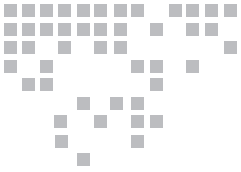

Chapter 2 第 2 章

UEFI 开发环境搭建

通过前面的学习，我们已经知道 UEFI 是一种标准，它没有给出具体的实现。软件厂商可以根据 UEFI 标准开发自己的 UEFI 实现，其中常用的开源实现是 EDK2。EDK2 是遵循 UEFI 标准和 PI 标准的跨平台固件开发环境。UEFI 的目标是完全取代 BIOS，因而它要能完全支持所有类型的 CPU[㊀]，并让所有的硬件厂商接受这种变化。来自不同厂商的开发者使用各种不同的开发环境开发自己的产品。为了让这些不同的开发者愉快地接受 EDK2 来开发自己平台上的 UEFI 固件或应用，EDK2 对每种平台都提供了对应的开发工具。EDK2 支持在多种操作系统下的开发，例如 Windows、Linux、Darwin、UNIX 等，并支持跨平台编译，如在 Windows 开发环境下可以编译出 Arm 平台上的 UEFI 应用程序。下面我们分别介绍在 Windows 和 Linux 下如何使用 EDK2 进行开发。

2.1 配置 Windows 开发环境

EDK2 目前支持 Windows 7、Windows 8、Windows 8.1。开发 UEFI 应用和驱动之前要建立开发环境，分为如下几步。

1）首先需安装 EDK2 依赖的开发工具：Windows SDK、C 编译器、IASL 编译器，然后下载 EDK2 源码，EDK2 源码包里包含了开发所需的源码和工具。

2）配置 EDK2 ：主要是设置开发工具的路径。然后就可以通过 EDK2 提供的源码和工

㊀ 目前 EDK2 支持 x86（32 位和 64 位）CPU、安腾 CPU 和 Arm CPU。

具开发 UEFI 应用和驱动。

3）编译 UEFI 模拟器和 UEFI 工程。

4）运行 UEFI 模拟器。

下面详细介绍动手开发之前必须进行的这几个步骤。

2.1.1　安装所需开发工具

首先需要安装 EDK2 所依赖的开发工具以及 EDK2 本身。以下是主要的安装步骤。

1）安装 Windows SDK。Windows SDK 可从如下地址下载：

http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=24826

2）安装 C 编译器。推荐安装微软公司的 Visual Studio 编译器或英特尔公司的 ICC 编译器，也可以安装 Cygwin 及其 gcc 编译器。

3）安装 IASL 编译器（https://www.acpica.org/downloads/binary_tools）。

4）IASL 用于编译 .asl 文件。.asl 是高级配置与电源接口（Advanced Configuration and Power Interface）源文件。

5）下载 EDK2 开发包：可以从 TianoCore 官方网站下载 EDK2 发行版（https://sourceforge.net/projects/edk2/files/UDK2014_Releases/UDK2014/UDK2014.Complete.MyWorkSpace.zip/download），也可以用 subversion 工具或命令行获得最新源码。下面是两个简单的示例。

1）用 subversion 命令行获得 EDK2 源码：

```
svn co https://svn.code.sf.net/p/edk2/code/trunk/edk2edk2
```

2）或者使用 TortoiseSvn 下载源码，如图 2-1 所示。

2.1.2　配置 EDK2 开发环境

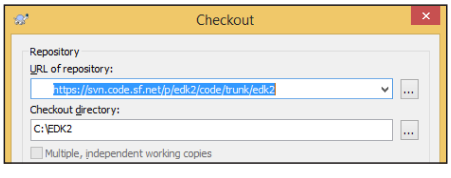

图 2-1　用 TortoiseSvn 下载 EDK2 源码

下面讲解如何配置 EDK2 开发环境。

1）首先进入 EDK2 目录并运行 edksetup.bat，此步骤用于建立 Conf 目录下的 target.txt、tools_def.txt 等文件。

```
C:\>EDK2Edksetup.bat
```

2）编辑 Conf\target.txt。

根据用户的编译器环境修改编译工具 TOOL_CHAIN_TAG。此处以 x86_64 平台的 32 位的 Visual Studio 2008 为例，需设置 TOOL_CHAIN_TAG=VS2008x86。

图 2-2 是设置了 TOOL_CHAIN_TAG 为 VS2008x86 的 target.txt 文件。

64位操作系统上的Conf\target.txt

```
ACTIVE_PLATFORM              = Nt32Pkg/Nt32Pkg.dsc
TARGET                       = DEBUG
TARGET_ARCH                  = IA32
TOOL_CHAIN_CONF              = Conf/tools_def.txt
TOOL_CHAIN_TAG               = VS2008x86
MAX_CONCURRENT_THREAD_NUMBER = 1
BUILD_RULE_CONF = Conf/build_rule.txt
```

