

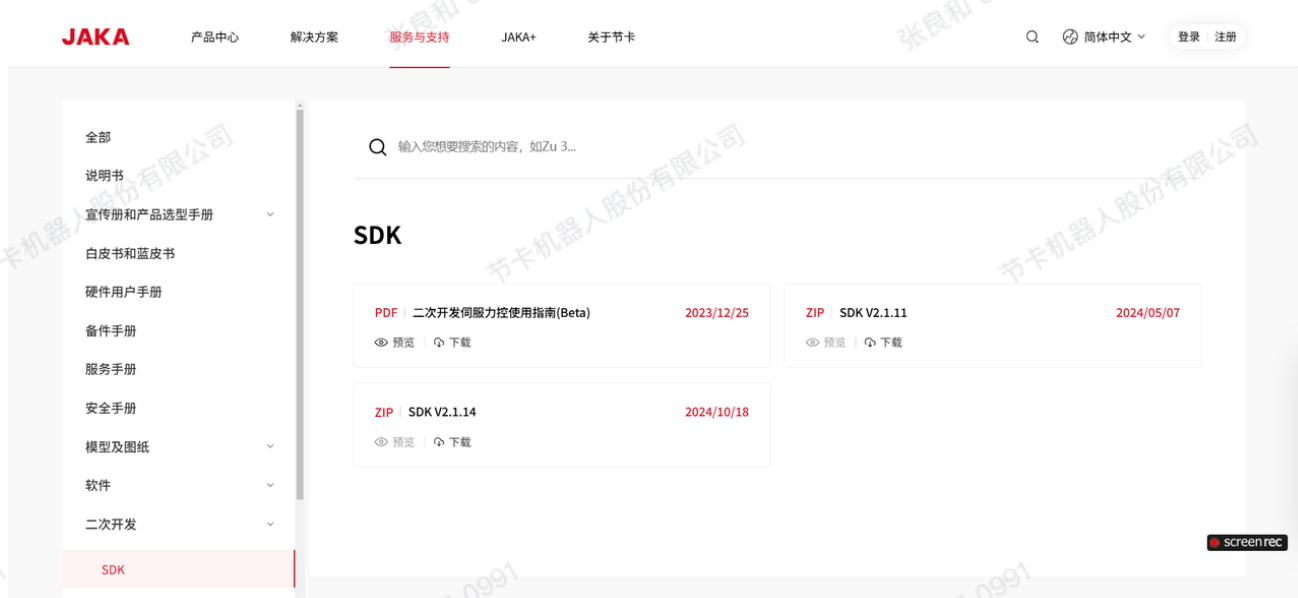
# 节卡 SDK 快速开始 - CN

## 1. 文档说明

JAKA SDK 是一个高效的工具包，旨在帮助开发者轻松与 JAKA 机器人进行交互。SDK 提供了一组接口，支持设备的连接、控制和数据传输等功能。通过本指南，您将能够在短时间内基于JAKA提供的SDK搭建自己的应用，并实现与 JAKA 机器人的基本交互。

## 2. SDK获取

JAKA SDK软件包可通过JAKA官网渠道下载获取。中文官方请通过JAKA官网首页（[资料中心 \(jaka.com\)](https://jaka.com)），点击[服务与支持]-> [资料中心] -> [二次开发] -> [SDK]，查看最新SDK以及历史版本信息。



## 3. 编写用户程序

为便于用户在不同的平台快速搭建自己的第一个应用，本文分别以三个示例说明在 Windows、Linux平台分别采用不同的编译工具创建应用的方法：

- 1) Windows上采用Microsoft Visual Studio创建C#应用；
- 2) Linux上采用CMake创建C++应用；
- 3) Linux上采用Qt Creator创建Qt应用。

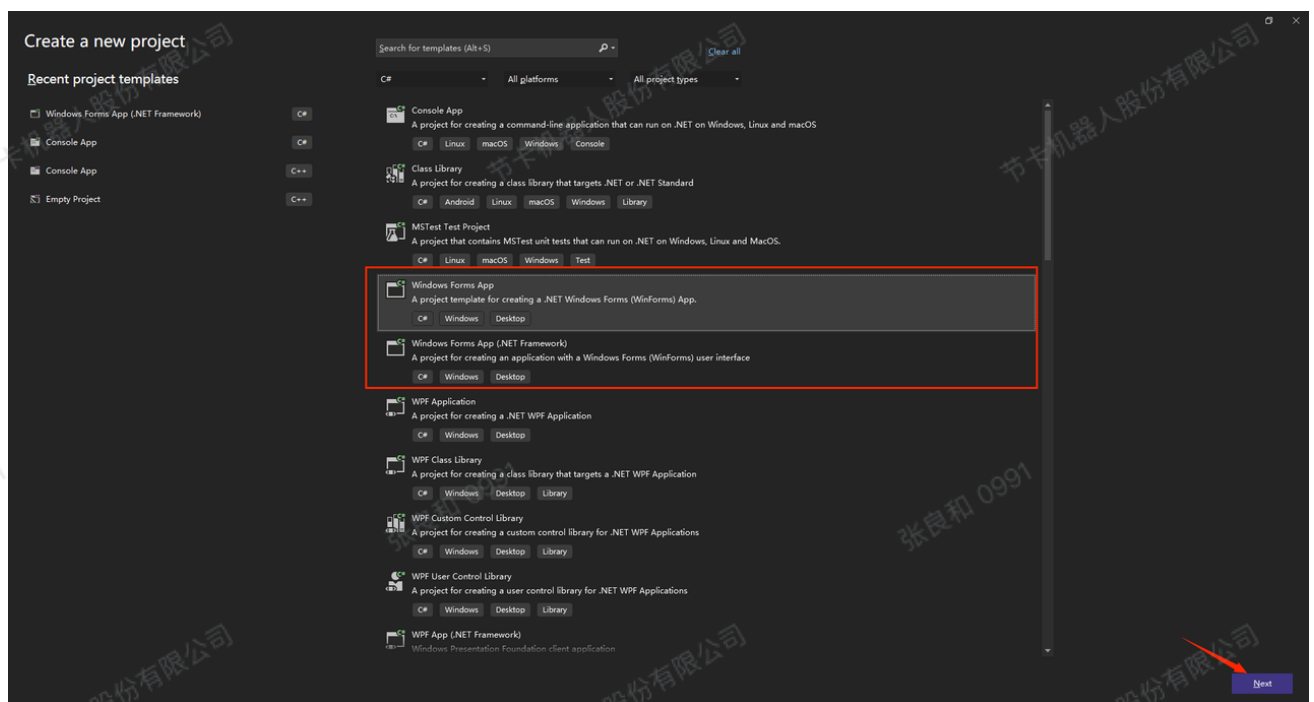
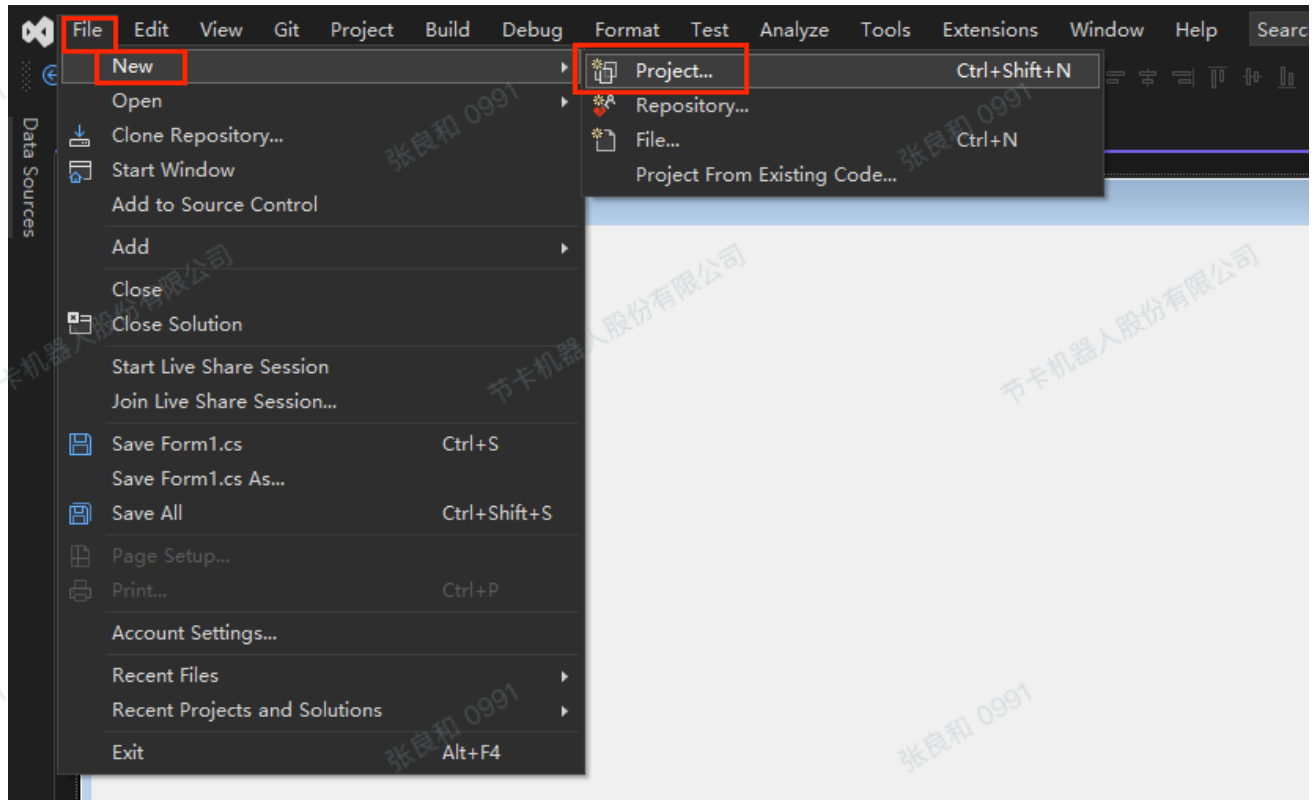
### 3.1 Windows上采用Visual Studio创建C#应用

