

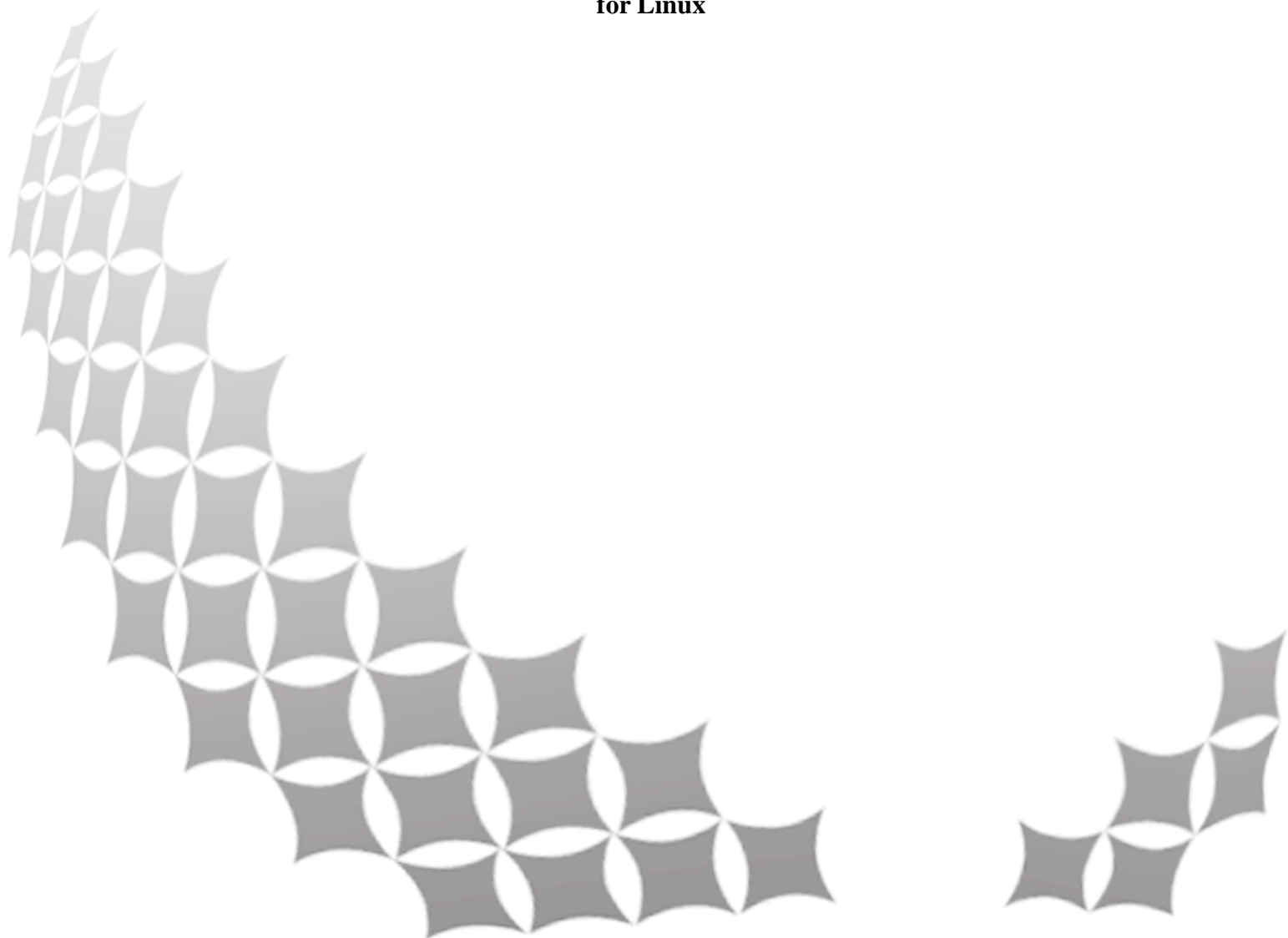
应用集成公共支撑软件

信息集成套件

开发者指南

Developer Manual

for Linux



目 录

第一章.....	3
产品简介.....	3
▶ 应用集成公共支撑软件说明文件.....	3
▶ 平台惯例.....	3
第二章.....	4
理解信息集成套件.....	4
▶ 什么是 DDS.....	4
▶ 什么是信息集成套件.....	4
▶ 信息集成套件特性.....	4
▶ 兼容性.....	6
▶ 开发环境.....	6
▶ 数据交互中间件.....	6
▶ 辅助开发工具.....	6
▶ IDL 至 C++ 的映射.....	7
第三章.....	8
开发实例应用.....	8
▶ 开发过程.....	8
▶ 步骤一：定义数据类型.....	8
▶ 步骤二：生成辅助文件.....	9
▶ 步骤四：应用软件环境配置.....	10
▶ 步骤五：应用组件开发.....	13
▶ 示例程序.....	13
▶ 发布端示例程序.....	13
▶ 订阅端示例程序.....	15

第一章

产品简介

应用集成公共支撑软件是 XX3B 型 QTZZ 系统通过统一规划和设计，提取并整合应用集成所需的共性支撑功能，为保证系统应用功能软件在异构平台上实现灵活集成而构建的统一的集成框架和运行环境。

应用集成公共支撑软件(信息集成套件)为实现装备软件化、软件构件化、软硬件解耦合提供底层技术支撑，使系统集成从传统的硬集成转变为以数据为中心的软集成，达到信息按需获取、应用功能即插即用、系统灵活重构的目的。

► 应用集成公共支撑软件说明文件

应用集成公共支撑软件的说明文件包括以下内容：

- ✓ **安装指南：**介绍如何在您的系统上安装应用集成公共支撑软件。
- ✓ **使用指南：**介绍有关使用应用集成公共支撑软件的信息。
- ✓ **开发者指南：**介绍如何使用 Visual Studio/Tornado/Workbench/ Eclipse，基于应用集成公共支撑软件开发相关应用程序。
