

Windows编译FFmpeg

一 编译环境

1. 下载MSYS2
2. 修改pacman的源
3. 启动cmd命令行
4. 安装编译环境
 - 4.1 安装mingw-w64
 - 4.2 安装git和make等工具
 - 4.3 编译环境的其他准备工作

二 编译第三方库

1. 编译x264
2. 编译fdk-aac
3. 编译mp3
4. 编译libvpx
5. 编译libass
 - 5.1 源码编译libass
 - 5.1.1 编译freetype
 - 5.1.2 编译harfbuzz
 - 5.1.3 编译fribidi
 - 5.1.4 编译libxml2
 - 5.1.5 编译fontconfig
 - 5.1.6 开始编译libass
 - 5.2 pacman安装包下载libass(建议)

三 编译FFmpeg

1. 下载FFmpeg
2. 编译FFmpeg
3. 简单使用ffplay播放测试

四 使用QT测试编译生成的ffmpeg

一 编译环境

1. 下载MSYS2

以下内容来自百度百科：

MSYS2 (Minimal SYStem 2)是一个MSYS的独立改写版本，主要用于 shell 命令行开发环境。同时它也是一个在Cygwin(POSIX 兼容性层) 和 MinGW-w64(从"MinGW-生成")基础上产生的，追求更好的互操作性的 Windows 软件。

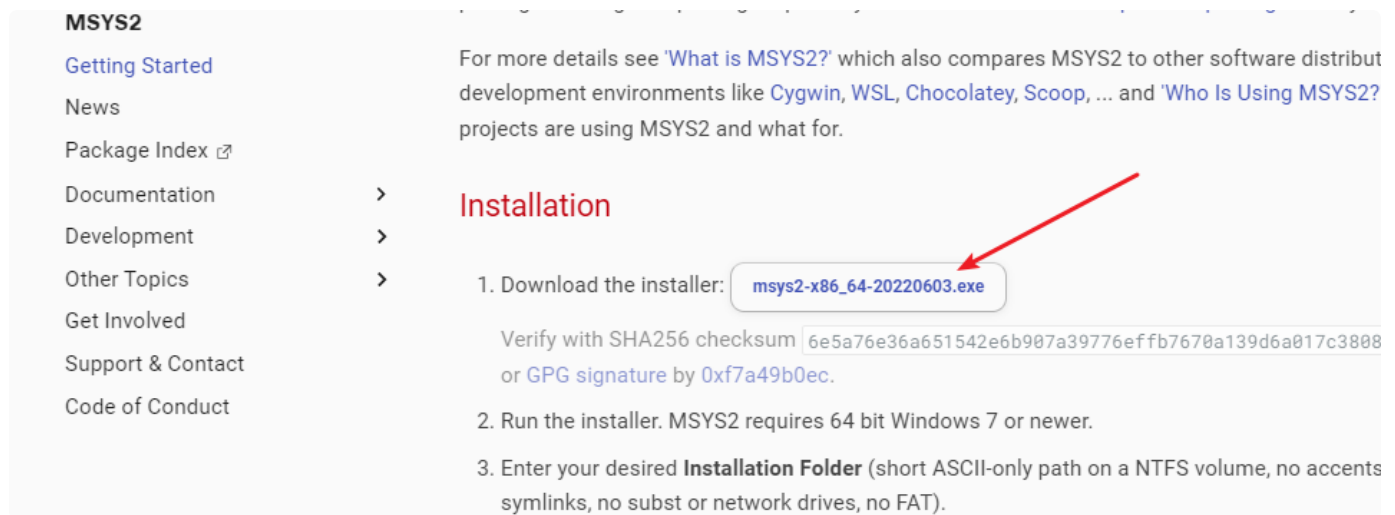
MSYS2 是MSYS的一个升级版，准确的说是集成了panman和Mingw-64的Cygwin升级版, 提供了bash shell等linux环境、版本控制软件 (git/hg) 和MinGW-w64 工具链。与MSYS最大的区别是移植了Arch linux的软件包管理系统Pacman(其实是与Cygwin的区别)。

MSYS2环境中的Pacman 和 Cygwin环境中的apt-cyg等价于Linux的apt-get。

例如这个github项目<https://github.com/nldzsz/ffmpeg-build-scripts>，就是使用Cygwin环境中的apt-cyg去编译ffmpeg的。而我们这里是使用MSYS2环境中的Pacman。

MSYS2下载地址：<https://www.msys2.org/>。

点击下载即可。



The screenshot shows the MSYS2 website with a sidebar on the left containing links like 'Getting Started', 'News', 'Package Index', 'Documentation', 'Development', 'Other Topics', 'Get Involved', 'Support & Contact', and 'Code of Conduct'. The main content area is titled 'Installation' and lists three steps: 1. Download the installer: [msys2-x86_64-20220603.exe](#) (highlighted with a red arrow), 2. Run the installer. MSYS2 requires 64 bit Windows 7 or newer. 3. Enter your desired **Installation Folder** (short ASCII-only path on a NTFS volume, no accents, symlinks, no subst or network drives, no FAT).

1. 创建一个新的目录，最好空间充足，不要放在C盘(例如我的是D:\English_Software\msys2)。
2. 然后右键，以管理员身份运行，等几分钟后，安装完成。

3. 在安装目录下，找到msys2_shell.cmd文件，进行以下修改。

```
▼ Bash | 复制代码
1  # cmd中rem表示注释，所以我们需要去掉注释，
2  # 这样可以让我们在msys2打开的控制台继承windows的PATH环境变量
3  rem set MSYS2_PATH_TYPE=inherit
4  改成
5  set MSYS2_PATH_TYPE=inherit
```

备注：

MSYS2可以选择msys或者MinGW-w64环境来编译，不过在msys下使用gcc编译出来的exe和dll依赖msys-2.0.dll，而MinGW-w64下编译出来的文件不需要依赖这个dll，从程序的运行效率来看，不依赖这个dll的程序效率应该更高。所以选择MinGW-w64来编译更佳。

2. 修改pacman的源

pacman是一个软件包管理器，用来在MSYS2中安装软件，但是默认的国外的源下载安装包时非常缓慢，

大概只有十几二十KB的速度，而且还容易下载中断出错，所以需要修改为国内源，国内源可以选择中科大的源。

按照MSYS2镜像提示修改。具体如下(注，etc前面的/代表你的msys2的安装目录)：

1) 编辑 /etc/pacman.d/mirrorlist.mingw32，在文件开头添加：

```
▼ Bash | 复制代码
1  Server = https://mirrors.tuna.tsinghua.edu.cn/msys2/mingw/i686/
2  Server = http://mirrors.ustc.edu.cn/msys2/mingw/i686/
```

2) 编辑 /etc/pacman.d/mirrorlist.mingw64，在文件开头添加：

```

1 Server = https://mirrors.tuna.tsinghua.edu.cn/msys2/mingw/x86_64/
2 Server = http://mirrors.ustc.edu.cn/msys2/mingw/x86_64/

```

3) 编辑 `/etc/pacman.d/mirrorlist.msys`，在文件开头添加：

```

1 Server = https://mirrors.tuna.tsinghua.edu.cn/msys2/msys/$arch/
2 Server = http://mirrors.ustc.edu.cn/msys2/msys/$arch/

```

此电脑 > OS (D:) > English_Software > msys2 > etc > pacman.d >

名称	修改日期	类型	大
gnupg	2022/6/13 下午 03:46	文件夹	
mirrorlist.clang32	2022/2/5 下午 08:12	CLANG32 文件	
mirrorlist.clang64	2022/2/5 下午 08:12	CLANG64 文件	
mirrorlist.mingw	2022/2/5 下午 08:12	MINGW 文件	
mirrorlist.mingw32	2022/2/5 下午 08:12	MINGW32 文件	
mirrorlist.mingw64	2022/2/5 下午 08:12	MINGW64 文件	
mirrorlist.msys	2022/2/5 下午 08:12	MSYS 文件	
mirrorlist.ucrt64	2022/2/5 下午 08:12	UCRT64 文件	

例如 `/etc/pacman.d/mirrorlist.mingw64`:

```

##
## 64-bit Mingw-w64 repository mirrorlist
##

## Primary
## msys2.org
Server = https://mirrors.tuna.tsinghua.edu.cn/msys2/mingw/x86_64/
Server = http://mirrors.ustc.edu.cn/msys2/mingw/x86_64/
Server = https://sourceforge.net/projects/msys2/files/REPOS/MINGW/x86_64/
Server = https://www2.futureware.at/~nickoe/msys2-mirror/mingw/x86_64/
Server = https://mirror.vandex.ru/mirrors/msys2/mingw/x86_64/

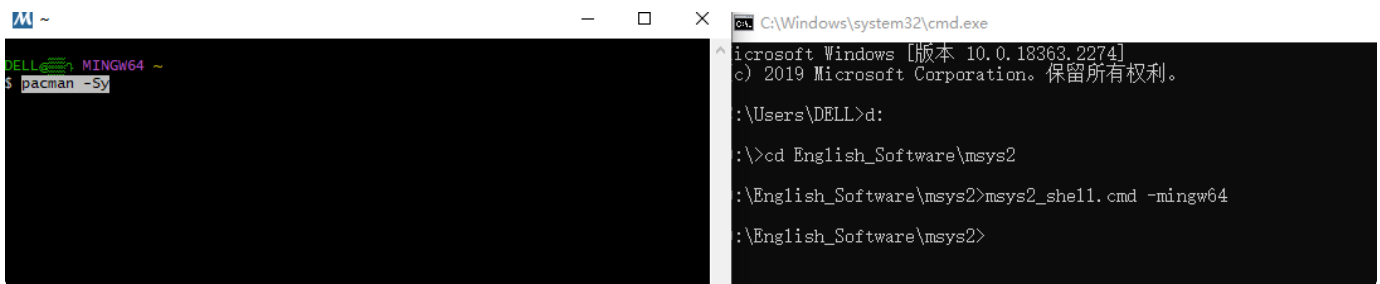
```