### 3.1.1 编译环境安装

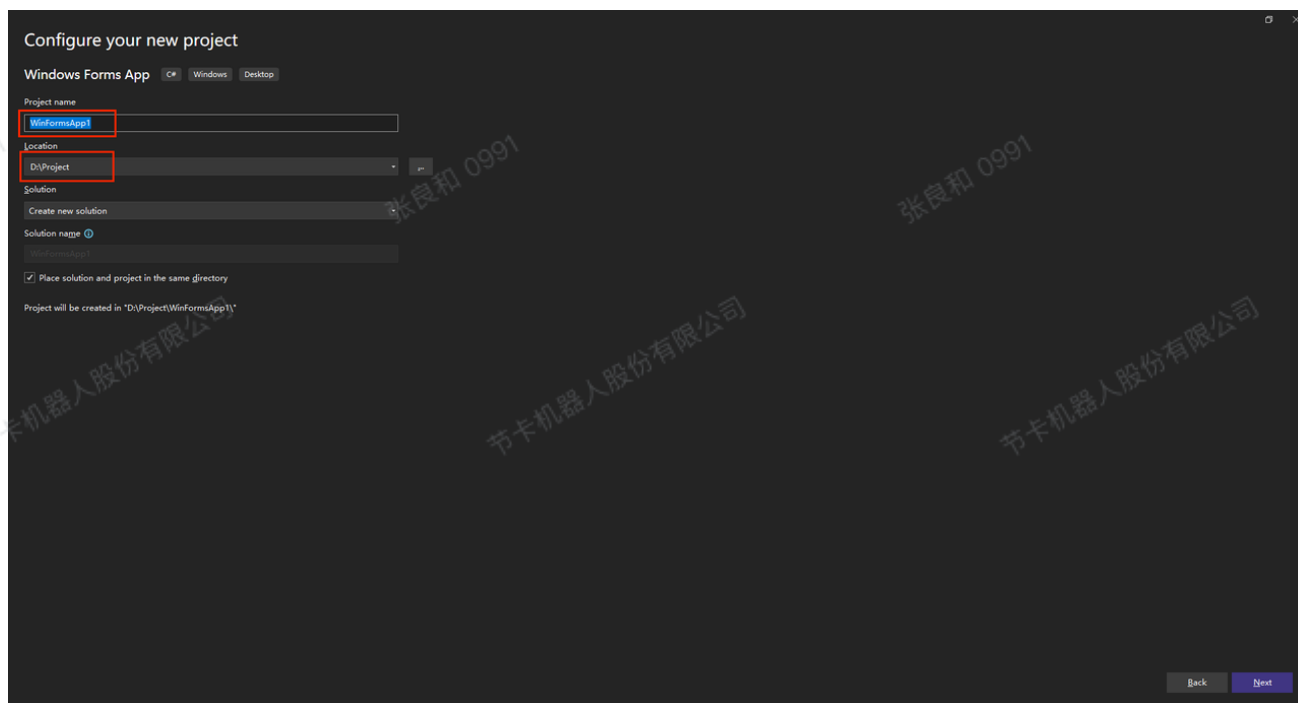
用户需下载安装Microsoft Visual Studio编程平台软件，请读者自行按照网上资源安装，以下介绍默认读者已经安装该软件。在部署C#开发环境的过程中，可能会要求更新.Net框架，请根据要求部署（此项目工程正常运行时 .NET框架为 .NET6.0 ）。

### 3.1.2 新建项目

打开软件，点击左上角的文件 ->新建 ->项目，如下图所示。本例选择创建窗体应用程序。



点击Next，如果需要跨平台应用程序，建议选择Windows Form App；如果仅仅运行在Windows上，可以选择Windows Forms App (.Net Framework)。进一步设置项目名称及存放位置后，点击Next直至完成项目创建。