- ✓ **程序员参考：**介绍应用集成公共支撑软件所提供的接口、配置文件的具体定义和使用方式。

► 平台惯例

应用集成公共支撑软件说明文件使用下列符号表示特定系统信息：

Windows：仅指 Windows XP 和 Windows 7

VxWorks：仅指 VxWorks5.5 和 VxWorks6.8 实时操作系统

Linux：仅指 NeoKylin Desktop V6.1 x86 和 Ubuntu 14.04 x86

第二章

理解信息集成套件

► 什么是 DDS

数据分发服务（Data Distribution Service）是 OMG 的有关分布式实时系统中数据发布的一个规范。该规范标准化了分布式实时系统中数据发布、传递和接受的接口和行为，定义了以数据为中心的发布/订阅（Data-Centric Publish-Subscribe）机制，提供了一个与平台无关的数据模型。

► 什么是信息集成套件

信息集成套件基于 OMG 组织制订的 DDS1.2 规范，使用订阅/发布机制进行应用功能软件间的信息传输通道组织和信息分发，使应用无需关心信息的来源和去向等信息传输细节，以统一的信息交互接口实现按需信息分发、按需信息获取，根据不同的系统信息流设计模式和要求，提供质量保证策略接口，保证高质量的实时信息传输。

以实时数据分发服务为核心，信息集成套件还提供了系统状态监视、数据记录、辅助开发、组件检测等服务，为系统提供实时、高效、灵活的数据交互手段，并在应用系统设计、开发、调试、运行的各个环节提供完整的支撑服务。

► 信息集成套件特性

信息集成套件具有以下部分中描述的一些关键特性。

● 完整的开发环境

信息集成套件为应用组件的开发者提供了完整的开发环境。用户在应用程序开发阶段可使用 IDL 语言编译器及辅助开发工具实现中间件相关代码的便捷嵌

入；在系统集成阶段可使用应用程序检测工具对程序进行预检测，以降低集成测试的故障发生概率和定位成本；系统实际运行阶段可使用系统监控、数据记录工具来获取系统的运行状态，并能够对数据进行离线分析。使用信息集成套件提供的完整开发环境，可以极大程度地降低应用组件设计、实现、测试和故障定位的复杂性。

