

# uDDS 数据分发服务软件 开发手册-Linux-MakeFile

南京磐优信息科技有限公司

2022 年 12 月

## 修订记录

版本号	修订状态	简要说明修订内容和范围	修订日期	修订人
V1.0	A	创建文件	20221122	王易
V1.0	A	增加缺少权限的错误处理	20230308	蒋梦杰

注：修订记录在文件提交后换版时使用，修订状态栏填写：A—增加，M—修改，D—删除

## 目 录

1 准备工作 .....	3
2 目录介绍 .....	3
3 快速开发过程 .....	3
4 进阶开发过程 .....	8
4.1 生成辅助文件 .....	8
4.2 创建发布端/订阅端。 .....	11
5 常见错误排查 .....	15
5.1 找不到指定文件 .....	15
5.2 Write error 200 .....	16
6 发布端示例程序 .....	17
7 订阅端示例程序 .....	21

## 1 准备工作

- 1) Linux 操作系统
- 2) 86\_64-linux-gnu-g++ 7.5.0 及以上版本编译器
- 3) 试用版 uDDS (uDDS-Linux-Trial)。
- 4) 解压 uDDS-Linux-Trial 文件。

## 2 目录介绍

uDDS-Linux-Trial 目录包含项目如下：

*uDDS*: uDDS 库和头文件。

*用户手册*: 适用于 Linux 平台使用 Makefile 文件进行编译的 uDDS 开发者指南。

*进阶开发资源/MakeFile*: 用于编译发布端和订阅端的 Makefile 文件。

*进阶开发资源/uDDSGen*: 处理 IDL 文件，生成辅助代码。

*进阶开发资源/示例代码*: 开发者指南中的示例代码。

*快速开发资源/DemoTool*: DDS 示例自动生成工具，快速生成 uDDS 示例项目，可直接编译运行。

## 3 快速开发过程

- 1) 打开 uDDS 文件夹；

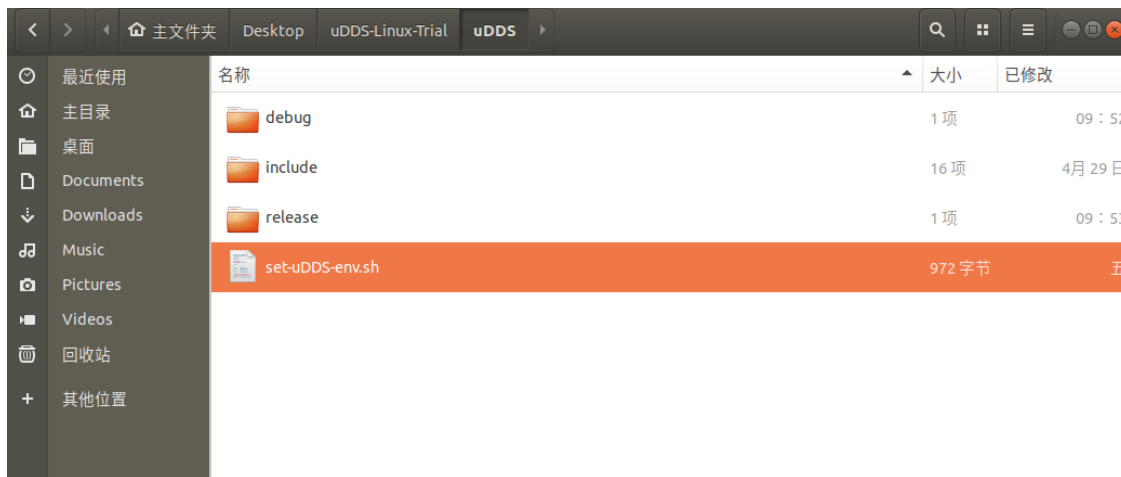


图 1 打开 uDDS 目录

- 2) 在 uDDS 文件目录下右键并点击在终端打开，或打开终端转到 uDDS 文件夹下并执行命令：

```
./ set-uDDS-env.sh
```



```

platforu@ubuntu: ~/Desktop/uDDS-Linux-Trial/uDDS
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
export uDDSHOME=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS
platforu@ubuntu:~/Desktop/uDDS-Linux-Trial/uDDS$
    
```

图 2 配置环境变量

3) /进入快速开发/DemoTool 目录。

如需更改 idl 内容请见 [4.1 生成辅助文件模块](#)，更改后的 xxx.idl 可以拷贝到该目录下，后续用 xxx.idl 来代替 UserDataTypes.idl 生成项目，继续快速开发。

