win7 下使用 Eclipse 在线调试 linux 应用程序

英创公司

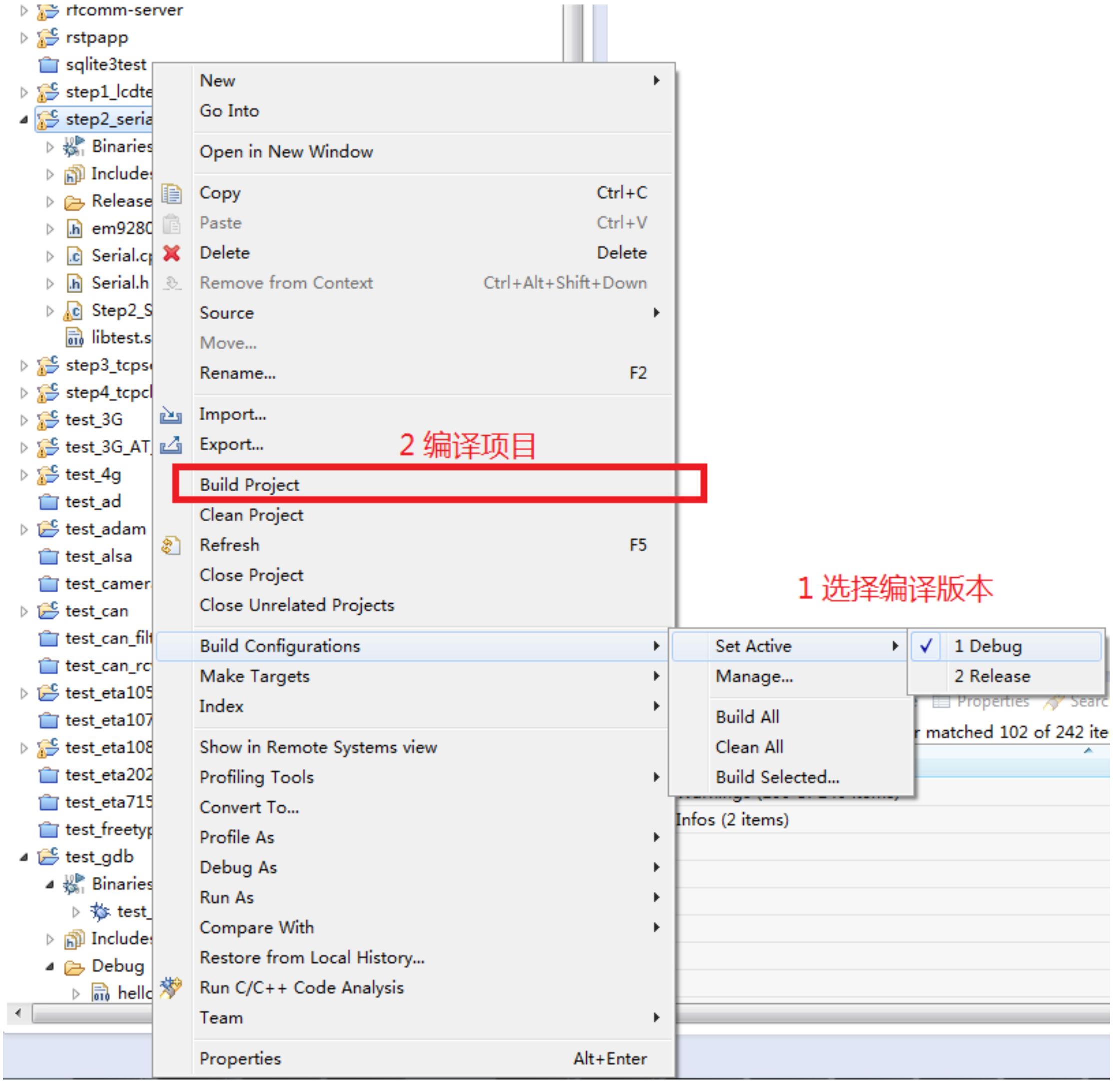
2017 年 3 月

英创公司提供的 Eclipse 是经过英创公司修改和配置的开发工具，可以在 Windows 操作系统下开发 Linux 应用程序，不需要用户编写复杂的 Makefile 文件，大大加快了应用程序的开发进度。除此之外，此 Eclipse 也提供了在线调试的功能，只要进行相应设置，就能在 Eclipse 中在线调试运行在英创主板上的 linux 应用程序。

下面在 win7 环境中，演示通过 Eclipse 在线调试 linux 应用程序，目标板是 em9287，运行的是 linux-4.1.14 操作系统，测试程序是 step2_serialtest，**目标板与 PC 在同一网络**。

一、进入调试界面的设置

1、编译 Debug 版本应用程序，放入 nfs 服务器



2 编译项目

1 选择编译版本

图 1 编译 Debug 版本的应用程序

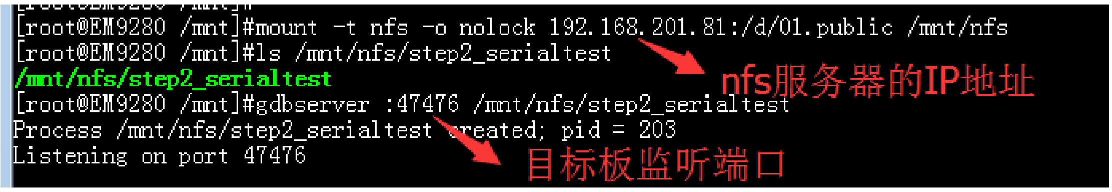
如图 1 所示，选中 step2_serialtest 项目，右键选择编译版本为 Debug，然后编译这个项目，最后将 Debug/ step2_serialtest 考备到 nfs 服务器的根目录下，本次实验使用的是 D:/01.public。

2、使用 gdbserver 运行 Debug 版程序

在目标板上挂载 nfs 服务，并使用 gdbserver 监听运行 Debug 版程序，如图 2。

```
mount -t nfs -o nolock 192.168.201.81:/d/01.public /mnt/nfs
gdbserver :47476 /mnt/nfs/step2_serialtest
```

47476 是 gdbserver 监听的端口，Eclipse 将连接到此端口



```
[root@EM9280 /mnt]#mount -t nfs -o nolock 192.168.201.81:/d/01.public /mnt/nfs
[root@EM9280 /mnt]#ls /mnt/nfs/step2_serialtest
/mnt/nfs/step2_serialtest
[root@EM9280 /mnt]#gdbserver :47476 /mnt/nfs/step2_serialtest
Process /mnt/nfs/step2_serialtest created; pid = 203
Listening on port 47476
```

