

QT CMake课程

QT CMake课程.....	1
1. QT课程更新	5
1.1. 1 cmake qt环境准备	5
1.1.1. QT.....	5
1.1.2. VS2022	5
1.1.3. qt addone.....	5
1.1.4. cmake.....	5
1.2. 2 免费部分	5
1.2.1. 不讲环境 将一个最简单的示例	5
1.2.2. 前置条件	5
1.3. 3 自动编译.....	5
1.3.1. 输出	5
1.3.2. release 和Debug	5
1.3.3. 去掉控制台 WIN32	5
1.4. 4 编译课程示例项目	5
2. CMake课程更新	5
2.1. 1 cmake qt环境准备	6
2.1.1. QT.....	6
2.1.2. VS2022	6
2.1.3. qt addone.....	6
2.1.4. cmake.....	6
2.2. 2 免费部分	6
2.2.1. 不讲环境 将一个最简单的示例	6
2.3. 3 自动编译.....	7
2.3.1. 输出	7
2.3.2. release 和Debug	7
2.3.3. 去掉控制台 WIN32	7
2.4. 4 自动安装软件	7
2.4.1. cmake install	7
2.5. x86 x64项目区分	7
3. QT 课程更新	7
3.1. cmake qt环境准备	7
3.1.1. QT.....	7
3.1.2. VS2022	7
3.1.3. qt addone.....	8
3.1.4. cmake.....	8
3.1.5. 下载	8

3.2.	qt5 + cmake编程示例（免费）	8
3.2.1.	QT & CMake项目说明	8
3.2.2.	项目代码	8
3.2.3.	moc 信号槽代码生成	14
3.2.4.	uic 界面代码生成	14
3.2.5.	rcc 资源代码生成	14
3.2.6.	加载qt库和头文件	15
3.3.	QT区分32位和64位程序	15
3.3.1.	cmake -S . -B build - DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017/lib/cmake/Qt5/ -A Win32	15
3.3.2.	cmake -S . -B build - DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/ -A x64	15
3.4.	QT区分Debug和Release版本	15
3.4.1.	cmake --build build --config Debug	15
3.4.2.	cmake --build build --config Release	15
3.5.	自动复制QT依赖库	15
3.5.1.	Win32	15
3.5.2.	x64	15
3.6.	XImageEdit使用cmake编译	16
3.6.1.	file(GLOB RC "*.rc")	16
3.6.2.	add_executable(\${PROJECT_NAME} WIN32 \${SOURCE} \${HEAD} \${QT} \${RC}) 16	
4.	CMake课程更新	16
4.1.	qt5 + cmake编程示例（免费）	16
4.1.1.	QT & CMake项目说明	16
4.1.2.	项目代码	16
4.1.3.	moc 信号槽代码生成	22
4.1.4.	uic 界面代码生成	22
4.1.5.	rcc 资源代码生成	22
4.1.6.	加载qt库和头文件	22
4.2.	QT区分32位和64位程序	22
4.2.1.	cmake -S . -B build - DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017/lib/cmake/Qt5/ -A Win32	22
4.2.2.	cmake -S . -B build - DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/ -A x64	23
4.3.	QT区分Debug和Release版本	23
4.3.1.	cmake --build build --config Debug	23
4.3.2.	cmake --build build --config Release	23
4.4.	自动复制QT依赖库	23
4.4.1.	Win32	23

