第 1 章

FFmpeg 环境搭建

本章介绍如何在Linux和Windows系统搭建FFmpeg（Fast Forward Moving Picture Expert Group）的开发环境，内容包括FFmpeg的用途及其发展历程、如何在Linux系统安装FFmpeg、如何在Windows系统安装FFmpeg、FFmpeg框架中的可执行程序与动态链接库，最后演示如何通过FFmpeg框架打印Hello World程序。

1.1 FFmpeg 简介

本节介绍FFmpeg开源平台的简单背景，包括FFmpeg名称的来源及其开发环境、FFmpeg的主要用途与支持的格式、FFmpeg的发展历程和许可说明等。

1.1.1 FFmpeg 是什么

FFmpeg的意思是快速掌握MPEG。其中MPEG全称为Moving Picture Expert Group，中文直译过来叫运动图像专家组格式。MPEG是一种流行的视频格式系列，目前常见的是第4版MPEG，即MPEG-4，采用MPEG-4格式的视频文件扩展名为.mp4。

缩写之后的FFmpeg拼读有点麻烦，前面三个字母依次按字母逐个念，后面三个字母连起来按照单词peg读。连起来的FFmpeg读音便是[ef ef em peg]。

FFmpeg是一个基于C语言的开源框架，它虽然在Linux环境下开发，但兼容其他操作系统，包括Windows、Mac OS X等。也就是说，把FFmpeg源码移植到Windows环境，也能正常编译和运行。

1.1.2 FFmpeg 的用途

FFmpeg是一个音视频处理平台，不仅能够处理音频、视频文件，还能够处理图像、字幕等文件。这里的处理动作包括查看参数、读取数据、保存数据、格式转换、加工编辑、渲染播放等交互操作，可谓是功能强悍。

基于FFmpeg的开源特性，它支持越来越多的媒体格式，无论来自哪个厂商或者哪个国家，总能找到对应的支持库。单就视频格式而言，从古老的ASF、RM、FLV，到后续的H264、H265、VP8、VP9，甚至中国的音视频标准编码AVS2，FFmpeg均可集成它们的编解码库。图像方面支持JPG、

PNG、GIF等常见的图片格式，音频方面支持MP3、WMA、AAC等常见的音频格式，甚至字幕格式SRT、ASS等都纳入了FFmpeg的支持范围。

FFmpeg的音视频处理功能非常强大，因此它已成为全球音视频开发者的首选框架。无论是国外的Adobe Premiere，还是国产的剪映，它们的音视频编辑功能都是基于FFmpeg实现的。可以说，学好FFmpeg编程是从事音视频开发行业的必备技能之一。

1.1.3 FFmpeg 的发展历程

FFmpeg最早由法国的天才程序员Fabrice Bellard发起，并在2004年由Michael Niedermayer主持维护，一直到2015年Michael Niedermayer宣布辞职。FFmpeg项目的官方网站地址为https://www.ffmpeg.org/，最新源码的下载页面为https://www.ffmpeg.org/download.html，也可访问https://github.com/FFmpeg/FFmpeg/tags获取各版本FFmpeg的源码下载。

2000年，FFmpeg的第一个版本发布。
2013年7月，FFmpeg 2.0发布。
2016年2月，FFmpeg 3.0发布，增加内置AAC编解码器。
2018年4月，FFmpeg 4.0发布，不再支持Windows XP，最低支持到Windows Vista。
2018年11月，FFmpeg 4.1发布，增加支持AVS2国标的编解码器。
2019年8月，FFmpeg 4.2发布，增加支持AV1视频和VP4视频的解码。
2020年6月，FFmpeg 4.3发布，增加支持转场滤镜xfade，全面支持Vulkan。
2021年4月，FFmpeg 4.4发布，增加支持AVS3国标的解码器。
2022年1月，FFmpeg 5.0发布，增加支持国产的龙芯架构。
2022年7月，FFmpeg 5.1发布，增加支持IPFS/IPNS协议。
2023年2月，FFmpeg 6.0发布，增加支持RGBE和WBMP两种图像格式。

FFmpeg采用LGPL或GPL许可证，其中GPL 2.0禁止商用。而LGPL允许开发闭源的商用软件，不过只能使用FFmpeg的动态库，并且需要标明用到了FFmpeg动态库；不允许把FFmpeg的静态库链接到商用软件中，除非这个软件也开放源码。

1.2 Linux 系统安装 FFmpeg

本节介绍在Linux系统搭建FFmpeg开发环境的具体步骤，包括FFmpeg开发对于Linux系统的软硬件要求、如何在Linux上安装已编译的FFmpeg套件、如何在Linux上自行编译与安装FFmpeg等。

1.2.1 Linux 开发机配置要求

FFmpeg本身就在Linux系统下开发，而且C代码向来以高效著称，因此对硬件方面的配置要求较低，主要对软件环境有要求。对Linux服务器的基本要求如下：

（1）编译器要求GNU Make 3.81及以上版本。
（2）GCC要求4.8及以上版本。

（3）磁盘剩余空间在5GB以上，越大越好。

由于个人计算机较少安装Linux系统，因此建议读者在云服务环境上实践FFmpeg编程。

以上服务器配置要求很低，国内的云服务提供商都能满足。以华为云的最低档配置为例，编译器为GNU Make 4.3，GCC版本为10.3.1，磁盘空间为40GB，完全能够满足入门FFmpeg开发的环境要求。本书配套的C代码在华为云的EulerOS（欧拉系统，兼容CentOS）上调试通过。

1.2.2　安装已编译的 FFmpeg 及其 SO 库

对于初学者来说，搭建FFmpeg的开发环境是个不小的拦路虎，因为FFmpeg用到了许多第三方开发包，所以要先编译这些第三方源码，之后才能为FFmpeg集成编译好的第三方库。

考虑到初学者刚开始仅仅调用FFmpeg的函数，不会马上修改FFmpeg的源码，因此只要给系统安装编译好的FFmpeg动态库，即可着手编写简单的FFmpeg程序。比如这个网站：https://github.com/BtbN/FFmpeg-Builds/releases提供了已经编译通过的FFmpeg开发包，囊括Linux、Windows等系统环境的开发版本。对该网站提供的Linux版FFmpeg安装包而言，需要事先安装不低于2.22版本的glibc库，否则编译FFmpeg程序会报错：undefined reference to '_ZGVdN4vv_pow@GLIBC_2.22'。下面介绍在Linux系统安装已编译的FFmpeg的详细步骤。

1. 安装glibc

因为GitHub编译好的FFmpeg依赖于glibc库，所以要在Linux环境安装版本号不低于2.22的glibc库。首先执行以下命令查看当前已安装的glibc版本。

```
rpm -qa | grep glibc
```

如果上面的命令返回的版本号不低于2.22，就无须再安装新版的glibc。只有返回的版本号低于2.22，才需要安装新版的glibc库。在华为云上实测发现，欧拉系统（Huawei Cloud EulerOS）自带的glibc版本号大于2.22，因此无须另外安装glibc库。

glibc库的安装步骤说明如下。

01 到https://ftp.gnu.org/gnu/glibc/下载2.23版本的glibc源码包。注意：虽然要求glibc版本不低于2.22，但是不宜安装过高版本的glibc，因为较高版本的glibc依赖于Python，安装Python环境又得费一番功夫，所以安装比2.22稍高一点的2.23版本就够了，也就是在https://ftp.gnu.org/gnu/glibc/glibc-2.23.tar.gz下载压缩包。

02 先解压glibc源码包，再进入glibc源码目录，然后创建build目录并进入该目录，也就是依次执行以下命令：

```
tar zxvf glibc-2.23.tar.gz
cd glibc-2.23
mkdir build
cd build
```