3. 启动cmd命令行

```
1 # 1. 进入D盘
2 d:
3 # 2. 进入msys2安装目录
4 cd English_Software\msys2
5 # 3. 启动MSYS2 MinGW64
6 msys2_shell.cmd -mingw64
7 # 注, 若想要使用msys编译, 则输入:
8 # msys2_shell.cmd
```

打开msys2的MinGW64窗口后, 在msys2的窗口输入:

```
1 pacman -Sy
```



执行成功。

```
DELL@MINGW64 ~
$ pacman -Sy
:: Synchronizing package databases...
mingw32           1617.1 KiB   514 KiB/s  00:03 [#####] 100%
mingw64           1626.9 KiB   537 KiB/s  00:03 [#####] 100%
ucrt64            1685.3 KiB   484 KiB/s  00:03 [#####] 100%
clang32           1584.6 KiB   235 KiB/s  00:07 [#####] 100%
clang64           1615.9 KiB   192 KiB/s  00:08 [#####] 100%
msys               394.6 KiB   887 KiB/s  00:00 [#####] 100%
DELL@MINGW64 ~
$ |
```

4. 安装编译环境

安装编译环境提前说明：

上面只是完成是msys2软件的安装，msys2只是提供了我们可以类似shell终端一样，可以执行命令的操作，而真正的编译环境，仍需要安装。

msys2 主要会有两类开发环境：

1) 使用MSYS2 自带的开发环境进行编译，需要安装一个依赖包`msys2-devel`。

2) 使用MinGW-w64 进行编译，它支持gcc、g++、gdb以及MingGW32位编译、MingGW64为编译的环境。例如有些Qt版本只支持32位，如果想编译64位的程序或者链接64位的dll，需要下载MinGW-64或者下载支持编译64位程序的指定Qt版本。

两者区别：

前者编译出来的可执行文件，要依赖 MSYS2 提供的动态链接库，而后者不需要。

下面详细说明一下：

1) MSYS2 下的 gcc 编译环境，编译的可执行文件要依赖于 `msys-2.0.dll`，这个 DLL 提供了 Linux

下编程的提供的函数和接口，例如 `fork` 函数。

这个编译环境对于编译基于 Linux 下编写的软件，是非常适合的。例如编译 GNU 提供的各种工具。例如，你想编译最新版本的 GNU `grep` 工具，MSYS2 下的这个环境是非常适合的。

2) 用 MinGW-w64 的编译环境，不再依赖于 `msys-2.0.dll`，如果源代码就是基于 windows 开发的，那

使用 MinGW 的编译环境比较好，编译出来的可执行文件，不用再依赖 MSYS 提供的动态链接库。当然，

前提是代码中不能使用 Linux 的东西，即 POSIX 的那套东西。

换句话说，如果在Linux开发，那么直接使用MSYS2自带的环境进行编译，得出的可执行文件是更合适的；

而在Windows开发，使用 MinGW64的环境进行编译，得出的可执行文件是更合适的，因为不用再链接 `msys-2.0.dll`，而`msys-2.0.dll`提供的主要是LINUX 下 POSIX相关的接口。

因为我们这里是在Windows开发，使用我们选择MinGW-w64进行编译。

4.1 安装mingw-w64

在刚刚打开的MSYS2 MinGW64窗口中输入：

```
1  # 一直按回车默认按照即可(1G左右, 可能需要10min-30min)。  
2  pacman -S mingw-w64-x86_64-toolchain  
3  
4  # 而如果选择msys编译则打开MSYS2 MSYS2, 在shell窗口中输入:  
5  #pacman -S msys2-devel  
6  # 或者  
7  #pacman -S make gcc diffutils pkg-config
```

备注：这一步可能网络比较慢，对于网速不好的，多试几次即可。

网络差时会报错。

```
error: failed retrieving file 'mingw-w64-x86_64-gcc-objc-12.1.0-2-any.pkg.tar.zst' from mirror.msys2.org : SSL connection timeout  
error: failed retrieving file 'mingw-w64-x86_64-gdb-multiarch-12.1-1-any.pkg.tar.zst' from mirror.msys2.org : Operation too slow. Less than 1 bytes/sec transferred the last 10 seconds  
error: failed retrieving file 'mingw-w64-x86_64-headers-git-10.0.0.r32.g89bacd2be-1-any.pkg.tar.zst' from mirror.msys2.org : Operation too slow. Less than 1 bytes/sec transferred the last 10 seconds  
warning: too many errors from mirror.msys2.org, skipping for the remainder of this transaction  
error: failed retrieving file 'mingw-w64-x86_64-gcc-12.1.0-2-any.pkg.tar.zst.sig' from mirrors.tuna.tsinghua.edu.cn : Operation too slow. Less than 1 bytes/sec transferred the last 10 seconds  
error: failed retrieving file 'mingw-w64-x86_64-gdb-multiarch-12.1-1-any.pkg.tar.zst' from repo.msys2.org : HTTP/2 stream 0 was not closed cleanly: PROTOCOL_ERROR (err 1)  
error: failed retrieving file 'mingw-w64-x86_64-gcc-objc-12.1.0-2-any.pkg.tar.zst' from repo.msys2.org : Operation too slow. Less than 1 bytes/sec transferred the last 10 seconds  
warning: failed to retrieve some files  
error: failed to commit transaction (unexpected error)  
Errors occurred, no packages were upgraded.
```

多试几次后即可成功下载完毕。

```
mingw-w64-x86_64-python-pygments: for syntax highlighting  
(43/48) installing mingw-w64-x86_64-libgccjit  
(44/48) installing mingw-w64-x86_64-libmangle-git  
(45/48) installing mingw-w64-x86_64-make  
(46/48) installing mingw-w64-x86_64-pkgconf  
(47/48) installing mingw-w64-x86_64-tools-git  
(48/48) installing mingw-w64-x86_64-winstorecompat-git  
DELL  MINGW64 ~  
$
```

4.2 安装git和make等工具

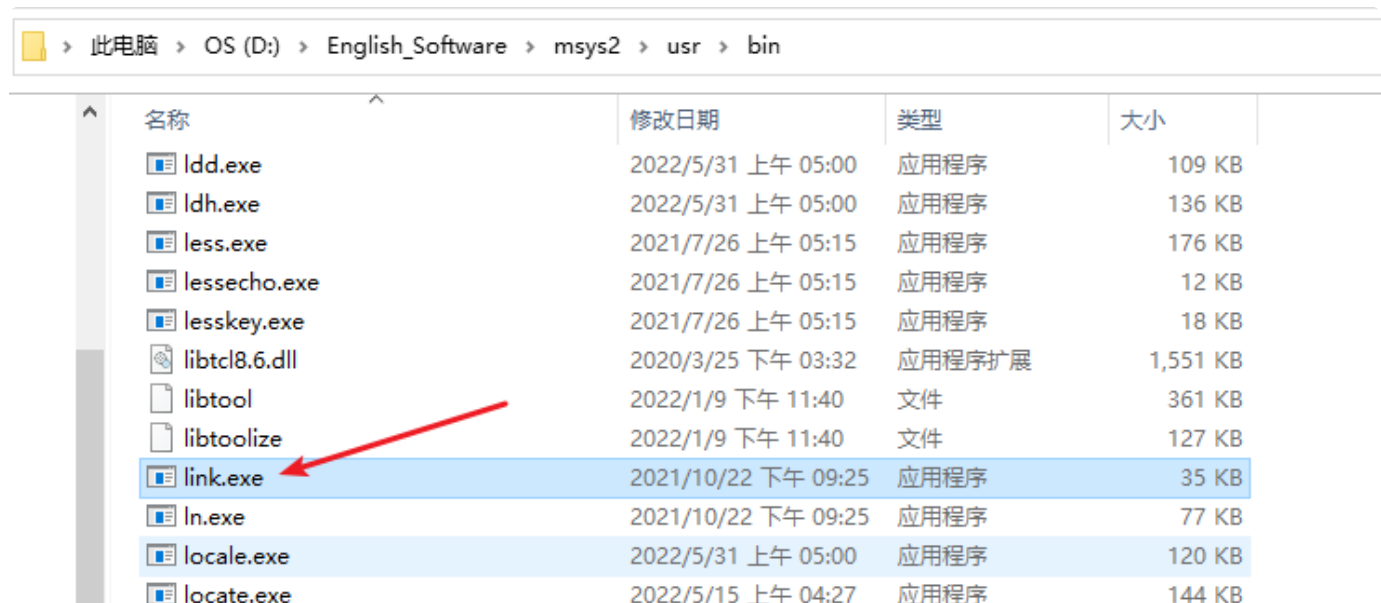
同样一直按回车默认安装，若出错同样多试几次。

Bash | 复制代码

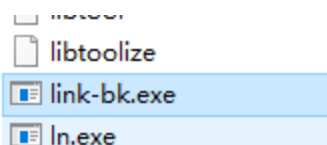
```
1  pacman -S git
2  pacman -S make
3  pacman -S automake
4  pacman -S autoconf
5  pacman -S perl
6  pacman -S libtool
7  pacman -S mingw-w64-i686-cmake
8  # 库配置工具，编译支持x264和x265等第三方库都会用到
9  pacman -S pkg-config
10 # 若想编译出ffplay，还需要安装SDL
11 pacman -S mingw-w64-x86_64-SDL2
```