4.4.2.	x64.....	23
4.5.	cmake install QT项目和依赖动态库	23
4.5.1.	J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe	23
4.5.2.	install(TARGETS \${PROJECT_NAME} CONFIGURATIONS Debug Release RUNTIME DESTINATION \${CMAKE_INSTALL_PREFIX}/\${CMAKE_VS_PLATFORM_NAME}/bin) install(CODE "execute_process(COMMAND \${windeployqt} \${CMAKE_INSTALL_PREFIX}/\${CMAKE_VS_PLATFORM_NAME}/bin/\${<TARGET_FILE _NAME:\${PROJECT_NAME}> WORKING_DIRECTORY \${CMAKE_BINARY_DIR})") ...	23
5.	QT CMake课程大纲.....	24
5.1.	qt5 + cmake编程示例	24
5.1.1.	项目代码	24
5.1.2.	moc 信号槽代码生成	30
5.1.3.	uic 界面代码生成.....	30
5.1.4.	rcc 资源代码生成.....	30
5.1.5.	加载qt库和头文件	30
5.2.	QT区分32位和64位程序	30
5.2.1.	cmake -S . -B build - DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017/lib/cmake/Qt5/ -A Win32	30
5.2.2.	cmake -S . -B build - DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/ -A x64	30
5.3.	QT区分Debug和Release版本	30
5.3.1.	cmake --build build --config Debug	30
5.3.2.	cmake --build build --config Release	31
5.4.	自动复制QT依赖库	31
5.4.1.	Win32.....	31
5.4.2.	x64.....	31
5.5.	cmake install QT项目和依赖动态库	31
5.5.1.	J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe	31
5.5.2.	install(TARGETS \${PROJECT_NAME} CONFIGURATIONS Debug Release RUNTIME DESTINATION \${CMAKE_INSTALL_PREFIX}/\${CMAKE_VS_PLATFORM_NAME}/bin) install(CODE "execute_process(COMMAND \${windeployqt} \${CMAKE_INSTALL_PREFIX}/\${CMAKE_VS_PLATFORM_NAME}/bin/\${<TARGET_FILE _NAME:\${PROJECT_NAME}> WORKING_DIRECTORY \${CMAKE_BINARY_DIR})") ...	31
5.6.	cmake打包qt程序.....	31
5.6.1.	J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe	31
5.6.2.	打包成zip文件.....	32
5.7.	去掉控制台 WIN32.....	32
6.	ffmpeg实训课更新	32
6.1.	1 cmake qt环境准备	32

6.1.1.	QT.....	32
6.1.2.	VS2022	32
6.1.3.	qt addone.....	32
6.1.4.	cmake.....	32
6.1.5.	下载	32
6.2.	2 免费部分	32
6.2.1.	不讲环境 将一个最简单的示例	32
6.3.	3 自动编译.....	32
6.3.1.	输出	32
6.3.2.	release 和Debug	32
6.3.3.	去掉控制台 WIN32	32
6.4.	4 自动安装软件.....	32
6.4.1.	cmake install	33
6.5.	5 自动编译ffmpeg	33
6.6.	6 编译项目和发布.....	33
6.7.	x86 x64项目区分	33
7.	微服务云盘实训更新	33
8.	编译Qt	33
8.1.	cmake -S . -B build - DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/	33
8.2.	J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe	33
8.3.	target_link_libraries(\${PROJECT_NAME} Qt5::Widgets xcodec)	33
8.4.	find_package(Qt5 COMPONENTS Widgets REQUIRED).....	33