03 在build目录下依次执行以下命令配置、编译与安装glibc。

```
../configure --prefix=/usr
make
make install
```

安装成功后，会在/usr/lib64目录下找到新的libc.so（还有libc.so.6和libc-2.23.so）和libmvec.so（还有libmvec.so.1和libmvec-2.23.so）等库文件。

2. 安装FFmpeg

安装glibc库后，再来安装已编译的FFmpeg，详细的安装步骤说明如下。

01 到https://github.com/BtbN/FFmpeg-Builds/releases下载Linux环境编译好的FFmpeg安装包，比如ffmpeg-master-latest-linux64-gpl-shared.tar.xz，注意区分32位系统和64位系统。

02 把下载好的FFmpeg安装包解压到/usr/local/ffmpeg目录，也就是依次执行以下命令：

```
cd /usr/local
tar xvf ffmpeg-master-latest-linux64-gpl-shared.tar.xz
mv ffmpeg-master-latest-linux64-gpl-shared ffmpeg
```

03 输入cd命令回到当前用户的初始目录，使用vi打开该目录下的.bash_profile，也就是依次执行以下命令：

```
cd
vi .bash_profile
```

04 把光标移动到文件末尾，按a键进入编辑模式，然后在文件末尾添加下面两行环境变量的配置：

```
export PATH=$PATH:/usr/local/ffmpeg/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/ffmpeg/lib
```

接着保存并退出文件，也就是先按Esc键退出编辑模式，再按“:”键，接着输入wq再按回车键，即可保存修改后的内容。

05 执行以下命令加载新的环境变量：

```
source .bash_profile
```

接着运行下面的环境变量查看命令：

```
env | grep PATH
```

发现控制台回显的PATH串包含/usr/local/ffmpeg/bin，同时LD_LIBRARY_PATH串包含/usr/local/ffmpeg/lib，说明FFmpeg的bin目录和lib目录的路径都已经加载到环境变量中了。

3. 运行测试命令

运行以下命令查看FFmpeg的版本信息：

```
ffmpeg -version
```

发现控制台回显如下的FFmpeg版本号，以及编译时的配置参数信息，说明FFmpeg程序成功运行起来了。

```
ffmpeg version N-110091-g261fb55e39-20230326 Copyright (c) 2000-2023 the FFmpeg
developers
built with gcc 12.2.0 (crosstool-NG 1.25.0.152_89671bf)
```

1.2.3　自行编译与安装 FFmpeg

由于别人编译的FFmpeg不可控，因此最好自己编译FFmpeg。尽管编译过程有点麻烦，但是整个安装过程都是受控的，更加符合开发者的定制需求。不过要在Linux系统编译FFmpeg，读者需要具备一定的Linux命令基础，Linux环境的FFmpeg编译与安装过程说明如下。

1. 安装NASM

NASM是一款汇编工具，因为x264库和x265库都依赖于该库，所以要给Linux系统安装NASM库。它的安装方式有两种，一种是使用yum直接安装，另一种是通过编译源码来安装。使用yum安装NASM的命令如下：

```
yum install nasm
```

如果使用yum安装NASM失败，就要通过编译源码来安装，源码方式的安装步骤说明如下。

01 到https://www.nasm.us/pub/nasm/releasebuilds/下载最新的NASM源码，比如2022年12月发布的nasm-2.16，该版本的源码下载地址是https://www.nasm.us/pub/nasm/releasebuilds/2.16/nasm-2.16.tar.gz。将下载好的压缩包上传到服务器并解压，也就是依次执行以下命令：

```
tar zxvf nasm-2.16.tar.gz
cd nasm-2.16
```

02 进入解压后的NASM目录，运行下面的命令配置NASM：

```
./configure
```

03 运行下面的命令编译NASM。

```
make
```

04 编译完成后，运行下面的命令安装NASM。

```
make install
```

2. 安装x264库

H.264格式的视频编解码用到了x264库，它的安装步骤说明如下。

01 到https://www.videolan.org/developers/x264.html下载最新的x264源码，比如新版的源码下载地址是https://code.videolan.org/videolan/x264/-/archive/master/x264-master.tar.gz。将下载好的压缩包上传到服务器并解压，也就是依次执行以下命令：

```
tar zxvf x264-master.tar.gz
cd x264-master
```

02 进入解压后的x264目录，运行下面的命令配置x264：

```
./configure --enable-shared --enable-static
```

03 运行下面的命令编译x264：

```
make
```

04 编译完成后，运行下面的命令安装x264：

```
make install
```

3. 安装CMake

H.265视频的编解码用到了x265库，因为x265库使用CMake编译，所以要事先在Linux下安装CMake工具。它的安装方式有两种，一种是使用yum直接安装，另一种是通过编译源码来安装。使用yum安装CMake的命令如下：

```
yum install cmake git
```

如果使用yum安装CMake失败，就要通过编译源码来安装，源码方式的安装步骤说明如下。

01 运行下面的命令，分别安装g++、openssl-devel和curl-devel，因为CMake依赖于这三个工具包。在安装过程中会提示[Y/n]确认是否继续安装，此时输入Y确定安装即可。

```
yum install g++ openssl-devel curl-devel
```

另外，还需执行下面的命令安装Git，虽然CMake不依赖于Git，但是只有安装了Git才能正常编译x265的动态库。

```
yum install git
```

02 到https://cmake.org/files/下载最新的CMake源码，比如2023年8月发布的cmake-3.27.3，该版本的源码下载地址是https://cmake.org/files/v3.27/cmake-3.27.3.tar.gz。将下载好的压缩包上传到服务器并解压，也就是依次执行以下命令：

```
tar zxvf cmake-3.27.3.tar.gz
cd cmake-3.27.3
```

03 进入解压后的CMake目录，运行下面的命令配置CMake。

```
./bootstrap  --system-curl
./configure
```

04 运行下面的命令编译CMake。

```
make
```

05 编译完成后，运行下面的命令安装CMake。

```
make install
```

注意以上是通过源码方式安装CMake，如果发现源码编译失败，就只能直接安装编译好的CMake程序。详细步骤说明如下：

01 到https://cmake.org/files/下载最新的CMake安装包，比如2023年8月发布的cmake-3.27.3，Linux 64位x86环境的安装包下载地址是https://cmake.org/files/v3.27/cmake-3.27.3-linux- x86_64.tar.gz。将下载好的压缩包上传到服务器并解压，也就是依次执行以下命令：

```
mv cmake-3.27.3-linux-x86_64.tar.gz /usr/local/
cd /usr/local/
tar zxvf cmake-3.27.3-linux-x86_64.tar.gz
mv cmake-3.27.3-linux-x86_64 cmake
```

02 给环境变量PATH添加CMake解压后的bin目录。输入cd命令回到当前用户的初始目录，使用vi打开该目录下的.bash_profile，也就是依次执行以下命令：

```
cd
vi .bash_profile
```

把光标移动到文件末尾，按a键进入编辑模式，然后在文件末尾添加下面一行环境变量的配置：

```
export PATH=$PATH:/usr/local/cmake/bin
```

接着保存并退出文件，也就是先按Esc键退出编辑模式，再按“:”键，接着输入wq再按回车键，即可保存修改后的内容。然后执行以下命令加载最新的环境变量：

```
source .bash_profile
```

03 运行以下命令检查CMake的版本号：

```
cmake -version
```

4. 安装x265库

H.265格式的视频编解码用到了x265库，它的安装步骤说明如下。

01 到https://bitbucket.org/multicoreware/x265_git/downloads/下载最新的x265源码，比如2021年3月发布的x265_3.5，该版本的源码下载地址是https://bitbucket.org/multicoreware/x265_git/downloads/x265_3.5.tar.gz。将下载好的压缩包上传到服务器并解压，也就是依次执行以下命令：

```
tar zxvf x265_3.5.tar.gz
cd x265_3.5
```