图 2-2　Conf\target.txt 配置文件

注意　TOOL_CHAIN_TAG 的设置要根据 Visual Studio 的安装路径来确定。

IA32 平台下 TOOL_CHAIN_TAG 的设置见表 2-1。

表 2-1　IA32 平台下的 TOOL_CHAIN_TAG

Visual Studio 版本	TOOL_CHAIN_TAG 的值
Visual Studio 2003（VC7）	VS2003
Visual Studio 2005（VC8）	VS2005
Visual Studio 2008（VC9.0）	VS2008
Visual Studio 2010（VC10.0）	VS2010

AMD64 平台下的 TOOL_CHAIN_TAG 的设置见表 2-2。

表 2-2　AMD64 平台下的 TOOL_CHAIN_TAG

Visual Studio 版本	TOOL_CHAIN_TAG 的值（64 位 VS）	TOOL_CHAIN_TAG 的值（32 位 VS）
Visual Studio 2003（VC7）	VS2003	VS2003x86
Visual Studio 2005（VC8）	VS2005	VS2005x86
Visual Studio 2008（VC9.0）	VS2008	VS2008x86
Visual Studio 2010（VC10.0）	VS2010	VS2010x86

如果用户安装的是 ICC 编译器，则需要将 TOOL_CHAIN_TAG 设置为 ICC 或 ICC11，具体配置需根据 ICC 版本号确定。例如，安装了 ICC11 编译器，则做如下设置：

```
TOOL_CHAIN_TAG = ICC11
```

如果用户安装的是 Cygwin，则需要将 TOOL_CHAIN_TAG 设置为 CYGWIN：

```
TOOL_CHAIN_TAG = CYGWIN
```

3）检查 Conf\tools_def.txt，确保编译器路径正确。

tools_def.txt 工具为 EDK2 自动生成的文件，里面预定义了几种常用的编译器。

例如，我们在步骤 2 中设置了 TOOL_CHAIN_TAG 为 VS2008x86，在 tools_def.txt 查找 VS2008x86_BIN，看其路径是否正确，如果路径不正确，需设置为 cl.exe 的正确路径。

图 2-3 为 tools_def.txt 文件中关于 VS2008x86 的部分，其中 DEFINE VS2008x86_BIN 定义了编译器 cl.exe 的路径。当编译 32 位 EDK 程序时，build 工具会使用 VS2008x86_BIN 下的 cl 编译器；当编译 x86_64 程序时，会使用 VS2008x86_BINX64 下的 cl 编译器。

DEFINE WIN_ASL_BIN_DIR 定义了 IASL 编译器的路径。

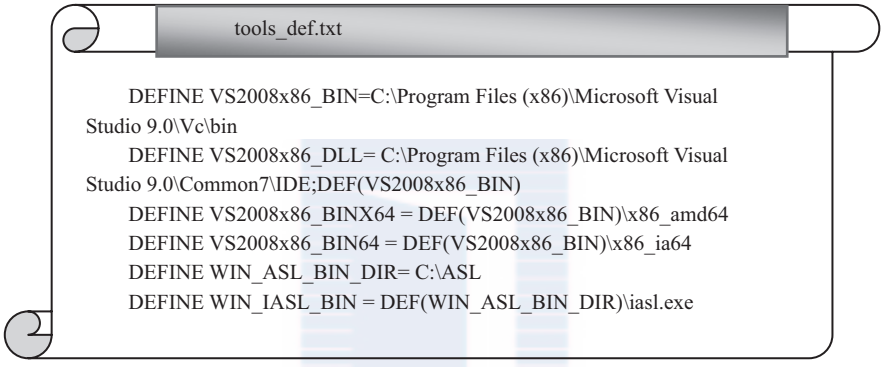

图 2-3　conf\tools_def.txt 文件 VS2008x86 部分，该文件定义了编译器的路径

2.1.3　编译 UEFI 模拟器和 UEFI 工程

下面讲解如何利用 EDK2 提供的工具链进行编译，并对其中的重要参数进行讲解。

编译 UEFI 代码是在 CMD 命令行中通过运行 EDK2 工具链命令完成的，分两种情况：一是编译 Nt32Pkg 工程，二是编译非 Nt32Pkg 的 UEFI 工程。

1. 编译 UEFI 模拟器，即 Nt32Pkg

首先需要运行 edksetup.bat--nt32 以设置 EDK2 环境变量，如下所示：

```
Setting environment for using Microsoft Visual Studio 2008 x86 tools.
C:\Program Files(x86)\Microsoft Visual Studio 9.0\VC>cd c:\EDK2
C:\EDK2>edksetup.bat --nt32
```