● 高性能通信服务

信息集成套件通过高效运行的核心代码和灵活多样的 QoS 策略，保证应用组件在苛刻的应用场景下能够实现良好的数据交互。

● 应用数据过滤

信息集成套件提供多种应用数据过滤方式。应用组件在 ZZ 系统中进行一对多的数据传输时，不同的订阅者对同一主题的数据内容、数据频率需求不尽相同。针对这一应用需求，信息集成套件可自动过滤掉订阅者不关心的数据，以减少应用程序的数据处理工作。

● 可靠性

信息集成套件为 ZZ 系统中的应用组件提供 P2P 的信息交互服务，在系统中不存在集中式的代理或服务器进程，从而保证整个系统服务不存在单点故障的风险。

● 独立升级和移植

信息集成套件提供多个操作系统的版本，均采用动态库的形式提供应用程序使用，并为不同操作系统下的应用程序提供统一的服务调用接口。应用组件和信息集成套件可分别进行独立升级。

● 扩展性

信息集成套件使用“订阅/发布”机制进行数据交互，建立全局的虚拟数据空间，在通信层面将应用逻辑与节点的物理信息解耦合。应用组件可根据需要在系统中的任意节点上部署或迁移，信息集成套件均能够自动完成发现过程，保证应用组件实现原有的通信功能。

● 丰富的 QoS 策略

信息集成套件提供丰富的 QoS 策略。QoS 策略能够独立或组合使用，以满足用户不同的应用场景下对通信质量灵活而复杂的需求。

► 兼容性

信息集成套件符合对象管理组织(OMG)的 DDS 规范(1.2 版)。有关详情，请参阅位于 <http://www.omg.org/> 中的 DDS 规范。

► 开发环境

信息集成套件由一套逻辑相互关联、功能各有侧重的中间件及工具软件组成。其中，与用户开发相关的工具主要包括：

- 数据交互中间件；
- 辅助开发工具。

► 数据交互中间件

数据交互中间件以动态库的形式提供应用软件调用，为分布式实时系统提供实时的数据分发服务。在开发过程中主要完成以下工作：

- 使用工具包中的 IDL 文件编译器 (idlcc.exe)，将 IDL 文件作为输入，并在 C++ 中产生必需的用户代码，方便用户进行应用开发；
- 使用文本编辑器对工具包中提供的配置文件模板进行编辑，便于对中间件的工作方式进行预设。

► 辅助开发工具

辅助开发工具以开发环境插件的形式，辅助用户基于信息集成套件进行应用组件开发，包括：

- 为用户提供界面式的应用程序配置界面，用户可设计应用程序所需的主题数据结构及发布、订阅关系；
- 根据用户配置，自动生成接口代码并添加到应用程序工程中；

- 为应用程序自动设置开发环境所需的配置参数；
- 自动生成配试程序，实现对应用程序通信功能的及时测试。

► IDL 至 C++的映射

信息集成套件符合 OMG IDL/C++语言映射规范。有关通过 idlc 编译程序实现的当前 IDL 至 C++语言的映射的汇总，请参阅程序员指南。对于每个 IDL 构造，都存在描述相应的 C++构造以及代码示例的小节。

有关映射规范的详情，请参阅 OMG IDL/C++语言映射规范。

第三章

开发实例应用

本节使用实例来描述如何基于信息集成套件进行应用软件的开发过程。该实例应用的 C++ 代码可以在信息集成套件安装完成后所在位置的 `demos` 目录下找到。

► 开发过程

当您基于信息集成套件开发分布式应用时，您必须首先确定应用组件需要进行发布/订阅的数据类型。以下是开发该实例所采取的步骤的概要描述：

- 1 使用接口定义语言（IDL）为每种数据类型编写说明；
- 2 使用 IDL 编译器生成用户代码；
- 3 将生成的代码添加到应用软件的开发工程中；
- 4 设置必须的应用软件开发工程属性；
- 5 为应用软件编辑对应的配置文件；
- 6 编写应用软件功能代码。

► 步骤一：定义数据类型

创建应用组件的第一步是使用 OMG 的 接口定义语言（IDL）说明您需要发布/订阅的数据类型，然后可以使用 (C++) `idlC` 编译器生成辅助开发代码。例如需要创建的 IDL 文件名为 `UserDataType.idl`，内容见图 1：