图 2 目标板运行 gdbserver

3、设置 Eclipse 调试功能

Step3-1：进入 Debug Configurations

左键选中项目 step2_serialtest，在菜单栏选择 Run >> Debug Configurations... ，如图 3 所示。

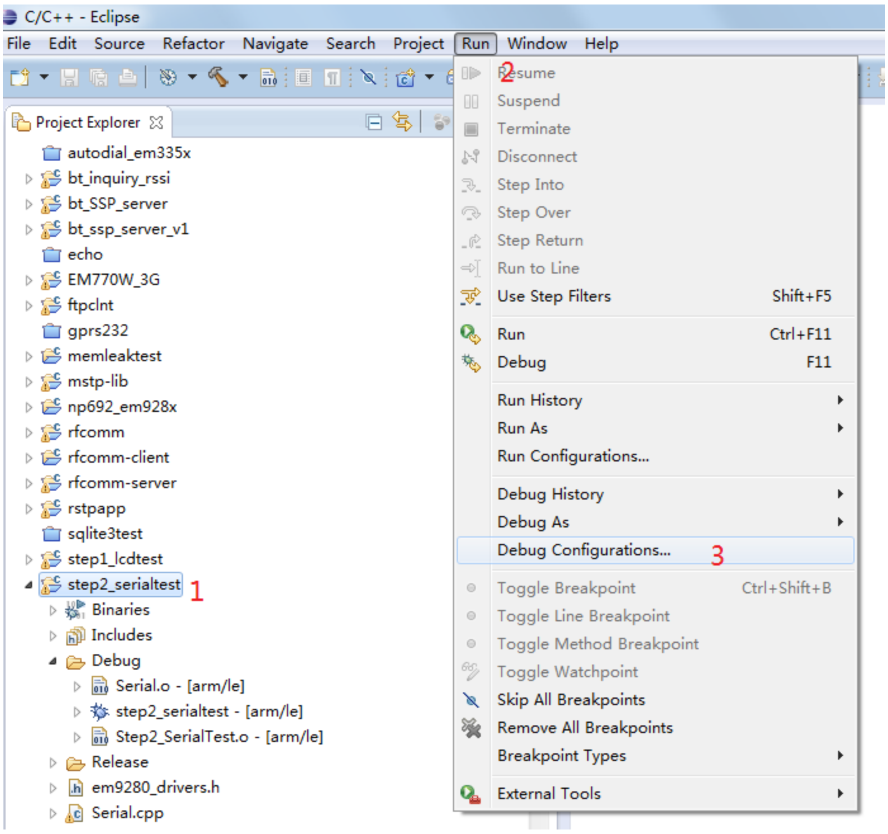
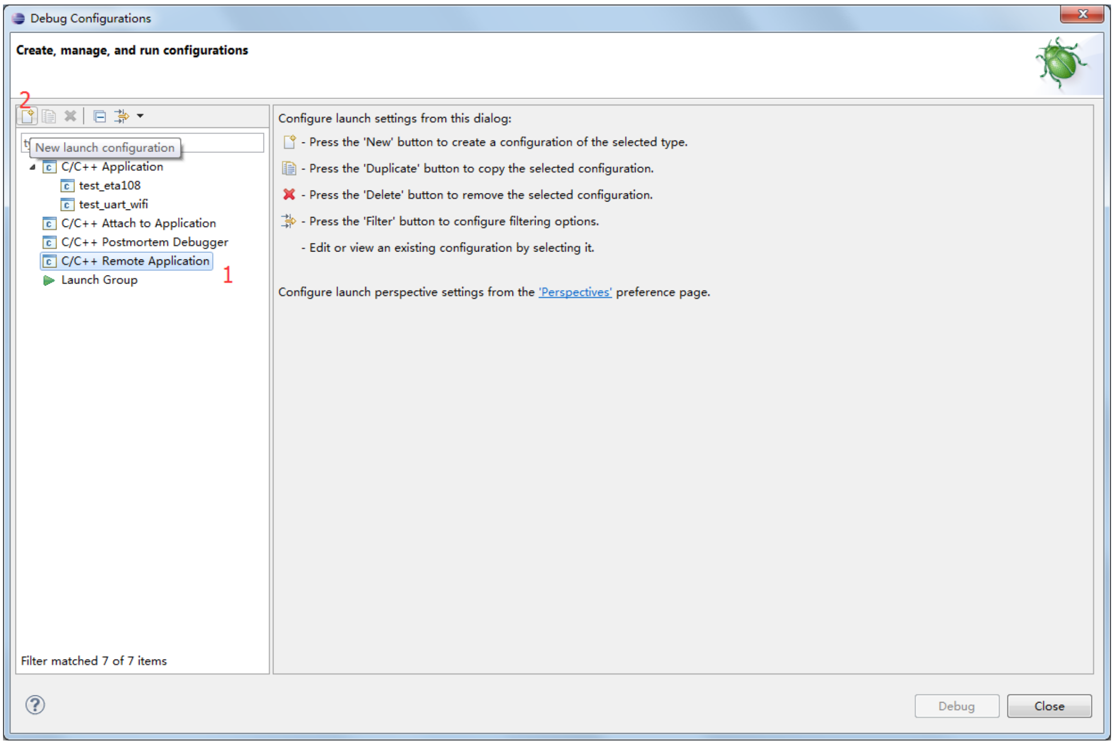


图 3 进入 Debug 配置

Step3-2：新建调试应用项目

在 Debug Configurations 对话框中选择 C/C++ Remote Application，点击左上角的新建图标，新建一个调试应用，如图 4。



Debug Configurations

Create, manage, and run configurations

New launch configuration

- C/C++ Application
  - test_eta108
  - test_uart_wifi
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- Launch Group