### 3.1.3 编译链接环境配置

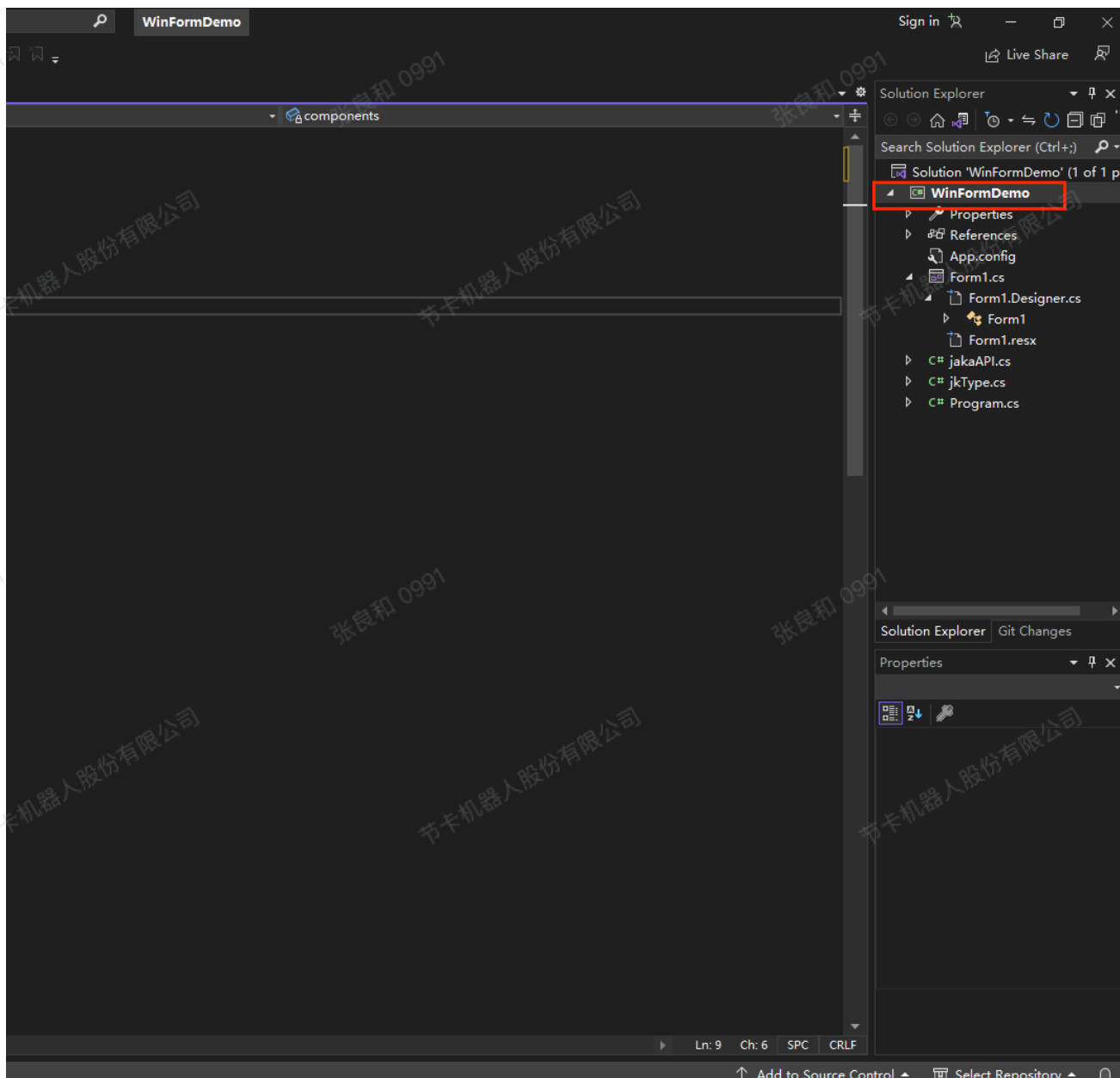
将从JAKA官网下载的SDK软件包解压后，可看到文件目录如下。

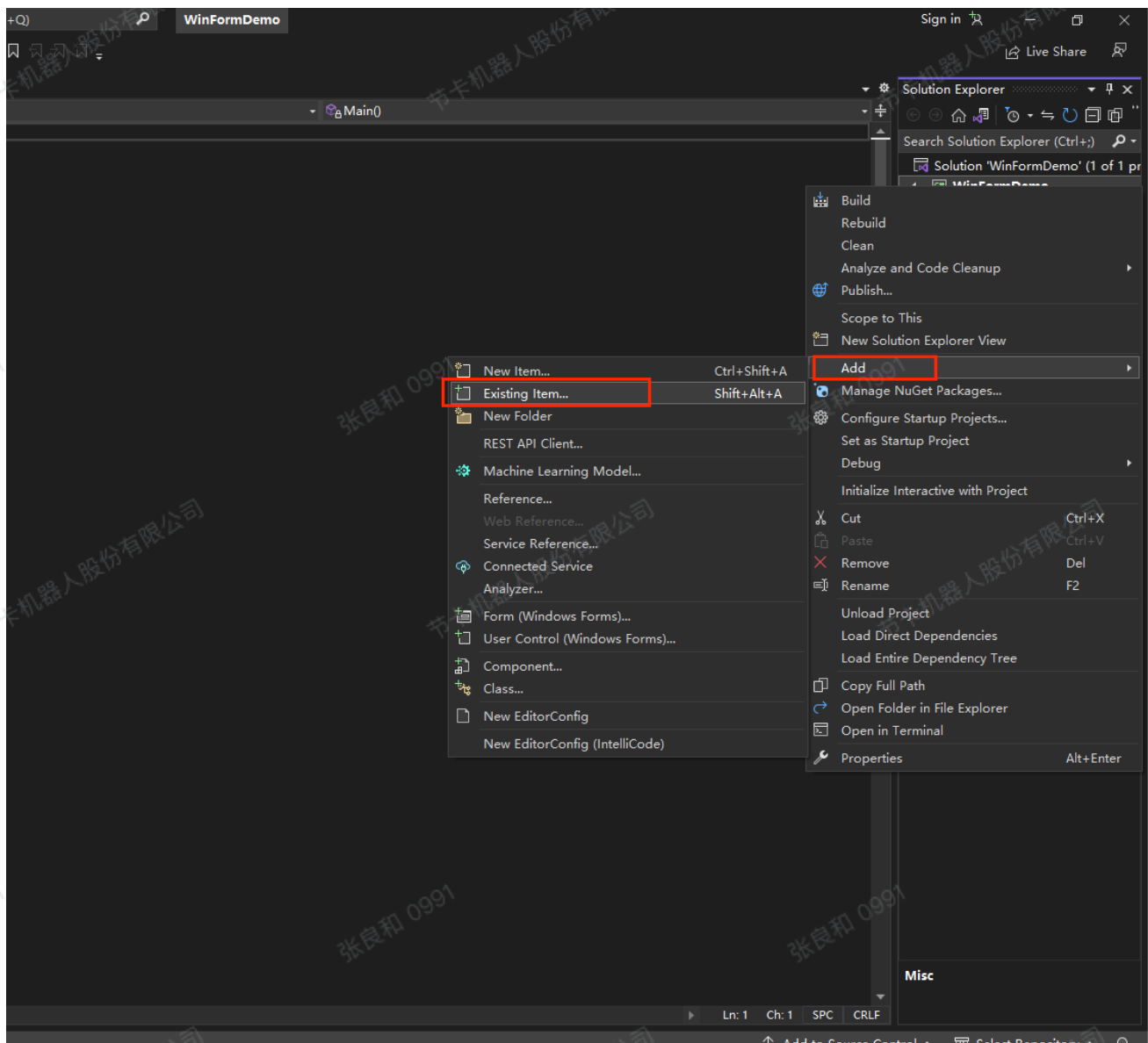
sdk-v2-1-11					在 sdk-v2-1-11 中搜索	
名称	修改日期	类型	大小			
doc	2024/05/06 17:58	文件夹				
SDK2.1.11	2024/05/06 10:32	文件夹				
changelog	2024/05/06 10:13	文件	4 KB			
commonly used Microsoft runtime li...	2024/06/20 17:24	应用程序	79,250 KB			
readme_EN.txt	2024/05/06 18:08	文本文档	3 KB			

对于Windows C#应用开发，可以找到Windows\csharp目录下的include（内含头文件）及x64（包含Windows 64位系统下的库文件）目录。

sdk-v2-1-11 > SDK2.1.11 > Windows > csharp					在 csharp 中搜索	
名称	修改日期	类型	大小			
include	2024/05/06 10:31	文件夹				
x64	2024/05/06 10:31	文件夹				

进入刚刚新建的项目中，右侧边栏的项目，点击添加--->现有项，将include文件夹中的两个头文件：jakaAPI.cs 和 jkType.cs添加进项目中。





将SDK包中 C#的动态库文件放到项目生成主程序目录下，一般在bin目录中。

名称	修改日期	类型	大小
WinFormDemo.exe	2024/11/6 16:17	应用程序	8 KB
WinFormDemo.exe.config	2024/11/6 14:23	CONFIG 文件	1 KB
WinFormDemo.pdb	2024/11/6 16:17	Program Debug...	30 KB
jakaAPI.dll	2024/10/12 10:43	应用程序扩展	1,142 KB
jakaAPI.exp	2024/10/12 10:43	Exports Library ...	61 KB
jakaAPI.lib	2024/10/12 10:43	Object File Library	101 KB

在用户程序中通过添加以下两行代码，以便在后续的代码使用JAKA SDK中定义的类型及方法。

```
using jkType;
using jakaApi;
```

```
Teach.cs [Design] Teach.cs x Teach.Designer.cs Program.cs
RobotTeach rshd
1 using System;
2 using System.Threading;
3 using System.Windows.Forms;
4 using jkType;
5 using jakaApi;
6 using System.Drawing;
7
8 namespace RobotTeach
9 {
10     3 references
11     public partial class Teach : Form
12     {
13         int rshd = -1;
14         bool login = false;
15         bool teach;
16         double r2d = 180/Math.PI;
17         double d2r = Math.PI/180;
18         JKTYPE.RobotStatus rs = new JKTYPE.RobotStatus();
19
20         #region log
21         public delegate void PrintMsgDelegate(string s);
22         public delegate void UiDelegate();
23         public delegate void UiupdateDelegate(JKTYPE.RobotStatus rs);
24         34 references
25         public void PrintMsg(string info)
26         {
27             textBox17.Text += string.Format("{0}\r\n", info);
28         }
29         //connect
30         1 reference
31         public void con()
32         {
33             button28.BackColor = Color.Green;
34             button28.Enabled = false;
35             button28.Text = "connect";
36             login = true;
37         }
38     }
39 }
```

### 3.1.4 编写应用程序

创建好程序之后，就可以根据自己的需求开发界面及对应的功能。完成代码编辑后，点击[生成]->[生成解决方案]进行编译生成可执行程序。最后点击工具栏启动按钮，可执行或调试程序。

示例给出了一个用于机器人示教的一个演示程序，功能包括：连接、上电上使能、关节空间运动控制、笛卡尔空间位姿控制等。具体实现客户可以参考示例代码。

## 3.2 Linux上采用CMake创建C++应用

### 3.2.1 下载安装Cmake工具

推荐到官网下载对应平台下的安装包[Cmake官网](<https://cmake.org/>)， 本示例使用的Cmake版本为3.7.2， 推荐使用此版本或以上版本。（具体安装操作请读着自行查阅网上资源， 下载并安装）以下说明默认读者已经完成此操作。

下载完成后，在终端输入 `cmake --version`，显示如下内容即代表安装成功。

复制代码

```
jakauser@ZuCAB2001:~$ cmake --version
cmake version 3.7.2

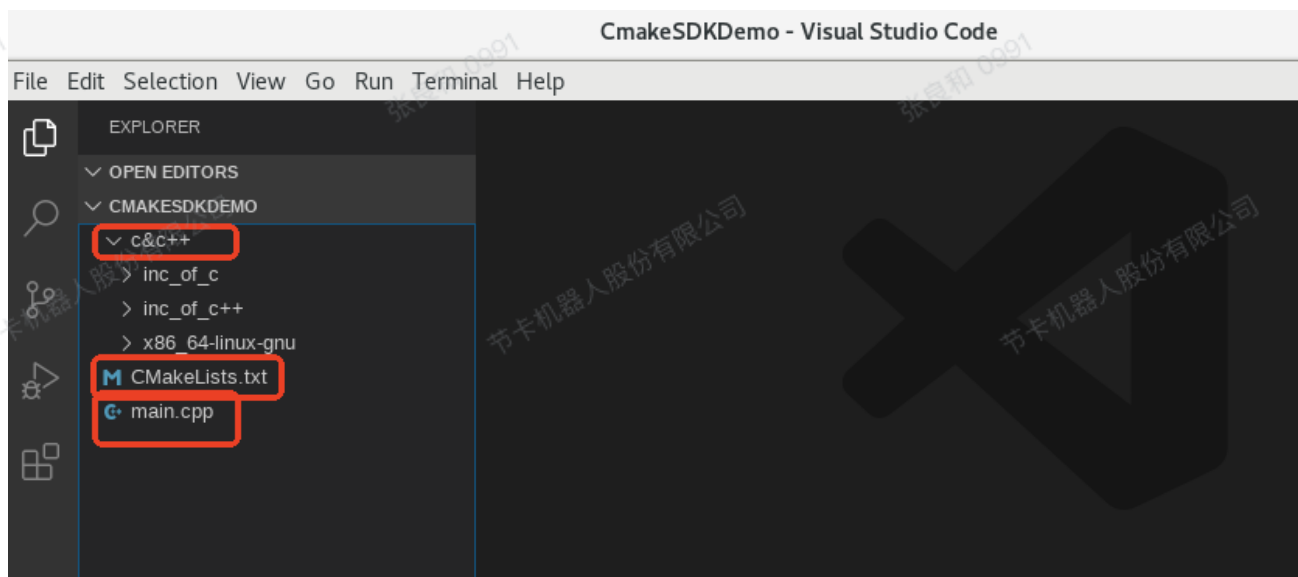
CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

### 3.2.2 下载安装Visual Studio Code

用户可前往Visual Studio Code官网（<https://code.visualstudio.com/Download>）查找软件安装包并下载安装。VS Code安装完成后，请在VS Code拓展中安装CMake拓展。

### 3.2.3 编写项目框架并配置CMake

在Linux下新建一个项目文件夹CmakeSDKDemo，创建以下文件，其中c&c++为节卡提供的SDK包里Linux下的资源文件，包括头文件inc及动态库文件。main.cpp为项目的主文件，CMakeLists.txt文件为Cmake配置文件，请注意大小写。



接下来用Cmake编写运行一个简单的hello world程序，首先在CMakeLists.txt下输入以下内容

复制代码