02 进入解压后的x265目录的build目录，运行以下命令配置x265库：

```
cd build
cmake ../source
```

03 运行以下命令编译x265：

```
make
```

04 编译完成后，运行以下命令安装x265：

```
make install
```

5. 加载环境变量PKG_CONFIG_PATH

在前面的安装步骤中，x264和x265的.pc文件都安装到了/usr/local/lib/pkgconfig目录，因为FFmpeg正是根据这些.pc文件来查找对应的第三方配置，所以需要把该路径添加到环境变量PKG_CONFIG_PATH中，方便FFmpeg自动查找.pc文件，详细的加载步骤说明如下。

01 输入cd命令回到当前用户的初始目录，使用vi打开该目录下的.bash_profile，也就是依次执行以下命令：

```
cd
vi .bash_profile
```

02 把光标移动到文件末尾，按a键进入编辑模式，然后在文件末尾添加下面一行环境变量的配置：

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
```

接着保存并退出文件，也就是先按Esc键退出编辑模式，再按“:”键，接着输入wq再按回车键，即可保存修改后的内容。

03 执行以下命令加载最新的环境变量：

```
source .bash_profile
```

接着运行下面的环境变量查看命令：

```
env | grep PKG_CONFIG_PATH
```

发现控制台回显的PKG_CONFIG_PATH串包含/usr/local/lib/pkgconfig，说明pkgconfig目录的路径已加载到环境变量中了。

6. 安装FFmpeg

以上工具都安装之后，再来安装FFmpeg，详细的安装步骤说明如下。

01 到https://github.com/FFmpeg/FFmpeg/tags下载指定版本的FFmpeg源码包，比如2022年9月发布的FFmpeg-5.1.2，该版本的源码下载地址是https://github.com/FFmpeg/FFmpeg/archive/refs/tags/n5.1.2.tar.gz。将下载好的压缩包上传到服务器并解压，也就是依次执行以下命令：

```
tar zxvf FFmpeg-n5.1.2.tar.gz
cd FFmpeg-n5.1.2s
```

02 进入解压后的FFmpeg目录，运行以下命令配置FFmpeg：

```
./configure --prefix=/usr/local/ffmpeg --enable-shared --disable-static --disable-doc
--enable-zlib --enable-libx264 --enable-libx265 --enable-iconv --enable-gpl
--enable-nonfree
```

注意configure后面的配置选项中，enable表示开启某项功能，disable表示关闭某项功能，常见的配置选项说明见表1-1。

表 1-1 FFmpeg 的配置命令选项说明

配置选项	说　明
--prefix=***	指定安装路径的目录前缀
--enable-shared	有编译动态链接库（Linux 系统为.so 文件，Windows 系统为.dll 文件）
--disable-static	不编译静态库，静态库为.a 文件
--disable-doc	不安装说明文档
--enable-gpl	允许使用基于 GPL（GNU General Public License，GNU 通用公共许可协议）的代码
--enable-nonfree	允许使用非自由代码
--enable-iconv	启用字符集编码转换库 iconv
--enable-zlib	启用 DEFLATE 压缩算法库 zlib，PNG 图片需要
--enable-libx264	启用 H.264 格式的编解码库 x264
--enable-libx265	启用 H.265 格式的编解码库 x265
--enable-libxavs2	启用 AVS2 格式的编码库 xavs2
--enable-libdavs2	启用 AVS2 格式的解码库 davs2

（续表）

配置选项	说　　明
--enable-libmp3lame	启用 MP3 格式的编解码库 mp3lame
--enable-libfreetype	启用字体引擎 FreeType，drawtext 滤镜需要
--enable-libass	启用字幕渲染器 libass，subtitles 滤镜需要
--enable-libfribidi	启用双向字符算法库 fribidi
--enable-libxml2	启用 XML 文档处理库 libxml2
--enable-fontconfig	启用字体配置工具 fontconfig
--enable-sdl2	启用音视频渲染库 SDL2
--extra-cflags="-I***"	额外的头文件路径***
--extra-ldflags="-L***"	额外的动态链接库路径***
--arch=x86_64	交叉编译时需要，指定处理器架构为 64 位的 x86
--cross-prefix=x86_64-w64-mingw32-	交叉编译时需要，交叉编译的工具名称前缀
--target-os=mingw32	交叉编译时需要，指定编译结果的目标操作系统

03 运行以下命令编译FFmpeg：

```
make
```

如果觉得make命令编译得太慢，可改成执行以下命令编译FFmpeg，“-j4”表示启动4个编译进程。在系统资源紧张的时候，多进程编译可能报错，此时再改回make命令即可。

```
make -j4
```

04 编译完成后，运行以下命令安装FFmpeg：

```
make install
```

05 把/usr/local/ffmpeg/bin添加到环境变量PATH中，把/usr/local/lib和/usr/local/ffmpeg/lib都添加到环境变量LD_LIBRARY_PATH中。也就是依次执行以下命令：

```
cd
vi .bash_profile
```

把光标移动到文件末尾，按a键进入编辑模式，然后在文件末尾添加下面三行环境变量的配置：

```
export PATH=$PATH:/usr/local/ffmpeg/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/ffmpeg/lib
```

接着保存并退出文件，也就是先按Esc键退出编辑模式，再按“:”键，接着输入wq再按回车键，即可保存修改后的内容。

然后执行以下命令加载最新的环境变量：

```
source .bash_profile
```

接着运行下面的环境变量查看命令：

```
env | grep PATH
```

发现控制台回显的PATH串包含/usr/local/ffmpeg/bin，同时LD_LIBRARY_PATH串包含/usr/local/lib和/usr/local/ffmpeg/lib，说明FFmpeg的bin目录和lib目录的路径都已经加载到环境变量中了。

06 运行以下命令查看FFmpeg的版本信息：