设置好环境变量后，就可以使用 EDK2 的工具链编译 UEFI 模拟器了。编译 UEFI 模拟器的命令非常简单，在命令行执行 build 命令即可，如下所示：

```
C:\EDK2> build
```

build 命令相当于 Visual Studio 的 nmake 命令，它分析 UEFI 的工程文件，根据分析结果

自动执行相应编译和连接命令。

不带参数的 build 命令等同于 build -a IA32 -p Nt32Pkg\Nt32Pkg.dsc，这是因为 build 命令使用了 conf\target.txt 中定义的默认参数 TARGET_ARCH（对应 -a 参数）和 ACTIVE_PLATFORM（对应 -p 参数）。

2. 编译非 Nt32Pkg 工程

首先设置环境变量，打开 Visual Studio 2008 command prompt，进入 EDK2 目录并运行 edksetup.bat，如下所示：

```
Setting environment for using Microsoft Visual Studio 2008 x86 tools.
C:\Program Files(x86)\Microsoft Visual Studio 9.0\VC>cd c:\EDK2
C:\>EDK2>edksetup.bat
```

设置好环境变量后，就可以使用 EDK2 提供的 build 工具编译 UEFI 代码了。例如，要编译 MdePkg：

```
C:\EDK2> build -a X64 -p MdePkg\MdePkg.dsc
```

3. build 命令

build 命令是编译 UEFI 工程常用的命令。它有三个重要参数：-a、-p 和 -m。

- -a 用来选择目标平台。可供选择的选项有 IA32（32 位 x86 CPU）、X64（64 位 x86_64 CPU）、IPF（ Itanium Processor Family）、ARM 和 EBC（EFI byte code）；默认的参数在 Conf/target.txt 中设置。
- -p 用来指定要编译的 package 或 Platform。-p 的参数是这个 package 或 Platform 的 .dsc 文件。默认的参数在 Conf/target.txt 中设置。
- -m 用来指定要编译的模块。如果不指定 -m 选项，build 将编译 .dsc 文件指定的所有模块。

例如，编译 32 位的 Shell 后，Shell.efi 存放于 edk2\Build\ShellPkg\DEBUG_VS2008x86\IA32\。

```
C:\EDK2> build -a IA32 -p ShellPkg/ShellPkg.dsc -m ShellPkg/Application/Shell/
    Shell.inf
```

编译 64 位的 Shell 后，Shell.efi 存放于 edk2\Build\ShellPkg\DEBUG_VS2008x86\X64\。

```
C:\EDK2> build -a X64 -p ShellPkg/ShellPkg.dsc -m ShellPkg/Application/Shell/
    Shell.inf
```

表 2-3 列出了 build 命令的常用参数及用法。

表 2-3　build 命令的常用参数

build 命令参数	参数用法
-a ARCH	选择目标平台，ARCH 可以是 IA32、X64、IPF、ARM 或 EBC，该选项将会取代 Conf\target.txt 文件中的 TARGET_ARCH
-DMACROS	定义宏，例如 -D NETWORK_ENABLE
-h	显示帮助信息
-j LOGFILE	将编译信息输出到文件
-b TARGET	选择编译成 DEBUG 还是 RELEASE 例如： -b DEBUG -b RELEASE
-t TOOLCHAIN	选择 tools_def.txt 中定义的编译工具，例如要使用 Visual Studio 2010： -t vs2010
-n ThreadNumber	编译器使用的线程数量
-p PlatformFile	通过指定 .dsc 文件指定要编译的 Package，该选项将会取代 Conf\target.txt 文件中的 ACTIVE_PLATFORM。例如，要编译 AppPkg，可以使用如下选项： -p AppPkg\AppPkg.dsc
-m ModuleFile	指定要编译的模块，build 工具将只编译此模块
-q	编译过程中只显示严重错误信息
-s	使用沉默模式执行 make 或 nmake
-u	跳过 AutoGen 这一步
-c	文件名不区分大小写

2.1.4　运行模拟器

使用以下命令运行 UEFI 模拟器：

```
C:\EDK2> build run
```

因为在 target.txt 中已经设置了 TARGET_ARCH 与 ACTIVE_PALTFORM，所以 build run 与 build -a IA32 -p Nt32Pkg\Nt32Pkg.dsc run 命令等同，可以用来运行 UEFI 模拟器。或者直接运行 Build\NT32\DEBUG_VS2008x86\IA32 目录下的 SecMain.exe。

通常模拟器最终会进入 UEFI Shell，有关 UEFI Shell 内容我们会在第 16 章详细讲述。EDK2 默认将目录 C:\edk2\Build\NT32\DEBUG_VS2008x86\IA32\ 映射为文件系统 FSNT0，在 Shell 中执行命令 FSNT0:，Shell 将会打开 FSNT0 分区并将当前目录切换到 FSNT0 分区的根目录，如下列代码所示：

```
Shell>FSNT0:
FSNT0:\>
```

