behavior.

F - A fix of a bug or problem feature.

N - New functionality has been introduced.

Some CRs are associated with more than one category and in some cases; the distinction between categories is a little indistinct. The classification should therefore be treated as a guide only.

5.1 Version 3.0

5.1.1 Change Requests

CR	T	Description
12298	F	prlgcc does not resolve symbolically linked directories or mount points on Windows. All Cygwin paths are now converted to full Windows paths.

5.2 Version 4.0

This release is used by and packaged into the Quick Start Utility.

5.2.1 Change Requests

CR	T	Description
14328	N	prlgcc generated personalities do not work with MINGW32 GCC. The force include files have been modified to be compatible with the MINGW compiler.
14134	F	prlgcc should not fail when the prlforceinclude directory already exists. The script now creates the directory if it does not exist.
14093	N	Make prlgcc run in non-interactive way. The script now runs without prompting the user - though this behaviour can be reverted using the - manual option.
13954	N	Everybody uses the wrong setting for -headers in prlgcc. Many users set the -headers setting to the gcc-headers directory rather than the output location for any substitute headers. This is now optional and defaults to the same location as the -personalities setting.
13953	F	prlgcc does not run well on Windows. A compiled Windows version of the script is now shipped in the package for native Windows compilers.

5.3 Version 4.1

5.3.1 Change Requests

CR	T	Description
11869	F	The G++ 'typeof' keyword can now be emulated using the C++ 11 decltype language feature.
14529	F	Support added for '__asm'.
15545	N	Updates to support new features and fixes in QA·C 8.1 and QA·C++ 3.0.1 including improvements in support for recent versions of GCC (<=4.7.1) and BOOST C++ libraries.
15834	C	PRLGCC now automatically detects the intrinsic type for size_t, ptrdiff_t and wchar_t as well as determining the correct size and alignments for the fundamental types.
15835	C	Remove support for QA·C++ 1.4.