```
ffmpeg -version
```

发现控制台回显如下的FFmpeg版本号，以及编译时的配置参数信息，说明FFmpeg程序成功运行起来了。

```
ffmpeg version 5.1.2 Copyright (c) 2000-2022 the FFmpeg developers
built with gcc 10.3.1 (GCC)
configuration: --prefix=/usr/local/ffmpeg --enable-shared --disable-static
--disable-doc --enable-zlib --enable-libx264 --enable-libx265 --enable-iconv --enable-gpl
--enable-nonfree
```

1.3 在 Windows 系统下安装 FFmpeg

本节介绍如何在Windows系统下搭建FFmpeg开发环境，包括FFmpeg开发对于Windows系统的软硬件要求，如何在Windows上安装FFmpeg依赖的软件，以及如何在Windows上安装已编译的FFmpeg套件等。

1.3.1 Windows 开发机配置要求

工欲善其事，必先利其器。要想保证FFmpeg的编译和运行速度，开发用的计算机配置就要跟上。使用Windows系统开发FFmpeg的话，对计算机硬件的基本要求如下。

（1）内存要求至少8GB，越大越好。
（2）CPU要求1.5GHz以上，越快越好。
（3）硬盘要求安装盘剩余空间10GB以上，越大越好。

下面是对操作系统的基本要求。

（1）必须是64位系统，不能是32位系统。
（2）Windows系统最低为Windows 10版本，本书配套的C代码在Windows 10上调试通过。

下面是对网络的基本要求。

（1）最好连公众网，因为校园网可能无法访问GitHub等网站。
（2）下载速度至少每秒1MB，越快越好。因为Visual Studio 2022与MSYS2的安装包大小都有几百MB，所以网络带宽一定要够大，否则连下载文件都要等很久。

1.3.2 安装依赖的 Windows 软件

虽然各大云厂商都提供了基于Linux的云服务，但毕竟是收费的，还有时间限制。由于Linux系统比较专业，个人计算机很少安装Linux，反而大都安装Windows系统，因此提高了FFmpeg的学习门槛，毕竟在Windows系统搭建FFmpeg的开发环境还是比较麻烦的。

不过如果有已经编译好的Windows版本FFmpeg开发包，就免去了烦琐的Windows编译过程。

如果改成直接安装已编译的FFmpeg开发包，还是相对容易的。无论是否自己编译FFmpeg源码，Windows系统均需事先安装Visual Studio和MSYS2，下面介绍在Windows系统安装已编译的FFmpeg的详细步骤。

1. 安装Visual Studio 2022

Visual Studio是Windows系统的微软官方开发环境，可用来开发C++编码的桌面程序，一些第三方工具也要依靠Visual Studio才能编译出.dll动态库，所以在Windows系统进行C++开发离不开Visual Studio。以下是在Windows系统安装Visual Studio 2022的详细步骤。

01 到https://visualstudio.microsoft.com/zh-hans/vs/community/下载Visual Studio 2022的社区版，打开该页面后，在左边找到“下载”按钮并单击，或者打开网页后下拉到底部，单击左边的“免费下载”按钮，浏览器就开始下载社区版的安装文件。

02 双击下载好的VisualStudioSetup.exe，弹出如图1-1所示的安装向导窗口。

单击窗口右下角的“继续”按钮，等待安装程序下载安装资源，如图1-2所示。

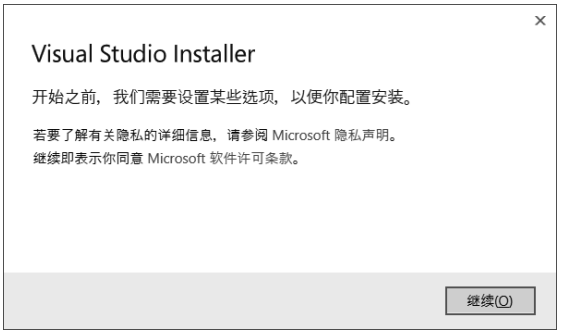

图 1-1　Visual Studio 的安装向导窗口

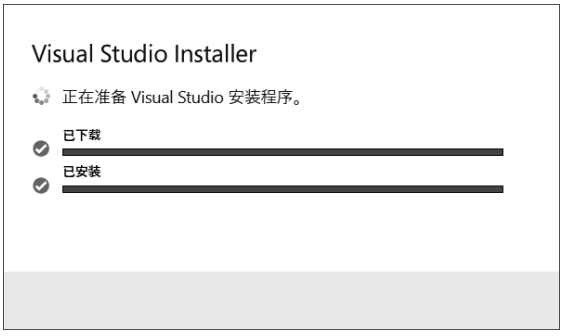

图 1-2　Visual Studio 的下载资源窗口

下载完毕会弹出完整的安装界面，如图1-3所示。

图 1-3　Visual Studio 的完整安装界面

在默认的“工作负荷”选项卡找到“使用C++的桌面开发”，勾选该复选框以便安装C++编程需要的各种插件。注意右侧的“可选”组件列表中，Windows 11 SDK（10.0.*****.0）这里要勾选不低于当前Windows版本号的SDK选项。在Windows命令行运行winver会弹出Windows版本窗口，版本窗口第二行括号中的“内部版本****”就是Windows的版本号。例如，Windows版本号为18363的话，图1-3右侧列表勾选SDK版本号大于或等于18363的任一选项均可；但是如果Windows版本号为22621，就要勾选SDK版本号大于或等于22621的选项。

单击“安装位置”选项卡，或者单击左下方的“更改”链接，可以修改Visual Studio的安装目录。然后单击安装界面右下角的“安装”按钮，转到正在安装的等待界面，如图1-4所示。

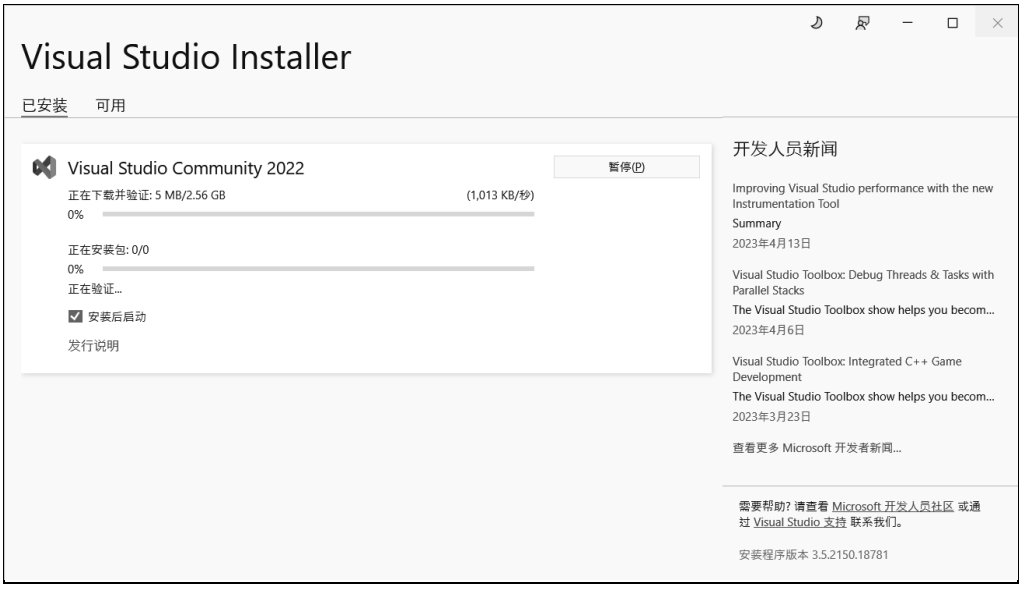

图 1-4　Visual Studio 的正在安装界面

等待安装程序的下载和安装操作，安装完毕的界面如图1-5所示。

图 1-5　Visual Studio 的安装完毕界面

接着双击Windows桌面上的Visual Studio 2022图标，弹出如图1-6所示的登录界面。

图 1-6　Visual Studio 的登录界面

单击界面右下方的“暂时跳过此项。”，跳转到如图1-7所示的设置界面。

图 1-7　Visual Studio 的设置界面

在设置界面选择合适的颜色主题，比如浅色主题，再单击右下角的“启动 Visual Studio”按钮，如图1-8所示。

图 1-8　Visual Studio 的启动界面

Visual Studio设置完成之后，弹出如图1-9所示的开始界面。

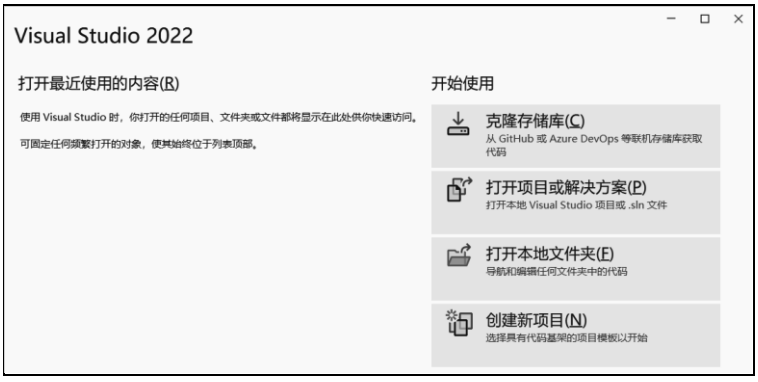

图 1-9　Visual Studio 的开始界面

至此，Visual Studio的安装过程就完成了。由于接下来的FFmpeg环境属于命令行操作，因此无须通过Visual Studio创建新项目。

2. 安装和配置MSYS2

MSYS2允许在Windows系统模拟Linux环境，它的命令行界面可以很好地仿真Linux终端，所以在Windows系统上编译和执行FFmpeg程序需要通过MSYS2的控制台操作。以下是在Windows系统安装MSYS2的详细步骤。

01 到https://github.com/msys2/msys2-installer/releases/下载MSYS2的安装包，打开该页面后，单击Assets文字以便展开安装包列表，接着单击MSYS2的Windows安装包链接，比如msys2-x86_64-20230318.exe，浏览器就开始下载Windows版本的安装文件。

02 双击下载好的msys2-x86_64-20230318.exe，弹出如图1-10所示的安装向导窗口。单击窗口下方的“下一步”按钮，转到安装目录窗口，如图1-11所示。

图 1-10　MSYS2 的安装向导窗口

图 1-11　MSYS2 的安装文件夹窗口

修改MSYS2的安装目录，单击窗口下方的下一步按钮，转到如图1-12所示的快捷方式窗口。继续单击“下一步”按钮，转到如图1-13所示的正在安装窗口。

图 1-12　MSYS2 的快捷方式窗口

图 1-13　MSYS2 的正在安装窗口

等待安装结束，转到如图1-14所示的安装完成窗口。至此，成功在Windows上安装了MSYS2。

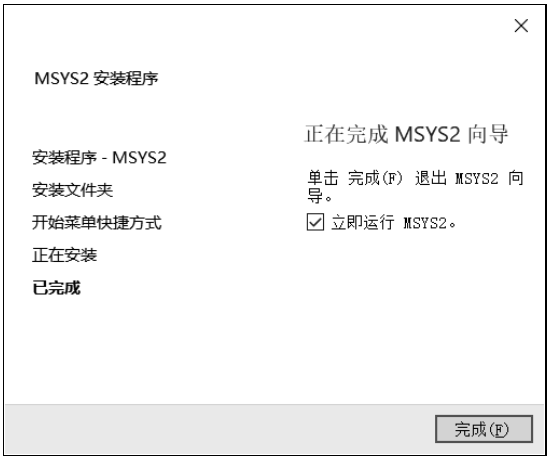

图 1-14　MSYS2 的安装完成窗口

03 MSYS2安装完毕后，打开安装目录下的msys2_shell.cmd，去掉该文件第17行代码的注释，也就是删除关键字rem，修改之后的代码如下：