- 4) 在 DemoTool 目录下右键并点击在终端打开，或者直接打开终端并转到 DemoTool 目录下，执行如下代码：

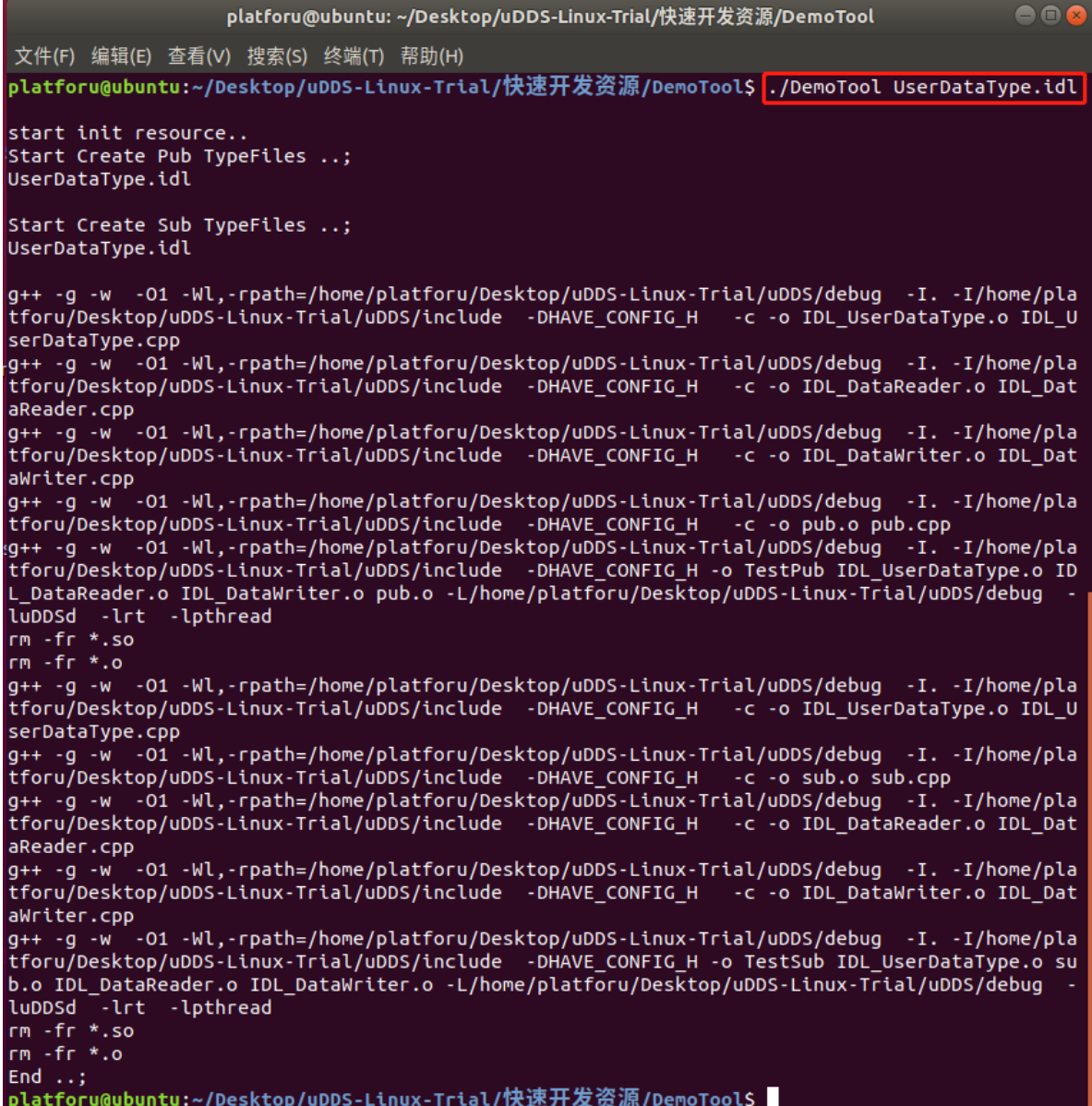
```
./DemoTool UserDataType.idl
```

DemoTool: DemoTool 程序名；

UserDataType.idl: 生成通信 demo 的 idl 文件。

**注意：**如果生成失败转至 5.3 查看是否是没有权限，给文件加赋予权限。

执行结果如图 3 所示。生成 demo 项目并编译生成可执行文件。



```

platform@ubuntu: ~/Desktop/uDDS-Linux-Trial/快速开发资源/DemoTool
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
platform@ubuntu:~/Desktop/uDDS-Linux-Trial/快速开发资源/DemoTools$ ./DemoTool UserDataType.idl
start init resource..
Start Create Pub TypeFiles ...;
UserDataType.idl

Start Create Sub TypeFiles ...;
UserDataType.idl

g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_UserDataType.o IDL_U
serDataType.cpp
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_DataReader.o IDL_Dat
aReader.cpp
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_DataWriter.o IDL_Dat
aWriter.cpp
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o pub.o pub.cpp
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -o TestPub IDL_UserDataType.o ID
L_DataReader.o IDL_DataWriter.o pub.o -L/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -
luDDSD -lrt -lpthread
rm -fr *.so
rm -fr *.o
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_UserDataType.o IDL_U
serDataType.cpp
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o sub.o sub.cpp
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_DataReader.o IDL_Dat
aReader.cpp
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_DataWriter.o IDL_Dat
aWriter.cpp
g++ -g -w -O1 -WL,-rpath=/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/pla
tform/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -o TestSub IDL_UserDataType.o su
b.o IDL_DataReader.o IDL_DataWriter.o -L/home/platform/Desktop/uDDS-Linux-Trial/uDDS/debug -
luDDSD -lrt -lpthread
rm -fr *.so
rm -fr *.o
End ...;
platform@ubuntu:~/Desktop/uDDS-Linux-Trial/快速开发资源/DemoTools$

```

图 3 生成项目

- 5) 进入 PUBSUB\_Makefile/Sub 文件夹（如下图所示）后，执行以下命令运行订阅端等待与发布端匹配并接收命令：

```
./TestSub
```