执行“?”命令将会列出所有 Shell 支持的命令，如图 2-4 所示。

```
FSNT0:\>?
alias           - Displays, Creates, or deletes UEFI Shell aliases.
attrib          - Displays or changes the attributes of files or directories.
...
```

图 2-4　Shell 中的“?”命令

图 2-5 显示了 UEFI 模拟器启动时的界面。图 2-6 显示了 UEFI 模拟器进入 Shell 时的界面。

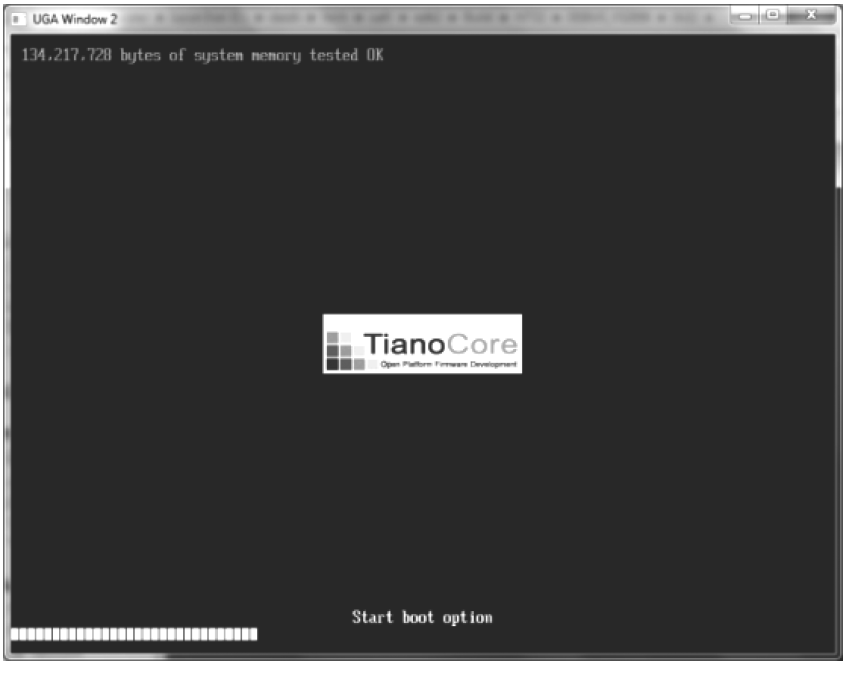

图 2-5　NT32 UEFI 模拟器启动界面图

图 2-5 中第二行显示了 Current running mode 1.1.2，这说明 UEFI Shell 是 EDK2 开发包中预先编译的 Shell。我们可以用如下方式进入刚才编译的 Shell。

1）将 Shell.efi 文件从 edk2\Build\ShellPkg\DEBUG_VS2008x86\IA32\ 复制到 Build\NT32\DEBUG_VS2008x86\IA32 目录。

