第 1 章
GCC 概述

本章主要对 GCC 的发展过程及 GCC 的特点进行简介，并给出了本书的主要内容简介。

1.1 GCC 的产生与发展

GCC（GNU Compiler Collection）是 GNU 工程（GNU Project）中的核心工具软件，其官方网址为 https://gcc.gnu.org/。GCC 支持多种前端的编程语言，包括 C、C++、Java、Ada 和 Fortran 等，其编译生成的目标代码可以在几乎所有的处理器平台上运行，是目前使用最广泛的编译系统之一。GCC 遵循 GNU GPL（GNU Public License）协议，由 FSF（Free Software Foundation）发布。GNU 和 GCC 的图标如图 1-1 所示。

初期的 GCC 仅仅作为 C 语言的编译器，即 GNU C Compiler。1987 年 GCC 1.0 发布，同年 12 月，GCC 开始支持 C++ 语言，随后，GCC 开始支持 Objective-C、Objective-C++、Fortran、Java 和 Ada 等语言。与此同时，GCC 也被逐渐移植到各种各样的主流处理器体系结构上，包括 i386、ix86_64、SPARCE、ARM 和 MIPS 等处理器平台。

a）GNU 图标　　b）GCC 图标

图 1-1　GNU 及 GCC 的图标

自从 1987 年 Richard Stallman 和 Len Tower 发布 GCC 的第一个版本 GCC 1.0 以来，目前 GCC 的最新版本已经更新到 GCC 6.0，https://gcc.gnu.org/releases.html 给出了 GCC 在各个时期推出的 GCC 版本，其中最重大的变化是在 1999 年 7 月，GCC 与 EGCS（Experimental/Enhanced GNU Compiler System）重新融合并发布了 GCC 2.95 版本。

相关的资料可以查阅以下官方网站信息：

```
GNU Compiler Collection: https://gcc.gnu.org/
Free Software Foundation: http://www.fsf.org/
GNU Project: https://gnu.org/
GNU Public License: https://www.gnu.org/licenses/licenses.en.html#GPL
```

1.2 GCC 的特点

GCC 作为目前较为成功的编译系统之一，具有非常突出的优点，主要包括：

（1）GCC 编译系统支持众多的前端编程语言，GCC 4.4.0 中 ${GCC_SOURCE}/gcc/ 目录下包含了前端编程语言处理的目录及其代码（其中，${GCC_SOURCE} 表示 GCC 源代码的主目录，下同），主要包括 C、C++、Ada、Fortran、Java、Objective-C、Objective-C++ 等语言的前端处理，可以使用如下命令查看这些目录：

```
[GCC@localhost gcc-4.4.0]$ ls -l gcc
drwxrwxr-x.  3 GCC GCC   69632 Apr 21  2009 ada
drwxrwxr-x.  2 GCC GCC    4096 Nov 27  2013 cp
drwxrwxr-x.  2 GCC GCC    4096 Nov  6 15:14 fortran
drwxrwxr-x.  2 GCC GCC    4096 Oct  9 17:34 java
drwxrwxr-x.  2 GCC GCC    4096 Apr 21  2009 objc
drwxrwxr-x.  2 GCC GCC    4096 Apr 21  2009 objcp
```

（2）GCC 支持众多的目标机器体系结构，具有良好的可移植性，GCC 4.4.0 的 ${GCC_SOURCE}/gcc/config/ 目录下包含了 GCC 对目标处理器的支持情况，其中包括了各种主流的处理器，例如，arm、i386、mips 以及 alpha 等，以下是 GCC 4.4.0 代码所支持的处理器列表：

```
alpha       arc           arm           avr           cris
crx         fr30          frv           h8300         i386
ia64        iq2000        m32c          m32r          m68hc11
m68k        mcore         mips          mmix          mn10300
pa          pdp11         picochip      rs6000        s390
score       sh            sparc         spu           stormy16
v850        vax           xtensa
```