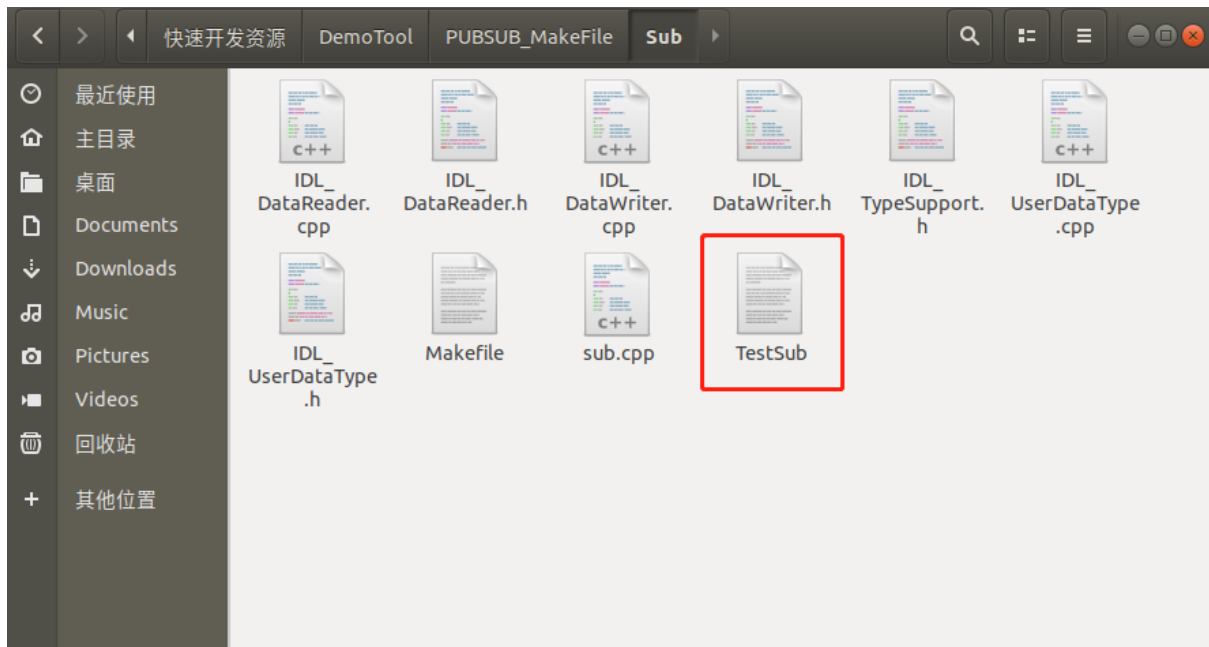


图 4 进入 Sub 文件夹

如下图所示为执行结果：

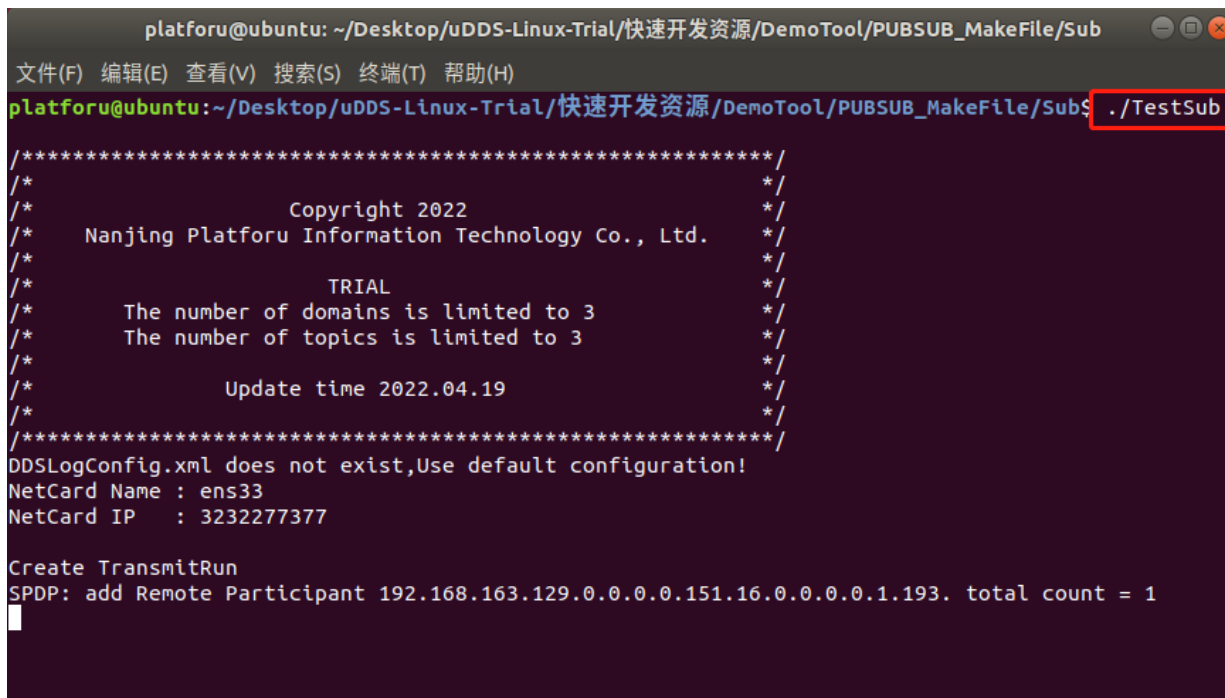


图 5 启动订阅端

- 6) 进入 PUBSUB\_Makefile/Pub 文件夹（如下图所示）后，执行以下命令运行发布端，与订阅端匹配：

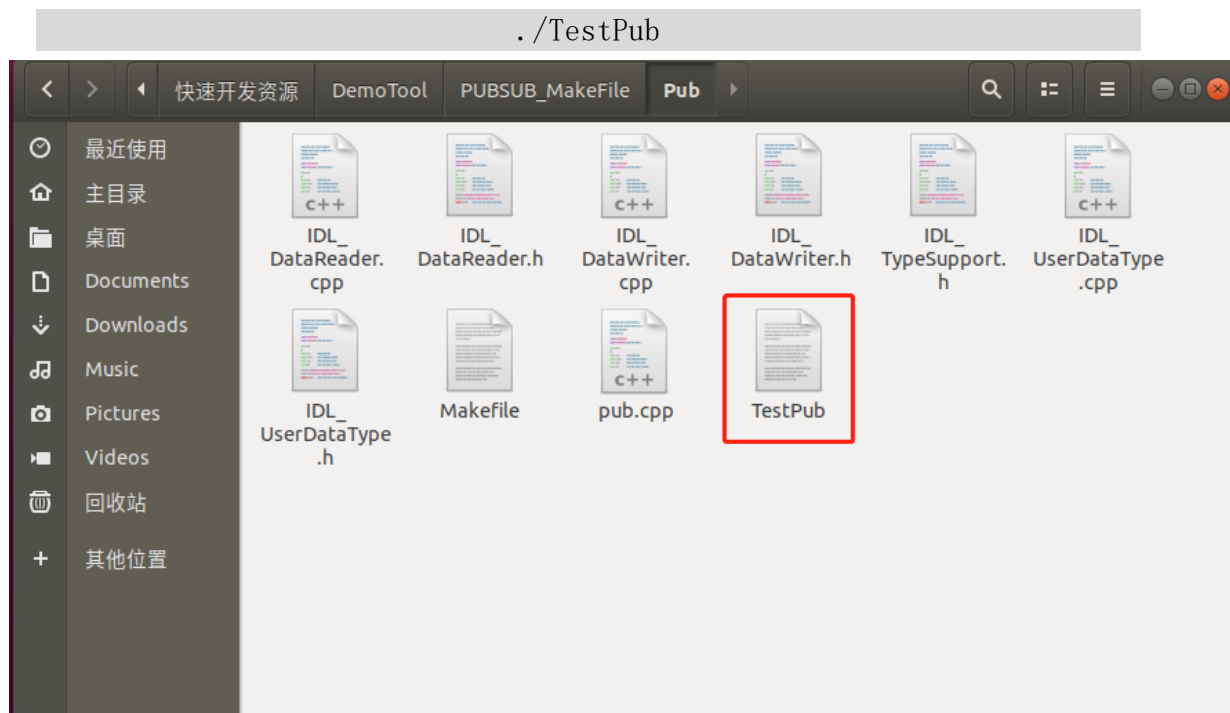


图 6 打开 Pub 文件夹

如下图所示为执行结果：

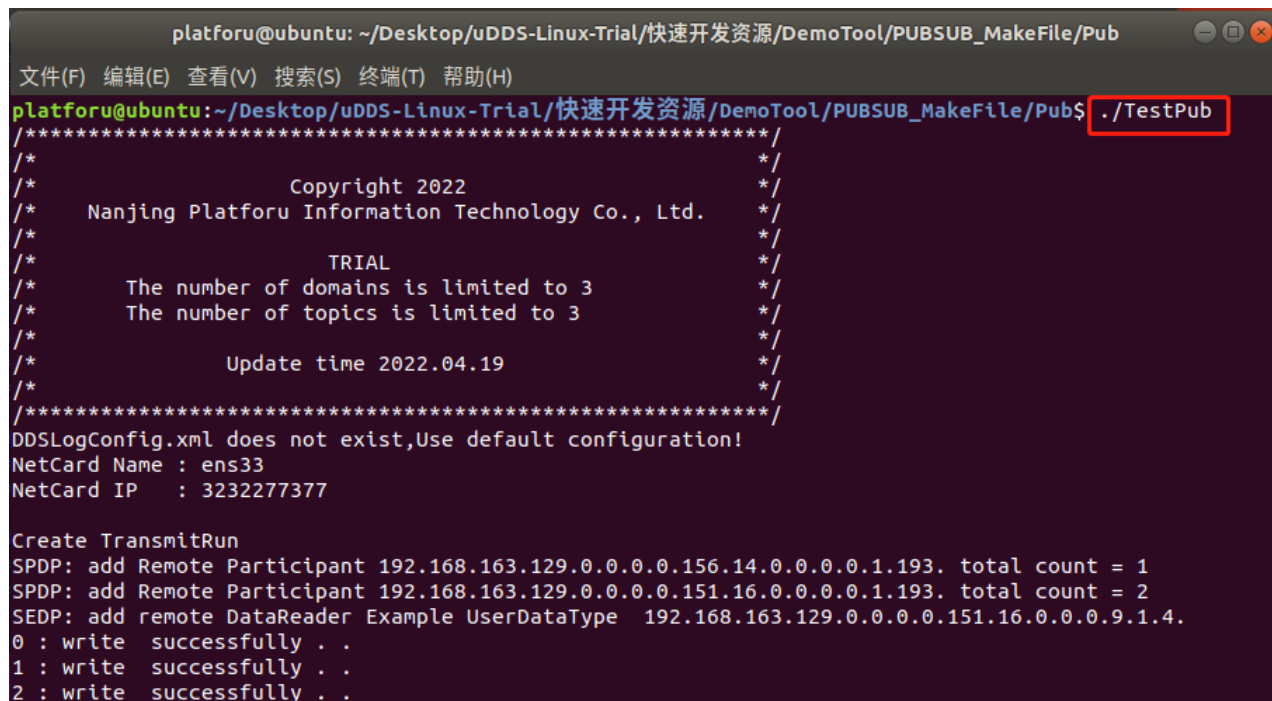


图 7 启动发布端



## 7) 发布端启动后与发布端收发数据如下图所示

```

platforu@ubuntu: ~/Desktop/uDDS-Linux-Trial/快速开发资源/DemoTool/PUBSUB_Ma...
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
platforu@ubuntu:~/Desktop/uDDS-Linux-Trial/快速开发资源/DemoTool/PUBSUB_Ma
keFile/Pub$ ./TestPub
/*****
/*
/*      Copyright 2022
/*      Nanjing Platforu Information Technology Co., Ltd.
/*
/*      TRIAL
/*      The number of domains is limited to 3
/*      The number of topics is limited to 3
/*
/*      Update time 2022.04.19
/*
*****/
DDSLogConfig.xml does not exist,Use default configuration!
NetCard Name : ens33
NetCard IP   : 3232277377

Create TransmitRun
SPDP: add Remote Participant 192.168.163.129.0.0.0.0.25.34.0.0.0.1.193.
total count = 1
SPDP: add Remote Participant 192.168.163.129.0.0.0.0.201.7.0.0.0.1.193.
total count = 2
SEDP: add remote DataReader Example UserDataTy 192.168.163.129.0.0.0.0.
201.7.0.0.9.1.4.
0 : write successfully .
1 : write successfully .
2 : write successfully .
3 : write successfully .
4 : write successfully .
5 : write successfully .
6 : write successfully .
7 : write successfully .
8 : write successfully .
9 : write successfully .
10 : write successfully .
11 : write successfully .
12 : write successfully .
13 : write successfully .

platforu@ubuntu:~/Desktop/uDDS-Linux-Trial/快速开发资源/DemoTool/PUBSUB_Ma...
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
platforu@ubuntu:~/Desktop/uDDS-Linux-Trial/快速开发资源/DemoTool/PUBSUB_Ma
keFile/Sub$ ./TestSub
/*****
/*
/*      Copyright 2022
/*      Nanjing Platforu Information Technology Co., Ltd.
/*
/*      TRIAL
/*      The number of domains is limited to 3
/*      The number of topics is limited to 3
/*
/*      Update time 2022.04.19
/*
*****/
DDSLogConfig.xml does not exist,Use default configuration!
NetCard Name : ens33
NetCard IP   : 3232277377

Create TransmitRun
SPDP: add Remote Participant 192.168.163.129.0.0.0.0.201.7.0.0.0.1.193. t
otal count = 1
SPDP: add Remote Participant 192.168.163.129.0.0.0.0.25.34.0.0.0.1.193. t
otal count = 2
SEDP: add remote DataWriter Example UserDataTy 192.168.163.129.0.0.0.0.2
5.34.0.0.0.4.1.3.
UserDataTy:
x = 0
y = 0
color =
UserDataTy:
x = 0
y = 0
color =
UserDataTy:
x = 0
y = 0
color =
UserDataTy:
x = 0
y = 0
color =