1. QT课程更新

1.1.1 cmake qt环境准备

1.1.1.1 QT

1.1.1.2 VS2022

1.1.1.3 qt addone

旧项目升级

1.1.1.4 cmake

生成vs 后打开

1.2.2 免费部分

1.2.2.1 不讲环境 将一个最简单的示例

1.2.2.2 前置条件

vs2022 c++ 安装完整

1.3.3 自动编译

1.3.3.1 输出

1.3.3.2 release 和Debug

1.3.3.3 去掉控制台 WIN32

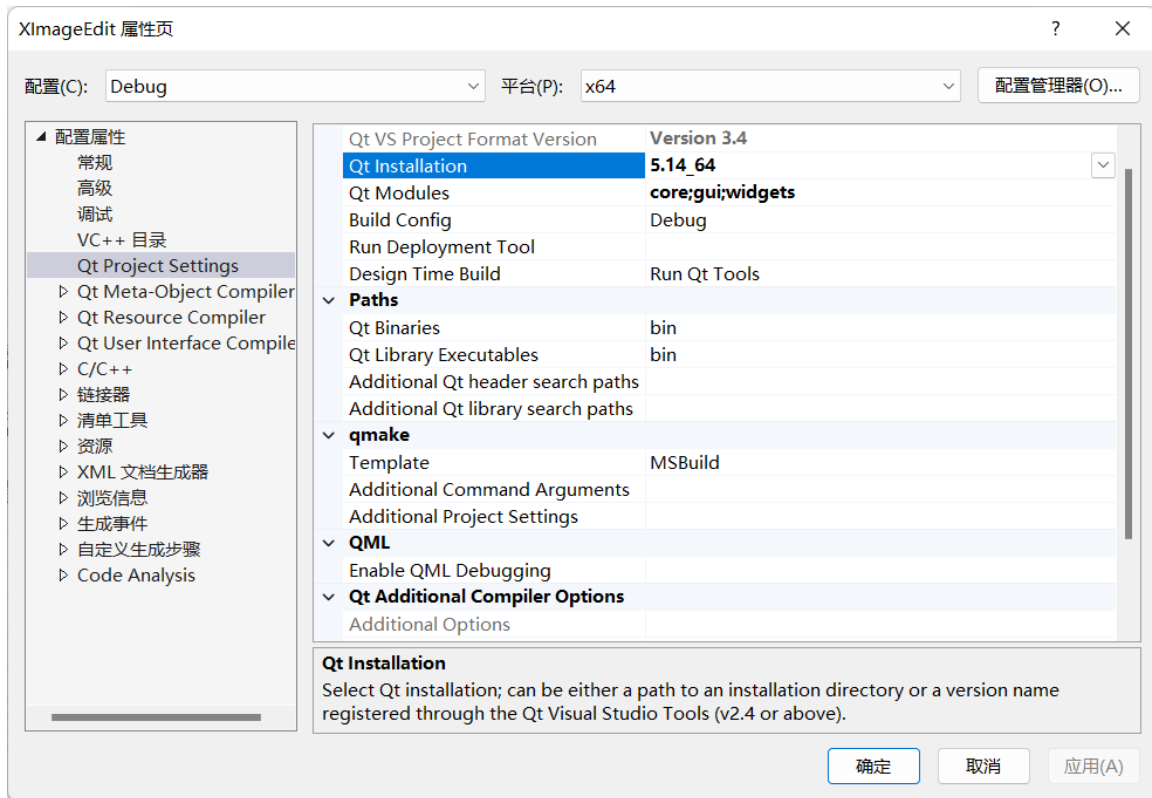
1.4.4 编译课程示例项目

2. CMake课程更新

2.1.1 cmake qt环境准备

2.1.1.1. QT

2.1.1.2. VS2022



2.1.3. qt addone

旧项目升级

2.1.4. cmake

生成vs 后打开

2.2.2 免费部分

2.2.1. 不讲环境 将一个最简单的示例

2.3.3 自动编译

2.3.1. 输出

2.3.2. release 和Debug

2.3.3. 去掉控制台 WIN32

2.4.4 自动安装软件

2.4.1. cmake install

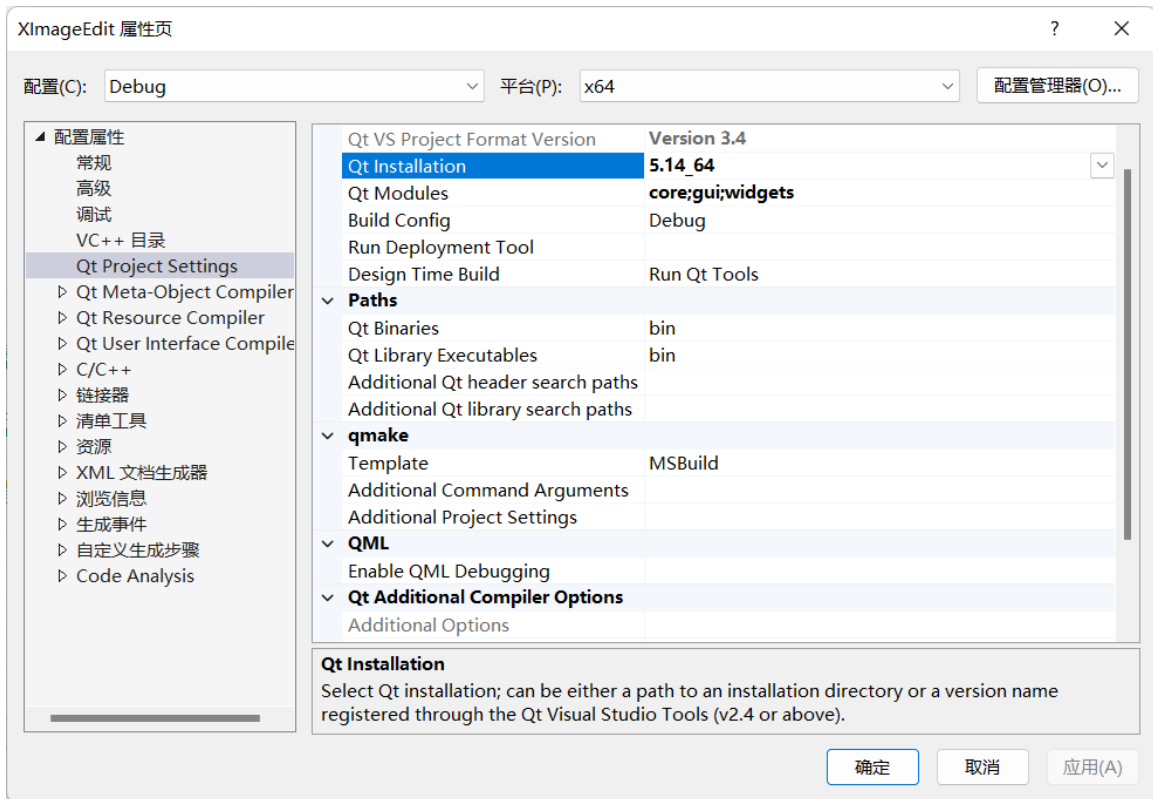
2.5. x86 x64项目区分

3. QT 课程更新

3.1. cmake qt环境准备

3.1.1. QT

3.1.2. VS2022



3.1.3. qt addone

旧项目升级

3.1.4. cmake

生成vs 后打开

3.1.5. 下载

<https://pan.baidu.com/s/12UfEFzR9ckAJdHOW6JjYJg?pwd=1234>

3.2. qt5 + cmake编程示例（免费）

3.2.1. QT & CMake项目说明

3.2.2. 项目代码

qtcmake.h


```
#pragma once
```

```
#include <QtWidgets/QWidget>
```

```
#include "ui_qtcmake.h"
```

```
class QtCmake : public QWidget
```

```
{
```

```
    Q_OBJECT
```

```
public:
```

```
    QtCmake(QWidget *parent = nullptr);
```

```
    ~QtCmake();
```

```
public slots:
```

```
    void TestCMake();
```

```
private:
```

```
    Ui::QtCmakeClass ui;
```

```
};
```

```
qtcmake.cpp
```

```
#include "qtcmake.h"
```

```
#include <QMessageBox>
```

```
QtCmake::QtCmake(QWidget *parent)
```

```
    : QWidget(parent)
```

```
{
```

```
    ui.setupUi(this);
```

```
}
```

```
QtCmake::~QtCmake()
```

```
{}
```

```
void QtCmake::TestCMake()
```

```
{
```

```
    QMessageBox::information(this, "", "Test CMake");  
}
```

main.cpp

```
#include "qtcmake.h"  
#include <QtWidgets/QApplication>  
  
int main(int argc, char *argv[])  
{  
    QApplication a(argc, argv);  
    QtCmake w;  
    w.show();  
    return a.exec();  
}
```

qtcmake.ui

```
<?xml version="1.0" encoding="UTF-8"?>  