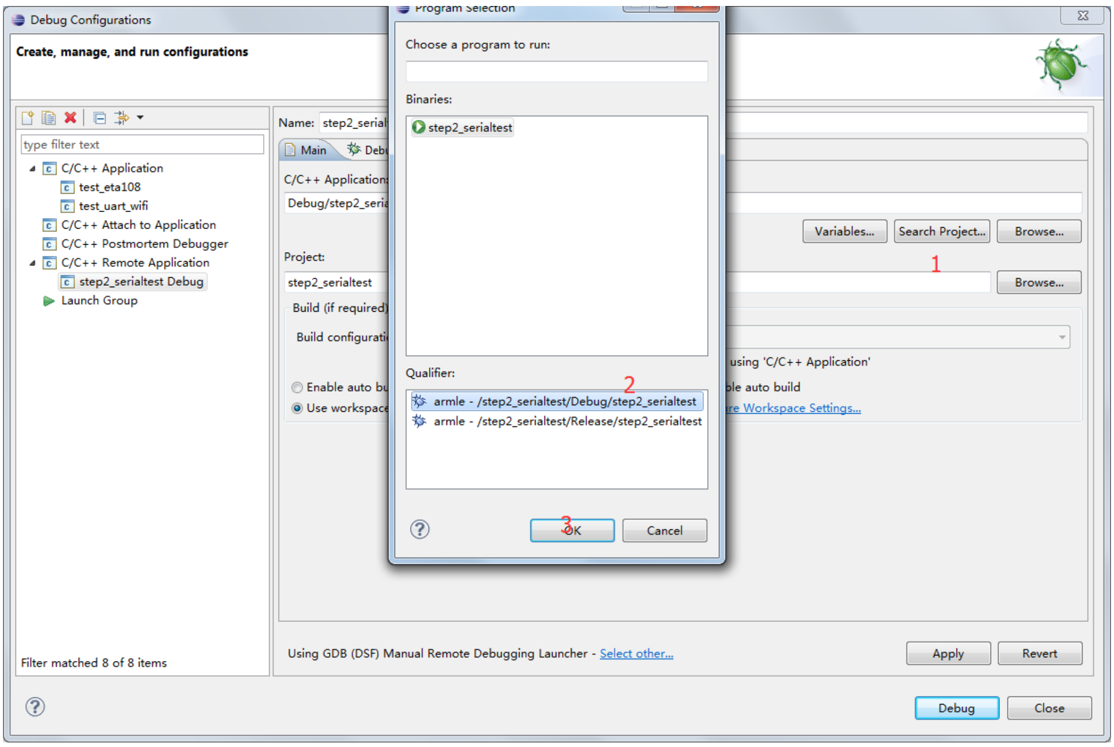
Filter matched 7 of 7 items

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

图 4 新建 C/C++远程调试项目



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
  - test_eta108
  - test_uart_wifi
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
  - step2_serialtest Debug
- Launch Group

Filter matched 8 of 8 items

Name: step2_serial

C/C++ Application:

Debug/step2_seria

Project:

step2_serialtest

Build (if required)

Build configurati

○ Enable auto bu

● Use workspace

using 'C/C++ Application'

le auto build

re Workspace Settings...

Using GDB (DSF) Manual Remote Debugging Launcher - Select other...

Program Selection

Choose a program to run:

Binaries:

step2_serialtest

Qualifier:

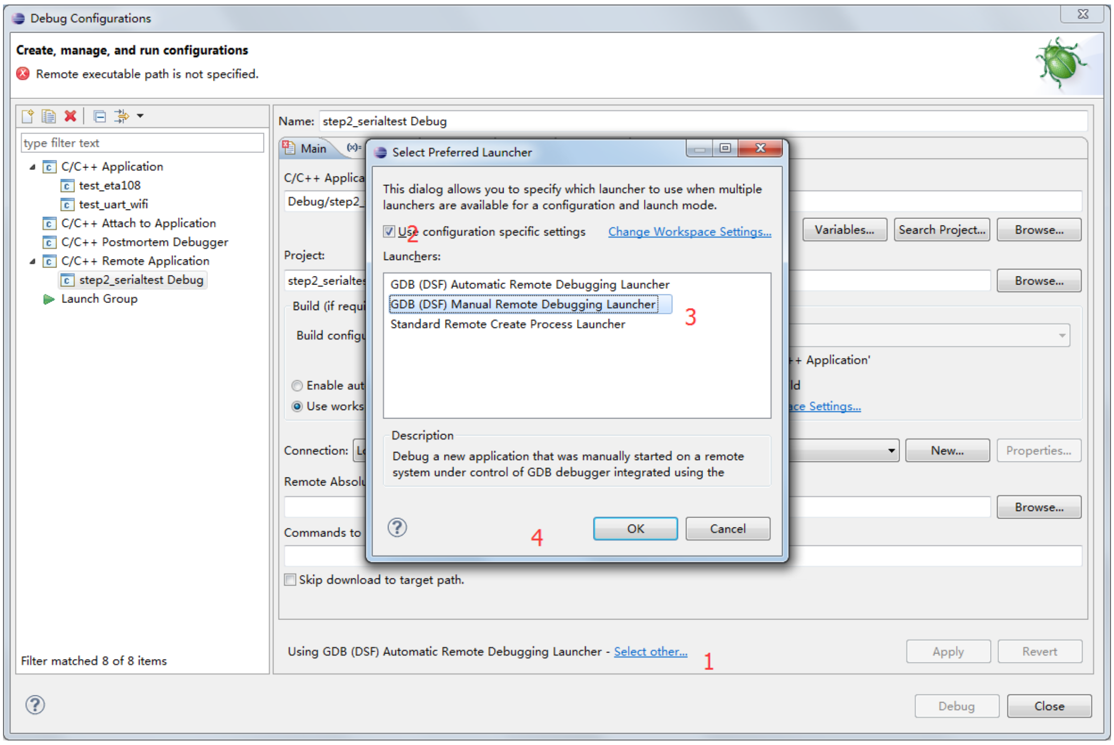
armle - /step2_serialtest/Debug/step2_serialtest

armle - /step2_serialtest/Release/step2_serialtest

图 5 选择调试的目标程序

Step3-3：设置 Main 页面

在新的对话框中的 Main 页面，Search Project，选择 Debug 版本的目标程序，点击 OK，如图 5；继续在 main 页面点击最下面的 Select other...，**在弹出的对话框中勾选 Use configuration specific settings，在 Launchers 中选择 GDB(DSF) Manual Remote Debugging Launcher**，最后选择 OK，如图 6



Debug Configurations

Create, manage, and run configurations

Remote executable path is not specified.

- C/C++ Application
  - test_eta108
  - test_uart_wifi
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
  - step2_serialtest Debug
- Launch Group

Filter matched 8 of 8 items

Name: step2_serialtest Debug

Select Preferred Launcher

This dialog allows you to specify which launcher to use when multiple launchers are available for a configuration and launch mode.

☑ Use configuration specific settings Change Workspace Settings...