```

图 8 数据收发示例

## 4 进阶开发过程

### 4.1 生成辅助文件

- 1) 进入 uDDSGen 文件, 创建一个 idl 文件并在 idl 文件里面添加需要使用的数据类型, 数据类型的定义形式为 IDL 文件, 用户可以创建一个文本文件, 然后将扩展名修改为 “.idl”, 并使用文本编辑器进行数据类型定义, 如图所示。

```

1 struct UserDataTy
2 {
3     short x;
4     long y;
5     string color;
6 };
7

```

图 9 用户定义数据类型

- 2) 使用控制台进入进阶开发资源/uDDSGen 目录并执行 uDDSGen 可执行程序，对用户定义的 idl 文件进行编译。例如编译器名称为 uDDSGen，用户定义的 idl 文件名称为 UserDataTypes.idl，执行如下命令：

```
./uDDSGen UserDataTypes.idl
```

uDDSGen: idl 文件编译程序；

UserDataTypes.idl: 要用户自定义 idl 文件的文件名

如图所示。



```

platforu@ubuntu: ~/Desktop/uDDS-Linux-Trial/进阶开发资源/uDDSGen
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
platforu@ubuntu:~/Desktop/uDDS-Linux-Trial/进阶开发资源/uDDSGen$ ./uDDSGen  UserDataTypes.idl
UserDataTypes.idl

/*****
/*                                     */
/*           Copyright 2022           */
/*   Nanjing Platforu Information Technology Co., Ltd.   */
/*                                     */
/*                                     */
/*           Update time 2022.09.19   */
/*                                     */
*****/
platforu@ubuntu:~/Desktop/uDDS-Linux-Trial/进阶开发资源/uDDSGen$

```

图 10 编译 idl 文件

- 3) 如果输出结果为用户编译的 idl 文件名称，如上图所示，则说明编译成功，编译器将在 IDL 目录下产生七个辅助文件放入“UserDataTypes\_当前时间”文件夹下，如图所示。

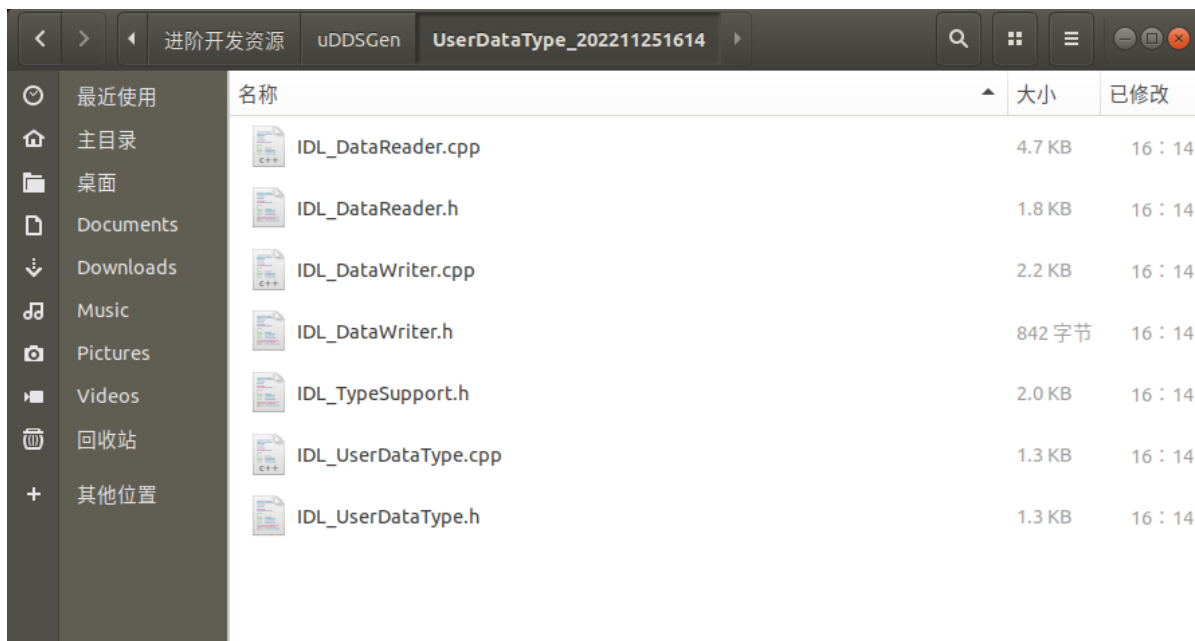


图 11 辅助文件

这七个文件的主要内容如下：

- IDL\_DataReader.h: UserDataTypes 数据类型订阅接口定义
- IDL\_DataReader.cpp: UserDataTypes 数据类型订阅接口实现
- IDL\_DataWriter.h: UserDataTypes 数据类型发布接口定义
- IDL\_DataWriter.cpp: UserDataTypes 数据类型发布接口实现
- IDL\_TypeSupport.h: UserDataTypes 数据类型实现需要头文件及类
- IDL\_UserDataType.h: UserDataTypes 数据类型的定义
- IDL\_UserDataType.cpp: UserDataTypes 数据类型的函数实现

## 4.2 创建发布端/订阅端。

使用生成的辅助代码进行用户程序开发

- 1) 新建两个目录并分别命名为“Pub”作为发布端工作目录、“Sub”作为订阅端工作目录（如下图所示）：

注：目录名用户可自定义；



图 12 生成 Pub 和 Sub 文件夹

- 2) 将 3.1 中生成的辅助文件复制到 Pub 文件夹及 Sub 文件夹下：

注：辅助文件不可修改。



图 13 复制辅助文件

- 3) 用户可根据测试用例代码及 idl 文件编写自己的代码调用辅助文件中的 uDDS API 接口并复制到 Pub 或 Sub 目录下，本文以测试用例 pub.cpp 及 sub.cpp 为例。将“进阶开发资源/测试用例代码”中 pub.cpp 复制到 Pub 目录下，将 sub.cpp 复制到 Sub 目录下。