4.3 编译环境的其他准备工作

1. 重命名 `msys2/usr/bin/link.exe` 为 `msys2/usr/bin/link-bk.exe`，避免和 MSVC 的 `link.exe` 抵触。



即：



2. 下载YASM

YASM是用来编译FFmpeg里面的一些汇编代码的。

在MSYS2 MinGW64窗口中输入：

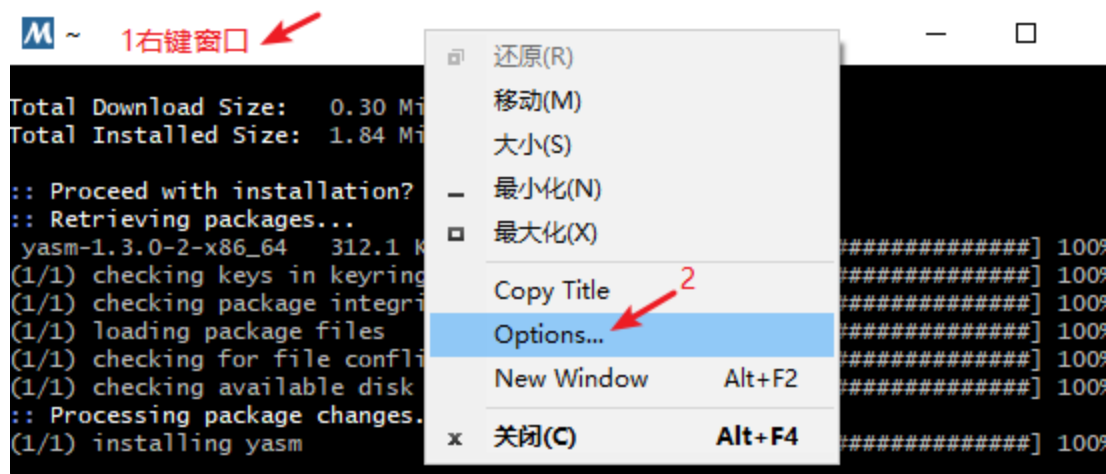
```
1  pacman -S yasm
2
3  # 检查编译环境工具
4  # 不是空结果，基本表明安装成功
5  which cl link yasm cpp
```

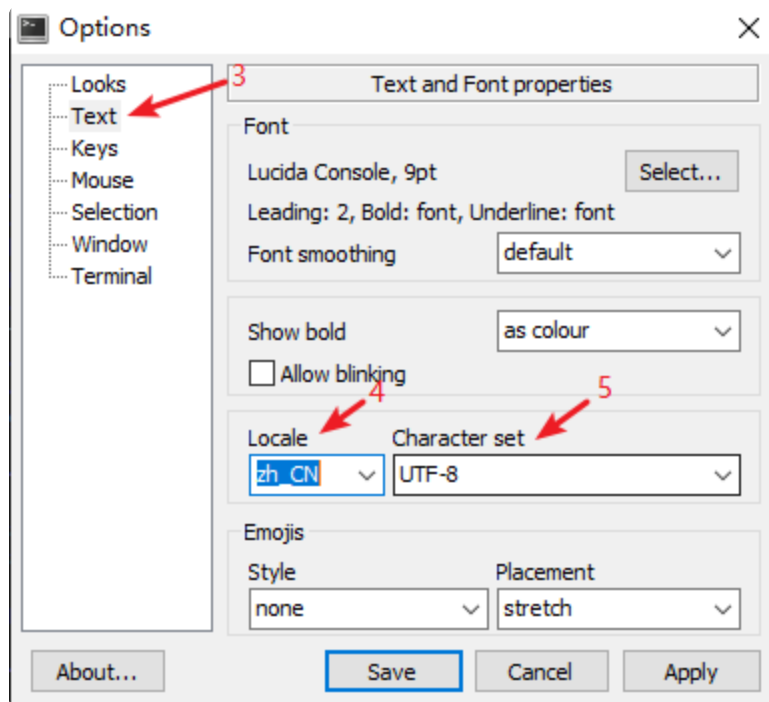
```
DELL MINGW64 ~
$ which cl link yasm cpp
/c/Program Files (x86)/Microsoft Visual Studio 14.0/VC/bin/cl
/c/Program Files (x86)/Microsoft Visual Studio 14.0/VC/bin/link
/usr/bin/yasm
/mingw64/bin/cpp
```

3. 修改支持中文显示

窗口右键->Options->Text, 然后locale选择: zh_CN, Character set 选择 UTF-8。

然后Apply, save即可。





4. 安装nasm

编译x264时需要依赖nasm。

```
Bash | 复制代码  
1  pacman -S nasm
```

二 编译第三方库

1. 编译x264

1) 在msys2的窗口输入以下命令：

```
1  # 1.在/home下创建一个用于临时构建的目录ffmpegBuild,
2  # 然后里面再创建一个extra目录,用于存放第三方库
3  cd /home
4  mkdir -p ffmpegBuild/extra
5  cd ffmpegBuild/extra
6
7  # 2. 下载x264
8  git clone http://git.videolan.org/git/x264.git
9  # 若卡,则使用码云的链接
10 #git clone https://gitee.com/mirrors_addons/x264.git
11
12 # 3. 在/home/ffmpegBuild中创建一个build目录,用于存放第三方库的生成的静态库或者动态库
13 # x264这里静态库、动态库都生成
14 mkdir -p /home/ffmpegBuild/build
15 cd /home/ffmpegBuild/extra/x264
16
17 # 4. 开始执行配置以及编译安装
18 ./configure --prefix=/home/ffmpegBuild/build/libx264 --host=x86_64-w64-mingw32 --enable-static --enable-shared --enable-pic --enable-strip --extra-ldflags=-Wl,--output-def=libx264.def
19 # 编译可能需要几分钟
20 make
21 make install
```

上面的目录结构:

此电脑 > OS (D:) > English_Software > msys2 > home > ffmpegBuild >				
名称	修改日期	类型	大小	
build	2022/6/14 上午 09:54	文件夹		
extra	2022/6/14 上午 09:18	文件夹		

配置执行会告诉你哪些配置被启用或者被禁用了。

```

DELL@MINGW64 /home/ffmpegBuild/extra/x264
$ ./configure --prefix=/home/ffmpegBuild/build/libx264 --host=x86_64-w64-mingw32
--enable-static --enable-shared --enable-pic --enable-strip --extra-ldflags=-Wl
,--output-def=libx264.def
platform:      X86_64
byte order:    little-endian
system:        WINDOWS
cli:           yes
libx264:       internal
shared:        yes
static:        yes
bashcompletion: no
asm:           yes
interlaced:    yes
avs:           yes
lavf:          no
ffms:          no
mp4:           no
gpl:           yes
thread:        win32
opencl:        yes
filters:       crop select_every
lto:           no
debug:         no
gprof:         no
strip:         yes
PIC:           yes
bit depth:     all
chroma format: all

You can run 'make' or 'make fprofiled' now.

DELL@MINGW64 /home/ffmpegBuild/extra/x264
$ |

```

注，此时看到生成的libx264.def是空的。

libx264.def	2022/6/14 上午 10:03	Export Definition...	0 KB
-------------	--------------------	----------------------	------

2) 生成libx264.lib

上面只是生成了libx264.def，以及对应的dll和头文件(/home/ffmpegBuild/build/libx264/bin下)。

但是其实并未生成libx264.lib，所以此时我们需要利用libx264.def去生成libx264.lib：

```

1  cp ./libx264.def /home/ffmpegBuild/build/libx264/lib/
2  cd /home/ffmpegBuild/build/libx264/lib
3
4  #生成64位lib(我们这里使用64位开发):
5  lib /machine:X64 /def:libx264.def
6  #生成32位lib:
7  #lib /machine:i386 /def:libx264.def

```

得到 **libx264.lib**，然后将 `/home/ffmpegBuild/build/libx264/bin/libx264-164.dll` (具体名字和x264版本有关) 改名或者复制一份为 `libx264.dll`。

> 此电脑 > OS (D:) > English_Software > msys2 > home > ffmpegBuild > build > libx264 > bin

名称	修改日期	类型	大小
libx264.dll	2022/6/14 上午 10:28	应用程序扩展	2,302 KB
libx264-164.dll	2022/6/14 上午 10:28	应用程序扩展	2,302 KB
x264.exe	2022/6/14 上午 10:28	应用程序	2,446 KB