<ui version="4.0">  
    <class>QtCmakeClass</class>  
    <widget class="QWidget" name="QtCmakeClass">  
        <property name="geometry">  
            <rect>  
                <x>0</x>  
                <y>0</y>  
                <width>875</width>  
                <height>600</height>  
            </rect>  
        </property>  
        <property name="windowTitle">  
            <string>QtCmake</string>  
        </property>
```

```

<property name="styleSheet">
  <string notr="true"/>
</property>
<widget class="QPushButton" name="pushButton">
  <property name="geometry">
    <rect>
      <x>210</x>
      <y>130</y>
      <width>481</width>
      <height>301</height>
    </rect>
  </property>
  <property name="styleSheet">
    <string notr="true">font: 22pt &quot;黑体&quot;;
background-image: url(/QtCmake/cmake_cpp.jpg);</string>
  </property>
  <property name="text">
    <string>Test CMake</string>
  </property>
</widget>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources>
  <include location="qtcmake.qrc"/>
</resources>
<connections>
  <connection>
    <sender>pushButton</sender>
    <signal>clicked()</signal>
    <receiver>QtCmakeClass</receiver>
    <slot>TestCMake()</slot>
  </connection>
</connections>

```

```

<hints>
  <hint type="sourcelabel">
    <x>601</x>
    <y>340</y>
  </hint>
  <hint type="destinationlabel">
    <x>1241</x>
    <y>410</y>
  </hint>
</hints>
</connection>
</connections>
<slots>
  <slot>TestCMake()</slot>
</slots>
</ui>

```

qtcmake.qrc

```

<RCC>
  <qresource prefix="QtCmake">
    <file>cmake_cpp.jpg</file>
  </qresource>
</RCC>

```

CMakeLists.txt

```

cmake_minimum_required(VERSION 3.20)
project(qt_cmake)

# cmake -S . -B build -DQT5=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64
# cmake -S . -B build -DQT5=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64 -
DCMAKE_INSTALL_PREFIX=out