```
cmake_minimum_required(VERSION 3.5)
project(JAKADemo)

add_executable(hello main.cpp)
```

- `cmake_minimum_required` : 指定CMake的最低版本要求；
- `project` : 定义项目名称；
- `add_executable` : 添加一个可执行目标，第一个参数是目标名称，第二个参数是源文件列表。

注：此项目默认使用的是gcc编译器，如下，请读者自行配置好C++编译器。

```
jakauser@ZuCAB2001:~/Desktop/sdk_test$ gcc --version
gcc (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

编写main.cpp主程序，输入以下代码：

```
#include <string>
#include <iostream>

int main(){
    std::cout << "hello world" << std::endl;
}
```

复制代码

之后，我们进行CMake配置。在终端输入cmake。配置整个项目。之后会在项目文件下生成构建所需的配置文件。

```
jakauser@ZuCAB2001:~/Desktop/CmakeSDKDemo$ cmake .
-- The C compiler identification is GNU 6.3.0
-- The CXX compiler identification is GNU 6.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/jakauser/Desktop/CmakeSDKDemo
```

生成完成后，在终端输入make命令，构建编译可执行程序，会在当前目录下生成一个hello的可执行程序。

```
jakauser@ZuCAB2001:~/Desktop/CmakeSDKDemo$ make
Scanning dependencies of target hello
[ 50%] Building CXX object CMakeFiles/hello.dir/main.cpp.o
[100%] Linking CXX executable hello
[100%] Built target hello
```

在终端输入 ./hello 命令。执行程序，结果如下

```
jakauser@ZuCAB2001:~/Desktop/CmakeSDKDemo$ ./hello
hello world
```

至此，用Cmake运行一个简单的helloworld程序就大功告成。



### 3.2.4 链接JAKA SDK动态库

在CMakeLists.txt文件下输入以下内容，各行配置作用请见注释。

▼ 复制代码

```
cmake_minimum_required(VERSION 3.7.2)    # Minimum CMake Version
project(sdk_test VERSION 1.0)           # Project Definition

# C++ Standard
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED true)

# Source Directory Information
message(${CMAKE_SOURCE_DIR})

# Executable Creation
add_executable(demo main.cpp)

# Set the header file path of the SDK included in the project
target_include_directories(demo PUBLIC
${CMAKE_SOURCE_DIR}/c&c++/inc_of_c++)

# Set the SDK dynamic library link path
target_link_libraries(demo ${CMAKE_SOURCE_DIR}/c&c++/x86_64-linux-
gnu/shared/libjakaAPI.so pthread)
```

编写main.cpp文件，实现通过SDK的控制示例代码如下：此实例实现的功能是打印SDK版本信息、获取当前tcp位姿、将机器人末端沿y轴移动一定距离。

▼ main.cpp 复制代码

```
#include <string>
#include <vector>
#include <iostream>
#include <chrono>
#include "JAKAZuRobot.h" // 确保包含此行
#include <thread>
int main(int argc, char** argv)
{
    JAKAZuRobot demo;

    demo.login_in("192.168.164.222");
    //sleep(2);
    demo.power_on();
```

```

//sleep(2)
demo.enable_robot();
//sleep(2);

CartesianPose tcp_pos;
int ret;
char ver[100];
demo.get_tcp_position(&tcp_pos);
demo.get_sdk_version(ver);

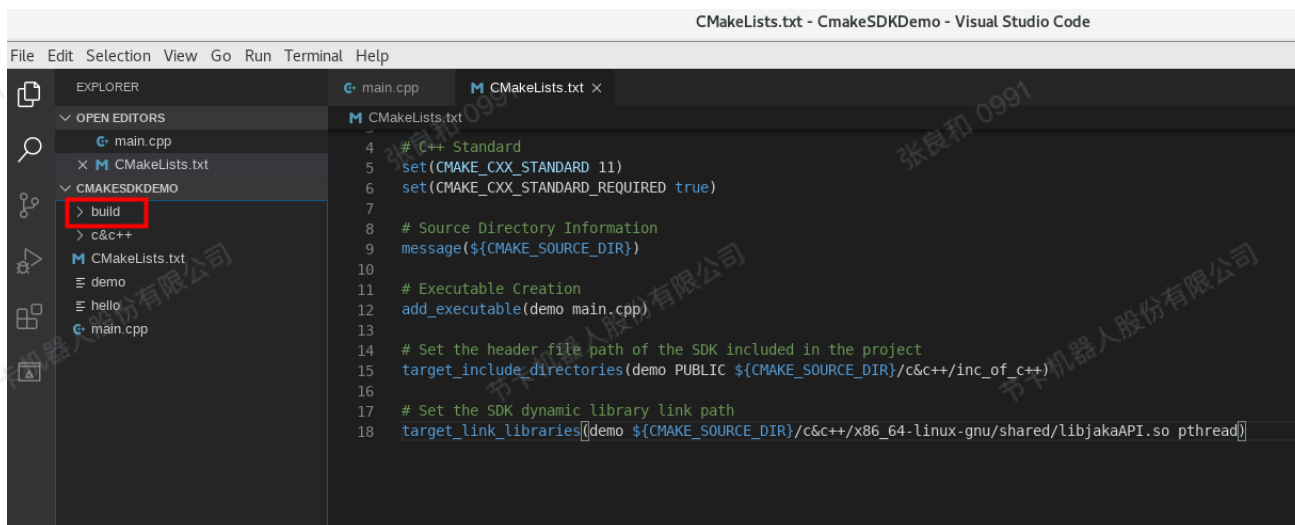
std::cout << "SDK version is :" << ver << std::endl;
std::cout << "tcp_pos is :\n x: " << tcp_pos.tran.x << "y: " <<
tcp_pos.tran.y << "z: " << tcp_pos.tran.z << std::endl;

std::cout << "tcp_pos is :\n rx: " << tcp_pos.rpy.rx << "ry: " <<
tcp_pos.rpy.ry << "rz: " << tcp_pos.rpy.rz << std::endl;
auto now = std::chrono::system_clock::now().time_since_epoch();
auto ms = std::chrono::duration_cast<std::chrono::milliseconds>
(now).count();
std::cout << "Current s: " << std::fixed << ((double)ms) / 1000.0f
<< std::endl;
tcp_pos.tran.y = tcp_pos.tran.y + 60.0;
ret = demo.linear_move(&tcp_pos, ABS, TRUE, 10, 10, 1, NULL);
std::cout << "ret==" << ret << std::endl;
std::cout << "linear_move finish! " << std::endl;
now = std::chrono::system_clock::now().time_since_epoch();

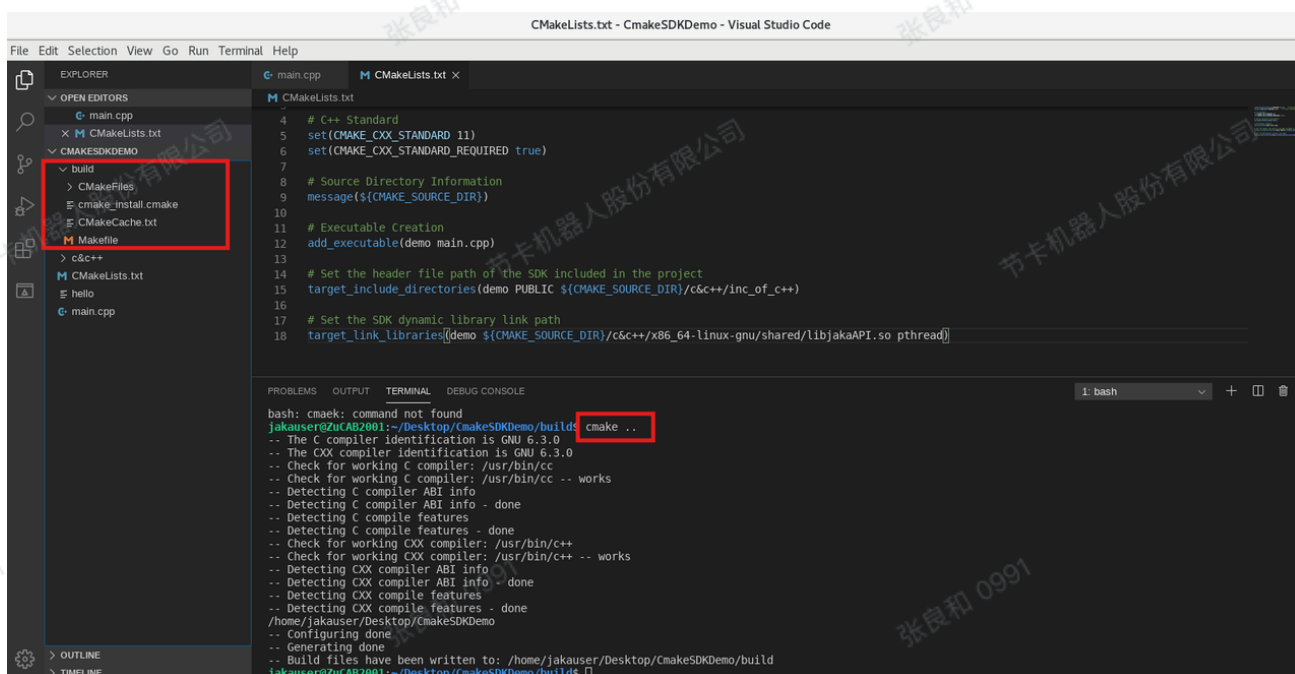
return 0;
}