复制一份，改名为libx264.dll，然后就可以放到项目中使用

备注：若在`./configure`配置时，没有加上`--enable-shared`选项，在bin目录下是没有对应的dll生成的。

如果想在程序中直接使用x264的话，将include中的.h头文件、`libx264.lib` 和 `libx264.dll` 复制到项目对应位置，并且在程序中添加头文件，然后就可以使用x264中的方法了。

```
DELL MINGW64 /home/ffmpegBuild/build/libx264/lib
$ lib /machine:X64 /def:libx264.def
Microsoft (R) Library Manager Version 14.00.24215.1
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
DELL MINGW64 /home/ffmpegBuild/build/libx264/lib
$ ls -l
total 3220
-rw-r--r-- 1 DELL None 3231054 Jun 14 10:28 libx264.a
-rw-r--r-- 1 DELL None 1532 Jun 14 10:36 libx264.def
-rw-r--r-- 1 DELL None 35228 Jun 14 10:28 libx264.dll.a
-rw-r--r-- 1 DELL None 7606 Jun 14 10:36 libx264.exp
-rw-r--r-- 1 DELL None 12860 Jun 14 10:36 libx264.lib
drwxr-xr-x 1 DELL None 0 Jun 14 10:28 pkgconfig
```

```
DELL MINGW64 /home/ffmpegBuild/build/libx264/lib
$ ls -lh
total 3.2M
-rw-r--r-- 1 DELL None 3.1M Jun 14 10:28 libx264.a
-rw-r--r-- 1 DELL None 1.5K Jun 14 10:36 libx264.def
-rw-r--r-- 1 DELL None 35K Jun 14 10:28 libx264.dll.a
-rw-r--r-- 1 DELL None 7.5K Jun 14 10:36 libx264.exp
-rw-r--r-- 1 DELL None 13K Jun 14 10:36 libx264.lib
drwxr-xr-x 1 DELL None 0 Jun 14 10:28 pkgconfig
```

```
DELL MINGW64 /home/ffmpegBuild/build/libx264/lib
$ |
```

libx264官网下载：<https://www.videolan.org/developers/x264.html>。

libx265官网下载：https://bitbucket.org/multicoreware/x265_git/src/master/，有x265需求的，可以进行兼容，因为没用到，所以这里不下载了。这个x265的性能是比其他x265库性能高的，因为之前有测试过，在宿主机中，x265性能差不多，但是放在docker里面，其他x265库性能变得很低，而这个网址的x265库的帧率、码率仍然和宿主机保持基本一致，所以这个库比较好。

2. 编译fdk-aac

```

1  # 1. 下载fdk-aac
2  cd /home/ffmpegBuild/extra/
3
4  # --depth 1的作用是我们只关注该项目最近一次的提交记录，而不管该项目以前的提交记录。
5  # 这样能让我们更快的下载该项目进行学习或者使用，如果你想参与该项目的开发，就最好不要加
   该参数。
6  # --depth 1的作用具体看：
   https://blog.csdn.net/qq_43827595/article/details/104833980
7  git clone --depth 1 https://gitee.com/mirrors/fdk-aac.git
8
9  # 2. 开始编译fdk-aac
10 cd fdk-aac
11 ./autogen.sh
12 ./configure --prefix=/home/ffmpegBuild/build/libfdk-aac --with-pic

```

上面make -j8时可能会比较久，估计半个小时，此时我们可以再打开一个新的MSYS2 MinGW64窗口往下执行，例如去编译mp3.

结果：

此电脑 > OS (D:) > English_Software > msys2 > home > ffmpegBuild > build > libfdk-aac > lib >

名称	修改日期	类型	大小
pkgconfig	2022/6/14 下午 01:34	文件夹	
libfdk-aac.a	2022/6/14 下午 01:34	A 文件	11,769 KB
libfdk-aac.dll.a	2022/6/14 下午 01:34	A 文件	13 KB
libfdk-aac.la	2022/6/14 下午 01:34	LA 文件	1 KB

此电脑 > OS (D:) > English_Software > msys2 > home > ffmpegBuild > build > libfdk-aac > bin

名称	修改日期	类型	大小
libfdk-aac-2.dll	2022/6/14 下午 01:34	应用程序扩展	9,667 KB

libfdk_aac官网下载：<https://sourceforge.net/projects/opencore-amr/files/fdk-aac/>。

3. 编译mp3

```

1  cd /home/ffmpegBuild/extra/
2  git clone --depth 1 https://gitee.com/hqiu/lame.git
3
4  cd lame/
5  # mp3的配置可能比较久，我这里差不多是半个小时
6  # 注意这里如果同时开启静态库和动态库的编译，
7  # 可能会出现cannot export lame_init_old symbol not defined的错误
8  ./configure --prefix=/home/ffmpegBuild/build/libmp3lame --disable-
  frontend --enable-static --disable-shared
9  make
10 make install

```

结果出现.a静态库文件即可，因为编译ffmpeg使用.a文件就行了：

电脑 > OS (D:) > English_Software > msys2 > home > ffmpegBuild > build > libmp3lame > lib

名称	修改日期	类型	大小
 libmp3lame.a	2022/6/14 上午 11:56	A 文件	489 KB
 libmp3lame.la	2022/6/14 上午 11:56	LA 文件	1 KB

libmp3lame官网下载（要求版本 $\geq 3.98.3$ ）：

<https://sourceforge.net/projects/lame/files/lame/>。

然后如果想生成动态库，将静态库和动态库的开关调换，然后重新配置即可(如果没有生成可以先不管)：

```

1  make clean
2  ./configure --prefix=/home/ffmpegBuild/build/libmp3lame --disable-
  frontend --disable-static --enable-shared
3  make
4  make install

```


4. 编译libvpx

```
▼ Bash | 复制代码
1  cd /home/ffmpegBuild/extra/
2  git clone --depth 1 https://github.com/webmproject/libvpx.git
3
4  cd libvpx
5  ./configure --prefix=/home/ffmpegBuild/build/libvpx --disable-examples --
   disable-unit-tests --enable-vp9-highbitdepth --as=yasm
6  # 这里三分钟左右即可编译完成
7  make -j4
8  make install
```

5. 编译libass

这里给出两种方法去生成libass，一种是以源码进行生成。

另一种是直接使用pacman去下载安装包生成，这种方法得到的libass是版本一样的，目前都是0.16.0版本(目前日期是2022-06-16)。

5.1 源码编译libass

libass库依赖第三方库freetype、fribidi、fontconfig。而freetype也会依赖harfbuzz库，fontconfig依赖libxml2。所以需要提前下载它们。

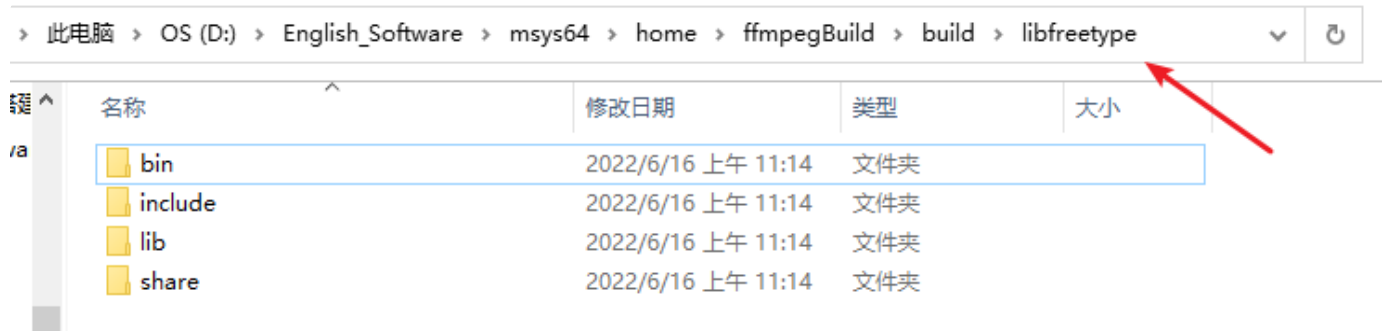
5.1.1 编译freetype

```

1  cd /home/ffmpegBuild/extra/
2  # 我是从gitlab下载源码的
3  git clone https://gitlab.freedesktop.org/freetype/freetype.git
4  # 也可以以tar.xz方式下载源码
5  # 链接为: https://download.savannah.gnu.org/releases/freetype/
6
7  cd freetype/
8  # 生成configure文件
9  ./autogen.sh
10 ./configure --prefix=/home/ffmpegBuild/build/libfreetype --with-pic --
    with-zlib --without-png --without-harfbuzz --without-bzip2 --without-
    fsref --without-quickdraw-toolbox --without-quickdraw-carbon --without-
    ats --disable-fast-install --disable-mmap --enable-static --enable-shared
    --host=x86_64-w64-mingw32 --build=x86_64-pc-msys
11 make -j4
12 make install

```

成功结果：



注意：如果想支持harfbuzz库，因为freetype和harfbuzz是相互依赖的，所以：

- 1) 首先在生成freetype时，先使用--without-harfbuzz参数。
- 2) 然后利用生成的freetype去编译harfbuzz。
- 3) 最后使用生成的harfbuzz库，开启harfbuzz选项即--with-harfbuzz(或者不写默认开启)，这样就能使freetype支持harfbuzz库了。

可参考：<https://blog.csdn.net/JohnYork/article/details/46434075>。

不想源码编译freetype的，可以直接从github下载官方编译好的头文件以及对应的静态库、动态库。

```
▼ Bash | 复制代码
1 # 也可以去github下载编译好的window版本，目前版本都是2.12.1，目前日期是2022-6-16
2 git clone https://github.com/ubawurinna/freetype-windows-binaries
```

5.1.2 编译harfbuzz

```
▼ Bash | 复制代码
1 cd /home/ffmpegBuild/extra
2 git clone --depth 1 https://github.com/harfbuzz/harfbuzz
3
4 cd harfbuzz/
5 ./autogen.sh
6 # 注意，当加上--with-freetype=yes选项时，若不加变量PKG_CONFIG_PATH，会报错误：
7 # configure: error: FreeType support requested but libfreetype2 not found
8 ./configure --prefix=/home/ffmpegBuild/build/harfbuzz --with-freetype=yes
   PKG_CONFIG_PATH=/home/ffmpegBuild/build/libfreetype/lib/pkgconfig
9
10 make -j4
11 make install
```