```
set MSYS2_PATH_TYPE=inherit
```

之所以去掉该行代码的注释，是为了让MSYS2继承Windows系统的PATH环境变量，否则后面编译会报错“Microsoft Visual Studio support requires Visual Studio 2013 Update 2 or newer”。

1.3.3　安装已编译的 FFmpeg 及其 DLL 库

Visual Studio 2022和MSYS2都安装之后，才能在MSYS2的命令行安装FFmpeg，详细的安装过程说明如下。

1. 安装C程序编译工具

依次选择Windows开始菜单的Visual Studio 2022→x64 Native Tools Command Prompt for VS 2022，打开Visual Studio 2022的命令行界面。

然后进入msys64目录，打开MSYS2的命令行窗口，也就是依次执行以下命令（注意msys64目录要改成自己计算机上的路径）：

```
cd E:\msys64
msys2_shell.cmd -mingw64
```

之所以要在msys2_shell.cmd后面添加“-mingw64”，是为了让MinGW运行于64位模式，而非默认的32位模式。MinGW的全称是Minimalist GNU for Windows，意思是专用于Windows系统的极简版GNU工具集，它允许在Windows系统执行Linux的编译命令。如果说MSYS2相当于Windows环境的Linux模拟器，那么MinGW给MSYS2准备了32位和64位两套编译器，而我们的目标是在Windows系统模拟64位的Linux开发环境。

虽然安装了MSYS2，但是一开始里面只支持cd和ls等基本命令，竟然连gcc和make这些编译命令都没有，因此要先给它安装常用的编译工具。在MSYS2的命令行输入以下命令安装几个编译工具：

```
pacman -S gcc make nasm pkg-config diffutils zlib vim unzip
```

以上的pacman –S命令后面跟着待安装的工具名称，这些工具的详细介绍见表1-2。

表 1-2　待安装的工具说明

Pacman 安装工具的命令	说　　明
pacman -S gcc	安装 gcc 和 g++编译器
pacman -S nasm	安装 NASM 汇编工具，编译 x264 和 x265 需要这个工具
pacman -S make	安装 Makefile 文件的编译工具
pacman -S diffutils	安装比较工具，FFmpeg 库在执行 configure 命令时会用到，若不安装会警告“cmp: command not found”
pacman -S pkg-config	安装.pc 文件的配置工具
pacman -S zlib	安装 zlib 库（压缩和解压缩），PNG 图片格式的编解码需要
pacman -S vim	安装 Vim 编辑器，可在命令行下打开并修改文件
pacman -S unzip	安装解压工具，用于解压缩 ZIP 文件

Pacman在安装过程中会提示[Y/n]确认是否继续安装，此时输入Y确定安装即可。等待Pacman将编译工具安装完毕，可以在/usr/bin下找到相应的可执行程序。

另外，还要执行以下命令安装交叉编译工具链：

```
pacman -S mingw-w64-x86_64-toolchain
```

执行以上命令提示“Enter a selection (default=all):”的时候，按回车键表示安装列表上的所有组件。接着提示Proceed with installation? [Y/n]的时候，输入Y表示确认安装操作。安装好工具链可以在/mingw64/bin下找到相应的可执行程序，注意把该目录下的x86_64-w64-mingw32-gcc-ar.exe改名为x86_64-w64-mingw32-ar.exe，把x86_64-w64-mingw32-gcc-nm.exe改名为x86_64-w64-mingw32-nm.exe，也就是去掉这两个文件名中的-gcc，另外要把strip.exe改名为x86_64-w64-mingw32-strip.exe。

虽然在第8章才会用到交叉编译工具链，但该工具链应当尽早安装。因为MSYS安装之后的密钥环有效期只有150天左右，而工具链中的个别组件会检查密钥环是否有效。如果密钥环过了150天之后才安装工具链，MSYS2就会报错signature from "David Macek <david.macek.0@gmail.com>" is unknown trust，意思是签名不被信任，这便是密钥环过期导致的。一旦发现密钥环过期，此时要么

更新密钥环，要么重新安装MSYS，这两种方式都比较麻烦，所以最好在安装完MSYS后立即安装交叉编译工具链。

2. 编译与安装FFmpeg

（1）到https://github.com/BtbN/FFmpeg-Builds/releases下载Windows环境编译好的FFmpeg安装包，比如ffmpeg-master-latest-win64-gpl-shared.zip。

要下载带Win64且带Shared的压缩包，因为带Shared的包提供了头文件和DLL动态库，而不带Shared的包只有可执行程序，没法用来编程开发。

（2）把ffmpeg-master-latest-win64-gpl-shared.zip解压到指定目录，并将解压后的目录改名为ffmpeg，比如E:\msys64\usr\local\ffmpeg\。

（3）右击Windows桌面上的“此电脑”图标，在弹出的快捷菜单中选择“属性”，接着单击新页面左侧的“高级系统设置”链接，在弹出的“系统属性”窗口中单击下方的“环境变量”按钮，然后编辑系统变量列表中的PATH变量，给它添加三个目录：

- 第一个是FFmpeg的可执行程序及其动态库目录，比如E:\msys64\usr\local\ffmpeg\bin，该目录中主要有ffmpeg、ffplay、ffprobe等程序，以及若干DLL动态库。
- 第二个是MSYS2的可执行程序目录，比如E:\msys64\usr\bin，该目录主要有cp、env、ls、mv、ps、rm、tar等基本命令程序，还包括后面安装的gcc、g++、gdb、make、pkg-config等编译调试程序。
- 第三个是MinGW64位模式的可执行程序及其动态库目录，比如E:\msys64\mingw64\bin，虽然该目录中的.exe文件基本都能在usr\bin目录中找到，但是该目录的众多DLL文件在其他目录找不到，所以也要把它加进PATH变量。

之所以给PATH变量添加这三个目录，是为了在命令行输入相关命令时，Windows能够自动找到对应的可执行程序，并链接需要的DLL动态库。

3. 执行测试命令

在MSYS2的控制台执行以下命令查看FFmpeg的版本信息：