（3）GCC 具有丰富的配套工具链支持。

GCC 不是一个孤立的编译工具，而是整个 GNU 工程中的一个组成部分。GNU 工程中的其他软件，包括 GNU C 库 glibc、GNU 的调试工具 gdb，以及 GNU 二进制工具链 binutils（GNU Binutils Toolchains，例如汇编工具 as，连接工具 ld，目标文件分析工具 objdump、objcopy 等）等都与 GCC 关系密切，互相依赖。

可以使用下述的 shell 命令查看 GNU 二进制工具链中主要包括的工具：

```
[GCC@localhost paag-gcc]$ rpm -ql binutils  | xargs ls -l | grep "/usr/bin"
-rwxr-xr-x. 1 root root  24352 Oct 15  2014 /usr/bin/addr2line
-rwxr-xr-x. 1 root root  54444 Oct 15  2014 /usr/bin/ar
-rwxr-xr-x. 1 root root 527220 Oct 15  2014 /usr/bin/as
-rwxr-xr-x. 1 root root  26356 Oct 15  2014 /usr/bin/c++filt
-rwxr-xr-x. 1 root root  99212 Oct 15  2014 /usr/bin/gprof
-rwxr-xr-x. 1 root root 588116 Oct 15  2014 /usr/bin/ld
-rwxr-xr-x. 1 root root  38800 Oct 15  2014 /usr/bin/nm
-rwxr-xr-x. 1 root root 212216 Oct 15  2014 /usr/bin/objcopy
-rwxr-xr-x. 1 root root 276528 Oct 15  2014 /usr/bin/objdump
-rwxr-xr-x. 1 root root  54448 Oct 15  2014 /usr/bin/ranlib
-rwxr-xr-x. 1 root root 288560 Oct 15  2014 /usr/bin/readelf
```

```
-rwxr-xr-x. 1 root root  27196 Oct 15  2014 /usr/bin/size
-rwxr-xr-x. 1 root root  25832 Oct 15  2014 /usr/bin/strings
-rwxr-xr-x. 1 root root 212244 Oct 15  2014 /usr/bin/strip
```

（4）GCC 提供可靠、高效、高质量的目标代码。

GCC 是目前使用的最为广泛的编译器系统之一，众多工业级应用的实践证明，GCC 编译系统生成的代码具有很高的可靠性和运行效率。

（5）GCC 对于并行编译的支持。

在 GCC 4.4.0 中，已经提供了对 OpenMP 的完整支持。

1.3 GCC 代码分析

GCC 作为目前 GNU 项目中应用最广泛的工具软件之一，是工程师设计编译系统最典型、最成功的范例，是高校学生学习编译系统最生动、最权威的设计实例，同时也是程序员进行高质量代码设计的有益参考。本书以 GCC 4.4.0 的源代码为例，对 GCC 的设计和实现进行分析和解读，主要涉及以下内容：

（1）GCC 的发展历史及特点；

（2）GCC 的总体结构；

（3）GCC 中各种中间表示（包括抽象语法树、GIMPLE、寄存器传输语言）的生成技术；

（4）GCC 中基于 GIMPLE 的优化处理，这一部分主要实现一些与目标机器无关的性能优化；

（5）GCC 中基于 RTL 的优化处理，这一部分主要实现一些与目标机器相关的性能优化；

（6）GCC 的移植技术，即如何让 GCC 支持新的目标机器。

本书将紧密围绕编译系统中的中间表示（IR，Intermediate Representation）这一核心概念，重点介绍 GCC 中的三种中间表示：抽象语法树（AST，Abstract Syntax Tree）、GIMPLE 和寄存器传输语言（RTL，Register Transfer Language），对其基本概念、存储结构及其生成过程等进行深入分析。由于 GCC 基于 GIMPLE 和 RTL 的优化处理数量非常多，每种优化处理都涉及一个比较独立的优化问题，很难在本书中一一详述，因此，本书只简单地介绍了 GCC 中基于 GIMPLE 及 RTL 的优化处理的基本组织方式，并对其中一些非常典型的优化处理进行了简介。最后，本书也给出了将 GCC 成功移植到西安邮电大学自主研发的阵列处理器上的一个实例。

限于篇幅，书中的大部分代码只给出了简化版本，读者在分析时需要结合源代码仔细研读。