```

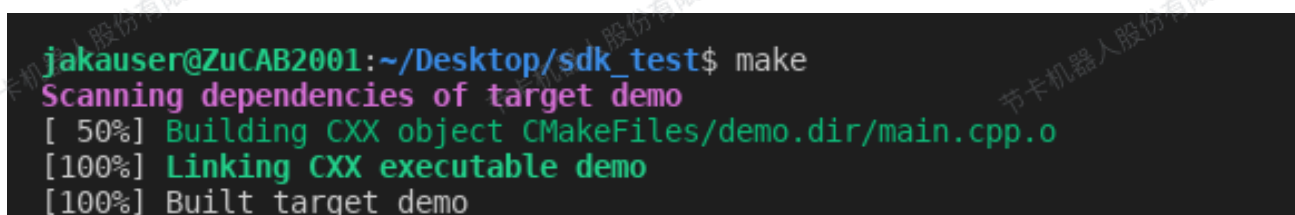
以上都准备好之后，通过cmake配置并构建整个项目。建议在项目根目录下新建build目录，用于存放构建生成的中间文件，避免导致文件内容复杂。



之后 cd build 进入到build目录，然后运行命令 `cmake ..`，构建整个项目，系统自动将生成的中间文件放到build目录下。（注意是`cmake ..`，代表CMakeLists.txt所在目录的上一级目录）。



之后在终端输入 `make` 构建编译可执行程序。



以上完成后，会在build目录下生成一个demo的可执行程序，读者可通过在终端运行 `./demo` 运行C++程序。

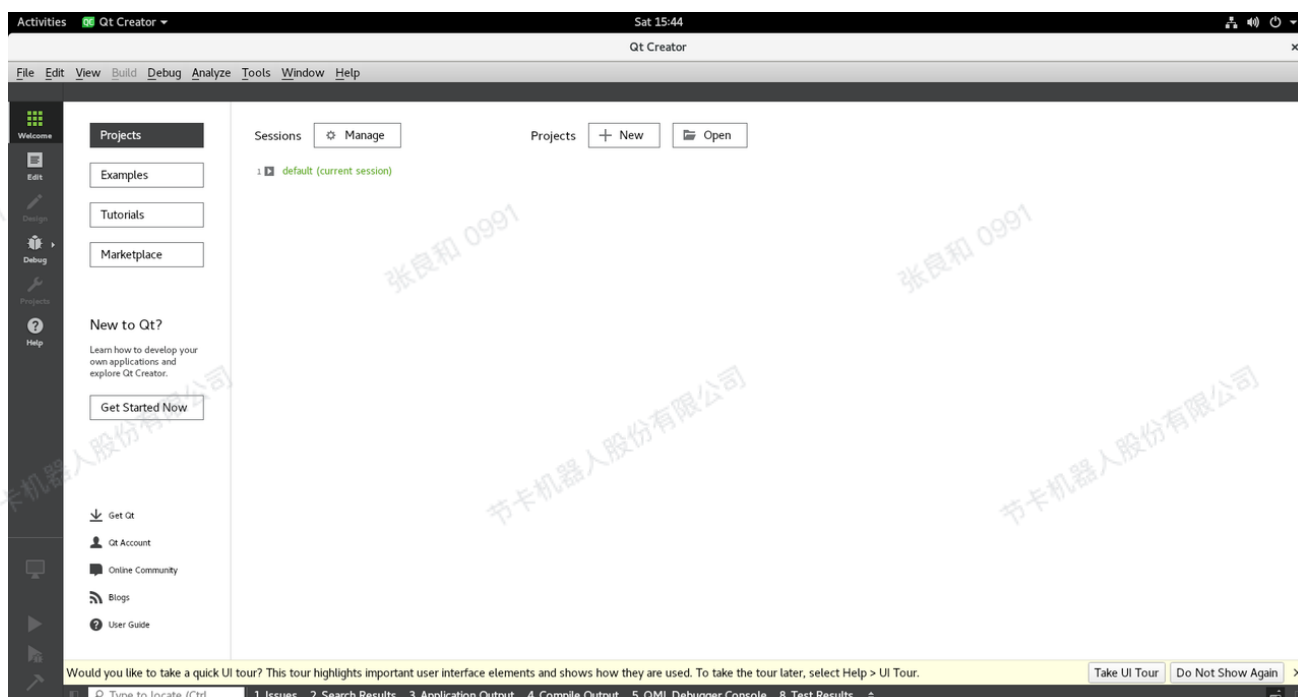
```
jakauser@ZuCAB2001:~/Desktop/sdk_test$ ./demo
try to connect: 172.30.3.172
connect success
SDK version is :libadd jakaAPI_version: V2.1.13stable_linux
tcp_pos is :
x: 697.529y: -294.055z: 1609.65
tcp_pos is :
rx: 3.07921ry: 0.143896rz: 2.59831
Current s: 1730968142.690000
ret==0
linear_move finish!
```

通过以上步骤，读者的C++程序便编写好了并可成功运行。

## 3.3 Linux上采用Qt Creator创建Qt应用

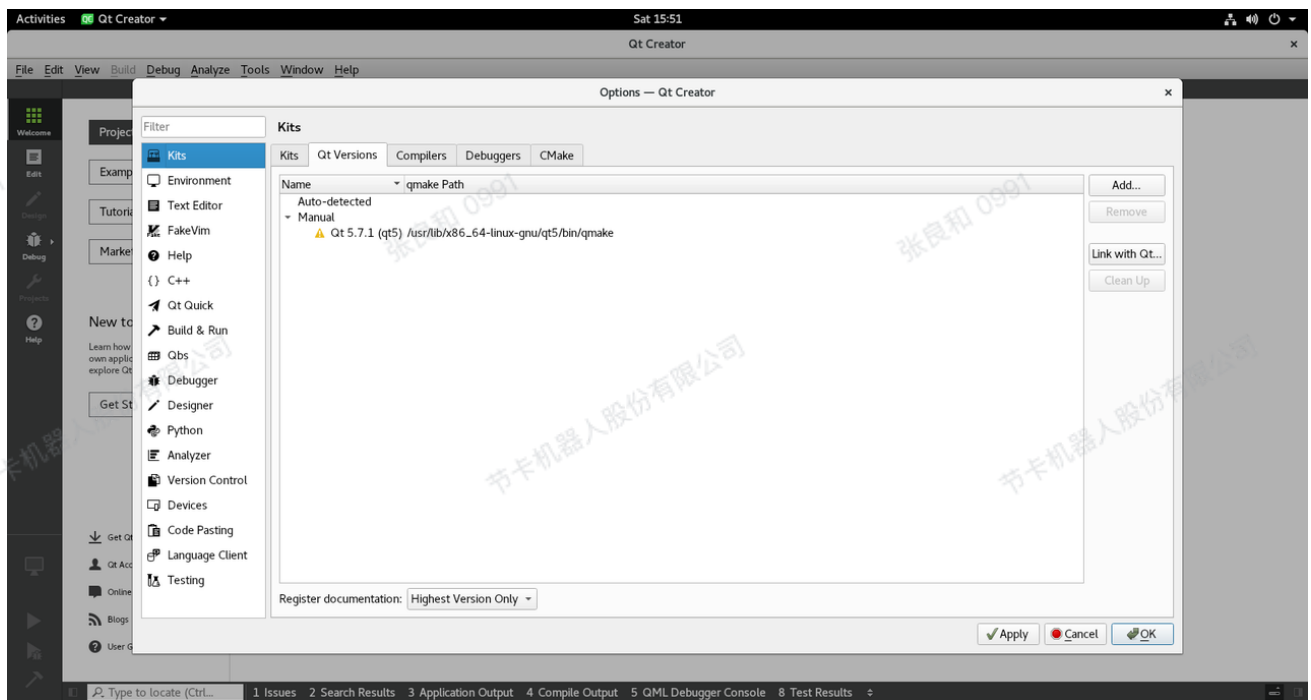
### 3.3.1 下载安装Qt Creator

用户可前往Qt Group官网（<https://www.qt.io/download-dev>）查找软件安装包并下载安装。Qt Creator安装完成后，启动Qt Creator，界面如下图所示。