```
    ffmpeg -version
```

发现控制台回显如下的FFmpeg版本与编译器版本信息，这说明FFmpeg程序已成功运行。

```
    ffmpeg version N-109444-geef763c705-20221222 Copyright (c) 2000-2022 the FFmpeg
developers
    built with gcc 12.2.0 (crosstool-NG 1.25.0.90_cf9beb1)
```

1.4　FFmpeg 的开发框架

本节介绍FFmpeg开发框架基本的使用说明，包括FFmpeg提供的三个可执行程序及其用法、FFmpeg提供的8个动态链接库及其用法、如何基于FFmpeg平台编写第一个FFmpeg程序等。

1.4.1　可执行程序

外界对于FFmpeg主要有两种使用途径：一种是在命令行运行FFmpeg的可执行程序，该方式适合没什么特殊要求的普通场景；另一种是通过代码调用FFmpeg的动态链接库，由于开发者可以在C代码中编排个性化的逻辑，因此该方式适合厂商专用的特制场景。

开源的FFmpeg框架提供了三个可执行程序，分别是ffmpeg、ffplay和ffprobe，它们的源码均位于fftools目录。下面对这三个程序分别展开详细介绍。

1. ffmpeg程序

ffmpeg程序主要有两个用途：一个是查询FFmpeg的支持信息，另一个是处理音视频的转换操作。关于音视频的转换命令，会在后面的章节中逐一介绍，这里只说明该程序能够查到哪些FFmpeg支持信息。前面在搭建FFmpeg开发环境的时候，提到可以用以下命令查看FFmpeg的版本信息：

```
ffmpeg -version
```

除此之外，ffmpeg程序还能查询它所支持的文件格式，比如以下命令可以查看FFmpeg支持的文件格式：

```
ffmpeg -formats
```

执行上面的命令，控制台回显长长的一串文件格式支持列表，列表开头如下：

```
File formats:
 D. = Demuxing supported
 .E = Muxing supported
 --
 D  3dostr         3DO STR
  E 3g2            3GP2 (3GPP2 file format)
  E 3gp            3GP (3GPP file format)
 D  4xm            4X Technologies
  E a64            a64 - video for Commodore 64
 D  aa             Audible AA format files
 D  aac            raw ADTS AAC (Advanced Audio Coding)
```

可见FFmpeg支持的文件格式分为两种类型：一种被标记为D，表示支持该类型文件的解析；另一种被标记为E，表示支持该类型文件的封装。继续下拉这一长串文件格式列表，既能找到古老的VCD格式，也能找到风靡一时的RM和FLV格式，还能找到MP3和MP4等常见格式，看来FFmpeg真的将常见的音视频格式一网打尽了。

ffmpeg程序还能够查看更多信息，详见表1-3。由于相关概念比较专业，因此这里不再一一展开，等到后续涉及时再来讲解。

表 1-3　ffmpeg 程序的一级命令用途说明

带参数的 ffmpeg 命令	该命令的用途
ffmpeg -codecs	查看支持的编解码器
ffmpeg -colors	查看支持的颜色名称及其 RGB 值
ffmpeg -decoders	查看支持的解码器

（续表）

带参数的 ffmpeg 命令	该命令的用途
ffmpeg -encoders	查看支持的编码器
ffmpeg -devices	查看支持的设备
ffmpeg -filters	查看支持的过滤器（又称滤波器、滤镜）
ffmpeg -formats	查看支持的文件格式
ffmpeg -demuxers	查看支持的解复用器（又叫解析器、拆包器）
ffmpeg -muxers	查看支持的复用器（又叫封装器、打包器）
ffmpeg -help	查看命令行的帮助信息
ffmpeg -layouts	查看支持的音频通道布局
ffmpeg -pix_fmts	查看支持的图像采样格式
ffmpeg -protocols	查看支持的通信协议
ffmpeg -sample_fmts	查看支持的音频采样格式

2. ffplay程序

ffplay程序相当于一个播放器，用来播放音视频文件。在播放音频时，ffplay不仅会让扬声器放出声音，还会在屏幕上展示该音频的波形画面。在播放视频时，ffplay会在屏幕上展示连续的视频画面，就像看电影、看电视那样。如果视频文件携带了音频数据，那么ffplay会让扬声器同时播放声音。

注意，在服务器上不能通过ffplay播放音视频文件，云服务执行ffplay命令会报错Could not initialize SDL - dsp: No such audio device，这是因为服务器只提供运算功能，没接入显示器和扬声器。只有把FFmpeg安装到个人计算机上，才能正常使用ffplay播放音视频。

以播放视频为例，前提条件完成了1.3节讲述的各项安装步骤，再在Windows系统的MSYS2窗口执行以下命令，使用ffplay程序播放名叫fuzhous.mp4的视频文件。

```
ffplay fuzhous.mp4
```

运行上面的命令之后，控制台一边弹出视频播放器窗口，如图1-15所示。

图 1-15　ffplay 的视频播放界面

一边回显以下文件日志信息：

```
filename=fuzhous.mp4, flags=1q=    0KB vq=    0KB sq=    0B f=0/0
proto_str=file
    Last message repeated 1 times
```

```
    Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'fuzhous.mp4':
      Metadata:
        major_brand     : isom
        minor_version   : 512
        compatible_brands: isomiso2avc1mp41
        encoder         : Lavf59.34.102
      Duration: 00:00:19.52, start: 0.000000, bitrate: 288 kb/s
      Stream #0:0[0x1](und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(progressive),
480x270 [SAR 1:1 DAR 16:9], 285 kb/s, 25 fps, 25 tbr, 12800 tbn (default)
```

根据以上日志信息，可知该视频持续时间为19.52秒，视频编码器采用h264，视频分辨率为480×270，fps帧率为每秒25帧。

再来看看播放音频，以下命令表示使用ffplay程序播放名为ship.mp3的音频文件。

```
ffplay ship.mp3
```

执行上面的命令，控制台一边弹出音频波形窗口，如图1-16所示。

图 1-16　ffplay 的音频播放界面

一边回显以下文件日志信息：

```
    filename=ship.mp3, flags=1 aq=    0KB vq=    0KB sq=    0B f=0/0
    proto_str=file
    Input #0, mp3, from 'ships.mp3':   0KB vq=    0KB sq=    0B f=0/0
      Metadata:
        title           : 渔舟唱晚
        artist          : 中国十大古典名曲
        genre           : Other
        encoder         : Lavf59.27.100
      Duration: 00:03:37.91, start: 0.025057, bitrate: 128 kb/s
      Stream #0:0: Audio: mp3, 44100 Hz, stereo, fltp, 128 kb/s
```

根据以上日志信息，可知晓该音频的标题和演唱者，以及音频的持续时间为3分37.91秒，音频编码器采用MP3，采样频率为44100Hz。

ffplay程序的更多命令行参数可通过以下命令查看，这里不再一一展开了。

```
ffplay -help
```

3. ffprobe程序

ffprobe程序是一个音视频分析工具，它既能分析音视频的文件参数、容器参数等信息，也能分析音视频文件中每个数据包的大小、类型、编解码器等信息。

以查看文件参数为例，以下命令表示使用ffprobe查看视频文件2018.mp4的格式信息。