若出现：

```
make[2]: 进入目录"/home/ffmpegBuild/extra/harfbuzz/src"
GEN      hb-buffer-deserialize-text.hh
git.mk: Generating .gitignore
/home/ffmpegBuild/extra/harfbuzz/missing:行81: ragel: 未找到命令
WARNING: 'ragel' is missing on your system.
You might have modified some files without having the proper
tools for further handling them. Check the 'README' file, it
often tells you about the needed prerequisites for installing
this package. You may also peek at any GNU archive site, in
case some other package contains this missing 'ragel' program.
make[2]: *** [Makefile:4187: hb-buffer-deserialize-text.hh] 错误 1
make[2]: 离开目录"/home/ffmpegBuild/extra/harfbuzz/src"
make[1]: *** [Makefile:513: all-recursive] 错误 1
make[1]: 离开目录"/home/ffmpegBuild/extra/harfbuzz"
make: *** [Makefile:445: all] 错误 2
```

则需要安装ragel。

▼

Bash | 复制代码




```
1 # 64位
2 pacman -S mingw-w64-x86_64-ragel
3 # 32位
4 #pacman -S mingw-w64-i686-ragel
```

然后再次make以及make install即可。

成功编译：

此电脑 > OS (D:) > English_Software > msys64 > home > ffmpegBuild > build > harfbuzz >

▼ ↺

名称	修改日期	类型	大小
 bin	2022/6/16 下午 01:41	文件夹	
 include	2022/6/16 下午 01:41	文件夹	
 lib	2022/6/16 下午 01:41	文件夹	

此时，我们可以重新编译freetype，以让freetype支持harbuzz。

不想支持的可以跳过这里。

```

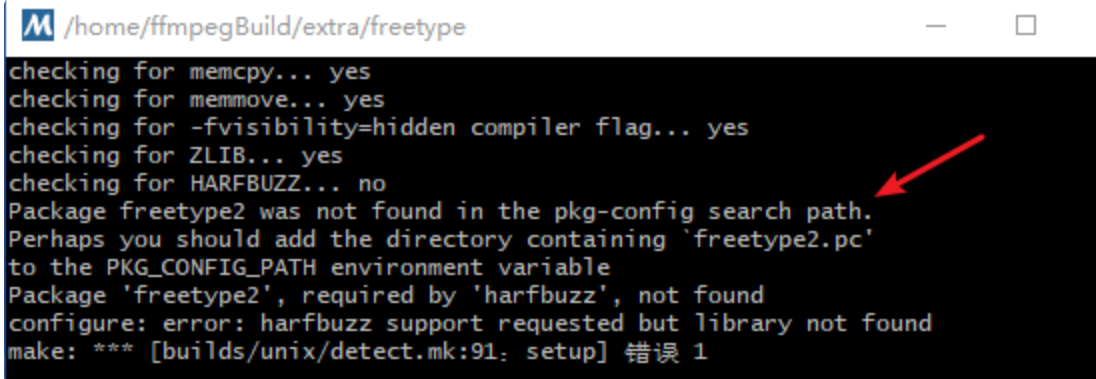
1  # 1. 首先我们把之前编译好的freetype备份
2  # 不过应该将cp换成mv都没问题，我这里用的是cp，是没问题的。
3  cp /home/ffmpegBuild/build/libfreetype/
   /home/ffmpegBuild/build/libfreetype-bk
4
5  # 2. 重新编译freetype
6  cd /home/ffmpegBuild/extra/freetype/
7  # 手动将harfbuzz、libfreetype的xxx.pc的路径导入环境变量
8  # 好像libfreetype的pkg的路径不需要导入，不过导入也没太大问题。
9  export
   PKG_CONFIG_PATH=/home/ffmpegBuild/build/harfbuzz/lib/pkgconfig:$PKG_CONFIG_PATH
10 export
   PKG_CONFIG_PATH=/home/ffmpegBuild/build/libfreetype/lib/pkgconfig:$PKG_CONFIG_PATH
11 ./configure --prefix=/home/ffmpegBuild/build/libfreetype --with-pic --
   with-zlib --without-png --with-harfbuzz=yes --without-bzip2 --without-
   fsref --without-quickdraw-toolbox --without-quickdraw-carbon --without-
   ats --disable-fast-install --disable-mmap --enable-static --enable-shared
   --host=x86_64-w64-mingw32 --build=x86_64-pc-msys
12
13 make -j4
14 make install

```

备注：如果没有执行语句

export

PKG_CONFIG_PATH=/home/ffmpegBuild/build/harfbuzz/lib/pkgconfig:\$PKG_CONFIG_PATH，会报以下的错误：



```

M /home/ffmpegBuild/extra/freetype
checking for memcpy... yes
checking for memmove... yes
checking for -fvisibility=hidden compiler flag... yes
checking for ZLIB... yes
checking for HARFBUZZ... no
Package freetype2 was not found in the pkg-config search path.
Perhaps you should add the directory containing 'freetype2.pc'
to the PKG_CONFIG_PATH environment variable
Package 'freetype2', required by 'harfbuzz', not found
configure: error: harfbuzz support requested but library not found
make: *** [builds/unix/detect.mk:91: setup] 错误 1

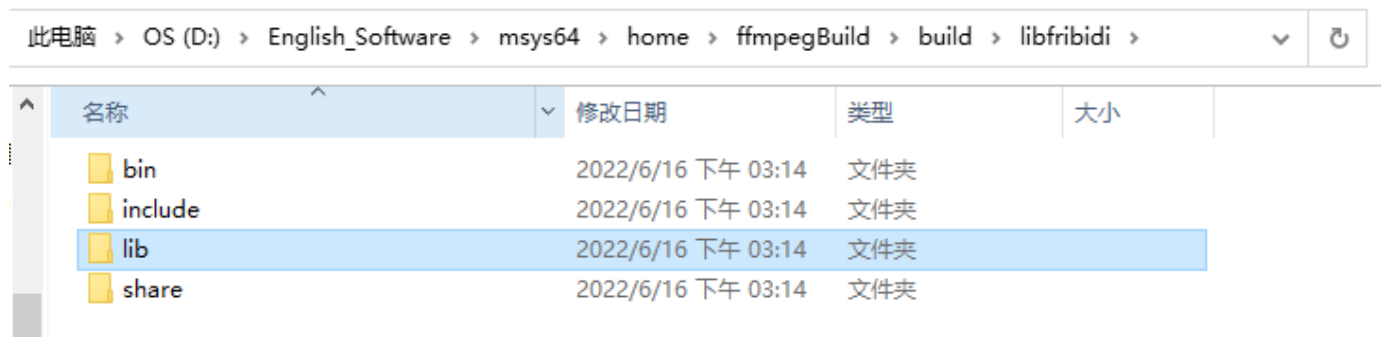
```

5.1.3 编译fribidi

Bash | 复制代码

```
1 cd /home/ffmpegBuild/extra/
2 git clone https://github.com/fribidi/fribidi
3
4 cd fribidi/
5 ./autogen.sh
6 ./configure --prefix=/home/ffmpegBuild/build/libfribidi --with-pic --
  enable-static --enable-shared
7 # 注意这里不能多核编译，否则会报c2man missing的错误。
8 # make -j8 # 会报错
9 make
10 make install
```

成功编译：



备注：

fribidi的make时不能多核编译，否则会报错。

官方的解释：<https://github.com/fribidi/fribidi/issues/45>。

通过官方的表示，主要看官方的这句话：

You don't have c2man, this is only fixed if all main pages were generated successfully(您没有c2man，只有在所有主页成功生成的情况下才会修复这个问题)。