2）在模拟器窗口的 Shell 中执行以下命令：

```
Shell>FSNT0:
FSNT0:\>shell
```

执行完毕后将会进入新的 UEFI Shell，新 Shell 如图 2-7 所示。

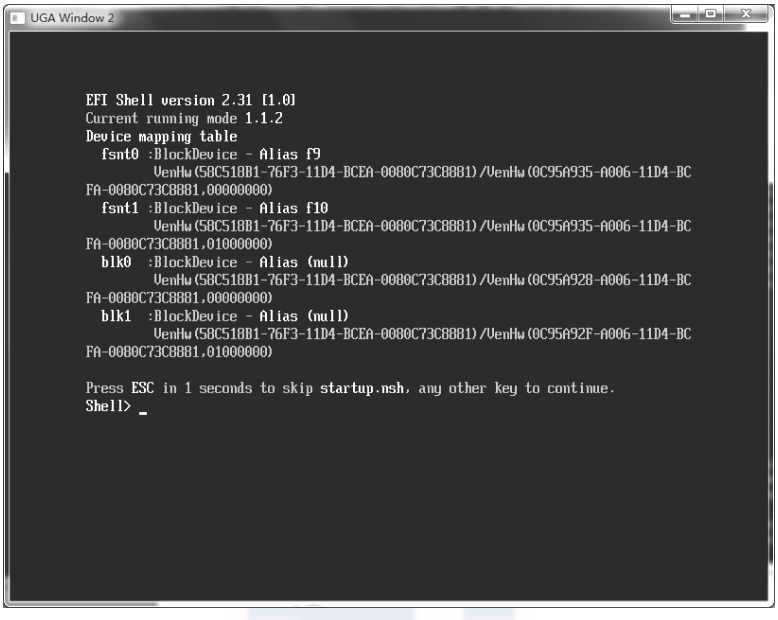

图 2-6　NT32 UEFI 模拟器 Shell 界面

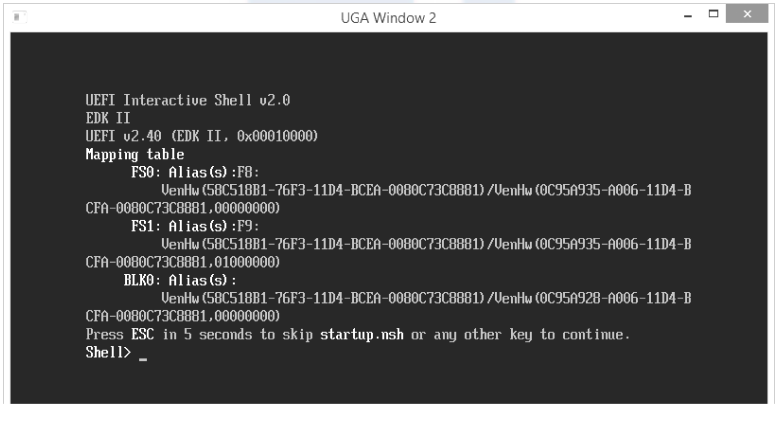

图 2-7　UEFI 2.4 标准下的 UEFI Shell

2.2　配置 Linux 开发环境

与配置 Windows 开发环境相似，配置 Linux 开发环境也包括如下步骤。

1）安装 gcc 编译器，并下载 EDK2 源码。

2）配置 EDK2，包括编译 EDK2 工具链和设置编译器路径。在配置 Windows 开发环境时，我们没有编译 EDK2 工具链，因为 EDK2 源码包中包含了 Windows 下的 EDK2 工具链可执行文件。但是，由于 Linux 发行版众多，因此，EDK2 源码包中没有包含 EDK2 工具链

可执行文件。然后就可以通过 EDK2 提供的源码和工具开发 UEFI 应用和驱动了。

3）编译 UEFI 模拟器和 UEFI 工程。

4）运行 UEFI 模拟器。

下面详细介绍配置 Linux 开发环境的这几个步骤。

2.2.1 安装所需开发工具

1）安装 gcc：在 Ubuntu 下可以使用如下命令安装 gcc。

```
EDK2:$apt-get install gcc
```

2）下载 EDK2 开发包。

① 可以从 TianoCore 官方网站下载 EDK2 发行版：

https://sourceforge.net/projects/edk2/files/UDK2014_Releases/UDK2014/UDK2014.Complete.MyWorkSpace.zip/download

② 也可以用代码管理工具获得最新源码。

3）用 subversion 命令行获得 EDK2 源码：

```
EDK2:$svn co https://svn.code.sf.net/p/edk2/code/trunk/edk2edk2
```

或用 git 命令行下载源码：

```
EDK2:$git clone https://github.com/tianocore/edk2
```

2.2.2 配置 EDK2 开发环境

1）编辑 Conf/target.txt。根据用户的编译器环境修改编译工具 TOOL_CHAIN_TAG。此处以 x86_32 Ubuntu 的 gcc 4.4 为例，需设置 TOOL_CHAIN_TAG=GCC44，如图 2-8 所示。

Conf/target.txt

```
ACTIVE_PLATFORM         = UnixPkg/UnixPkg.dsc
TARGET                  = DEBUG
TARGET_ARCH             = IA32
TOOL_CHAIN_CONF         = Conf/tools_def.txt
TOOL_CHAIN_TAG          =GCC44
MAX_CONCURRENT_THREAD_NUMBER = 1
BUILD_RULE_CONF = Conf/build_rule.txt
```

图 2-8 Conf/target.txt 文件，操作系统为 x86_32 Ubuntu，编译器为 gcc 4.4

2）检查 Conf\tools_def.txt（见图 2-9），确保编译器路径正确。

Conf/tools_def.txt

```
DEFINE GCC44_IA32_PREFIX     = /usr/bin/
DEFINE GCC44_X64_PREFIX      = /usr/bin/

DEFINE GCC45_IA32_PREFIX     = /usr/bin/
DEFINE GCC45_X64_PREFIX      = /usr/bin/

DEFINE GCC46_IA32_PREFIX     = /usr/bin/
```

图 2-9　Conf\tools_def.txt 文件 GCC44 相关部分，该文件定义了编译器的路径

3）编译 EDK2 工具链。

Linux 环境里 EDK2 工具链位于 BaseTools/BinWrappers/PosixLike/ 目录下。EDK2 工具链包括 build、GenFv、GenFw 等工具。如果该目录下没有这些工具，就要编译 BaseTools 来获得这些工具。