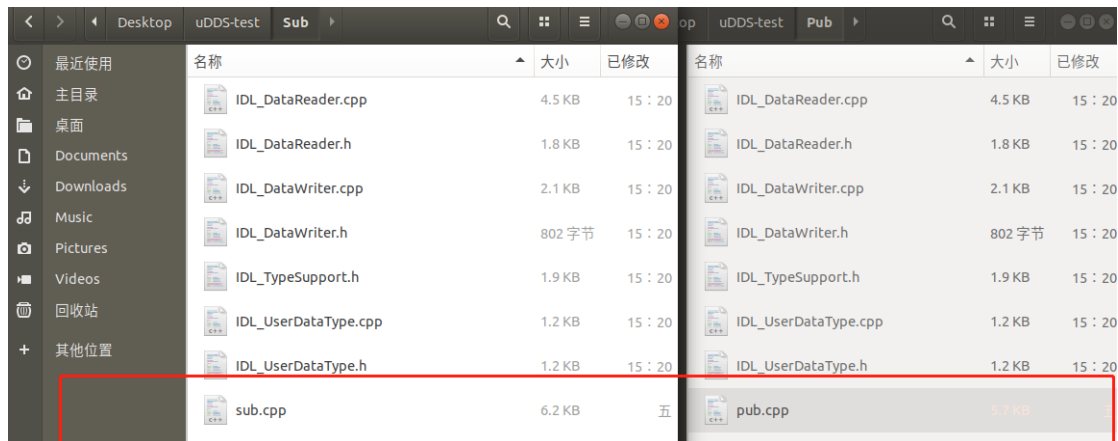


图 14 复制测试用例代码

- 4) 将“进阶开发资源/MakeFile”目录下 Makefile 文件复制到 Pub 和 Sub 文件夹下：

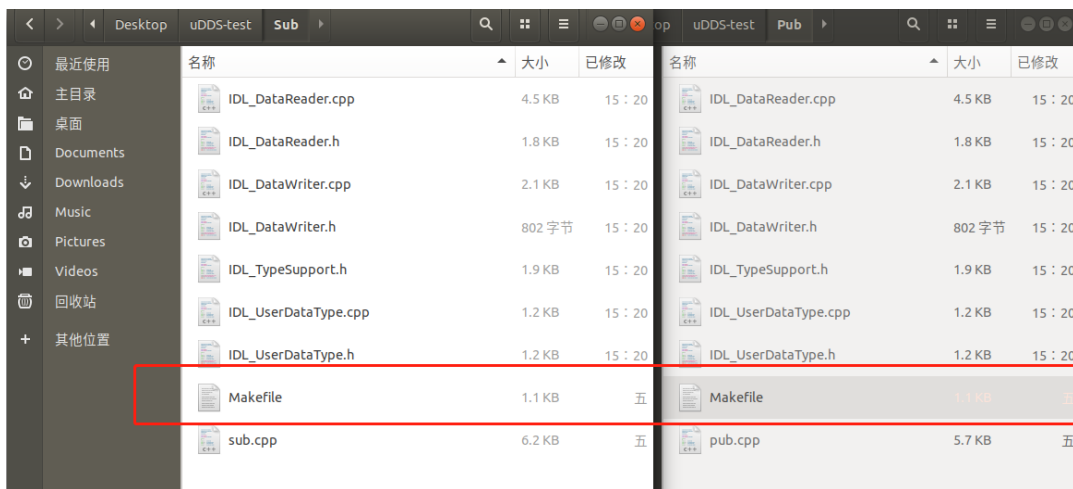
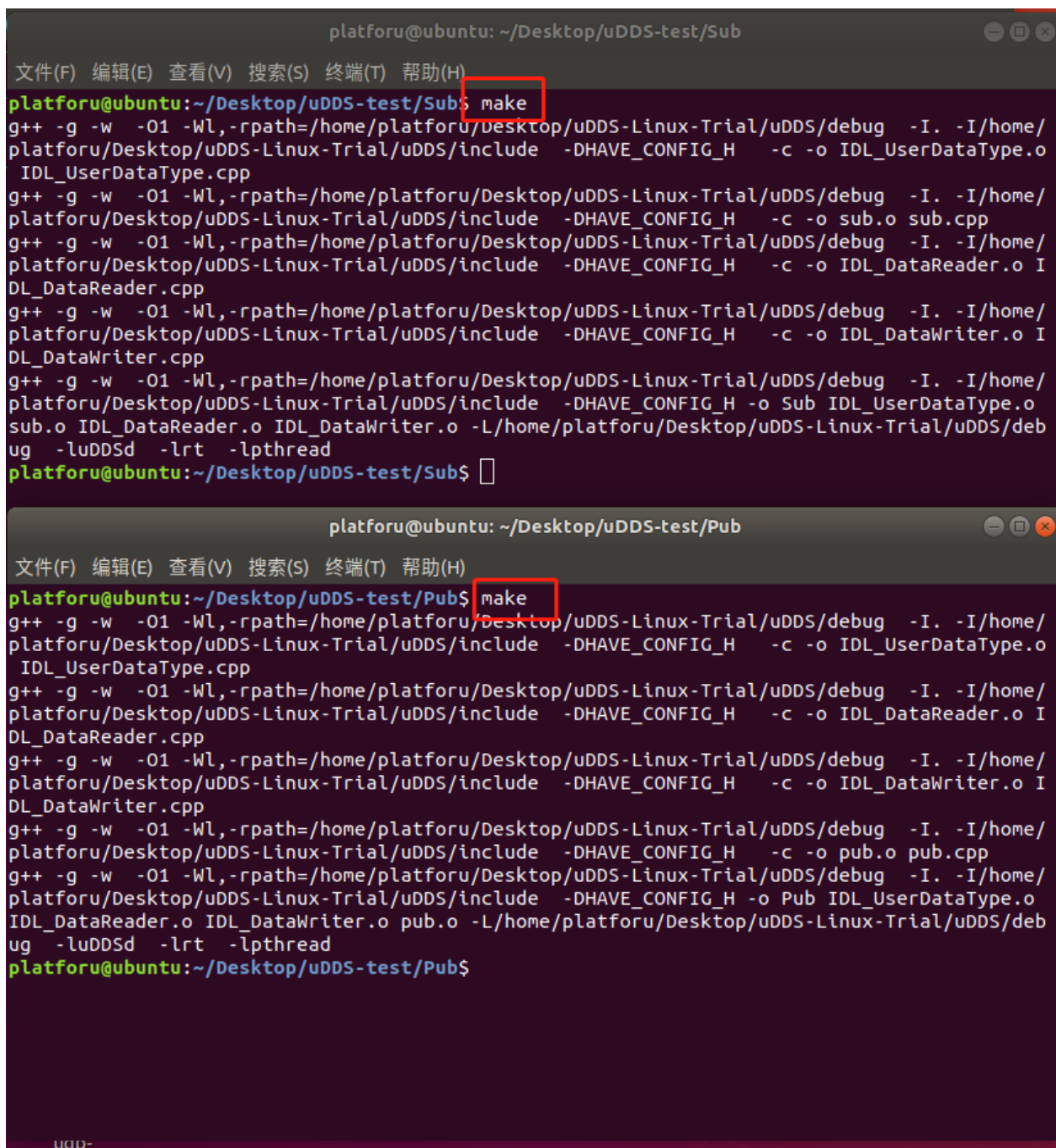


图 15 复制 Makefile 文件

5) 分别在 Pub 文件夹和 Sub 文件夹下打开终端并执行 make 命令：



The image displays two terminal windows. The top window is titled 'platforu@ubuntu: ~/Desktop/uDDS-test/Sub' and shows the command 'make' being executed. The output lists the compilation of IDL\_UserDataTypes.o, sub.o, IDL\_DataReader.o, and IDL\_DataWriter.o, followed by linking them into the final executable 'Sub' with various flags like -luDDSd, -lrt, and -lpthread. The bottom window is titled 'platforu@ubuntu: ~/Desktop/uDDS-test/Pub' and also shows 'make' being executed. Its output shows the compilation of IDL\_UserDataTypes.o, IDL\_DataReader.o, IDL\_DataWriter.o, and pub.o, followed by linking them into the final executable 'Pub' with the same flags as the Sub window.

```

platforu@ubuntu: ~/Desktop/uDDS-test/Sub
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
platforu@ubuntu:~/Desktop/uDDS-test/Sub$ make
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_UserDataTypes.o IDL_UserDataTypes.cpp
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o sub.o sub.cpp
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_DataReader.o IDL_DataReader.cpp
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_DataWriter.o IDL_DataWriter.cpp
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -o Sub IDL_UserDataTypes.o sub.o IDL_DataReader.o IDL_DataWriter.o -L/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -luDDSd -lrt -lpthread
platforu@ubuntu:~/Desktop/uDDS-test/Sub$

platforu@ubuntu: ~/Desktop/uDDS-test/Pub
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
platforu@ubuntu:~/Desktop/uDDS-test/Pub$ make
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_UserDataTypes.o IDL_UserDataTypes.cpp
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_DataReader.o IDL_DataReader.cpp
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o IDL_DataWriter.o IDL_DataWriter.cpp
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o pub.o pub.cpp
g++ -g -w -O1 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -o Pub IDL_UserDataTypes.o IDL_DataReader.o IDL_DataWriter.o pub.o -L/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -luDDSd -lrt -lpthread
platforu@ubuntu:~/Desktop/uDDS-test/Pub$
  
