1 Linux 环境

在 FDbus(5.4.0)安装.pdf 中，已经获得了 Linux 安装动态库和头文件的地址，文章中分别为"/usr/local/usr/lib"和"/usr/local/usr/include"，此处介绍两种方式包含(头文件)和链接(动态库)。

1.1 CMakeLists.txt

1.1.1 FDbus client demo

在 FDbus client demo 同级目录下撰写 CMakeLists.txt：

```
cmake_minimum_required(VERSION 3.10)

project(ClientDemo)

set(CXX_STANDARD 11)

aux_source_directory(${CMAKE_CURRENT_SOURCE_DIR}/ SRC)

include_directories(/usr/local/usr/include)      #指定包含之前获得的头文件目录

set(EXECUTABLE_OUTPUT_PATH ${CMAKE_CURRENT_SOURCE_DIR}/app)

link_directories(/usr/local/usr/lib)      #指定包含之前获得的动态库目录

add_executable(ClientApp ${SRC})

target_link_libraries(ClientApp pthread fdbus fdbus-clib)      #链接 fdbus 和 fdbus-clib 两个动态库
```

同级目录下新建 build 目录，进入 build，执行：

```
cmake ..
```

此时在 build 目录生成了 makefile 文件，在此目录执行 make。

回到项目根目录，可以看到多了一个 app 目录：

```
aminuos@aminuos-virtual-machine:~/桌面/fdbus demo by feishu/client$ pwd
/home/aminuos/桌面/fdbus demo by feishu/client
aminuos@aminuos-virtual-machine:~/桌面/fdbus demo by feishu/client$ ls
 app   CMakeLists.txt  'FDBus client demo'  'FDBus client demo.cpp'
```

进入 app，看到成功生成了 ClietnApp 可执行文件：

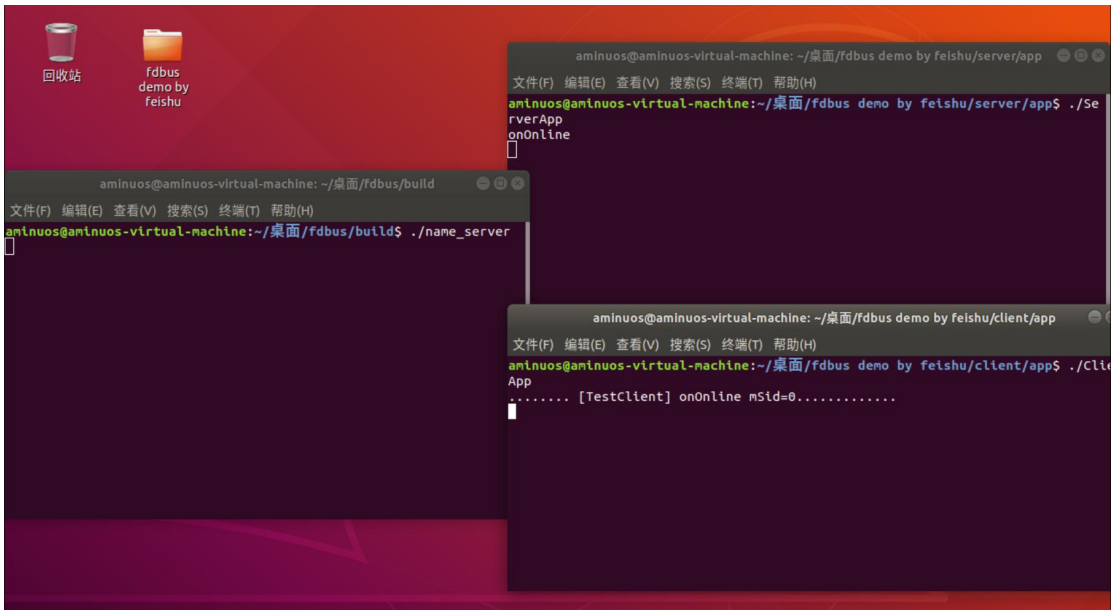
```
aminuos@aminuos-virtual-machine:~/桌面/fdbus demo by feishu/client$ cd app/
aminuos@aminuos-virtual-machine:~/桌面/fdbus demo by feishu/client/app$ ls
ClientApp
```

1.1.2 FDbus server demo

修改 project 和 executable 参数为 Server 的即可，其它此处不再赘述。

1.1.3 运行

进入之前 clone 的 fdbus 项目的 build 目录中，运行 name_server
分别运行 ServerApp 和 ClientApp
效果如图：



1.2 Visual Studio Code

安装 Visual Studio Code
打开项目目录

1.2.1 IntelliSense 配置

ctrl+shift+p，输入 C/C++，选择编辑配置(JSON)
includePath 参数添加"/usr/local/usr/include/**"

```
{
  "configurations": [
    {
      "name": "Linux",
      "includePath": [
        "${workspaceFolder}/**",
        "/usr/local/usr/include/**"
      ],
      "defines": [],
      "compilerPath": "/usr/bin/gcc",
      "cStandard": "c11",
      "cppStandard": "gnu++14",
      "intelliSenseMode": "linux-gcc-x64"
    }
  ],
  "version": 4
}
```