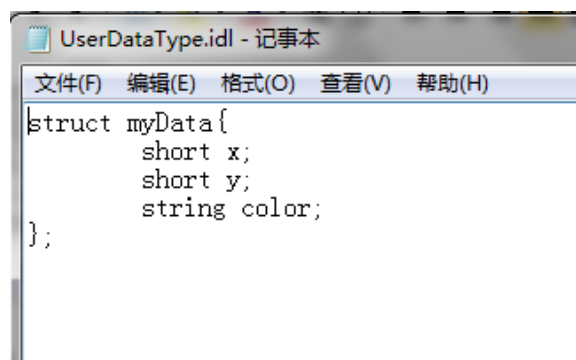


图 1 用户自定义数据类型示例

► 步骤二：生成辅助文件

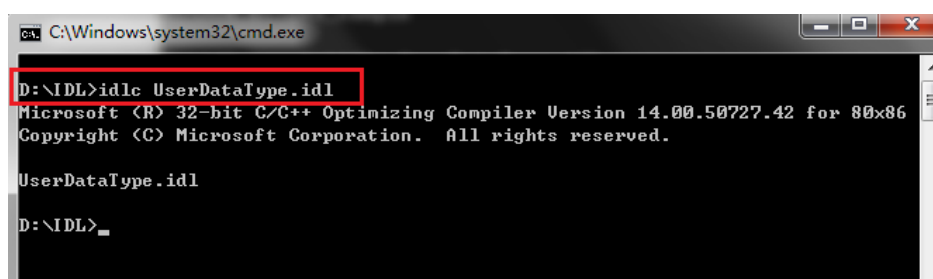


图 2 使用 idlc.exe 编译用户自定义数据类型

C++: idl 编译器可以从 UserDataTypeIdl 文件产生十个文件：

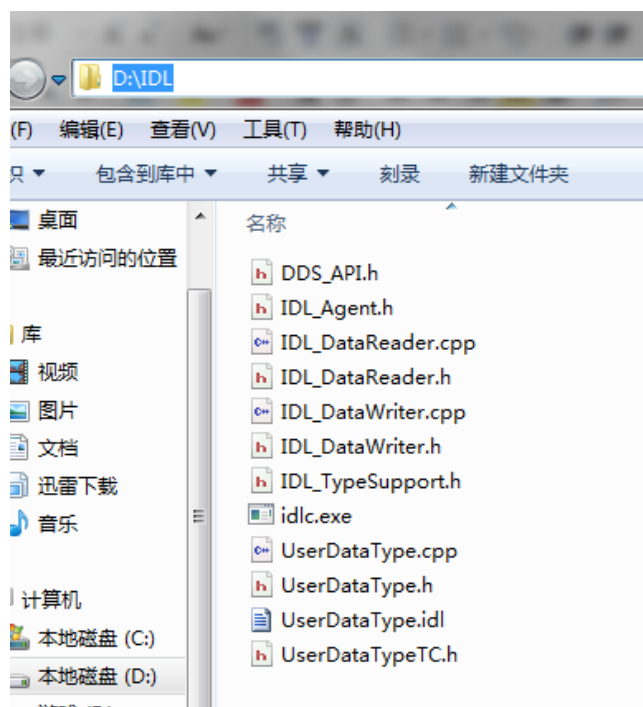


图 3 使用 IDL 编译器编译生成的文件

这十个文件的主要内容如下：

- myData.h: myData 数据类型的定义
- myData.cpp: myData 数据类型的函数实现
- myDataTC.h: myData 数据类型的 TypeCode 方法
- DDS_API.h: 用户接口定义
- IDL_Agent.h: 使用 BLUEDCS 库需要头文件及类
- IDL_DataReader.h: myData 数据类型订阅接口定义
- IDL_DataReader.cpp: myData 数据类型订阅接口实现
- IDL_DataWriter.h: myData 数据类型发布接口定义
- IDL_DataWriter.cpp: myData 数据类型发布接口实现
- IDL_TypeSupport.h: myData 数据类型实现需要头文件及类

注意:

1.使用系统内置类型_DDS_STRING 时,首先需要使用 IDL 编译器生成文件,否则无法使用。

► 步骤三：应用软件环境配置

开发环境: CDT 8.5.0 for Eclipse Luna

1. 创建项目、导入代码
直接将代码文件复制到项目的路径文件夹下即可,再刷新 Eclipse 的项目窗口。

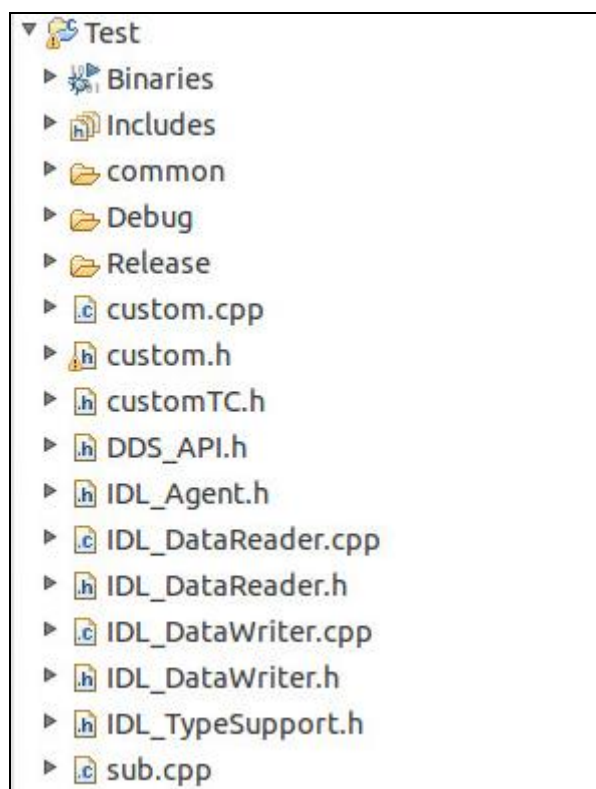


图 4 Eclipse 项目窗口

2. 配置头文件

在项目属性中的 G++ Compile 中设置，添加头文件。

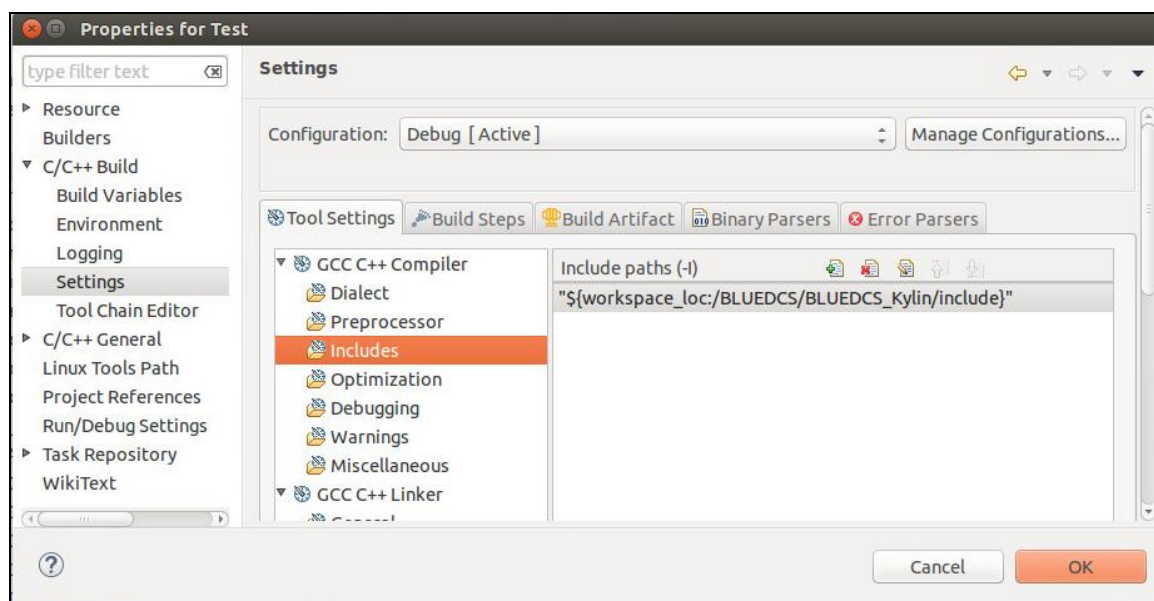


图 5 添加头文件路径

3. 配置依赖库

在项目属性->设置中的 GCC C++ linker 中配置；
这里需要添加 DDS 的依赖库，例如动态库文件为 libBLUEDCS.so，那么添加的库为 BLUEDCS（注意不需要加 lib 前缀）。