```

图 16 执行 make 语句

执行命令后得到发布端可执行程序“TestPub”和订阅端可执行程序“TestSub”：

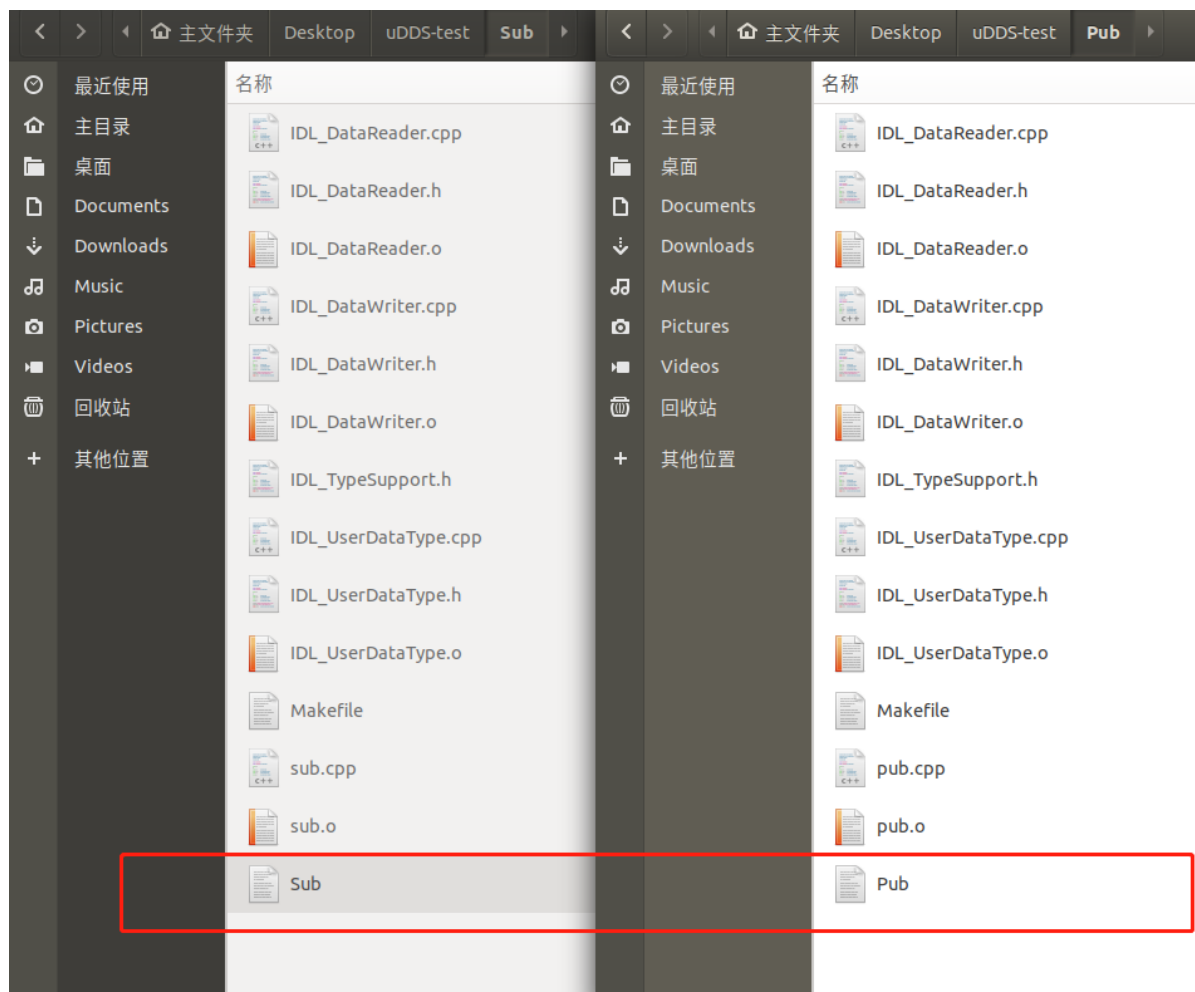


图 17 生成可执行程序在当前目录

## 5 常见错误排查

下面内容有助于解决一些问题，这些问题是在您拿到压缩文件到运行演示结束，过程中可能会遇到。如果这一阶段您没有出现问题，可以忽略。

### 5.1 找不到指定文件

如图所示。

```
platforu@ubuntu:~/Desktop/uDDS-Linux-Trial/DemoTool$ ./DemoTool d.idl
start init resource...
cp: 无法获取 'd.idl' 的文件状态(stat): 没有那个文件或目录
cp: 无法获取 'd.idl' 的文件状态(stat): 没有那个文件或目录
Start Create Pub TypeFiles ..;
d.idl
can not open!Start Create Sub TypeFiles ..;
d.idl
can not open!g++ -g -w -O3 -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug
-I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o pub.o
pub.cpp
pub.cpp:10:10: fatal error: IDL_TypeSupport.h: 没有那个文件或目录
#include "IDL_TypeSupport.h"
          ^~~~~~
compilation terminated.
<内置>: recipe for target 'pub.o' failed
make: *** [pub.o] Error 1
g++ -g -w -WL,-rpath=/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/debug -I. -I/home/platforu/Desktop/uDDS-Linux-Trial/uDDS/include -DHAVE_CONFIG_H -c -o sub.o sub.cpp
sub.cpp:10:10: fatal error: IDL_TypeSupport.h: 没有那个文件或目录
#include "IDL_TypeSupport.h"
          ^~~~~~
compilation terminated.
<内置>: recipe for target 'sub.o' failed
make: *** [sub.o] Error 1
End ..;
platforu@ubuntu:~/Desktop/uDDS-Linux-Trial/DemoTool$
```