```
ffprobe -show_format 2018.mp4
```

执行上面的命令，控制台回显如下的文件格式信息，斜杆后面是额外添加的说明注释。

```
[FORMAT]
filename=2018.mp4                          // 文件名
nb_streams=2                               // 流的数量。为2表示包含视频流和音频流
nb_programs=0
format_name=mov,mp4,m4a,3gp,3g2,mj2        // 格式名称
format_long_name=QuickTime / MOV           // 完整的格式名称
start_time=0.000000                        // 开始时间，单位为秒
duration=253.332993                        // 结束时间，单位为秒
size=42853286                              // 文件大小，单位为字节
bit_rate=1353263                           // 比特率，即每秒传输的比特数量（1字节有8比特）
probe_score=100
TAG:major_brand=isom
TAG:minor_version=512
TAG:compatible_brands=isomiso2avc1mp41
TAG:encoder=Lavf57.71.100
[/FORMAT]
```

因为ffprobe程序返回的文件信息直接显示在控制台，密密麻麻的，令人看得眼花缭乱，所以实际上很少使用ffprobe分析音视频，而是采用第三方专业的桌面软件加以分析，后面讲到相关格式时再介绍这些软件。

1.4.2　动态链接库

FFmpeg不仅提供了ffmpeg、ffplay和ffprobe三个可执行程序，还提供了8个工具库，方便开发者调用库中的函数，从而实现更精准的定制化需求。这8个库通常采用动态链接的方式，在Linux系统上动态库表现为SO文件，在Windows系统上动态库表现为DLL文件。这8个库的名字是avcodec、avdevice、avfilter、avformat、avutil、postproc、swresample、swscale，库名开头的a表示audio，也就是音频，库名开头的v表示video，也就是视频。下面分别对这些库展开介绍。

1. avcodec

avcodec是FFmpeg的音视频编解码库，它的源码位于libavcodec目录，在Linux系统的文件名形如libavcodec.so，在Windows系统的文件名形如avcodec-**.dll。

avcodec包含各种音频的编码库和解码库，以及各种视频的编码库和解码库。通过avcodec可以将原始的音视频数据压缩为符合某种码流规则的数据压缩包，也可以将数据压缩包按照指定的码流规则解压为原始的音视频数据。尽管avcodec内置了大部分的音视频编解码库，可是有些码流需要集成第三方的编解码库，比如视频格式H.264要求集成第三方的x264，视频格式H.265要求集成第三方的x265，音频格式MP3要求集成第三方的mp3lame等，libavcodec目录下的诸多lib***.c代码就是用来集成第三方编解码库的。

早期的FFmpeg对于音频格式AAC要求集成第三方的fdk-aac，不过最新的FFmpeg已经集成了自己的AAC编解码库，因此即使没集成fdk-aac也能正常进行AAC格式的编解码。还有一些媒体格式，虽然FFmpeg内置了该格式的编解码库，但因为依赖于特定库，所以编译时要把特定库链接进来，比如图像格式PNG的编解码就依赖于zlib库。

2. avdevice

avdevice是FFmpeg的音视频设备库，它的源码位于libavdevice目录，在Linux系统的文件名形如libavdevice.so，在Windows系统的文件名形如avdevice-**.dll。

avdevice包含音视频的各种输入输出设备库，其中输入设备指的是采集音视频信号的设备，输出设备指的是渲染音视频画面的设备。当然，FFmpeg不会直接操作设备硬件，而是通过第三方的软件包来实现，比如采集媒体信号用到了Windows平台的VFW（Video for Windows，捕捉器），以及VFW的升级版DirectShow捕捉器；渲染媒体画面用到了Windows平台的GDI（Graphics Device Interface，接收器），以及跨平台的SDL2（Simple DirectMedia Layer，媒体开发库）。当然，FFmpeg也支持音效处理库OpenAL（Open Audio Library）和图形处理库OpenGL（Open Graphics Library）。

3. avfilter

avfilter是FFmpeg的音视频滤镜库，它的源码位于libavfilter目录，在Linux系统的文件名形如libavfilter.so，在Windows系统的文件名形如avfilter-**.dll。

avfilter包含加工编辑音频和视频的各种滤镜包，其中音频滤镜的源码文件名形如af_***.c，视频滤镜的源码文件名形如vf_***.c。音频滤镜多用于调整参数、混合音频等处理，视频滤镜多用于变换视频、特效画面、添加部件等处理。

部分高级滤镜要求FFmpeg集成第三方支持库，例如水印滤镜drawtext需要集成FreeType库，字幕滤镜subtitles需要集成ASS库。

4. avformat

avformat是FFmpeg的音视频格式库，它的源码位于libavformat目录，在Linux系统的文件名形如libavformat.so，在Windows统的文件名形如avformat-**.dll。

avformat包含各类媒体文件格式库，以及各种网络通信协议库。其中格式库不仅包含视频格式MP4、AVI、MOV、3GP等，音频格式MP3、WAV、AAC、PCM等，还包含图像格式JPEG、GIF、PNG、YUV等。协议库不仅包含文件协议file，常规的通信协议HTTP、FTP、TCP、UDP等，还包含流媒体传输协议RTSP、RTMP、HLS、SRT等。

由于FFmpeg把协议层的传输操作和不同格式的解析操作都封装好了，因此它们对开发者而言是透明的，从而减轻了开发者适配不同协议和格式的负担。

5. avutil

avutil是FFmpeg的音视频工具库，它的源码位于libavutil目录，在Linux系统的文件名形如libavutil.so，在Windows系统的文件名形如avutil-**.dll。

avutil包含常见的通用工具和各类算法库，其中通用工具包括字典读写、日志记录、缓存交互、线程处理，以及加解密库AES、MD5、SHA、BASE64等；各类算法包括排队算法、排序算法、哈希表、二叉树等。除此之外，avutil也囊括色彩空间、音频采样等方面的公共函数。

6. postproc

postproc是FFmpeg的音视频后期效果处理库，它的源码位于libpostproc目录，在Linux系统的文件名形如libpostproc.so，在Windows系统的文件名形如postproc-**.dll。

postproc主要用于进行后期的效果处理，如果代码中使用了滤镜，编译时就要链接这个库，因为滤镜用到了postproc的一些基础函数。

7. swresample

swresample是FFmpeg的音频重采样库，它的源码位于libswresample目录，在Linux系统的文件名形如libswresample.so，在Windows系统的文件名形如swresample-**.dll。

swresample主要用于音频重采样的相关功能，比如把音频从单声道变为多声道，变更音频的采样频率，转换音频的数据格式等。

8. swscale

swscale是FFmpeg的视频图像转换库，它的源码位于libswscale目录，在Linux系统的文件名形如libswscale.so，在Windows系统的文件名形如swscale-**.dll。

swscale主要用于图像缩放、色彩空间转换等功能，其中色彩空间转换有时也被称作像素格式转换，比如把视频帧从YUV格式转换为RGB格式。

1.4.3　第一个 FFmpeg 程序

在验证FFmpeg是否成功安装时，可通过命令ffmpeg -version查看FFmpeg版本号。如果能够正确回显FFmpeg的版本信息，就表示FFmpeg已经成功安装。不过，对于开发者来说，最佳的验证方式是通过编写C代码。特别是看到自己亲手编写的代码输出Hello World时，这标志着成功迈出FFmpeg开发的第一步。下面就来介绍如何编写第一个FFmpeg程序。

众所周知，C语言有个printf函数，可以把文字信息输出到控制台，比如下面的C代码调用printf函数打印Hello World（完整代码见chapter01/helloc.c）：

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Hello World\n");
    return 0;
}
```