Launchers:

- GDB (DSF) Automatic Remote Debugging Launcher
- GDB (DSF) Manual Remote Debugging Launcher
- Standard Remote Create Process Launcher

Description

Debug a new application that was manually started on a remote system under control of GDB debugger integrated using the

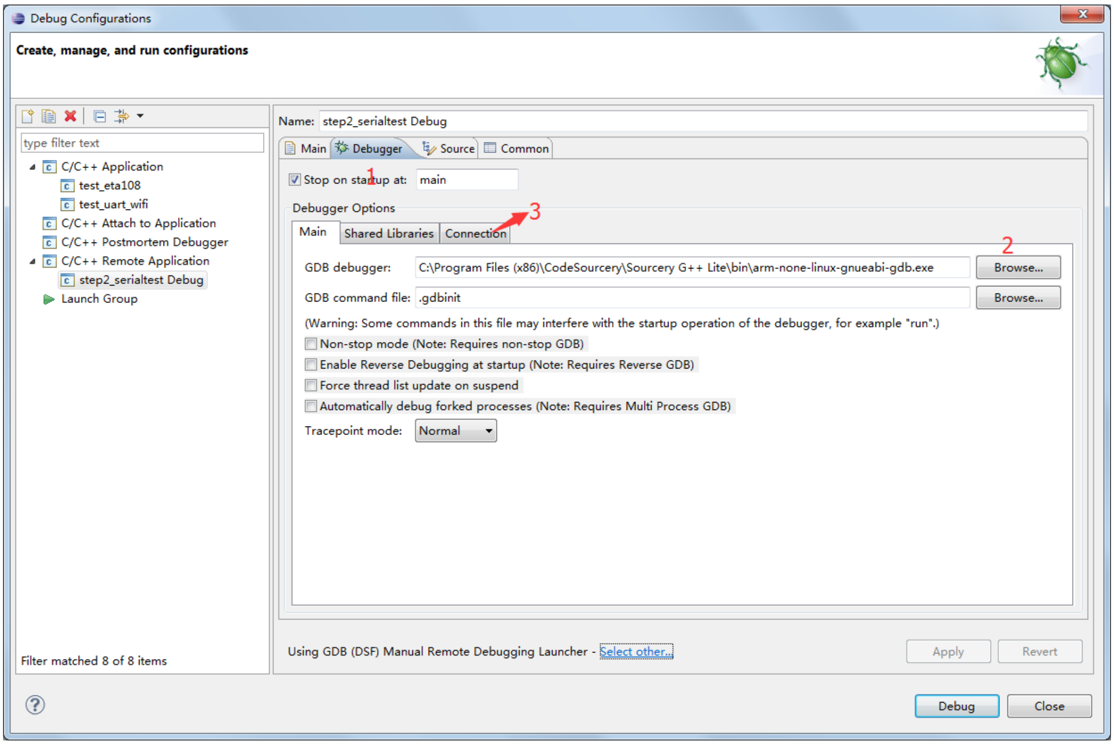
☐ Skip download to target path.

Using GDB (DSF) Automatic Remote Debugging Launcher - Select other...

图 6 选择 Launcher

Step3-4：设置 Debugger 页面

点击 Debugger 页面，在 **GDB debugger** 中通过 **Browse** 选择 **arm-none-linux-gnueabi-gdb.exe 的安装路径**，再点击 connection 继续设置，如图 7。



Name: step2_serialtest Debug

☑ Stop on startup at: main

GDB debugger: C:\Program Files (x86)\CodeSourcery\Sourcery G++ Lite\bin\arm-none-linux-gnueabi-gdb.exe

GDB command file: .gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

☐ Non-stop mode (Note: Requires non-stop GDB)

☐ Enable Reverse Debugging at startup (Note: Requires Reverse GDB)

☐ Force thread list update on suspend

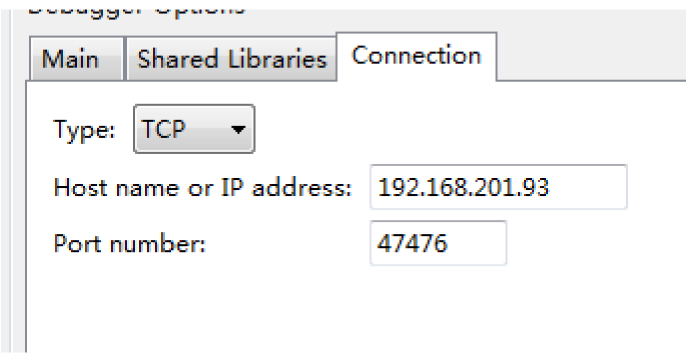
☐ Automatically debug forked processes (Note: Requires Multi Process GDB)

Tracepoint mode: Normal

Using GDB (DSF) Manual Remote Debugging Launcher - Select other...

图 7 选择 gdb 工具路径

在 Connection 中 Type：选择 TCP；Host name or IP address 和 Port number 中分别**输入目标板（EM9287）的 IP 地址和 gdbserver 监听端口号**，如图 8，最后点击 Apply。



Type: TCP

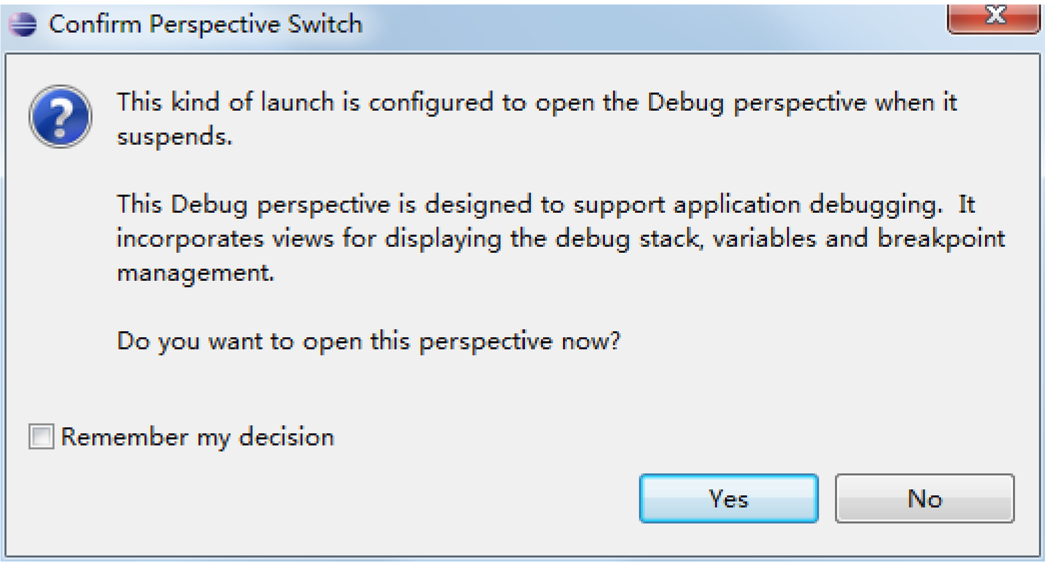
Host name or IP address: 192.168.201.93

Port number: 47476

图 8 设置与 gdbserver 建立连接的参数

4、进入调试界面及界面切换

点击对话框最下面的 Debug 开始调试，弹出的对话框选择 yes（可以勾选前面的 Remember my decision，后边将不会再弹出此对话框），进入调试界面，至此，可以进行调试工作了，如图 9 和图 10。