图 18 找不到指定文件

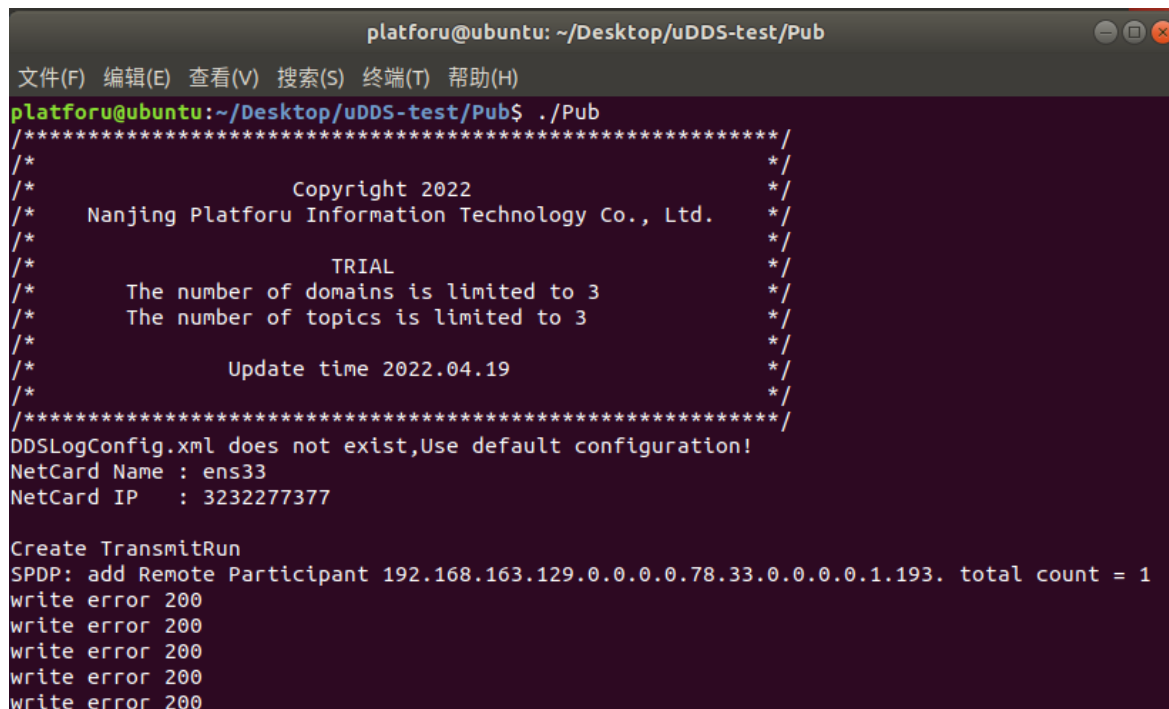
问题原因：idl 文件输入有误。

解决方法：确认一下 DemoTool 文件下，用户输入的 idl 文件名字是否与 UserDataTypes.idl 对应。



## 5.2 Write error 200

如图所示。



```

platforu@ubuntu: ~/Desktop/uDDS-test/Pub
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
platforu@ubuntu:~/Desktop/uDDS-test/Pub$ ./Pub
/*****/
/*
/*      Copyright 2022
/*      Nanjing Platforu Information Technology Co., Ltd.
/*
/*      TRIAL
/*      The number of domains is limited to 3
/*      The number of topics is limited to 3
/*
/*      Update time 2022.04.19
/*
/*****/
DDSLogConfig.xml does not exist,Use default configuration!
NetCard Name : ens33
NetCard IP   : 3232277377

Create TransmitRun
SPDP: add Remote Participant 192.168.163.129.0.0.0.0.78.33.0.0.0.0.1.193. total count = 1
write error 200
write error 200
write error 200
write error 200
write error 200

```

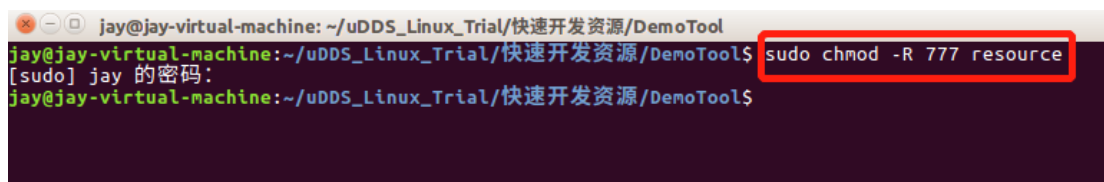
图 19 write error 200

问题原因：没有对应的接收端演示程序。

解决方法：启动一个接收端演示程序即可。

## 5.3 权限不够

如下图所示：



```

jay@jay-virtual-machine: ~/uDDS_Linux_Trial/快速开发资源/DemoTool
jay@jay-virtual-machine:~/uDDS_Linux_Trial/快速开发资源/DemoTool$ sudo chmod -R 777 resource
[sudo] jay 的密码:
jay@jay-virtual-machine:~/uDDS_Linux_Trial/快速开发资源/DemoTool$

```

图 20 权限不够

问题原因：没有对文件的使用权限。

解决方法：给没有权限的文件赋予权限。

## 6 发布端示例程序

```

/*****
*****发布端程序pub.cpp*****
*****/
#include <stdio.h>
#include <stdlib.h>
#include <cstring>
#if defined(_WIN32)
#else
#include <unistd.h>
#endif

/* IDL_TypeSupport.h中包含所有依赖的头文件 */
#include "IDL_TypeSupport.h"

/* 删除所有实体 */
static int publisher_shutdown(DomainParticipant *participant)
{
    ReturnCode_t retcode;
    int status = 0;

    if (participant != NULL) {
        retcode = participant->delete_contained_entities();
        if (retcode != RETCODE_OK) {
            fprintf(stderr, "delete_contained_entities
error %d\n", retcode);
            status = -1;
        }

        retcode =
DomainParticipantFactory::get_instance()->delete_participant(participant);
        if (retcode != RETCODE_OK) {
            fprintf(stderr, "delete_participant error %d\n",
retcode);
            status = -1;
        }
    }

    return status;
}

/* 发布者函数 */
extern "C" int publisher_main(int domainId, int sample_count)
{
    DomainParticipant *participant = NULL;
    Publisher *publisher = NULL;
    Topic *topic = NULL;
    DataWriter *writer = NULL;
    UserDataWriter * UserDataWriter = NULL;

```

```

UserDataTypes *instance = NULL;
ReturnCode_t retcode;
InstanceHandle_t instance_handle = HANDLE_NIL;
const char *type_name = NULL;
int count = 0;

/* 1. 创建一个participant, 可以在此处定制participant的QoS */
/* 建议1: 在程序启动后优先创建participant, 进行资源初始化*/
/* 建议2: 相同的domainId只创建一次participant, 重复创建会占用大量资源 */
*/
participant =
DomainParticipantFactory::get_instance()->create_participant(
    domainId, PARTICIPANT_QOS_DEFAULT/* participant默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (participant == NULL) {
    fprintf(stderr, "create_participant error\n");
    publisher_shutdown(participant);
    return -1;
}

/* 2. 创建一个publisher, 可以在创建publisher的同时定制其QoS */
/* 建议1: 在程序启动后优先创建publisher */
/* 建议2: 一个participant下创建一个publisher即可, 无需重复创建 */
publisher = participant->create_publisher(
    PUBLISHER_QOS_DEFAULT /* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (publisher == NULL) {
    fprintf(stderr, "create_publisher error\n");
    publisher_shutdown(participant);
    return -1;
}

/* 3. 在创建主题之前注册数据类型 */
/* 建议1: 在程序启动后优先进行注册 */
/* 建议2: 一个数据类型注册一次即可 */
type_name = UserDataTypesTypeSupport::get_type_name();
retcode = UserDataTypesTypeSupport::register_type(
    participant, type_name);
if (retcode != RETCODE_OK) {
    fprintf(stderr, "register_type error %d\n", retcode);
    publisher_shutdown(participant);
    return -1;
}

/* 4. 创建主题, 并定制主题的QoS */
/* 建议1: 在程序启动后优先创建Topic */
/* 建议2: 一种主题名创建一次即可, 无需重复创建 */
topic = participant->create_topic(
    "Example UserDataTypes"/* 主题名 */,
    type_name /* 类型名 */, TOPIC_QOS_DEFAULT/* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);

```

```

if (topic == NULL) {
    fprintf(stderr, "create_topic error\n");
    publisher_shutdown(participant);
    return -1;
}

/* 5. 创建datawriter, 并定制datawriter的QoS */
/* 建议1: 在程序启动后优先创建datawriter */
/* 建议2: 创建一次即可, 无需重复创建 */
/* 建议3: 在程序退出时再进行释放 */
/* 建议4: 避免打算发送数据时创建datawriter, 发送数据后删除, 该做法消耗
资源, 影响性能 */
writer = publisher->create_datawriter(
    topic, DATAWRITER_QOS_DEFAULT/* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (writer == NULL) {
    fprintf(stderr, "create_datawriter error\n");
    publisher_shutdown(participant);
    return -1;
}
UserDataWriter = UserDataWriter::narrow(writer);
if (UserDataWriter == NULL) {
    fprintf(stderr, "DataWriter narrow error\n");
    publisher_shutdown(participant);
    return -1;
}

/* 6. 创建一个数据样本 */
/* 建议: 该数据为new出来的, 使用后用户需要调用delete_data进行释放内存
*/
instance = UserDataWriter::create_data();
if (instance == NULL) {
    fprintf(stderr, "UserDataWriter::create_data
error\n");
    publisher_shutdown(participant);
    return -1;
}

//此处可以修改数据的值
instance->x = 3;
instance->y = 4;
instance->color = const_cast<char *>("red");

/* 7. 主循环, 发送数据 */
for (count = 0; (sample_count == 0) || (count < sample_count); ++count)
{
    retcode = UserDataWriter->write(*instance,
instance_handle);
    if (retcode != RETCODE_OK) {
        fprintf(stderr, "write error %d\n", retcode);
    }
}