大概意思是说, 如果你没有c2man, 那么只能等所有的主页完成才能往下编译(主页应该只doc目录的内容), 所以并行编译fribidi的话, 就会报错。 c2man应该是被Linux系统舍弃了的一个命令。

```
/home/ffmpegBuild/extra/fribidi
Running c2man
Running c2man
../missing:行81: c2man: 未找到命令
../missing:行81: c2man: 未找到命令
../missing:行81: c2man: 未找到命令
../missing:行81: c2man: 未找到命令
WARNING: 'c2man' is missing on your system.
You might have modified some files without having the proper
tools for further handling them. Check the 'README' file, it
WARNING: 'c2man' is missing on your system.
WARNING: 'c2man' is missing on your system.
WARNING: 'c2man' is missing on your system.
often tells you about the needed prerequisites for installing
You might have modified some files without having the proper
You might have modified some files without having the proper
You might have modified some files without having the proper
this package. You may also peek at any GNU archive site, in
tools for further handling them. Check the 'README' file, it
tools for further handling them. Check the 'README' file, it
tools for further handling them. Check the 'README' file, it
case some other package contains this missing 'c2man' program.
often tells you about the needed prerequisites for installing
often tells you about the needed prerequisites for installing
often tells you about the needed prerequisites for installing
this package. You may also peek at any GNU archive site, in
this package. You may also peek at any GNU archive site, in
this package. You may also peek at any GNU archive site, in
case some other package contains this missing 'c2man' program.
case some other package contains this missing 'c2man' program.
case some other package contains this missing 'c2man' program.
WARNING: 'c2man' is missing on your system.
You might have modified some files without having the proper
tools for further handling them. Check the 'README' file, it
often tells you about the needed prerequisites for installing
this package. You may also peek at any GNU archive site, in
case some other package contains this missing 'c2man' program.
WARNING: 'c2man' is missing on your system.
You might have modified some files without having the proper
WARNING: 'c2man' is missing on your system.
WARNING: 'c2man' is missing on your system.
tools for further handling them. Check the 'README' file, it
You might have modified some files without having the proper
You might have modified some files without having the proper
often tells you about the needed prerequisites for installing
tools for further handling them. Check the 'README' file, it
tools for further handling them. Check the 'README' file, it
this package. You may also peek at any GNU archive site, in
often tells you about the needed prerequisites for installing
often tells you about the needed prerequisites for installing
case some other package contains this missing 'c2man' program.
this package. You may also peek at any GNU archive site, in
this package. You may also peek at any GNU archive site, in
case some other package contains this missing 'c2man' program.
case some other package contains this missing 'c2man' program.
mv: mv mv mv ::无法获取 'c2man.stamp.tmp'无法获取无法获取无法获取 的文件状态(sta
t)'c2man.stamp.tmp''c2man.stamp.tmp''c2man.stamp.tmp': 的文件状态(stat) 的文件
状态(stat) 的文件状态(stat)No such file or directory: : :
No such file or directoryNo such file or directoryNo such file or directory

make[3]: 离开目录"/home/ffmpegBuild/extra/fribidi/doc"
make[3]: *** [Makefile:602: c2man.stamp] 错误 1
make[3]: *** [Makefile:602: c2man.stamp] 错误 1
make[3]: *** [Makefile:602: c2man.stamp] 错误 1
make[3]: *** [Makefile:602: c2man.stamp] 错误 1
make[3]: 离开目录"/home/ffmpegBuild/extra/fribidi/doc"
make[3]: 离开目录"/home/ffmpegBuild/extra/fribidi/doc"
make[3]: 离开目录"/home/ffmpegBuild/extra/fribidi/doc"
make[3]: 离开目录"/home/ffmpegBuild/extra/fribidi/doc"
mv mv mv: : 无法获取 无法获取'c2man.stamp.tmp'无法获取'c2man.stamp.tmp' 的文件状
态(stat)'c2man.stamp.tmp' 的文件状态(stat): 的文件状态(stat): No such file or d
irectory: No such file or directory
```

5.1.4 编译libxml2

```

1  cd /home/ffmpegBuild/extra/
2
3  # 从gitlab下载源码
4  git clone --depth 1 https://gitlab.gnome.org/GNOME/libxml2/
5  cd libxml2/
6
7  ./autogen.sh
8  # 这里没加静态库、动态库的选项，默认生成动态库
9  ./configure --prefix=/home/ffmpegBuild/build/libxml2 --without-python
10 make
11 make install

```

成功编译：

电脑 > OS (D:) > English_Software > msys64 > home > ffmpegBuild > build > libxml2 >

名称	修改日期	类型	大小
bin	2022/6/16 下午 03:55	文件夹	
include	2022/6/16 下午 03:55	文件夹	
lib	2022/6/16 下午 03:55	文件夹	
share	2022/6/16 下午 03:55	文件夹	

备注：

如果不想编译libxml2，在配置fontconfig即./configure时，将--enable-libxml2参数换成--disable-libxml2即可。

5.1.5 编译fontconfig


```

1  cd /home/ffmpegBuild/extra/
2
3  # 从gitlab下载源码
4  git clone --depth 1 https://gitlab.freedesktop.org/fontconfig/fontconfig
5  cd fontconfig/
6
7  # 需要先安装gperf
8  pacman -S gperf
9
10 ./autogen.sh
11 export
12   PKG_CONFIG_PATH=/home/ffmpegBuild/build/libxml2/lib/pkgconfig:$PKG_CONFIG_PATH
13 # 这一步如果在编译harfbuzz已经执行过，那么就不用再执行了
14 # export
15   PKG_CONFIG_PATH=/home/ffmpegBuild/build/libfreetype/lib/pkgconfig:$PKG_CONFIG_PATH
16
17 ./configure --prefix=/home/ffmpegBuild/build/libfontconfig --with-pic --
18   disable-docs --enable-libxml2 --enable-static --enable-shared
19 make -j8
20 make install

```

成功编译：

OS (D:) > English_Software > msys64 > home > ffmpegBuild > build > libfontconfig >

名称	修改日期	类型	大小
bin	2022/6/16 下午 04:13	文件夹	
etc	2022/6/16 下午 04:13	文件夹	
include	2022/6/16 下午 04:12	文件夹	
lib	2022/6/16 下午 04:13	文件夹	
share	2022/6/16 下午 04:13	文件夹	

5.1.6 开始编译libass

上面编译完libass所有的依赖库后，那么开始正式编译libass。

为什么要编译libass？因为subtitle filter需要用到libass去解码字体。

```

1  cd /home/ffmpegBuild/extra/
2
3  git clone --depth 1 https://github.com/libass/libass
4  cd libass/
5
6  # 查看上面的库已经导入到pkg的路径
7  echo $PKG_CONFIG_PATH
8  # 如果下面4个没导入的, 则导入
9  export
   PKG_CONFIG_PATH=/home/ffmpegBuild/build/libfontconfig/lib/pkgconfig:$PKG_
   CONFIG_PATH
10 export
   PKG_CONFIG_PATH=/home/ffmpegBuild/build/libfribidi/lib/pkgconfig:$PKG_CON
   FIG_PATH
11 # 按上面的步骤, 这两个已经导入, 所以不用再导入, 没有导入则要
12 #export
   PKG_CONFIG_PATH=/home/ffmpegBuild/build/libfreetype/lib/pkgconfig:$PKG_CO
   NFIG_PATH
13 #export
   PKG_CONFIG_PATH=/home/ffmpegBuild/build/harfbuzz/lib/pkgconfig:$PKG_CONFI
   G_PATH
14
15 ./autogen.sh
16 # 注意: 如果加上 --enable-harfbuzz选项, 会警告没有该选项(有的博客可能会加上, 这应该
   是写错了),
17 # ./configure -h可以看到确实没有, 但不会影响编译
18 ./configure --prefix=/home/ffmpegBuild/build/libass --with-pic --enable-
   static --enable-shared --enable-fontconfig
19 make -j8
20 make install

```

成功结果:

电脑 > OS (D:) > English_Software > msys64 > home > ffmpegBuild > build > libass >

名称	修改日期	类型	大小
 bin	2022/6/16 下午 04:45	文件夹	
 include	2022/6/16 下午 04:45	文件夹	
 lib	2022/6/16 下午 04:45	文件夹	

5.2 pacman安装包下载libass(建议)

直接使用pacman去下载：

```
1 # 打开该链接即可看到libass是可以直接在pacman下载的，并且可以看到对应依赖库：
2 # https://packages.msys2.org/package/mingw-w64-x86_64-libass
3 pacman -S mingw-w64-x86_64-libass
```

可以看到，pacman会自动把所需要的依赖库全都下载。

```
DELL@MINGW64 /home/ffmpegBuild/extra/freetype
$ pacman -S mingw-w64-x86_64-libass
resolving dependencies...
looking for conflicting packages...
warning: dependency cycle detected:
warning: mingw-w64-x86_64-harfbuzz will be installed before its mingw-w64-x86_64-
freetype dependency

Packages (11) mingw-w64-x86_64-brotli-1.0.9-4
mingw-w64-x86_64-fontconfig-2.14.0-1
mingw-w64-x86_64-freetype-2.12.1-1
mingw-w64-x86_64-fribidi-1.0.12-1
mingw-w64-x86_64-glib2-2.72.2-1
mingw-w64-x86_64-graphite2-1.3.14-2
mingw-w64-x86_64-harfbuzz-4.3.0-1
mingw-w64-x86_64-libpng-1.6.37-6 mingw-w64-x86_64-pcre-8.45-1
mingw-w64-x86_64-wineditline-2.205-3
mingw-w64-x86_64-libass-0.16.0-1

Total Download Size:    7.61 MiB
Total Installed Size:  47.70 MiB

:: Proceed with installation? [Y/n] y
```

该方法比较简单，但是后续需要自己去下载对应的dll，这里不再讲述。

三 编译FFmpeg

1. 下载FFmpeg

```
1 cd /home/ffmpegBuild/extra
2 git clone git://source.ffmpeg.org/ffmpeg.git
3 # 网速太慢可以使用码云的链接
4 # git clone https://gitee.com/mirrors/ffmpeg.git
5
6 cd ffmpeg
7 # 查看版本
8 git branch -a
9 # 这里我们选择4.3版本
10 git checkout remotes/origin/release/4.3
```

2. 编译FFmpeg

如果有编译过ffmpeg，应当先把中间的.o等文件删掉。

```
1 make clean
2 make distclean
```

然后再进行编译。

去到cd /home/ffmpegBuild/extra/ffmpeg目录下，创建一个build.sh脚本(实际上不想创建脚本的话，是可以直接执行下面的内容的)：

1) 如果libass是源码编译的，脚本使用以下内容：

```
1  ./configure \  
2  --prefix=/home/ffmpegBuild/build/ffmpeg-4.3 \  
3  --arch=x86_64 \  
4  --enable-shared \  
5  --enable-gpl \  
6  --enable-libfdk-aac \  
7  --enable-nonfree \  
8  --enable-libx264 \  
9  --enable-libmp3lame \  
10 --enable-filter=subtitles \  
11 --enable-libass \  
12 --extra-cflags="-I/home/ffmpegBuild/build/libfdk-aac/include" \  
13 --extra-ldflags="-L/home/ffmpegBuild/build/libfdk-aac/lib" \  
14 --extra-cflags="-I/home/ffmpegBuild/build/libvpx/include" \  
15 --extra-ldflags="-L/home/ffmpegBuild/build/libvpx/lib" \  
16 --extra-cflags="-I/home/ffmpegBuild/build/libx264/include" \  
17 --extra-ldflags="-L/home/ffmpegBuild/build/libx264/lib" \  
18 --extra-cflags="-I/home/ffmpegBuild/build/libmp3lame/include" \  
19 --extra-ldflags="-L/home/ffmpegBuild/build/libmp3lame/lib" \  
20 --extra-cflags="-I/home/ffmpegBuild/build/libass/include" \  
21 --extra-ldflags="-L/home/ffmpegBuild/build/libass/lib" \  
22
```