Confirm Perspective Switch

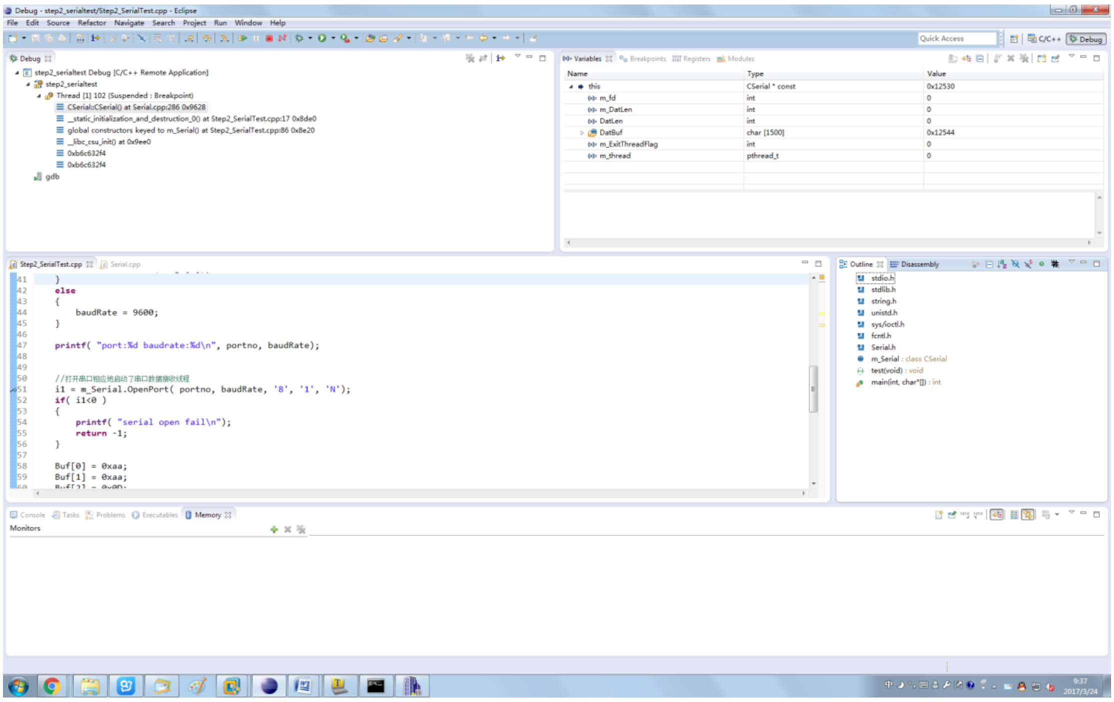
This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

□ Remember my decision

图 9 切换调试界面提示框



```
step2_serialtest Debug [C/C++ Remote Application]
    step2_serialtest
        Thread [1] 102 (Suspended : Breakpoint)
            CSerial::CSerial() at Serial.cpp:286 0x9628
            __static_initialization_and_destruction_0() at Step2_SerialTest.cpp:17 0x8de0
            global constructors keyed to m_Serial() at Step2_SerialTest.cpp:86 0x8e20
            __libc_csu_init() at 0x9ee0
            0xb6c632f4
            0xb6c632f4
    gdb
```

Name	Type	Value
this	CSerial * const	0x12530
m_fd	int	0
m_DatLen	int	0
DatLen	int	0
DatBuf	char [1500]	0x12544
m_ExitThreadFlag	int	0
m_thread	pthread_t	0

```
    }
    else
    {
        baudRate = 9600;
    }

    printf( "port:%d baudrate:%d\n", portno, baudRate);


    //打开串口相应地启动了串口数据接收线程
    i1 = m_Serial.OpenPort( portno, baudRate, '8', '1', 'N');
    if( i1<0 )
    {
        printf( "serial open fail\n");
        return -1;
    }

    Buf[0] = 0xaa;
    Buf[1] = 0xaa;
    Buf[2] = 0x0D;
```

```
stdio.h
stdlib.h
string.h
unistd.h
sys/ioctl.h
fcntl.h
Serial.h
m_Serial : class CSerial
test(void) : void
main(int, char*[]) : int
```

图 10 Eclipse 调试界面

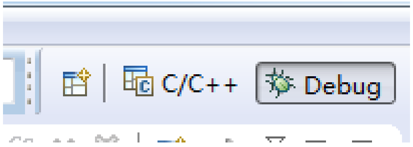


图 11 C/C++编辑界面和调试界面切换

如图 11 所示，点击右上角的 C/C++与 Debug 可以在开发编辑界面和调试界面之间切换。

5、后续使用说明

上述设置只需要一次，以后可以直接点击下图所示的小爬虫图标直接进入调试界面，小爬虫右边的倒三角可以选择要调试的项目，如图 12；或者通过 Run >> Debug History 选择要调试的项目，如图 13。