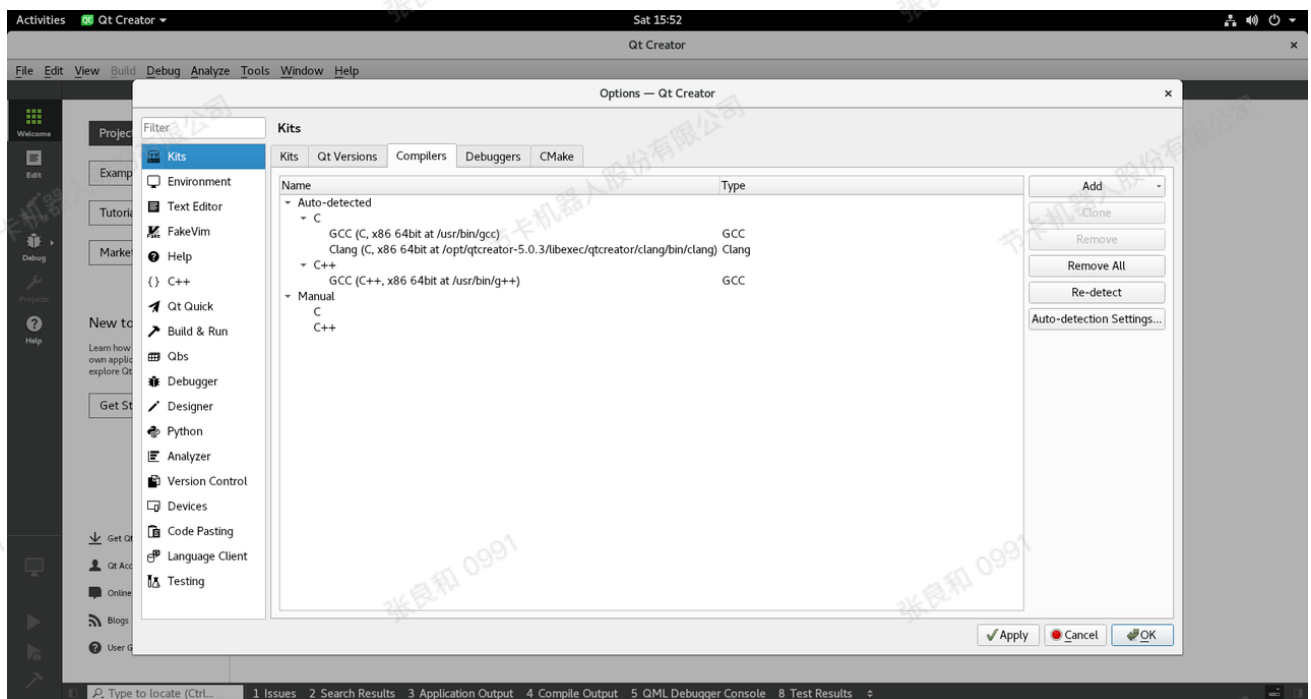


Qt安装后，需要进一步完成Qt的构建配置。打开 Qt Creator，进入 [Tools] -> [Options]，在弹出的对话框中选择 Kits 标签。

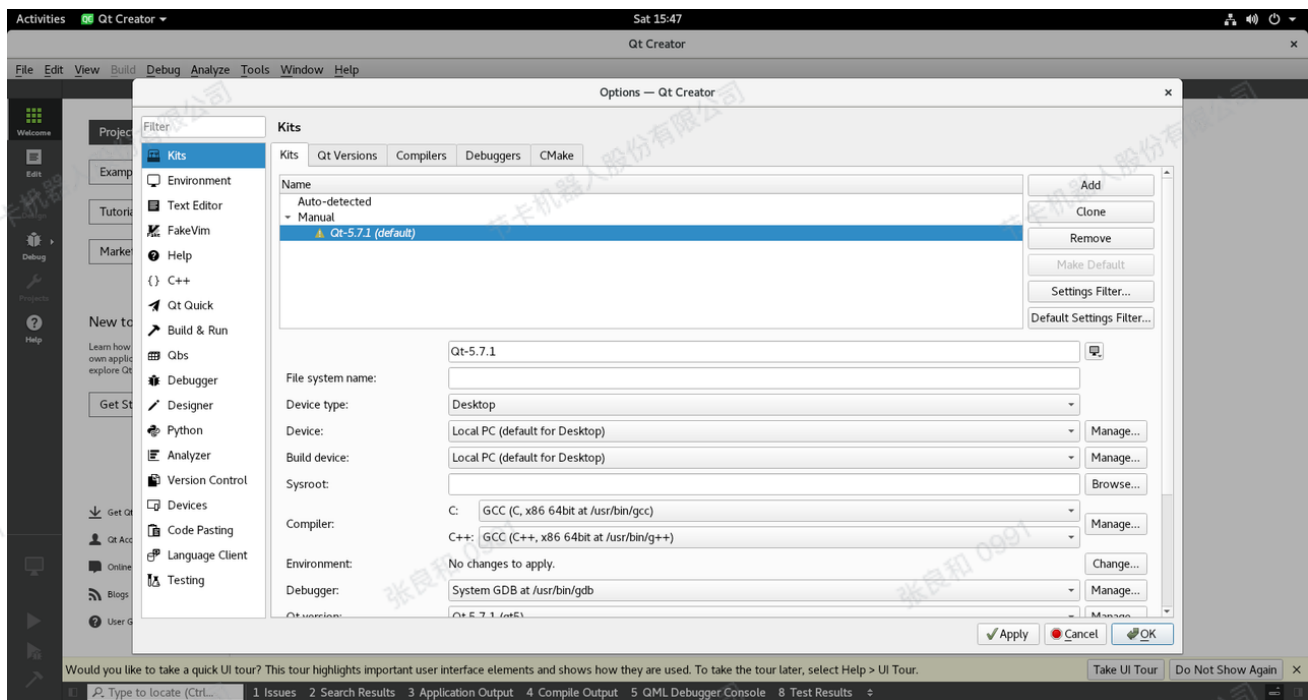
- 在 Qt Versions 部分，确保已列出你安装的 Qt 版本。如果没有显示任何版本，用户需要手动添加。