并且按照libass源码编译的话，在执行前，要将libass的pkg路径导入到环境变量。否则会报：ERROR: libass not found using pkg-config的错误。

```
1  export  
   PKG_CONFIG_PATH=/home/ffmpegBuild/build/libass/lib/pkgconfig:$PKG_CONFIG_  
   PATH
```

2) 如果libass不是源码编译，而是直接使用pacman -S mingw-w64-x86_64-libass下载的，则脚本使用以下内容：

```
1  ./configure \  
2  --prefix=/home/ffmpegBuild/build/ffmpeg-4.3 \  
3  --arch=x86_64 \  
4  --enable-shared \  
5  --enable-gpl \  
6  --enable-libfdk-aac \  
7  --enable-nonfree \  
8  --enable-libx264 \  
9  --enable-libmp3lame \  
10 --enable-filter=subtitles \  
11 --enable-libass \  
12 --extra-cflags="-I/home/ffmpegBuild/build/libfdk-aac/include" \  
13 --extra-ldflags="-L/home/ffmpegBuild/build/libfdk-aac/lib" \  
14 --extra-cflags="-I/home/ffmpegBuild/build/libvpx/include" \  
15 --extra-ldflags="-L/home/ffmpegBuild/build/libvpx/lib" \  
16 --extra-cflags="-I/home/ffmpegBuild/build/libx264/include" \  
17 --extra-ldflags="-L/home/ffmpegBuild/build/libx264/lib" \  
18 --extra-cflags="-I/home/ffmpegBuild/build/libmp3lame/include" \  
19 --extra-ldflags="-L/home/ffmpegBuild/build/libmp3lame/lib"
```

关于更多ffmpeg的编译选项可以参考：

<https://blog.csdn.net/HW140701/article/details/124493962>。

然后执行脚本，并make、make install即可。

```
1  sh build.sh  
2  make -j8  
3  make install
```

如果脚本报下图错误，说明拷贝出现编码不一致了，手动输入脚本的内容即可。

```
$ sh build.sh
Unknown option "".
See ./configure --help for available options.
build.sh:行2: --prefix=/home/ffmpegBuild/build/ffmpeg-4.3: No such file or directory
build.sh:行3: --arch=x86_64: 未找到命令
build.sh:行4: --enable-shared: 未找到命令
build.sh:行5: --enable-gpl: 未找到命令
build.sh:行6: --enable-libfdk-aac: 未找到命令
build.sh:行7: --enable-nonfree: 未找到命令
```

但是目前在make -j8时出现以下问题，：

```
imgutils_init.o libavutil/x86/l1s.o libavutil/x86/l1s_init.o libavutil/x86/pixelu
utils.o libavutil/x86/pixelutils_init.o libavutil/x86_font_data.o libavutil/xtea.
o > libavutil/avutil-56.def
D:\English_Software\msys2\mingw64\bin\ar.exe: library.????197609.23927.lib: Inv
alid argument
Could not create temporary library.
make: *** [ffbuild/library.mak:102: libavutil/avutil-56.dll] 错误 1
```

然后我根据整个报错提示排查，看到./compat/windows/makedef文件：

```
DELL@MINGW64 /home/ffmpegBuild/extra/ffmpeg
# make
EXTERN_PREFIX="" AR="ar" NM="nm -g" ./compat/windows/makedef libavutil/libavutil
.ver libavutil/adler32.o libavutil/aes.o libavutil/aes_ctr.o libavutil/audio_fif
o.o libavutil/avsscanf.o libavutil/avstring.o libavutil/base64.o libavutil/blowf
ish.o libavutil/bprint.o libavutil/buffer.o libavutil/camellia.o libavutil/cast5
.o libavutil/channel_layout.o libavutil/color_utils.o libavutil/cpu.o libavutil/
crc.o libavutil/des.o libavutil/dict.o libavutil/display.o libavutil/downmix_inf
o.o libavutil/encryption_info.o libavutil/error.o libavutil/eval.o libavutil/fif
o.o libavutil/file.o libavutil/file_open.o libavutil/fixd_dsp.o libavutil/float
_dsp.o libavutil/frame.o libavutil/hash.o libavutil/hdr_dynamic_metadata.o libav
util/hmac.o libavutil/hwcontext.o libavutil/hwcontext_d3d11va.o libavutil/hwcont
ext_dxva2.o libavutil/imgutils.o libavutil/integer.o libavutil/intmath.o libavut
il/lfg.o libavutil/l1s.o libavutil/log.o libavutil/log2_tab.o libavutil/lzo.o li
bavutil/mastering_display_metadata.o libavutil/mathematics.o libavutil/md5.o lib
avutil/mem.o libavutil/murmur3.o libavutil/opt.o libavutil/parseutils.o libavuti
l/pixdesc.o libavutil/pixelutils.o libavutil/random_seed.o libavutil/rational.o
libavutil/rc4.o libavutil/reverse.o libavutil/ripemd.o libavutil/samplefmt.o lib
avutil/sha.o libavutil/sha512.o libavutil/slicethread.o libavutil/spherical.o li
bavutil/stereo3d.o libavutil/tea.o libavutil/threadmessage.o libavutil/time.o li
bavutil/timecode.o libavutil/tree.o libavutil/twofish.o libavutil/tx.o libavutil
/utls.o libavutil/x86/cpu.o libavutil/x86/cpuid.o libavutil/x86/fixd_dsp.o lib
avutil/x86/fixd_dsp_init.o libavutil/x86/float_dsp.o libavutil/x86/float_dsp_in
it.o libavutil/x86/imgutils.o libavutil/x86/imgutils_init.o libavutil/x86/l1s.o
libavutil/x86/l1s_init.o libavutil/x86/pixelutils.o libavutil/x86/pixelutils_ini
t.o libavutil/x86_font_data.o libavutil/xtea.o > libavutil/avutil-56.def
D:\English_Software\msys64\mingw64\bin\ar.exe: library.????197609.184.lib: Inva
lid argument
Could not create temporary library.
make: *** [ffbuild/library.mak:102: libavutil/avutil-56.dll] 错误 1
DELL@MINGW64 /home/ffmpegBuild/extra/ffmpeg
```

下面看到，打印"Could not create temporary library."是因为上面一个命令执行失败了，\$?代表上一个命令的执行结果，为0代表执行成功，非0代表执行错误。

所以很明显，就是\$AR rcs \${libname} \$@ >/dev/null 或者 lib.exe -out:\${libname} \$@ >/dev/null命令执行失败了，而错误就是说ar.exe不认识这个库名。

```
# Create a lib temporarily to dump symbols from.
# It's just much easier to do it this way
libname=$(mktemp -u "library").lib

trap 'rm -f -- $libname' EXIT

if [ -n "$AR" ]; then
    $AR rcs ${libname} $@ >/dev/null
else
    lib.exe -out:${libname} $@ >/dev/null
fi

if [ $? != 0 ]; then
    echo "Could not create temporary library." >&2
    exit 1
fi
```

而库名就是由mktemp()函数创建的，其中使用到了主机名这\${HOSTNAME}个环境变量，而我的环境变量是中文名字，ar.exe不支持中文名字，所以就出现了非法参数的错误。


```

# mktemp isn't POSIX, so supply an implementation
mktemp() {
    echo "${2%%XXX*}.${HOSTNAME}.${UID}.$$"
}

if [ $# -lt 2 ]; then
    echo "Usage: makedef <version_script> <objects>" >&2
    exit 0
fi

vscript=$1
shift

if [ ! -f "$vscript" ]; then
    echo "Version script does not exist" >&2
    exit 1
fi

for object in "$@"; do
    if [ ! -f "$object" ]; then
        echo "Object does not exist: ${object}" >&2
        exit 1
    fi
done

# Create a lib temporarily to dump symbols from.
# It's just much easier to do it this way
libname=$(mktemp -u "library").lib

trap 'rm -f -- $libname' EXIT

if [ -n "$AR" ]; then
    $AR rcs ${libname} $@ >/dev/null
else
    lib.exe -out:${libname} $@ >/dev/null
fi

if [ $? != 0 ]; then
    echo "Could not create temporary library." >&2
    exit 1
fi