要编译 BaseTools，首先需保证系统中已经安装了 make、gcc 等编译工具。下面的命令用于安装编译 BaseTools 所需的编译工具：

```
EDK2$sudo apt-get install build-essential uuid-dev
```

然后进入 BaseTools 目录并使用 make 命令编译，如下所示：

```
EDK2$ cd BaseTools
EDK2/BaseTools$ make
```

编译后，BaseTools/BinWrappers/PosixLike/ 目录如图 2-10 所示。

```
EDK2$ls BaseTools/BinWrappers/PosixLike/
BootSectImage  EfiRom     GenFv             GenVtf             Split          VfrCompile
BPDG           GenCrc32   GenFw             GnuGenBootSector   TargetTool     VolInfo
build          GenDepex   GenPage           LzmaCompress       TianoCompress
Ecc            GenFds     GenPatchPcdTable  LzmaF86Compress    Trim
EfiLdrImage    GenFfs     GenSec            PatchPcdValue      UPT
EDK2$
```

图 2-10　Linux 环境下 EDK2 工具链

2.2.3　编译 UEFI 模拟器和 UEFI 工程

UEFI 开发环境已经配置完毕，下面就可以编译 UEFI 模拟器和 UEFI 应用程序了。

1）首先设置环境变量。打开命令行工具，并进入 EDK2 目录，然后执行 source edksetup.sh，如下所示：

```
EDK2$ source edksetup.sh
```

2）环境变量配置好后就可以编译 UFEI 模拟器和其他 UEFI 工程了。

① 编译 UEFI 模拟器。

Linux 下的模拟器是 UnixPkg，编译模拟器就是编译 UnixPkg。可以使用 build 命令或带 -p 参数的 build 命令编译 UnixPkg，如下所示：

```
EDK2$build
```

或

```
EDK2$build -p UnixPkg/UnixPkg.dsc
```

编译后，模拟器 SecMain 将输出到 EDK2/Build/Unix/DEBUG_GCC44/IA32 目录。

build 命令的用法在 2.1 节配置 Windows 开发环境时已经做了详细说明，Linux 环境下 build 命令与之完全相同，在此不再重复说明。

② 编译其他 UEFI 工程。

编译其他 UEFI 工程也是通过 build 命令完成的。下面是使用 build 命令编译 Shell.efi 的示例。

【示例 2-1】 用如下命令可以编译 32 位的 Shell。编译后，Shell.efi 存放于 Build/Unix/DEBUG_GCC44/IA32/ 目录下。

```
EDK2$build -a IA32 -p ShellPkg/ShellPkg.dsc -m ShellPkg/Application/Shell/
    Shell.inf
```

【示例 2-2】 用如下命令可以编译 64 位的 Shell。编译后，Shell.efi 存放于 Build/Unix/DEBUG_GCC44/X64/ 目录下。

```
EDK2$build -a X64 -p ShellPkg/ShellPkg.dsc -m ShellPkg/Application/Shell/Shell.inf
```

2.2.4 运行模拟器

有两种方式可以运行 UEFI 模拟器。

第一种方式是在 Linux 图形界面的命令行中运行 build run 命令。

```
EDK2$build run
```

第二种方式是执行 EDK2/Build/Unix/DEBUG_GCC44/IA32 目录下的 SecMain 命令，如下所示。

```
EDK2$Build/Unix/DEBUG_GCC44/IA32/SecMain
```

图 2-11 是 Linux 下的 UEFI 模拟器。

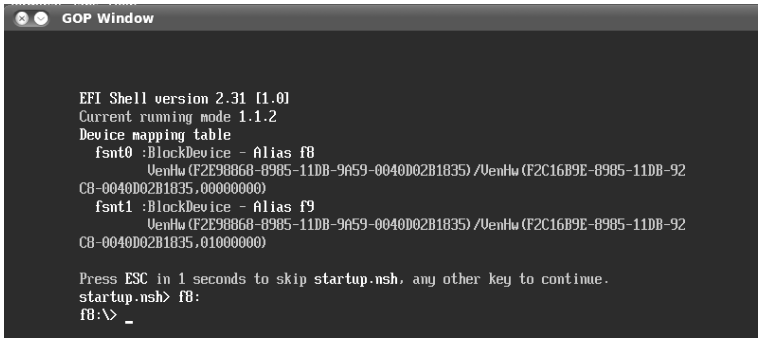

图 2-11　Linux 下的 UEFI 模拟器

2.3　OVMF 的制作和使用

OVMF(Open Virtual Machine Firmware，开放虚拟机固件）是用于虚拟机上的 UEFI 固件。在开发过程中，我们需要不断地测试所开发的产品。在模拟器中测试非常方便，但模拟器功能有限，并且模拟器只能测试 32 位程序。另外，在真实的 UEFI 环境中，测试又往往比较烦琐。在虚拟机中测试无疑是一种方便、快捷的方式，它既能较好地模拟真实环境，又可以做到快速、方便。EDK2 提供了制作虚拟机固件的方法，称为 OVMF。下面介绍如何编译和使用虚拟机固件。