- 在Compilers部分确认编译器配置。通常，Qt Creator 会自动识别已安装的编译器，但如果  
没有，用户可以手动配置。

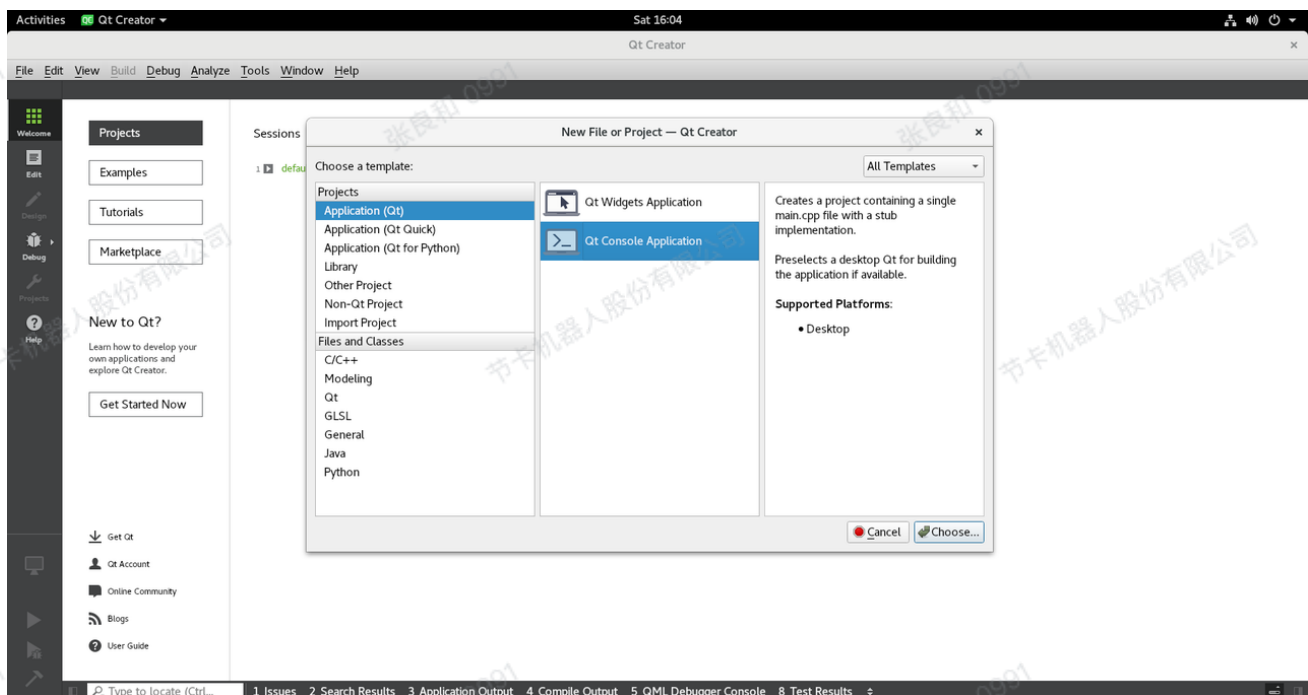


- 当 Qt 和编译器配置好之后，进入 Kits 标签中，点击左下角的 Add 按钮进行创建一个新的  
Kit 或者选择已有的 Kit。

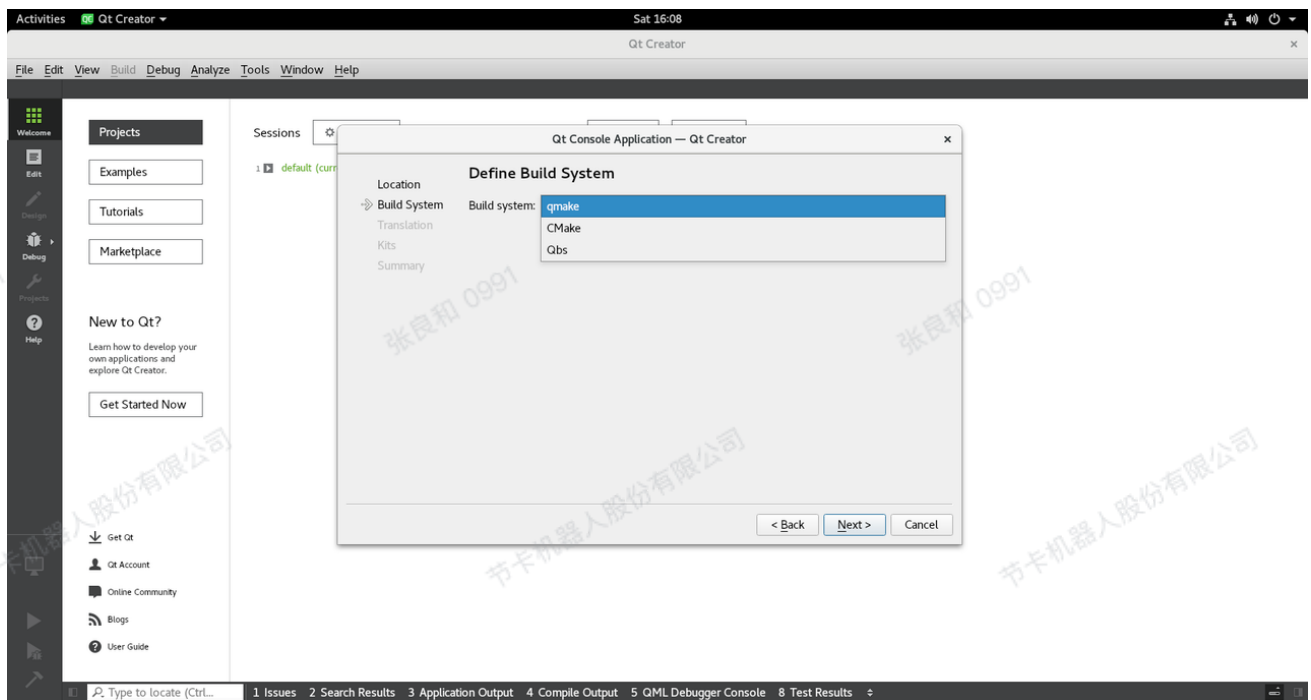


### 3.3.2 创建Qt应用程序

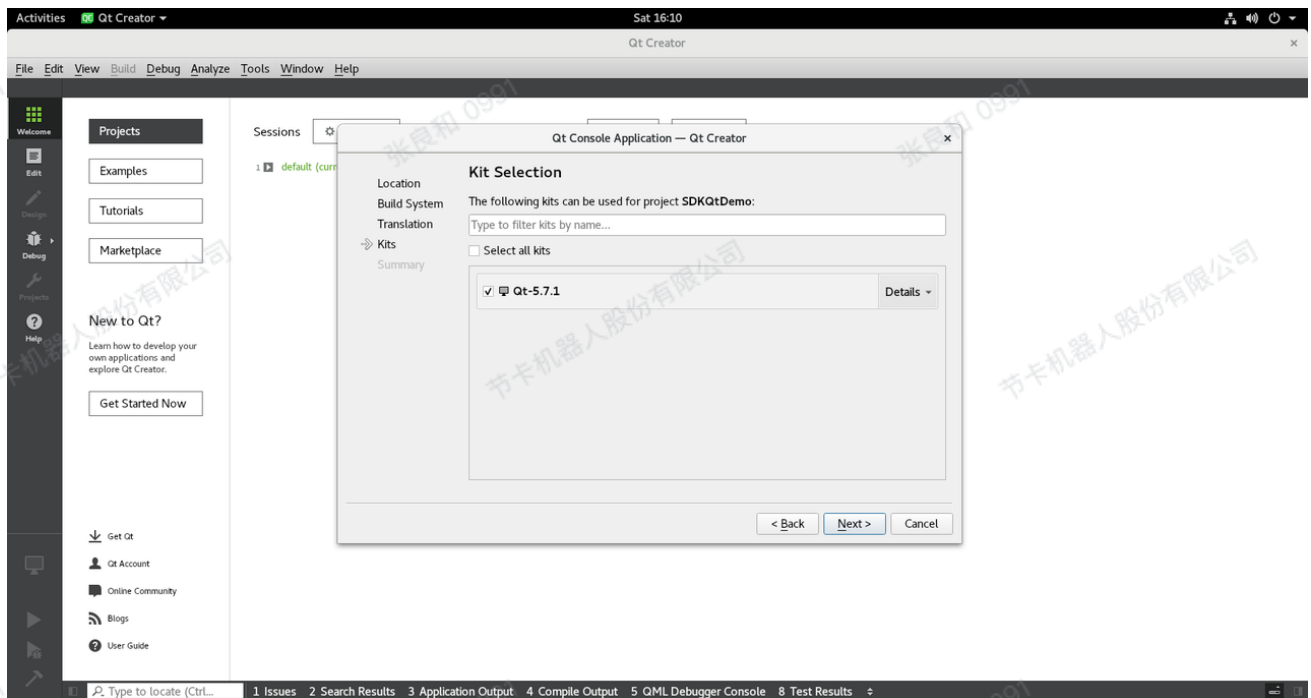
运行Qt Creator，菜单栏点击[File] -> [New File or Project ...]创建新的Qt应用程序。以下示例选择Application (Qt) -> Qt Console Application，即Qt控制台应用程序，然后根据引导进行配置操作。



配置项目名称以及存储目录后，需选择Build System（构建系统）。Qt支持多种构建系统，本例选择qmake。



在Kits选取界面，选择之前配置的Kit后，点击Next并最终完成项目创建。

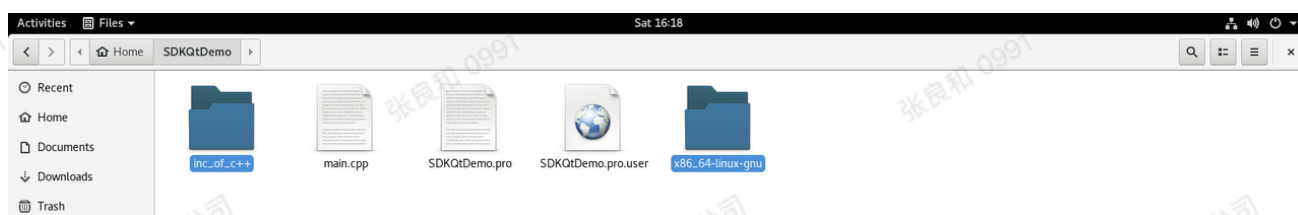


项目创建后，初步界面如下所示。



### 3.3.3 配置SDK链接

将下载的JAKA SDK中的头文件以及库文件添加到项目中。本例为Linux系统，因此将JAKA SDK文件夹中的两个文件夹inc\_of\_c++及x86\_64-linux-gnu拷贝至项目目录。



在Qt Creator中双击项目配置文件SDKQtDemo.pro，将使用JAKA SDK所需的头文件与库文件添加至项目中。本示例添加配置如下：

- 1) 添加配置行INCLUDEPATH += inc\_of\_c++，将头文件目录添加至项目；
- 2) 添加配置行LIBS += -L\$PWD/x86\_64-linux-gnu/shared -ljakaAPI，将libjakaAPI.so文件添加至项目。其中\$(PWD)/x86\_64-linux-gnu/shared指定libjakaAPI.so文件所在的目录，jakaAPI为库名称（即libjakaAPI.so）。





至此即完成Qt程序使用JAKA SDK所需的环境搭建及配置工作。

### 3.3.4 编辑程序调用SDK

可通过在程序中添加#include "JAKAZuRobot.h", 引入JAKA SDK头文件后, 即可调用相关接口。本示例编写以下代码, 供参考。注: 示例中所使用IP地址为机器人IP地址, 用户需根据情况进行调整。

▼ mian.cpp

复制代码

```
#include <QCoreApplication>
#include "JAKAZuRobot.h"
#include <string>
#include <vector>
#include <iostream>
#include <chrono>
#include <thread>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    JAKAZuRobot demo;

    demo.login_in("192.168.1.100");
    //sleep(2);
    demo.power_on();
    //sleep(2)
    demo.enable_robot();
    //sleep(2);

    CartesianPose tcp_pos;
    int ret;
    char ver[100];
    demo.get_tcp_position(&tcp_pos);
```

```

demo.get_sdk_version(ver);

std::cout << "SDK version is :" << ver << std::endl;
std::cout << "tcp_pos is :\n x:" << tcp_pos.tran.x << "y: " << tcp_pos.tran.y << "z: " <<
tcp_pos.tran.z << std::endl;

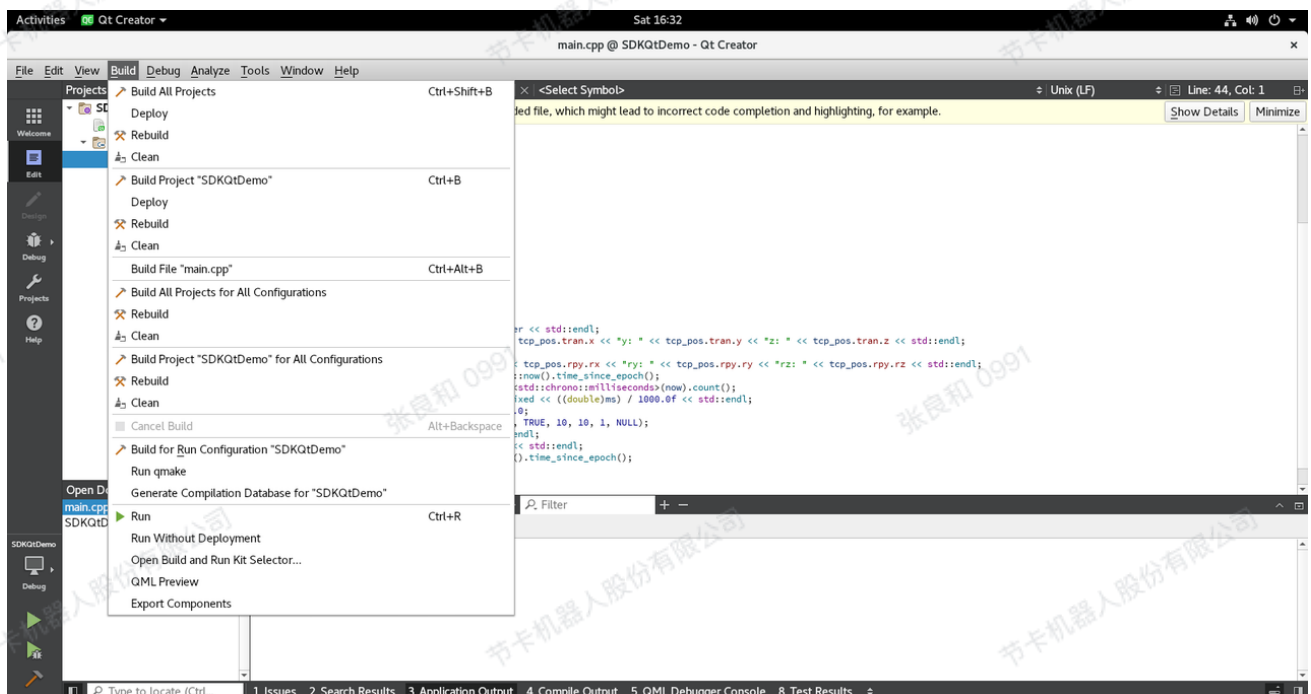
std::cout << "tcp_pos is :\n rx: " << tcp_pos.rpy.rx << "ry: " << tcp_pos.rpy.ry << "rz: "
<< tcp_pos.rpy.rz << std::endl;
auto now = std::chrono::system_clock::now().time_since_epoch();
auto ms = std::chrono::duration_cast<std::chrono::milliseconds>(now).count();
std::cout << "Current s: " << std::fixed << ((double)ms) / 1000.0f << std::endl;
tcp_pos.tran.y = tcp_pos.tran.y + 60.0;
ret = demo.linear_move(&tcp_pos, ABS, TRUE, 10, 10, 1, NULL);
std::cout << "ret==" << ret << std::endl;
std::cout << "linear_move finish! " << std::endl;
now = std::chrono::system_clock::now().time_since_epoch();

return a.exec();
}