```

解决方法：

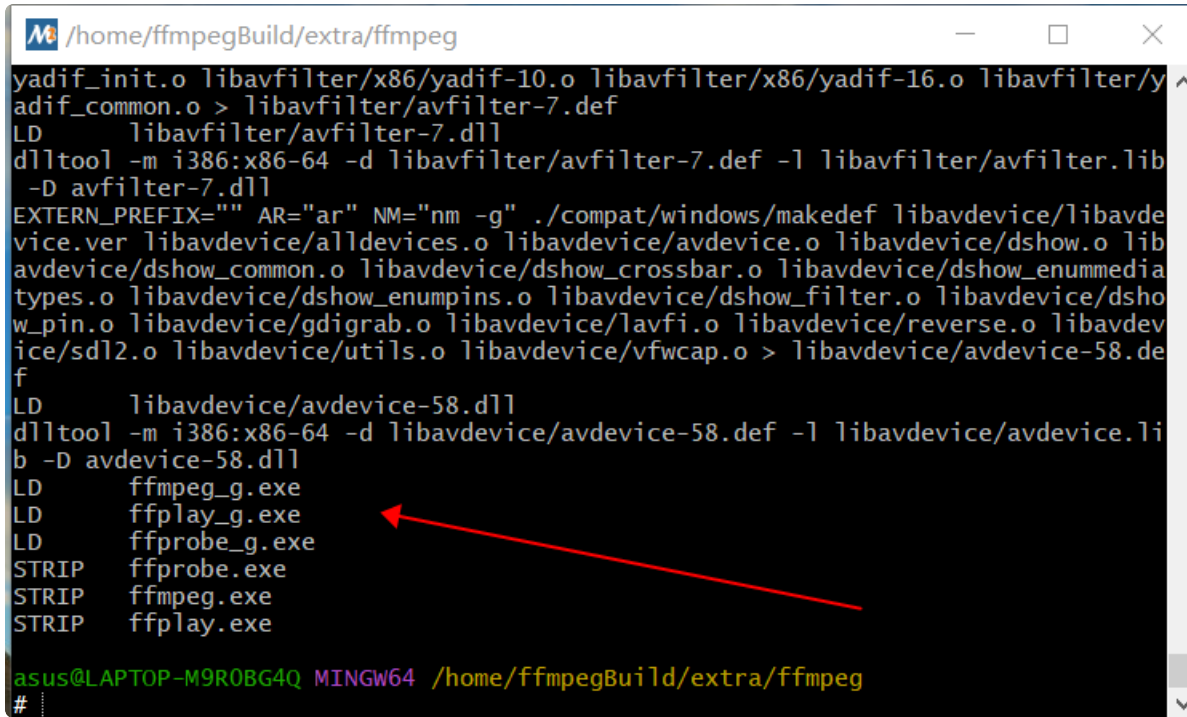
```

1  mktemp() {
2      echo "${2%%XXX*}.${UID}.${UID}.$$"
3  }
4  换成:
5  mktemp() {
6      echo "${2%%XXX*}.${UID}.${UID}.$$"
7  }

```

然后重新make即可编译成功，然后再安装即可。

后面我也换了另一台电脑成功编译，make -j8结果如图所示：



```
/home/ffmpegBuild/extra/ffmpeg
yadif_init.o libavfilter/x86/yadif-10.o libavfilter/x86/yadif-16.o libavfilter/y
adif_common.o > libavfilter/avfilter-7.def
LD      libavfilter/avfilter-7.dll
dlltool -m i386:x86-64 -d libavfilter/avfilter-7.def -l libavfilter/avfilter.lib
-D avfilter-7.dll
EXTERN_PREFIX="" AR="ar" NM="nm -g" ./compat/windows/makedef libavdevice/libavde
vice.ver libavdevice/alldevices.o libavdevice/avdevice.o libavdevice/dshow.o lib
avdevice/dshow_common.o libavdevice/dshow_crossbar.o libavdevice/dshow_enummedia
types.o libavdevice/dshow_enumpins.o libavdevice/dshow_filter.o libavdevice/dsho
w_pin.o libavdevice/gdigrab.o libavdevice/lavfi.o libavdevice/reverse.o libavdev
ice/sdl2.o libavdevice/utils.o libavdevice/vfwcap.o > libavdevice/avdevice-58.de
f
LD      libavdevice/avdevice-58.dll
dlltool -m i386:x86-64 -d libavdevice/avdevice-58.def -l libavdevice/avdevice.li
b -D avdevice-58.dll
LD      ffmpeg_g.exe
LD      ffplay_g.exe
LD      ffprobe_g.exe
STRIP   ffprobe.exe
STRIP   ffmpeg.exe
STRIP   ffplay.exe

asus@LAPTOP-M9R0BG4Q MINGW64 /home/ffmpegBuild/extra/ffmpeg
#
```

make install结果：

上面编译libass时，参考了这篇博客：<https://blog.csdn.net/theartedly/article/details/110980820>。

Cygwin编译方法请参考：

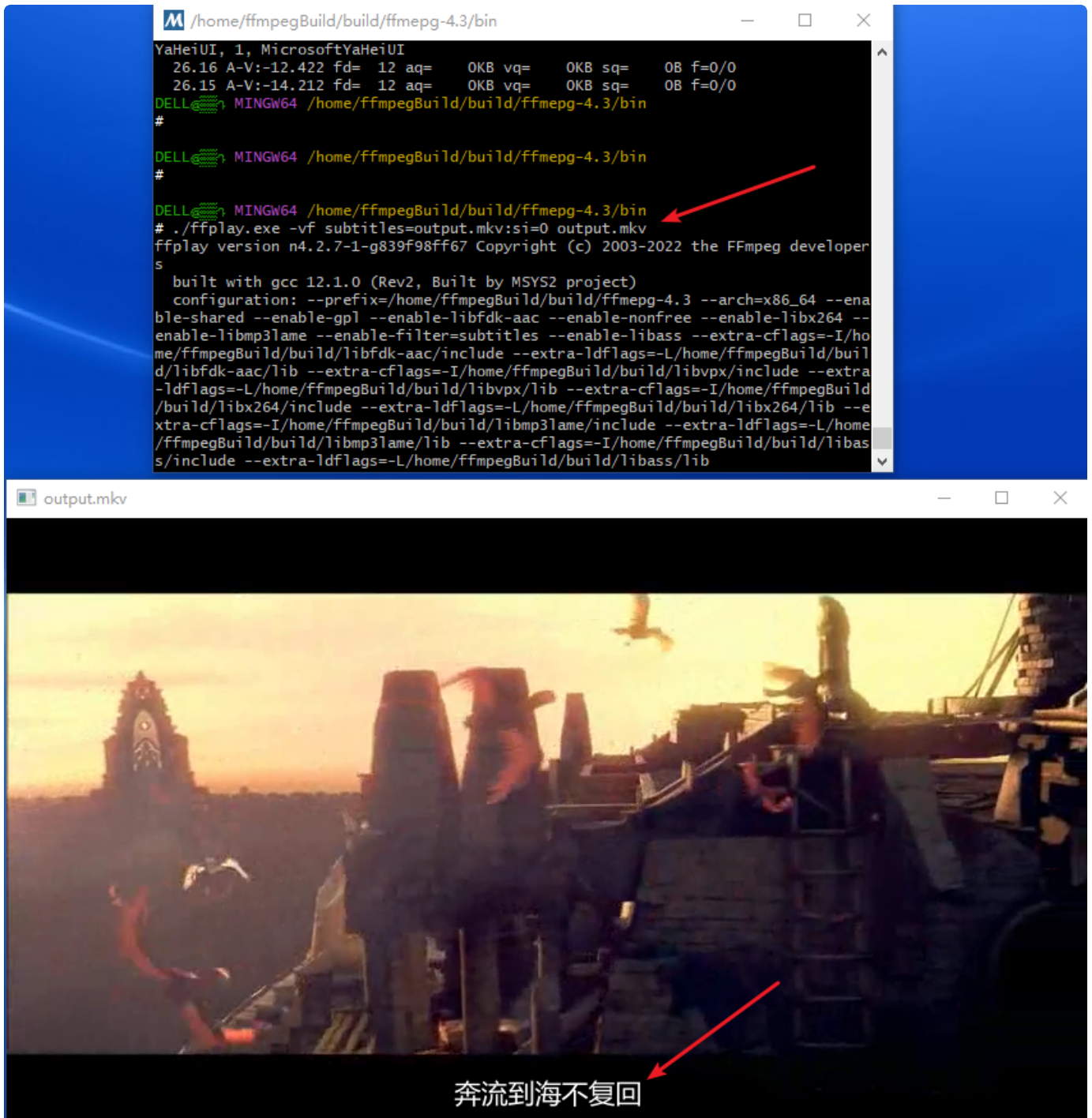
https://blog.csdn.net/qq_21743659/article/details/109379921

3. 简单使用ffplay播放测试

去到/home/ffmpegBuild/build/ffmpeg-4.3/bin，点击ffplay.exe，它会告诉你缺少哪个dll，首先去/home/ffmpegBuild/build下将上面编译出来的dll拷贝到ffplay.exe同目录下。

如果还有缺少的，可以去C盘的Windows/system32，以及去 msys64的安装目录即/mingw64/bin(64位)下去找。例如libintl-8.dll就是在/mingw64/bin下面找到的。

```
1 # 播放视频，注意如果不加上过滤器选项，字幕是不会显示出来的
2 ./ffplay.exe output.mkv
3
4 # 播放视频，并且播放字幕，字幕下标为第0个字幕流
5 ./ffplay.exe -vf subtitles=output.mkv:si=0 output.mkv
```



备注：其实不加上libass库，使用ffplay也是能播放字幕的，但是无法使用ffmpeg对视频添加字幕。

```
1  # 转换为通用格式ass，然后调用libass库渲染。
2  ./ffplay -f lavfi
    "color,subtitles=output.mkv:force_style='FontName=DejaVu
    Serif,FontSize=28,MarginV=60,PrimaryColour=&HAA00FF00'[out0];amovie=outpu
    t.mkv[out1]"
```

四 使用QT测试编译生成的ffmpeg

使用ffmpeg给视频添加字幕的测试程序可以参考这篇博客：

https://blog.csdn.net/qq_21743659/article/details/109305411。