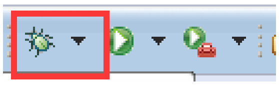
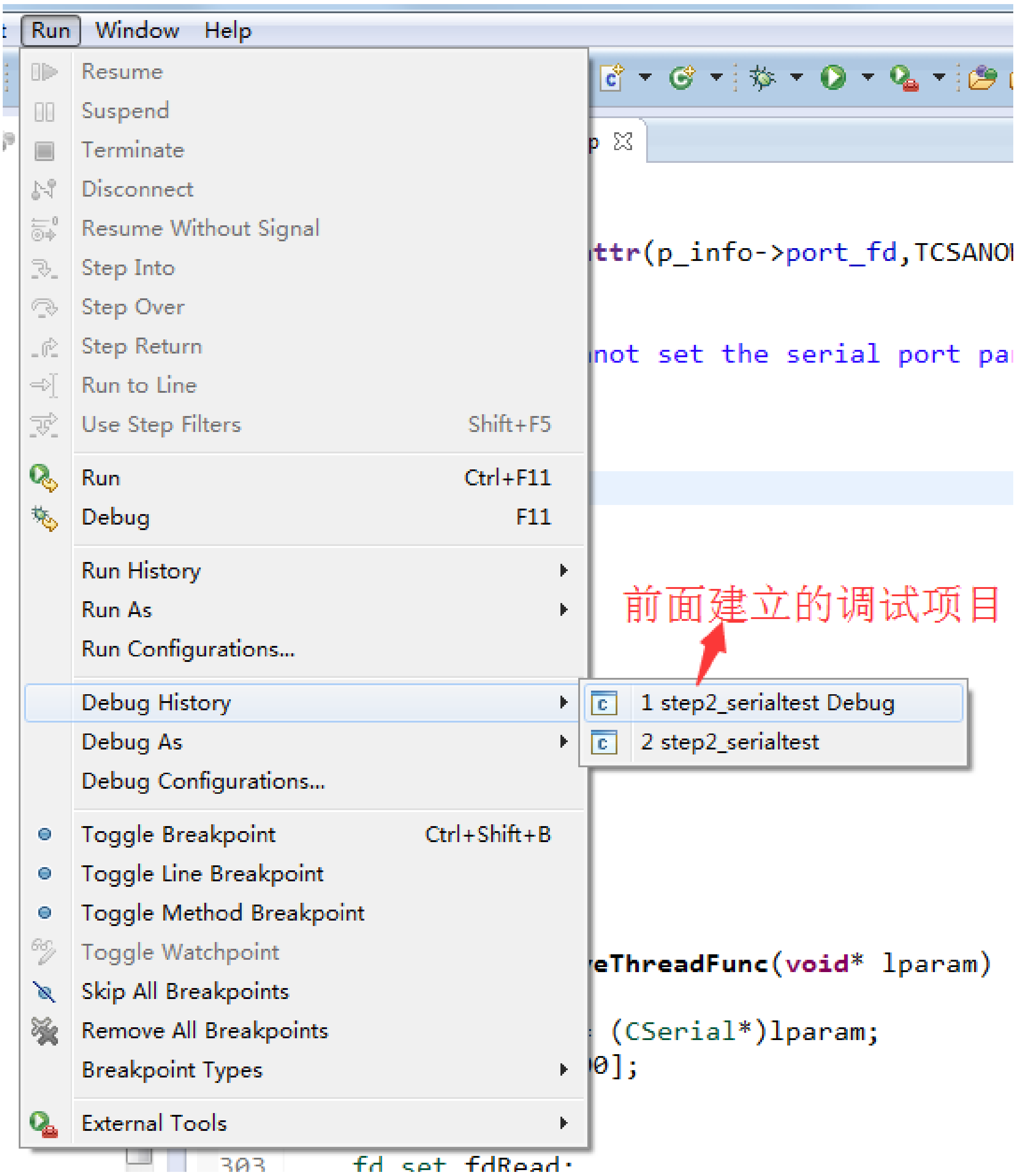


图 12 运行调试程序工具栏图标



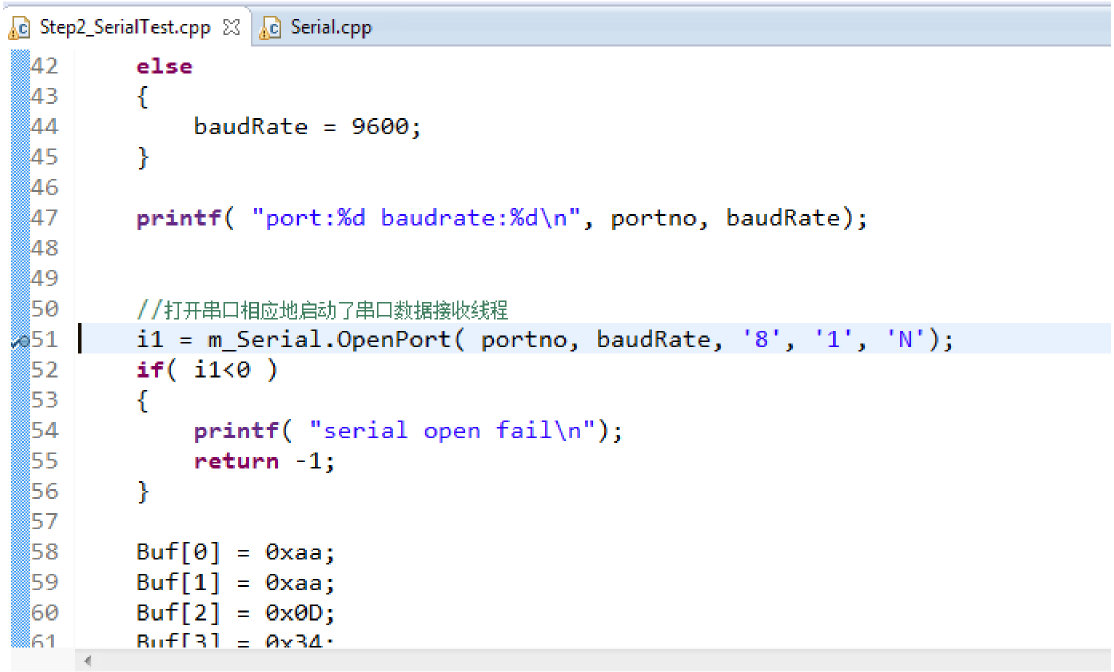
```
ttr(p_info->port_fd,TCSANO
not set the serial port pa
eThreadFunc(void* lparam)
 (CSerial*)lparam;
0];
303     fd_set fdRead;
```