1. 制作 OVMF

编译 OVMF 包，分如下两种情况。

1）用如下命令编译 64 位 OVMF 固件。

```
C:\EDK2>build -a X64 -p OvmfPkg\OvmfPkgX64.dsc
```

编译后的固件为 edk2\Build\OvmfX64\DEBUG_VS2008x86\FV\ 目录下的 OVMF.fd。

2）用如下命令编译 32 位 OVMF 固件。

```
C:\EDK2>build -a IA32 -p OvmfPkg\OvmfPkgIa32.dsc
```

编译后的固件为 edk2\Build\OvmfIa32\DEBUG_VS2008x86\FV\ 目录下的 OVMF.fd。

2. 在 QEMU 虚拟机中使用 OVMF

QEMU 是目前广泛使用的计算机仿真器和虚拟机。在 QEMU 虚拟机中，用户可以使用自定义的固件，利用这个特性我们可以测试 OVMF。首先，将存放于 edk2\Build\OvmfPkgIa32\DEBUG_VS2008x86\FV 下的文件 OVMF.fd 复制到 QEMU 的安装目录。然

后，在 QEMU 虚拟机中加载 OVMF，有两种方式：一种是使用 QEMU Manager（管理器）运行 QEMU 虚拟机，在图形界面中指定 OVMF 固件；另一种是使用 QEMU 命令行指定 OVMF 固件。

（1）使用 QEMU Manager

在 QEMU Manager 控制面板选择 Advanced → BIOS Filename，然后选择 OVMF.fd，如图 2-12 所示。

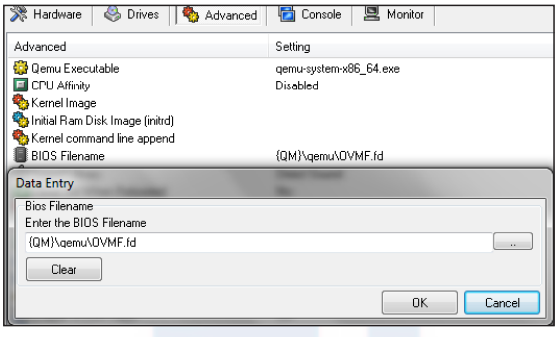

图 2-12　在 QEMU Manager 中选择固件 OVMF.fd

运行虚拟机，运行后将进入 UEFI Shell 界面，如图 2-13 所示。

（2）在 CMD 命令行运行 QEMU 命令

在 CMD 命令行运行如下 QEMU 命令。

```
C:\Program files\Qemu\>qemu-system-x86_64.exe-bios "OVMF.fd" -M "pc" -m 256 -cpu
    "qemu64" -vga cirrus -serial vc -parallel vc -name "UEFI" -boot order=dc
```

该命令运行后同样会进入图 2-13 所示的 UEFI Shell 界面。

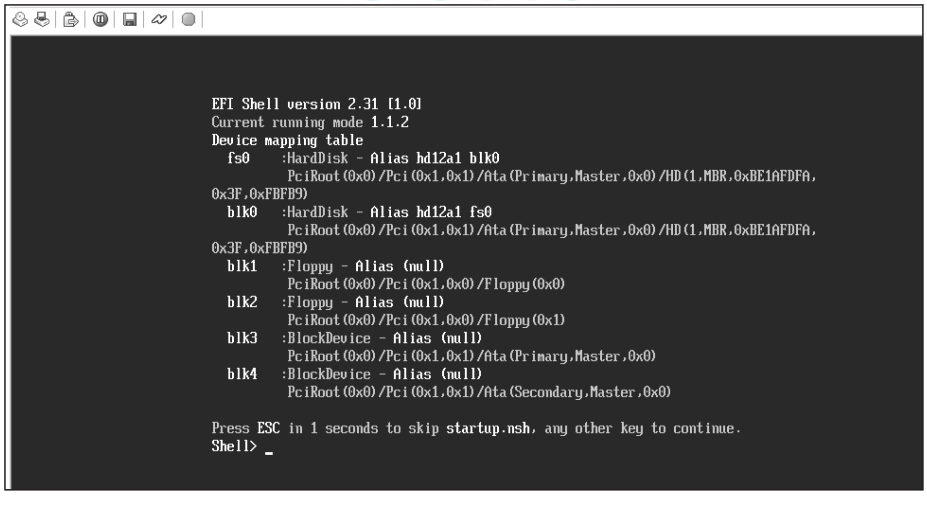

图 2-13　基于固件 OVMF.fd 的虚拟机

2.4　UEFI 的启动

虽然现在已经进入了 UEFI 时代，但是因为 UEFI 刚刚开始取代 BIOS，目前依然有很多运行着的 BIOS 系统。为了方便开发者从 BIOS 转移到 UEFI，EDK2 提供了 DUET，用于在 BIOS 系统上模拟 UEFI 执行环境。下面介绍一下 DUET。