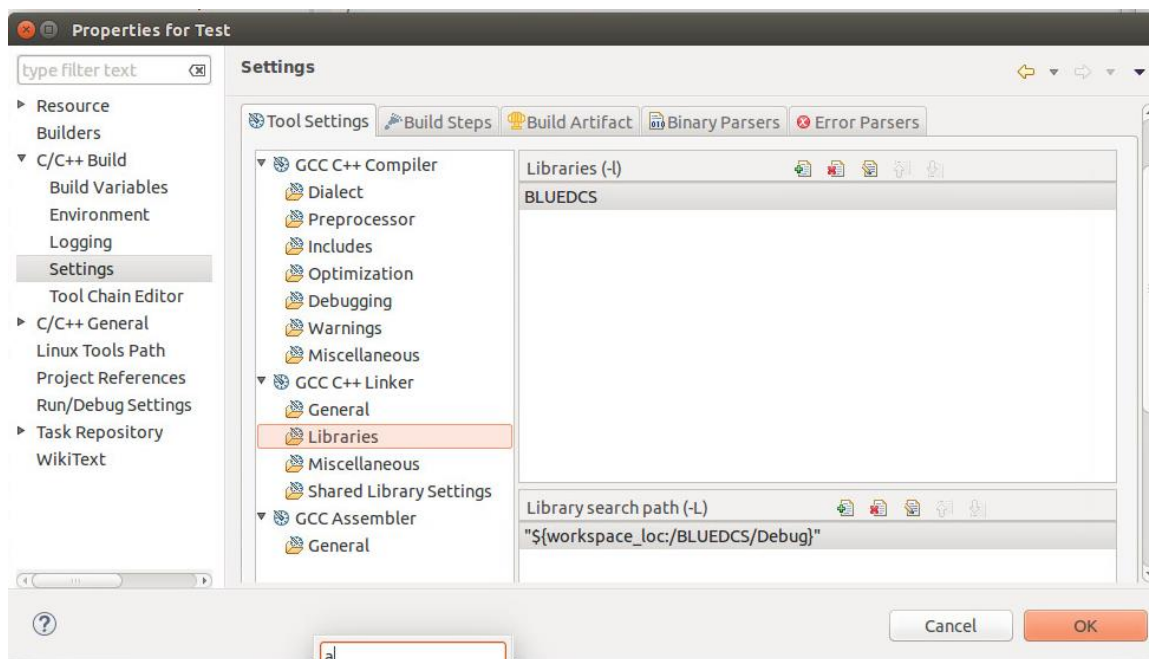


图 6 添加依赖的 DDS 库

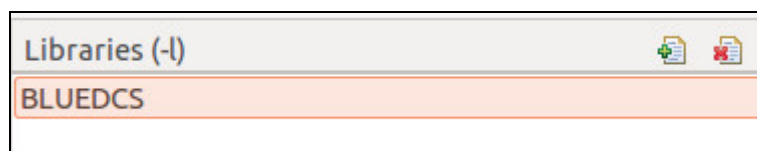


图 7 添加 DDS 依赖库的具体界面

4. 配置动态库搜索路径

目的是在程序运行时可以找到动态库（注意这与 win 下自动寻找当前路径下的动态库不同）；

在 GCC C++ linker 的 Miscellaneous 中配置，例如路径为/BLUEDCS/Debug；

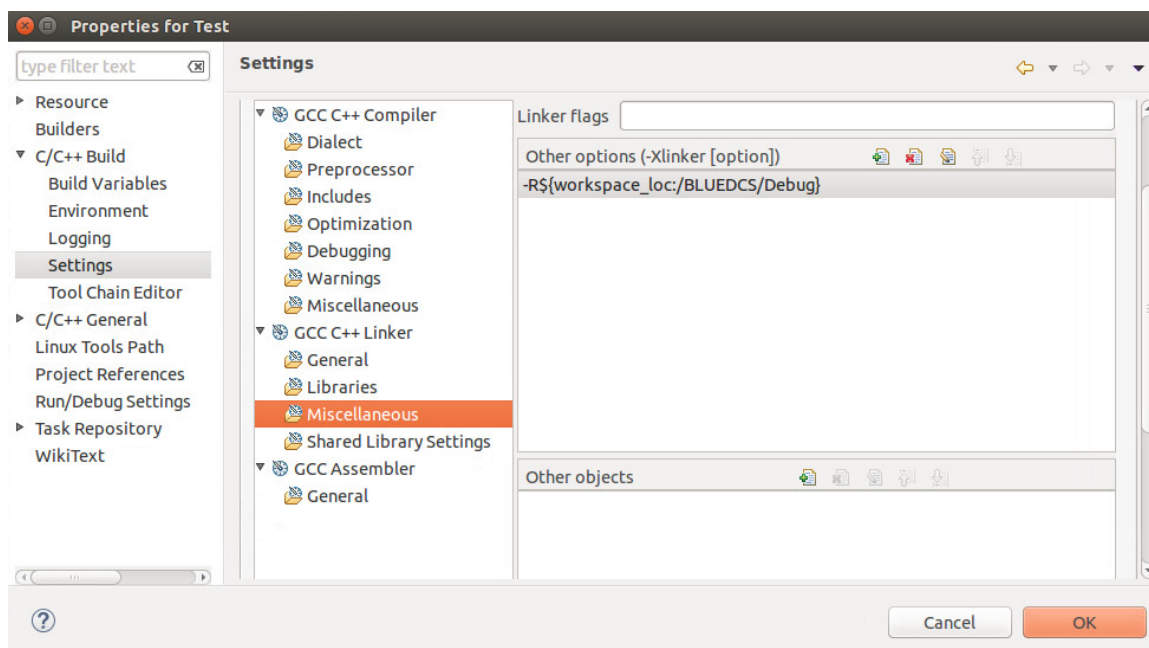


图 8 配置动态库搜索路径

添加时注意，需要加上-R 的前缀，之后是动态库的路径。

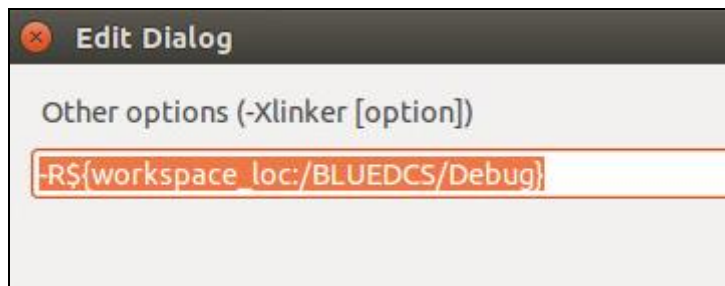


图 9 添加动态库路径

添加后如下：



图 10 添加之后的显示

这样程序在运行或调试时都会在这个路径下去寻找动态库。

还有其他两种方法：

- 1) 修改系统的默认搜索路径，类似添加环境变量 LD_LIBRARY_PATH;
- 2) 将 DDS 动态库放入/lib 或者/usr/lib 下。

5. 拷贝 common 文件夹

将 common 文件夹拷贝至可执行程序的同一路径之下即可。否则会出现 domainInit 错误的问题。（注意如果使用 Eclipse 调试的方式运行，可执行程序的路径是 Debug 文件夹）

► 步骤四：应用组件开发

根据用户的实际需求，进行应用组件代码的开发。开发人员可参考以下示例程序。

► 示例程序

► 发布端示例程序

```

/*****
*****发布端程序publisher.cpp*****
*****/

#include "dds_api.h"
#include <time.h>
#include <iostream>