图 13 菜单栏运行调试程序

二、使用 Eclipse 进行调试

1、设置/取消断点

使用 Eclipse 进行调试，与其他图形界面的集成开发环境一样，直接在代码编辑界面的左侧双击即可设置/取消断点，如图 14。

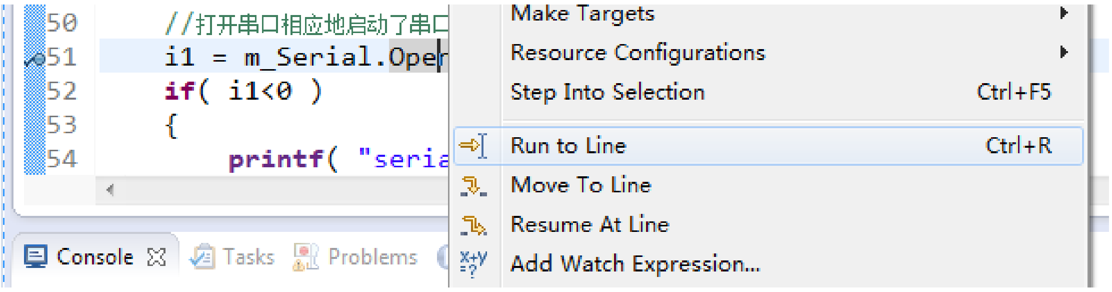


```
    else
    {
        baudRate = 9600;
    }

    printf( "port:%d baudrate:%d\n", portno, baudRate);


    //打开串口相应地启动了串口数据接收线程
    i1 = m_Serial.OpenPort( portno, baudRate, '8', '1', 'N');
    if( i1<0 )
    {
        printf( "serial open fail\n");
        return -1;
    }

    Buf[0] = 0xaa;
    Buf[1] = 0xaa;
    Buf[2] = 0x0D;
    Buf[3] = 0x34;
```

图 14 设置断点

2、单步调试及运行到光标所在行

可以在断点所在行右键选择 Run to Line（ctrl+r），直接执行到此行，如图 15。



```
    //打开串口相应地启动了串口
    i1 = m_Serial.Oper
    if( i1<0 )
    {
        printf( "seria
```

图 15 运行到光标所在行

也可以使用调试工具栏的按键（F6），一步一步执行；使用（F8）可以让程序一直运行，直到遇到断点，如图 16。

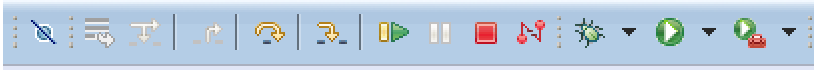
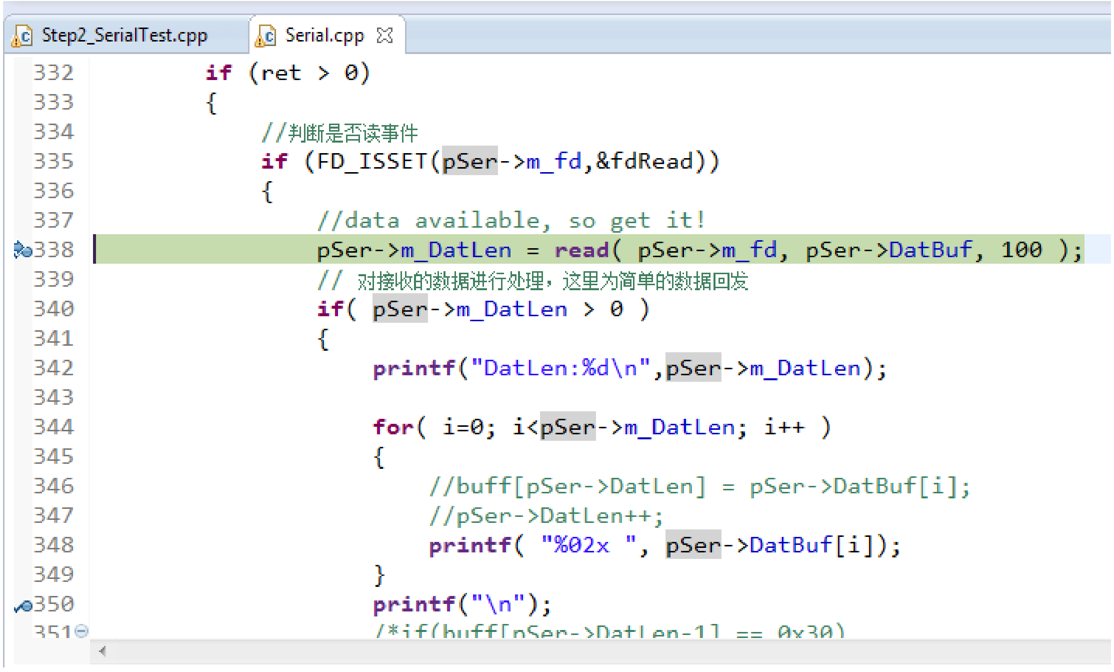


图 16 调试工具栏

3、多线程运行

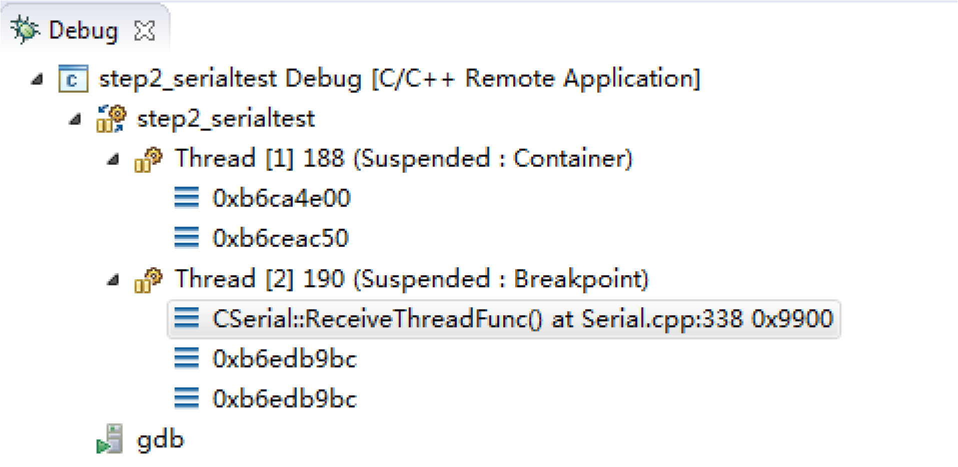
在 Serial.cpp 中 read 函数处设置断点，点击▶，让程序一直运行，此时主线程一直发送数据，而接收线程一直监听是否有数据到来，等待触发断点，如图 17。



```
        if (ret > 0)
        {
            //判断是否读事件
            if (FD_ISSET(pSer->m_fd,&fdRead))
            {
                //data available, so get it!
                pSer->m_DatLen = read( pSer->m_fd, pSer->DatBuf, 100 );
                // 对接收的数据进行处理，这里为简单的数据回发
                if( pSer->m_DatLen > 0 )
                {
                    printf("DatLen:%d\n",pSer->m_DatLen);

                    for( i=0; i<pSer->m_DatLen; i++ )
                    {
                        //buff[pSer->DatLen] = pSer->DatBuf[i];
                        //pSer->DatLen++;
                        printf( "%02x ", pSer->DatBuf[i]);
                    }
                    printf("\n");
                    /*if(buff[pSer->DatLen-1] == 0x30)
```