```

### 3.3.5 编译并运行程序

程序编辑完成并保存以后，点击菜单[Build]->[Run qmake]，生成构建所需的配置文件。运行qmake结束后，会提示成功完成或异常（如有）。



然后点击[Build] -> [Build All Projects]编译源代码并链接生成可执行文件。Qt Creator底部Compile Output窗口会输出编译结果。



成功构建后，可点击Qt Creator左侧工具栏Run按钮，可启动程序运行。本例运行效果如下所示。



## 4. 以Addon部署在机器人控制器的说明

为便于部分用户采用SDK编程并部署在控制器内执行以替代复杂的图形化编程作业，本章节通过Addon的方式部署并运行使用节卡SDK编写的应用程序。由于目前节卡Addon功能是运行Python主文件且默认使用Python2.7版本，但目前节卡SDK是用Python3以上编译的，在Addon下直接运行Python的SDK库会出现问题。故此文档主要讲解的是如何通过Addon方式去调用客户C++语言编写的SDK应用程序，平台为Linux系统。

故总体思路可以分为两个步骤：

- 1) 基于C/C++开发可在JAKA控制器中运行的用户程序；
- 2) 开发一个Addon运行在JAKA控制器中并最终调用执行上述基于SDK的用户作业程序。

## 4.1 环境准备

整个过程用户需要准备两个开发环境分别完成上述两个步骤：

- 1) 基于SDK的用户程序开发环境；
- 2) JAKA Addon开发环境。

### 4.1.1 基于SDK的用户程序开发环境

该部分请参考3.2 Linux上采用CMake创建C++应用章节介绍过程进行准备。

### 4.1.2 Addon环境准备

首先准备一个JAKA Addon开发的demo（见下面链接，可获取下载），解压缩后有以下目录及文件，其中最主要的是红框标出来的两个文件，Addon\_Demo.py为Python主程序，是整个Addon运行时的入口程序。Addon\_Demo\_config.ini为Addon配置文件，里面是关于此Addon的一些配置项。

关于Addon的具体介绍详见节卡文档中心相关网页：

<https://www.jaka.com/docs/guide/addOn/1.2-AboutDev.html>。

名称	修改日期
client	2022/4/20 20:55
AddOn_Demo.py	2022/5/12 17:41
AddOn_Demo_config.ini	2022/5/9 14:45
readme.txt	2022/5/9 16:20
server_config.json	2022/5/9 16:47
server_config.py	2022/5/12 16:57

## 4.2 开发及部署

以下演示的是如何在Linux下部署节卡SDK C++的开发环境，此开发部署需要用到CMake工具，请读者自行下载并安装、配置，以下讲解默认读者已经配置完成并且有一定的开发基础。

### 4.2.1 编写用户程序

用户作业程序的编写可参考 3.2 Linux上采用CMake创建C++应用章节中描述的过程，创建可在机器人控制器环境下运行可执行程序。在本例中，假设用户开发的作业程序可执行文件为

demo。

## 4.2.2 Python运行C++程序

此时，读者需要在Addon的python主程序下，编写Python代码调用C++的可执行程序。以下是两种可供用户实现调用的方式（注意：控制器系统环境中默认的Python运行版本为Python 2.7）。

### 1) 通过subprocess 子进程模块运行

```
import subprocess
import os
if __name__ == '__main__':
    # way 1
    args = ['./build/demo']
    result = subprocess.call(args) # python2
```

### 2) 通过os模块运行

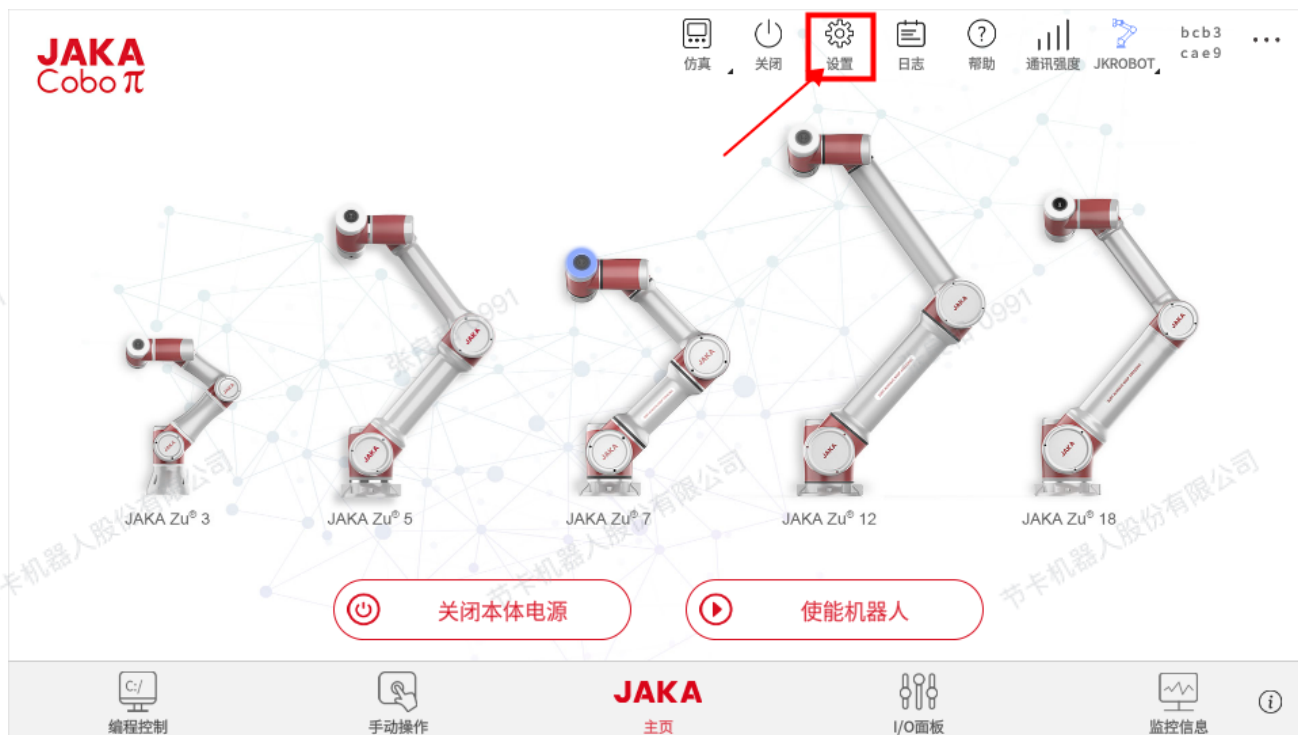
```
import subprocess
import os
if __name__ == '__main__':
    # way 2
    args = ['./build/demo']
    os.system('./build/demo')
```

以上，就上通过JAKA的Addon框架运行SDK程序的示例。其中./build/demo为用户开发的作业程序可执行文件在所开发Addon下的路径。

## 4.3 以Addon方式运行用户程序

用户编写完Addon后需要将项目打包成tar.gz格式，形成Addon软件包。然后打开JAKA App 软件，用户可通过[设置]-->[系统设置]-->[附件程序]，进入附加程序管理界面，上传该Addon包后即可完成安装。安装完成后，用户可以在该界面控制Addon启动或终止。

如下图所示：



选择本地目录中打包好的Addon包，然后等待文件上传成功。



通过状态列下的切换按钮来开启或关闭此Addon



## 4.4 注意事项

- 1) 基于SDK开发用户应用程序时, 需注意当前控制柜的系统架构。JAKA控制器基于Debian Linux可能存在两种配置, x86及x86\_64, 具体可通过JAKA App连接控制器查看控制器版本后缀获得;
- 2) 用户编写程序运行在控制器, 因此需要合理规划资源占用, 包括CPU、内存以及磁盘空间等使用, 避免对机器人控制系统造成影响。