```

```
#include <iomanip>
#include <fstream>
#include <string>
#include <unistd>

#define compName    "DDSTest"    //组件名
#define domainId    1            //数据域
#define topicName    "myDataTest" //主题名
using namespace std;

//程序主函数
int main()
{
    _RETURNCODE_T ret;

    //将发送端组件加入数据域
    ret = DomainInit(domainId,compName);

    if (ret != RETURNCODE_OK) //出错处理
    {
        cout<<"Domain: "<<domainId <<" Init failed"<<endl;
        return 0;
    }

    //应用组件QoS设置
    _DATA_WRITER_QOS *qos = new _DATA_WRITER_QOS();
    qos->Reliability.Kind = BEST_EFFORT;

    //创建发布者
    DataWriter * dataWriter = NULL;
    dataWriter = CreateDataWriter(compName,domainId, topicName, "myData", NULL, qos);
    if (dataWriter == NULL) //出错处理
    {
        cout<<"Create Data Writer failed"<<endl;
        DomainRelease(domainId);
        return 0;
    }

    //安全的类型转换
    myDataDataWriter *myDataTypeDataWriter = NULL;
    myDataTypeDataWriter = myDataDataWriter::Narrow(dataWriter);

    if (myDataTypeDataWriter == NULL) //出错处理
    {
```

```

        cout<<"myDataDataWriter Narrow failed"<<endl;
        DeleteDataWriter(dataWriter);
        DomainRelease(domainId);
        return 0;
    }

    //定义用户数据
    myData data;
    data.color = new char[30];

    cout<<"Please enter the point and color you want to send,such as 3 4 red"<<endl;
    while(1)
    {
        cin>>data.x>>data.y>>data.color;    //获取用户数据

        //将用户输入的数据调用Write接口发送
        ret = myDataTypeDataWriter->Write(data);

        if (ret != RETURNCODE_OK) //出错处理
        {
            cout <<"Write error: " << ret << endl;
        }
    }

    //安全删除dataWriter
    DeleteDataWriter(dataWriter);

    //安全退出数据域
    DomainRelease(domainId);

    return 0;
}

```

► 订阅端示例程序

```

/*****
*****订阅端程序subscriber.cpp*****
*****/
#include "subscriber.h"

//重载数据接收回调函数
_RETURNCODE_T HelloListener::On_Data_Available(DataReader* dataReader)

```

```
{
    _RETURNCODE_T ret = RETURNCODE_OK;

    //安全的类型转换
    myDataDataReader* myDataReader = NULL;
    myDataReader = myDataDataReader::Narrow(dataReader);

    if (myDataReader == NULL) //出错处理
    {
        cout << "Narrow failed at On_Data_Available of HelloListener" << endl;
        return -1;
    }

    //获取用户数据
    myData data;
    ret = myDataReader->Read_Next_Sample(data);

    if (ret != RETURNCODE_OK) //出错处理
    {
        return -1;
    }

    cout<<"Received " << data.x<<" " <<data.y<<" " <<data.color<<endl; //显示接收到的数据

    return RETURNCODE_OK;
}

//程序主函数
int main()
{
    _RETURNCODE_T ret;
    DataReader *dataReader = NULL;

    //将接收端组件加入数据域
    ret = DomainInit(domainId,compName);

    if (ret != RETURNCODE_OK) //出错处理
    {
        cout<<"Domain:"<<domainId<<" Init failed"<<endl;
        return 0;
    }

    //初始化QoS策略，使用尽力而为的传输
    _DATA_READER_QOS *qos = new _DATA_READER_QOS();
```



```

qos->Reliability.Kind = BEST_EFFORT;

//创建订阅者并绑定回调函数
HelloListener *listener = new HelloListener(0);
dataReader = CreateDataReader(compName, domainId, topicName, "myData", listener, qos);

if (dataReader == NULL) //出错处理
{
    cout<<"Create Data Reader failed"<<endl;
    DomainRelease(domainId);
    return 0;
}

cout<<"Press any button to exit..."<<endl;
int a;
cin >> a;

return 0;
}

/*****
*****订阅端程序subscriber.h *****
*****/

#ifndef _SUBSCRIBER_H
#define _SUBSCRIBER_H


#include "dds_api.h"
#include <time.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <unistd>

#define compName    "DDSTest"    //应用组件名
#define domainId    1            //数据域
#define topicName    "myDataTest" //主题名
using namespace std;

class HelloListener : public DataReaderListener
{
public:
    HelloListener(long num)

```

```
{  
    receiveCount = num;  
}  
  
_RETURNCODE_T On_Data_Available(DataReader* dataReader);  
  
long receiveCount;  
  
};  
#endif
```



地址：北京市海淀区丰贤东路1号

邮编：100094

电话：010-59516390

传真：010-59516300

联系人：韩澎 18911990571

杨猛 18911990524