1. DUET 简介

DUET（Developer's UEFI Emulation）是基于 Legacy BIOS 系统的 UEFI 模拟器，主要为 UEFI 开发者提供一个在传统 BIOS 系统上的 UEFI 运行环境。DUET 支持基于 MBR 的启动方式，从 MBR 启动后进入 UEFI 执行环境。

下面开始制作传统 BIOS 平台下的模拟 UEFI 启动盘。

如果目标平台是 Legacy BIOS，则需要在 U 盘中制作 MBR 和引导文件，可按如下步骤来进行。

1）编译 DuetPkg，在 CMD 命令行输入如下命令。

```
C:\EDK2>build -a IA32 -p DuetPkg\DuetPkgIa32.dsc
```

或者

```
C:\EDK2>build -a X64 -p DuetPkg\DuetPkgX64.dsc
```

2）通过如下命令生成引导文件。

```
C:\EDK2>  cd DuetPkg
C:\EDK2\DuetPkg>postbuild.bat Ia32
```

或者

```
C:\EDK2\DuetPkg>postbuild.bat X64
```

3）插入 U 盘，假设 J: 是 U 盘盘符，通过如下命令向 U 盘写入 MBR。

```
C:\EDK2\DuetPkg>  createbootdisk usb J: FAT32 IA32
```

或者

```
C:\EDK2\DuetPkg>  createbootdisk usb J: FAT32 X64
```

4）拔出并重新插入 U 盘，通过如下命令向 U 盘复制 UEFI 文件。

```
C:\EDK2\DuetPkg>   createbootdisk usb J: FAT32 IA32 step2
```

或者

```
C:\EDK2\DuetPkg>  createbootdisk usb J: FAT32 X64 step2
```

此命令向 U 盘根目录复制了 efildr20（该文件用于引导系统进入 UEFI 环境），并向 efi\boot 目录复制了引导文件 bootia32.efi（源文件位于 ShellBinPkg\UefiShell\Ia32\Shell.efi）或 bootx64.efi（源文件位于 ShellBinPkg\UefiShell\X64\Shell.efi）。

总结一下，要制作传统 BIOS 平台下的模拟 UEFI 启动盘（64 位），需在 CMD 命令行执行如下命令。

```
C:\EDK2>build -a X64 -p DuetPkg\DuetPkgX64.dsc
C:\EDK2>  cd DuetPkg
C:\EDK2\DuetPkg>postbuild.bat X64
C:\EDK2\DuetPkg>  createbootdisk usb J: FAT32 X64
```

拔出并重新插入 U 盘，继续执行如下命令。

```
C:\EDK2\DuetPkg>  createbootdisk usb J: FAT32 X64 step2
```

32 位模拟 UEFI 启动盘与 64 位相似，此处不再赘述。

有时需要使用最新的 Shell，此时要编译 ShellPkg，然后将编译好的 Shell.efi 复制到 U 盘 efi\boot 目录并重命名为引导文件（bootia32.efi 或 bootx64.efi）。

接下来就可以用 U 盘来引导进入 UEFI 世界了。

2. 制作 UEFI 平台下的 USB 启动盘

如果目标平台是 UEFI 平台，那么启动盘的制作就变得非常简单，可按如下步骤来进行。

1）格式化 U 盘为 FAT（FAT、FAT16 或 FAT32）格式。

2）在 U 盘上建立目录 efi\boot。

3）将 efi 的应用程序复制到 efi\boot 目录，并改名为 bootx64.efi 或者 bootia32.efi。

与 Legacy BIOS 需要 MBR 来引导操作系统不同，UEFI 的启动文件是 FAT 盘内 efi\boot 目录中的 bootx64.efi 或 bootia32.efi。

2.5 本章小结

本章主要是为开发 UEFI 应用和驱动准备开发环境，主要内容包括：

- Windows 和 Linux 下的 UEFI 开发环境的搭建（主要是 EDK2 的配置）。
- build 命令的用法。
- 制作和使用 OVMF 固件。
- 利用 DUET 制作传统 BIOS 系统上的 UEFI 启动盘。

第 3 章我们将开始介绍 UEFI 工程模块，主要包括 UEFI 应用程序工程、UEFI 驱动工程和 UEFI 类库工程。