把上面的代码保存为helloc.c文件，接着运行以下GCC命令编译可执行程序，命令格式为“gcc 源代码文件名称 -o 可执行程序名称”。

```
gcc helloc.c -o helloc
```

编译通过后，执行下面的命令（“./”表示位于当前目录），即可在控制台看到程序输出了一行文字Hello World，表示C程序正常运行。

```
./hello
```

FFmpeg框架使用av_log函数替代printf函数，顾名思义，该函数用于打印日志，默认会把日志信息打印到控制台上，相当于printf函数的日志打印功能。av_log函数的用法很简单，只要包含头文件libavutil/avutil.h，然后调用av_log函数，指定日志等级和日志内容就行。编写带日志功能的代码的详细步骤说明如下。

01 创建名为helloffmpeg.c的C代码文件，填入下面的代码内容（完整代码见chapter01/helloffmpeg.c）：

```
#include <stdio.h>
#include <libavutil/avutil.h>

int main(int argc, char *argv[]) {
    av_log(NULL, AV_LOG_INFO, "Hello World\n");
    return 0;
}
```

02 保存并退出该文件，执行以下命令编译helloffmpeg.c：

```
gcc helloffmpeg.c -o helloffmpeg -I/usr/local/ffmpeg/include -L/usr/local/ffmpeg/lib
-lavformat -lavdevice -lavfilter -lavcodec -lavutil -lswscale -lswresample -lpostproc -lm
```

编译命令中的-I指定了头文件的存放目录，-L指定了链接库的存放目录，-l指定了编译过程需要链接哪些库，比如上面的命令要求链接avformat、avdevice、avfilter、avcodec、avutil、swscale、swresample、postproc这8个FFmpeg动态库，另外还链接了系统自带的m库（表示math库，也就是数学函数库）。

03 运行编译好的helloffmpeg程序，也就是执行以下命令：

```
./helloffmpeg
```

发现控制台回显日志信息Hello World，这表明测试程序运行正常，说明FFmpeg开发环境已经成功搭建。

04 刚才的测试程序helloffmpeg.c采用C语言编写，并且使用GCC编译。若要采用C++编程，则需改成下面的helloffmpeg.cpp代码（完整代码见chapter01/helloffmpeg.cpp）：

```
#include <iostream>  // C++使用iostream代替stdio.h

// 因为FFmpeg源码使用C语言编写，所以若是在C++代码中调用FFmpeg，则要通过标记extern "C"{…}把
FFmpeg的头文件包含进来
extern "C"
{
#include <libavutil/avutil.h>
}

int main(int argc, char** argv) {
    av_log(NULL, AV_LOG_INFO, "Hello World\n");
    return 0;
}
```

鉴于C++代码采用G++编译，那么编译helloffmpeg.cpp的编译命令如下所示：

```
g++ helloffmpeg.cpp -o helloffmpeg -I/usr/local/ffmpeg/include -L/usr/local/ffmpeg/lib
-lavformat -lavdevice -lavfilter -lavcodec -lavutil -lswscale -lswresample -lpostproc -lm
```

编译完毕后，同样生成名为helloffmpeg的可执行程序，如此就实现了C++代码集成FFmpeg函数的目标。

不过extern "C"标记只能在CPP代码中使用，如果在C代码中写入extern "C"并且使用GCC来编

译的话，GCC会报错error: expected identifier or '(' before string constant，意思是它不认识这个标记。这种情况属于GCC和G++的编译差别，主要体现在下列几个方面：

（1）G++在编译C代码和CPP代码时，都采用C++的语法来编译；而GCC对于C代码采用C语言的语法编译，对于CPP代码采用C++的语法编译（GCC也能编译C++程序）。

（2）GCC不能自动链接C++库，而G++会自动链接C++库，所以通过GCC编译C++代码时，要记得链接stdc++库。

如果希望C代码既能通过GCC编译，也能通过G++编译，就要引入宏__cplusplus。一旦定义了这个宏，表示当前采用C++的语法编译，就得添加extern "C"标记；否则表示当前采用C语言的语法编译，无须添加extern "C"标记。可将helloffmpeg.c的代码补充完善，并将修改后的代码另存为hellofull.c，具体代码示例如下（完整代码见chapter01/hellofull.c）：

```
#include <stdio.h>

//libavutil/common.h要求定义，否则会报错：error missing -D__STDC_CONSTANT_MACROS
#define __STDC_CONSTANT_MACROS

// 之所以增加__cplusplus的宏定义，是为了同时兼容GCC编译器和G++编译器
#ifdef __cplusplus
extern "C"
{
#endif
#include <libavutil/avutil.h>
#ifdef __cplusplus
};
#endif

int main(int argc, char** argv) {
    av_log(NULL, AV_LOG_INFO, "Hello World\n");
    return 0;
}
```

然后分别运行下面两行GCC和G++编译命令，发现均能编译通过，表示该代码同时兼容C语言的语法和C++的语法。

```
    gcc hellofull.c -o hellofull -I/usr/local/ffmpeg/include -L/usr/local/ffmpeg/lib
-lavformat -lavdevice -lavfilter -lavcodec -lavutil -lswscale -lswresample -lpostproc -lm
    g++ hellofull.c -o hellofull -I/usr/local/ffmpeg/include -L/usr/local/ffmpeg/lib
-lavformat -lavdevice -lavfilter -lavcodec -lavutil -lswscale -lswresample -lpostproc -lm
```

注意，前面调用av_log函数时，第二个参数都填作AV_LOG_INFO，该参数值表示标准信息，用于标明当前日志的日志等级，详细的日志等级说明见表1-4。

表 1-4　FFmpeg 的日志等级说明

日志等级	日志文字的颜色	说　　明
AV_LOG_FATAL	红色	致命错误
AV_LOG_ERROR	红色	错误信息

（续表）

日志等级	日志文字的颜色	说　　明
AV_LOG_WARNING	黄色	警告信息
AV_LOG_INFO	默认颜色，比如白色	标准信息
AV_LOG_VERBOSE	绿色	详细信息
AV_LOG_DEBUG	绿色	开发者添加的调试信息
AV_LOG_TRACE	灰色	开发过程中极其冗长的调试

之所以要设置这些日志等级，是为了更好地管理输入日志，主要体现在以下两个方面：

（1）给不同等级的日志文字显示不同的颜色，有利于快速找到警告、错误等重要日志。

（2）能够通过av_log_set_level函数来设置打印的日志等级，默认只会打印AV_LOG_INFO和更高级别的日志。如果调用av_log_set_level函数设置了其他的日志级别，那么只会打印该级别以及更高级别的日志信息。

为了更好地观察日志的输出情况，打开helloffmpeg.c，在av_log之前增加调用av_log_set_level函数，修改后的代码如下：

```
#include <stdio.h>
#include <libavutil/avutil.h>

int main(int argc, char** argv) {
    av_log_set_level(AV_LOG_TRACE);
    av_log(NULL, AV_LOG_INFO, "Hello World\n");
    return 0;
}
```

然后更改av_log函数的日志等级参数，编译并运行程序，即可观察对应的日志输出情况。

1.5　小　　结

本章主要介绍了FFmpeg开发环境的搭建过程，首先简要介绍了FFmpeg的起源、用途及其发展历程，接着介绍了在Linux系统搭建FFmpeg开发环境的详细步骤，然后介绍了在Windows系统搭建FFmpeg开发环境的详细步骤，最后介绍了FFmpeg的开发框架并基于FFmpeg平台编写了第一个FFmpeg程序。

通过本章的学习，读者应该能够掌握以下3种开发技能：

（1）学会在Linux系统搭建FFmpeg的开发环境。

（2）学会在Windows系统搭建FFmpeg的开发环境。

（3）学会在FFmpeg平台上编写Hello World程序。