```

```

# cmake -S . -B build -
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/
# J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe
if(NOT Qt5_DIR)
    set(Qt5_DIR ${QT5}/lib/cmake/Qt5)
endif()
set(windeployqt ${QT5}/bin/windeployqt.exe)
if(Qt5)
    set(QT5 ${Qt5})
endif()

message("Qt5_DIR = ${Qt5_DIR}")
file(GLOB SOURCE "*.cpp" "*.c")
file(GLOB HEAD "*.hpp" "*.h")
file(GLOB QT "*.uic" "*.qrc")

set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)
set(CMAKE_AUTOUIC ON)

add_executable(${PROJECT_NAME} WIN32 ${SOURCE} ${HEAD} ${QT})
find_package(Qt5 COMPONENTS Widgets REQUIRED)
target_link_libraries(${PROJECT_NAME}
    Qt5::Widgets
)

add_custom_command(OUTPUT test_output
    WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
    DEPENDS ${PROJECT_NAME}
    COMMAND ${windeployqt} $<TARGET_FILE:${PROJECT_NAME}>

```

```
COMMENT "test for add_custom_command")
```

```
add_custom_target(copy_qt  
COMMAND ${windeployqt} $<TARGET_FILE:${PROJECT_NAME}>  
)
```

```
message("CMAKE_INSTALL_PREFIX = ${CMAKE_INSTALL_PREFIX}")
```

```
install(TARGETS ${PROJECT_NAME}  
        CONFIGURATIONS Debug  
        RUNTIME DESTINATION Debug/bin)  
install(CODE "message(hello)")  
install(CODE "message(${CMAKE_INSTALL_PREFIX})")
```

```
install(CODE "execute_process(COMMAND  
${windeployqt}  
${CMAKE_INSTALL_PREFIX}/Debug/bin/${<TARGET_FILE_NAME:${PROJECT_N  
AME}>  
WORKING_DIRECTORY ${CMAKE_BINARY_DIR}  
)")
```

3.2.3. moc 信号槽代码生成

```
set(CMAKE_AUTOMOC ON)
```

3.2.4. uic 界面代码生成

```
set(CMAKE_AUTOUIC ON)
```

```
file(GLOB QT "*.uic" "*.qrc")
```

3.2.5. rcc 资源代码生成

```
set(CMAKE_AUTORCC ON)
```

3.2.6. 加载qt库和头文件

```
cmake -S . -B build -  
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/  
  
find_package(Qt5 COMPONENTS Widgets REQUIRED)  
  
target_link_libraries(${PROJECT_NAME}  
Qt5::Widgets  
)
```

3.3. QT区分32位和64位程序

3.3.1. cmake -S . -B build -
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017/lib/cmake/Qt5/ -A Win32

3.3.2. cmake -S . -B build -
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/ -A x64

3.4. QT区分Debug和Release版本

3.4.1. cmake --build build --config Debug

3.4.2. cmake --build build --config Release

3.5. 自动复制QT依赖库

3.5.1. Win32

```
J:\Qt\Qt5.14.2\5.14.2\msvc2017\bin\windeployqt.exe  
build/Release/qt_cmake.exe
```

3.5.2. x64

```
J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe  
build/Debug/qt_cmake.exe
```

3.6. XImageEdit使用cmake编译

3.6.1. file(GLOB RC "*.rc")

3.6.2. add_executable(\${PROJECT_NAME} WIN32 \${SOURCE} \${HEAD} \${QT}
\${RC})

4. [CMake课程更新](#)

4.1. qt5 + cmake编程示例 （免费）

4.1.1. QT & CMake项目说明

4.1.2. 项目代码

qtcmake.h

```
#pragma once
```

```
#include <QtWidgets/QWidget>
```

```
#include "ui_qtcmake.h"
```

```
class QtCmake : public QWidget
```

```
{
```

```
    Q_OBJECT
```

```
public:
```

```
    QtCmake(QWidget *parent = nullptr);
```

```
    ~QtCmake();
```

```
public slots:
```

```
    void TestCMake();
```

```
private:
```

```
    Ui::QtCmakeClass ui;
```

```
};
```


qtcmake.cpp

```
#include "qtcmake.h"
#include <QMessageBox>
QtCmake::QtCmake(QWidget *parent)
    : QWidget(parent)
{
    ui.setupUi(this);
}

QtCmake::~QtCmake()
{}

void QtCmake::TestCMake()
{
    QMessageBox::information(this, "", "Test CMake");
}
```

main.cpp

```
#include "qtcmake.h"
#include <QtWidgets/QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QtCmake w;
    w.show();
    return a.exec();
}
```

qtcmake.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>QtCmakeClass</class>
  <widget class="QWidget" name="QtCmakeClass">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>875</width>
        <height>600</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>QtCmake</string>
    </property>
    <property name="styleSheet">
      <string notr="true"/>
    </property>
    <widget class="QPushButton" name="pushButton">
      <property name="geometry">
        <rect>
          <x>210</x>
          <y>130</y>
          <width>481</width>
          <height>301</height>
        </rect>
      </property>
      <property name="styleSheet">
        <string notr="true">font: 22pt &quot;黑体&quot;;
background-image: url(/QtCmake/cmake_cpp.jpg);</string>
      </property>

```