```

```

    }
    else
        fprintf(stderr, "%d : write successfully . . \n",
count);
#if defined(_WIN32)
        Sleep(1000); //沉睡1秒
#else
        sleep(1); //沉睡1秒
#endif
    }

    /* 8. 删除数据样本 */
    retcode = UserDataTypesupport::delete_data(instance);
    if (retcode != RETCODE_OK) {
        fprintf(stderr, "UserDataTypesupport::delete_data
error %d\n", retcode);
    }

    /* 9. 删除所有实例 */
    return publisher_shutdown(participant);
}

int main(int argc, char *argv[])
{
    int domain_id = 0;
    int sample_count = 0; /* 无限循环 */

    if (argc >= 2) {
        domain_id = atoi(argv[1]); /* 发送至域domain_id */
    }
    if (argc >= 3) {
        sample_count = atoi(argv[2]); /* 发送sample_count次 */
    }

    return publisher_main(domain_id, sample_count);
}

```

## 7 订阅端示例程序

```

/*****
*****订阅端程序sub.cpp*****
*****/
#include <stdio.h>
#include <stdlib.h>
#include <cstring>
#if defined(_WIN32)
#else
#include <unistd.h>
#endif

/* IDL_TypeSupport.h中包含所有依赖的头文件 */
#include "IDL_TypeSupport.h"

/* UserDataListener继承于DataReaderListener,
   需要重写其继承过来的方法on_data_available(), 在其中进行数据监听读取操作 */
class UserDataListener : public DataReaderListener {
public:
    virtual void on_data_available(DataReader* reader);
};

/* 重写继承过来的方法on_data_available(), 在其中进行数据监听读取操作 */
void UserDataListener::on_data_available(DataReader* reader)
{
    UserDataReader *UserDataReader = NULL;
    UserDataSeq data_seq;
    SampleInfoSeq info_seq;
    ReturnCode_t retcode;
    int i;

    /* 利用reader, 创建一个读取UserData类型的UserDataReader */
    UserDataReader = UserDataReader::narrow(reader);
    if (UserDataReader == NULL) {
        fprintf(stderr, "DataReader narrow error\n");
        return;
    }

    /* 获取数据, 存放至data_seq, data_seq是一个队列 */
    retcode = UserDataReader->read(
        data_seq, info_seq, DDS_LENGTH_UNLIMITED, DDS_ANY_SAMPLE_STATE,
        DDS_ANY_VIEW_STATE, DDS_ANY_INSTANCE_STATE);
    if (retcode == RETCODE_NO_DATA) {
        return;
    }
    else if (retcode != RETCODE_OK) {
        fprintf(stderr, "take error %d\n", retcode);
        return;
    }
}

```

```

        /* 打印数据 */
        /* 建议1: 避免在此进行复杂数据处理 */
        /* 建议2: 将数据传送到其他数据处理线程中进行处理 */
        /* 建议3: 假如数据结构中有string类型, 用完后需手动释放 */
        for (i = 0; i < data_seq.length(); ++i) {
            UserDataDataTypeSupport::print_data(&data_seq[i]);
        }
    }

    /* 删除所有实体 */
    static int subscriber_shutdown(
        DomainParticipant *participant)
    {
        ReturnCode_t retcode;
        int status = 0;

        if (participant != NULL) {
            retcode = participant->delete_contained_entities();
            if (retcode != RETCODE_OK) {
                fprintf(stderr, "delete_contained_entities error %d\n",
retcode);

                status = -1;
            }

            retcode =
DomainParticipantFactory::get_instance()->delete_participant(participant);
            if (retcode != RETCODE_OK) {
                fprintf(stderr, "delete_participant error %d\n",
retcode);

                status = -1;
            }
        }
        return status;
    }

    /* 订阅者函数 */
    extern "C" int subscriber_main(int domainId, int sample_count)
    {
        DomainParticipant *participant = NULL;
        Subscriber *subscriber = NULL;
        Topic *topic = NULL;
        UserDataListener *reader_listener = NULL;
        DataReader *reader = NULL;
        ReturnCode_t retcode;
        const char *type_name = NULL;
        int count = 0;
        int status = 0;

        /* 1. 创建一个participant, 可以在此处定制participant的QoS */
        /* 建议1: 在程序启动后优先创建participant, 进行资源初始化*/
    }

```

```

/* 建议2: 相同的domainId只创建一次participant, 重复创建会占用大量资源 */
participant = DomainParticipantFactory::get_instance()->create_participant(
    domainId, PARTICIPANT_QOS_DEFAULT/* participant默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (participant == NULL) {
    fprintf(stderr, "create_participant error\n");
    subscriber_shutdown(participant);
    return -1;
}

/* 2. 创建一个subscriber, 可以在创建subscriber的同时定制其QoS */
/* 建议1: 在程序启动后优先创建subscriber*/
/* 建议2: 一个participant下创建一个subscriber即可, 无需重复创建 */
subscriber = participant->create_subscriber(
    SUBSCRIBER_QOS_DEFAULT/* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (subscriber == NULL) {
    fprintf(stderr, "create_subscriber error\n");
    subscriber_shutdown(participant);
    return -1;
}

/* 3. 在创建主题之前注册数据类型 */
/* 建议1: 在程序启动后优先进行注册 */
/* 建议2: 一个数据类型注册一次即可 */
type_name = UserDataDataTypeSupport::get_type_name();
retcode = UserDataDataTypeSupport::register_type(
    participant, type_name);
if (retcode != RETCODE_OK) {
    fprintf(stderr, "register_type error %d\n", retcode);
    subscriber_shutdown(participant);
    return -1;
}

/* 4. 创建主题, 并定制主题的QoS */
/* 建议1: 在程序启动后优先创建Topic */
/* 建议2: 一种主题名创建一次即可, 无需重复创建 */
topic = participant->create_topic(
    "Example UserDataType"/* 主题名, 应与发布者主题名一致 */,
    type_name, TOPIC_QOS_DEFAULT/* 默认QoS */,
    NULL /* listener */, STATUS_MASK_NONE);
if (topic == NULL) {
    fprintf(stderr, "create_topic error\n");
    subscriber_shutdown(participant);
    return -1;
}

/* 5. 创建一个监听器 */
reader_listener = new UserDataListener();

/* 6. 创建datareader, 并定制datareader的QoS */

```



```

/* 建议1: 在程序启动后优先创建datareader*/
/* 建议2: 创建一次即可, 无需重复创建 */
/* 建议3: 在程序退出时再进行释放 */
/* 建议4: 避免打算接收数据时创建datareader, 接收数据后删除, 该做法消耗资源,
影响性能 */
reader = subscriber->create_datareader(
    topic, DATAREADER_QOS_DEFAULT/* 默认QoS */,
    reader_listener/* listener */, STATUS_MASK_ALL);
if (reader == NULL) {
    fprintf(stderr, "create_datareader error\n");
    subscriber_shutdown(participant);
    delete reader_listener;
    return -1;
}

/* 7. 主循环 , 监听器会默认调用on_data_available()监听数据 */
for (count = 0; (sample_count == 0) || (count < sample_count); ++count) {
    //保持进程一直运行
#ifdef _WIN32
        Sleep(1000); //沉睡1秒
#else
        sleep(1); //沉睡1秒
#endif
}

/* 8. 删除所有实体和监听器 */
status = subscriber_shutdown(participant);
delete reader_listener;

return status;
}

int main(int argc, char *argv[])
{
    int domain_id = 0;
    int sample_count = 0; /* 无限循环 */

    if (argc >= 2) {
        domain_id = atoi(argv[1]); /* 发送至域domain_id */
    }
    if (argc >= 3) {
        sample_count = atoi(argv[2]); /* 发送sample_count次 */
    }
    return subscriber_main(domain_id, sample_count);
}

```



# uDDS 数据分发服务软件

让智能系统的数据交互更简单

联系人：胡敬羽

联系方式：18100610350

E-mail: [hujingyu@platforu.com](mailto:hujingyu@platforu.com)

公司地址：南京市江北新区星火路 15 号智芯科技大厦 7 楼