这样在文件编辑页面就能认出所添加的 fdbus 所需要的头文件了

1.2.1 g++调试文件配置

如图，点击调试 C/C++文件

在弹出的选择调试配置的界面选择："C/C++:g++ 生成和调试活动文件"，此时会生成一个 tasks.json，编辑它。

2.1 在 args(参数)列表添加包含库目录："-I/usr/local/usr/include"

2.2 添加链接库目录："-L/usr/local/usr/include"

2.3 添加链接库"-lpthread","-lfdbus","-lfdbus-clib"

```
{
  "tasks": [
    {
      "type": "cppbuild",
      "label": "C/C++: g++ 生成活动文件",
      "command": "/usr/bin/g++",
      "args": [
        "-fdiagnostics-color=always",
        "-g",
        "${file}",
        "-o",
        "${fileDirname}/${fileBasenameNoExtension}",
        "-I/usr/local/usr/include",
        "-L/usr/local/usr/lib",
        "-lpthread",
        "-lfdbus",
        "-lfdbus-clib"
      ],
      "options": {
        "cwd": "${fileDirname}"
      },
      "problemMatcher": [
```

1.2.2 调试

再次点击调试就可以了。

2 Windows 环境

2.1 FDBus client demo

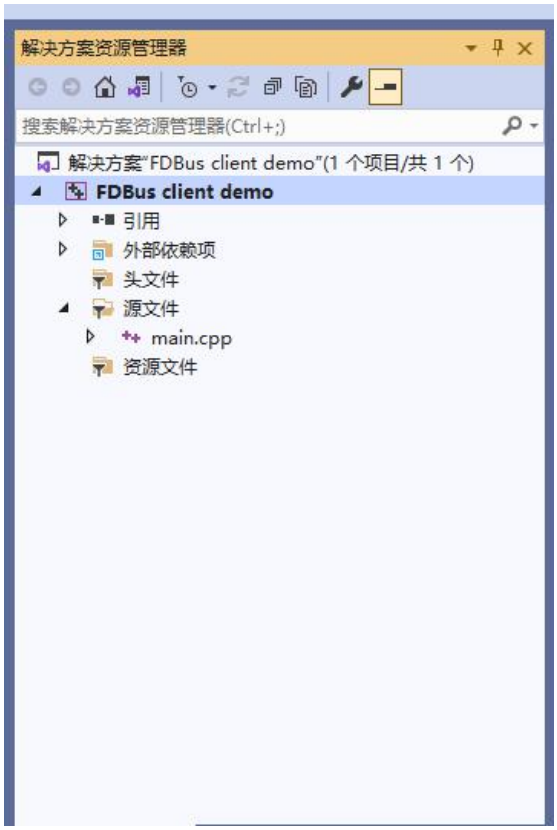
以 Microsoft Visual Studio 2019 为例

打开 Microsoft Visual Studio 2019 创建项目

加入 clientDemo 源文件

2.1.1 配置项目属性

如图，右键当前项目，打开项目属性配置



项目属性列表

配置步骤：

C/C++

1.1 常规->附加包含目录->选择之前安装 fdbus 时创建的 include 目录

1.2 常规->代码生成->运行库->选择多线程调试 (/MTd)

1.3 常规->预处理器->添加__WIN32__

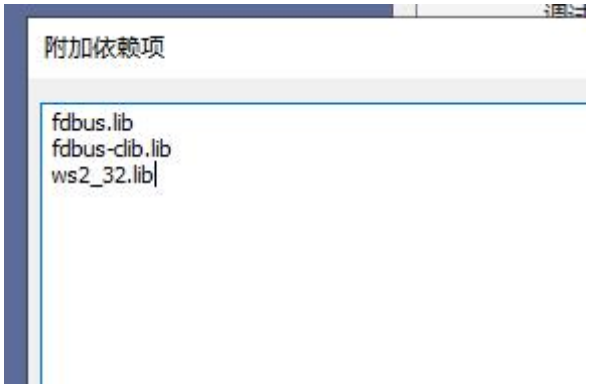
仔细观察，demo 中有

```
#ifdef __WIN32__
    ...
#endif
```

链接器

2.1 常规->附加库目录->选择之前安装 fdbus 时创建的 lib 目录

2.2 常规->输入->附加依赖项 fdbus.lib，fdbus-clib.lib，ws2_32.lib



前两个库文件为 fdbus 的动态库，由于还使用了 Windows 的 socket，所以还得添加 ws2_32.lib(为 Windows 的 api 库不用在 2.1 中的附加库目录中指出目录了)，或者在源文件添加#pragma comment(lib,"ws2_32.lib");。

2.1 FDBus server demo

新建项目，添加飞书的 serverDemo 源文件，其他同上。

2.2 运行

运行的可执行文件同 Linux。

效果如下：