```

    <property name="text">
        <string>Test CMake</string>
    </property>
</widget>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources>
    <include location="qtcmake.qrc"/>
</resources>
<connections>
    <connection>
        <sender>pushButton</sender>
        <signal>clicked()</signal>
        <receiver>QtCmakeClass</receiver>
        <slot>TestCMake()</slot>
    <hints>
        <hint type="sourcelabel">
            <x>601</x>
            <y>340</y>
        </hint>
        <hint type="destinationlabel">
            <x>1241</x>
            <y>410</y>
        </hint>
    </hints>
    </connection>
</connections>
<slots>
    <slot>TestCMake()</slot>
</slots>
</ui>

```

qtcmake.qrc

```
<RCC>
  <qresource prefix="QtCmake">
    <file>cmake_cpp.jpg</file>
  </qresource>
</RCC>
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.20)
project(qt_cmake)

# cmake -S . -B build -DQT5=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64
# cmake -S . -B build -DQT5=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64 -
DCMAKE_INSTALL_PREFIX=out
# cmake -S . -B build -
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/
# J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe
if(NOT Qt5_DIR)
  set(Qt5_DIR ${QT5}/lib/cmake/Qt5)
endif()
set(windeployqt ${QT5}/bin/windeployqt.exe)
if(Qt5)
  set(QT5 ${Qt5})
endif()

message("Qt5_DIR = ${Qt5_DIR}")
file(GLOB SOURCE "*.cpp" "*.c")
file(GLOB HEAD "*.hpp" "*.h")
file(GLOB QT "*.uic" "*.qrc")
```

```

set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)
set(CMAKE_AUTOUIC ON)

add_executable(${PROJECT_NAME} WIN32 ${SOURCE} ${HEAD} ${QT})
find_package(Qt5 COMPONENTS Widgets REQUIRED)
target_link_libraries(${PROJECT_NAME}
Qt5::Widgets
)

add_custom_command(OUTPUT test_output
WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
DEPENDS ${PROJECT_NAME}
COMMAND ${windeployqt} $<TARGET_FILE:${PROJECT_NAME}>
COMMENT "test for add_custom_command")

add_custom_target(copy_qt
COMMAND ${windeployqt} $<TARGET_FILE:${PROJECT_NAME}>
)

message("CMAKE_INSTALL_PREFIX = ${CMAKE_INSTALL_PREFIX}")

install(TARGETS ${PROJECT_NAME}
CONFIGURATIONS Debug
RUNTIME DESTINATION Debug/bin)
install(CODE "message(hello)")
install(CODE "message(${CMAKE_INSTALL_PREFIX})")

install(CODE "execute_process(COMMAND
${windeployqt}

```

```

    ${CMAKE_INSTALL_PREFIX}/Debug/bin/${CMAKE_TARGET_FILE_NAME:${PROJECT_NAME}}
  )
  WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
)

```

4.1.3. moc 信号槽代码生成

```
set(CMAKE_AUTOMOC ON)
```

4.1.4. uic 界面代码生成

```
set(CMAKE_AUTOUIC ON)
```

```
file(GLOB QT "*.uic" "*.qrc")
```

4.1.5. rcc 资源代码生成

```
set(CMAKE_AUTORCC ON)
```

4.1.6. 加载qt库和头文件

```
cmake -S . -B build -
```

```
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/
```

```
find_package(Qt5 COMPONENTS Widgets REQUIRED)
```

```
target_link_libraries(${PROJECT_NAME}
```

```
Qt5::Widgets
```

```
)
```

4.2. QT区分32位和64位程序

4.2.1. cmake -S . -B build -

```
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017/lib/cmake/Qt5/ -A Win32
```

4.2.2. cmake -S . -B build -

DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/ -A x64

4.3. QT区分Debug和Release版本

4.3.1. cmake --build build --config Debug

4.3.2. cmake --build build --config Release

4.4. 自动复制QT依赖库

4.4.1. Win32

J:\Qt\Qt5.14.2\5.14.2\msvc2017\bin\windeployqt.exe
build/Release/qt_cmake.exe

4.4.2. x64

J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe
build/Debug/qt_cmake.exe

4.5. cmake install QT项目和依赖动态库

4.5.1. J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe

4.5.2. install(TARGETS \${PROJECT_NAME}

CONFIGURATIONS Debug Release

RUNTIME DESTINATION

\${CMAKE_INSTALL_PREFIX}/\${CMAKE_VS_PLATFORM_NAME}/bin)

install(CODE "execute_process(COMMAND

\${windeployqt}

\${CMAKE_INSTALL_PREFIX}/\${CMAKE_VS_PLATFORM_NAME}/bin/\${TARGET_FILE

NAME:\${PROJECT_NAME}>

```
WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
)")
```

5. QT CMake课程大纲

5.1. qt5 + cmake编程示例

5.1.1. 项目代码

qtcmake.h

```
#pragma once
```

```
#include <QtWidgets/QWidget>
```

```
#include "ui_qtcmake.h"
```

```
class QtCmake : public QWidget
```

```
{
```

```
    Q_OBJECT
```

```
public:
```

```
    QtCmake(QWidget *parent = nullptr);
```

```
    ~QtCmake();
```

```
public slots:
```

```
    void TestCMake();
```

```
private:
```

```
    Ui::QtCmakeClass ui;
```

```
};
```

qtcmake.cpp

```
#include "qtcmake.h"
```

```
#include <QMessageBox>
```



```

QtCmake::QtCmake(QWidget *parent)
    : QWidget(parent)
{
    ui.setupUi(this);
}

QtCmake::~QtCmake()
{}

void QtCmake::TestCMake()
{
    QMessageBox::information(this, "", "Test CMake");
}

```

main.cpp

```

#include "qtcmake.h"
#include <QtWidgets/QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QtCmake w;
    w.show();
    return a.exec();
}

```

qtcmake.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
<class>QtCmakeClass</class>
<widget class="QWidget" name="QtCmakeClass">
    <property name="geometry">

```

```

<rect>
  <x>0</x>
  <y>0</y>
  <width>875</width>
  <height>600</height>
</rect>
</property>
<property name="windowTitle">
  <string>QtCmake</string>
</property>
<property name="styleSheet">
  <string notr="true"/>
</property>
<widget class="QPushButton" name="pushButton">
  <property name="geometry">
    <rect>
      <x>210</x>
      <y>130</y>
      <width>481</width>
      <height>301</height>
    </rect>
  </property>
  <property name="styleSheet">
    <string notr="true">font: 22pt &quot;黑体&quot;;
background-image: url(:/QtCmake/cmake_cpp.jpg);</string>
  </property>
  <property name="text">
    <string>Test CMake</string>
  </property>
</widget>
</widget>

```

```

<layoutdefault spacing="6" margin="11"/>
<resources>
  <include location="qtcmake.qrc"/>
</resources>
<connections>
  <connection>
    <sender>pushButton</sender>
    <signal>clicked()</signal>
    <receiver>QtCmakeClass</receiver>
    <slot>TestCMake()</slot>
  <hints>
    <hint type="sourcelabel">
      <x>601</x>
      <y>340</y>
    </hint>
    <hint type="destinationlabel">
      <x>1241</x>
      <y>410</y>
    </hint>
  </hints>
</connection>
</connections>
<slots>
  <slot>TestCMake()</slot>
</slots>
</ui>

```

qtcmake.qrc

```

<RCC>
  <qresource prefix="QtCmake">
    <file>cmake_cpp.jpg</file>

```

</qresource>

</RCC>

CMakeLists.txt

cmake_minimum_required(VERSION 3.20)

project(qt_cmake)

cmake -S . -B build -DQT5=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64

cmake -S . -B build -DQT5=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64 -

DCMAKE_INSTALL_PREFIX=out

cmake -S . -B build -

DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/

J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe

if(NOT Qt5_DIR)

 set(Qt5_DIR \${QT5}/lib/cmake/Qt5)

endif()

set(windeployqt \${QT5}/bin/windeployqt.exe)

if(Qt5)

 set(QT5 \${Qt5})

endif()

message("Qt5_DIR = \${Qt5_DIR}")

file(GLOB SOURCE "*.cpp" "*.c")

file(GLOB HEAD "*.hpp" "*.h")

file(GLOB QT "*.uic" "*.qrc")

set(CMAKE_AUTOMOC ON)

set(CMAKE_AUTORCC ON)

set(CMAKE_AUTOUIC ON)

```

add_executable(${PROJECT_NAME} WIN32 ${SOURCE} ${HEAD} ${QT})
find_package(Qt5 COMPONENTS Widgets REQUIRED)
target_link_libraries(${PROJECT_NAME}
Qt5::Widgets
)

```

```

add_custom_command(OUTPUT test_output
WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
DEPENDS ${PROJECT_NAME}
COMMAND ${windeployqt} $<TARGET_FILE:${PROJECT_NAME}>
COMMENT "test for add_custom_command")

```

```

add_custom_target(copy_qt
COMMAND ${windeployqt} $<TARGET_FILE:${PROJECT_NAME}>
)

```

```

message("CMAKE_INSTALL_PREFIX = ${CMAKE_INSTALL_PREFIX}")

```

```

install(TARGETS ${PROJECT_NAME}
CONFIGURATIONS Debug
RUNTIME DESTINATION Debug/bin)
install(CODE "message(hello)")
install(CODE "message(${CMAKE_INSTALL_PREFIX})")

```

```

install(CODE "execute_process(COMMAND
${windeployqt}
${CMAKE_INSTALL_PREFIX}/Debug/bin/$<TARGET_FILE_NAME:${PROJECT_N
AME}>
WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
)")

```

5.1.2. moc 信号槽代码生成

```
set(CMAKE_AUTOMOC ON)
```

5.1.3. uic 界面代码生成

```
set(CMAKE_AUTOUIC ON)
```

```
file(GLOB QT "*.uic" "*.qrc")
```

5.1.4. rcc 资源代码生成

```
set(CMAKE_AUTORCC ON)
```

5.1.5. 加载qt库和头文件

```
cmake -S . -B build -
```

```
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/
```

```
find_package(Qt5 COMPONENTS Widgets REQUIRED)
```

```
target_link_libraries(${PROJECT_NAME}
```

```
Qt5::Widgets
```

```
)
```

5.2. QT区分32位和64位程序

5.2.1. cmake -S . -B build -

```
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017/lib/cmake/Qt5/ -A Win32
```

5.2.2. cmake -S . -B build -

```
DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/ -A x64
```

5.3. QT区分Debug和Release版本

5.3.1. cmake --build build --config Debug

5.3.2. cmake --build build --config Release

5.4. 自动复制QT依赖库

5.4.1. Win32

```
J:\Qt\Qt5.14.2\5.14.2\msvc2017\bin\windeployqt.exe  
build/Release/qt_cmake.exe
```

5.4.2. x64

```
J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe  
build/Debug/qt_cmake.exe
```

5.5. cmake install QT项目和依赖动态库

5.5.1. J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe

5.5.2. install(TARGETS \${PROJECT_NAME}

CONFIGURATIONS Debug Release

RUNTIME DESTINATION

\${CMAKE_INSTALL_PREFIX}/\${CMAKE_VS_PLATFORM_NAME}/bin)

```
install(CODE "execute_process(COMMAND
```

```
${windeployqt}
```

```
${CMAKE_INSTALL_PREFIX}/${CMAKE_VS_PLATFORM_NAME}/bin/${<TARGET_FILE_NAME:${PROJECT_NAME}>
```

```
WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
```

```
")
```

5.6. cmake打包qt程序

5.6.1. J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe

5.6.2. 打包成zip文件

5.7. 去掉控制台 WIN32

6. ffmpeg实训课更新

6.1.1 cmake qt环境准备

6.1.1.1. QT

6.1.1.2. VS2022

6.1.1.3. qt addone

旧项目升级

6.1.1.4. cmake

生成vs 后打开

6.1.1.5. 下载

<https://pan.baidu.com/s/12UfEFzR9ckAJdHOW6JjYJg?pwd=1234>

6.2.2 免费部分

6.2.1. 不讲环境 将一个最简单的示例

6.3.3 自动编译

6.3.1. 输出

6.3.2. release 和Debug

6.3.3. 去掉控制台 WIN32

6.4.4 自动安装软件

6.4.1. cmake install

6.5.5 自动编译ffmpeg

6.6.6 编译项目和发布

6.7. x86 x64项目区分

7. 微服务云盘实训更新

8. 编译Qt

8.1. cmake -S . -B build -

DQt5_DIR=J:/Qt/Qt5.14.2/5.14.2/msvc2017_64/lib/cmake/Qt5/

8.2. J:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin\windeployqt.exe

8.3. target_link_libraries(\${PROJECT_NAME}

Qt5::Widgets xcodec

)

8.4. find_package(Qt5 COMPONENTS Widgets REQUIRED)