图 17 等待断点触发

在电脑端向 step2_serialtest 使用的串口发送数据，将会触发断点，Debug 窗口页面也可以看到目前执行到了第二个线程的 ReceiveThreadFunc()，如图 18。

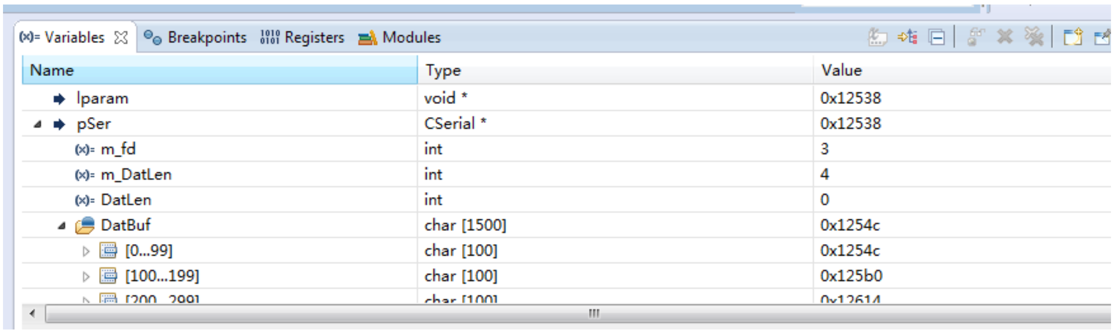


```
step2_serialtest Debug [C/C++ Remote Application]
    step2_serialtest
        Thread [1] 188 (Suspended : Container)
            0xb6ca4e00
            0xb6ceac50
        Thread [2] 190 (Suspended : Breakpoint)
            CSerial::ReceiveThreadFunc() at Serial.cpp:338 0x9900
            0xb6edb9bc
            0xb6edb9bc
    gdb
```

图 18 Debug 窗口程序运行状态

4、Variables 窗口

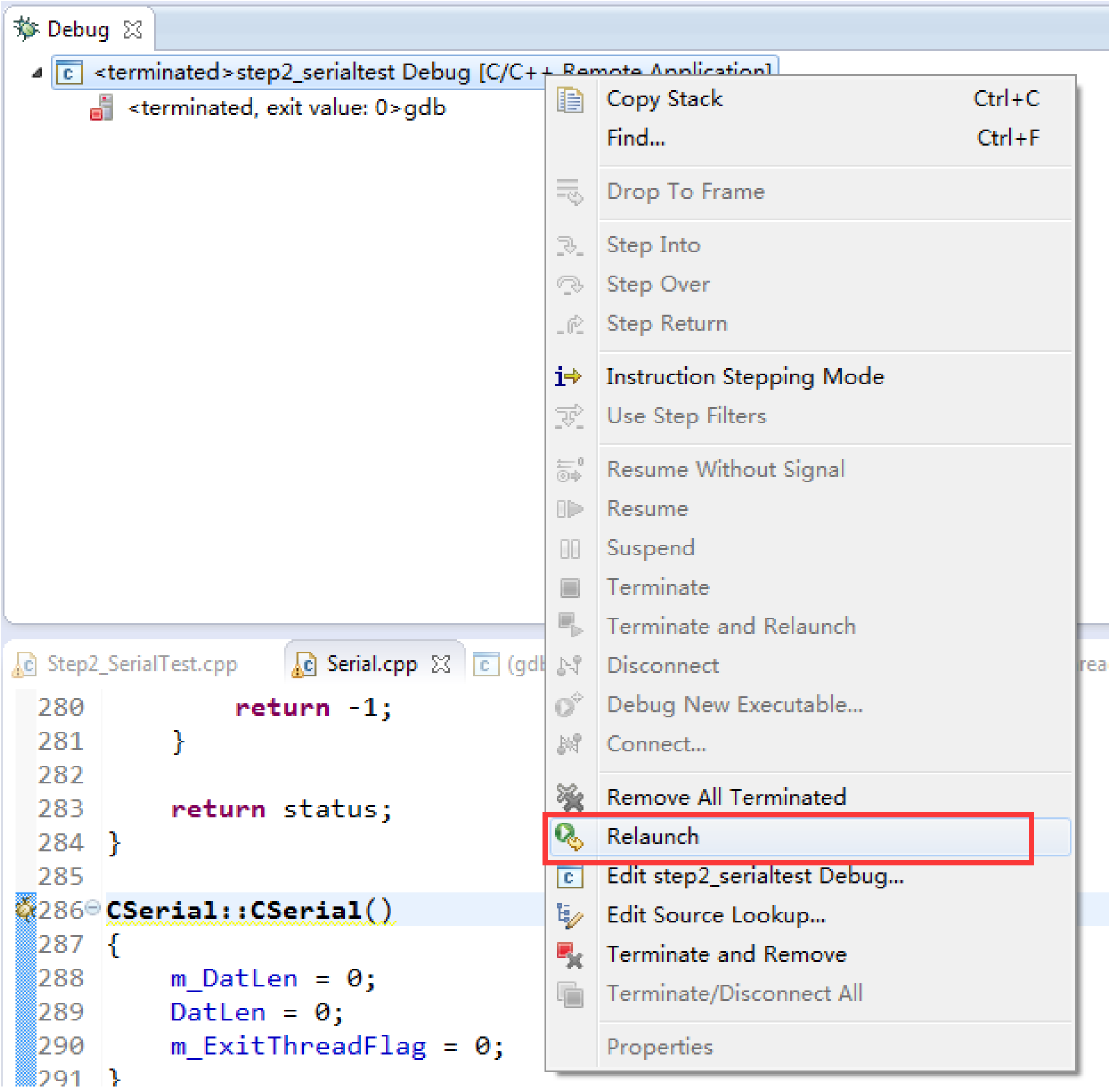
通过 Variables 窗口可以观察程序运行中的所有变量值，如图 19。



Name	Type	Value
lparam	void *	0x12538
pSer	CSerial *	0x12538
m_fd	int	3
m_DatLen	int	4
DatLen	int	0
DatBuf	char [1500]	0x1254c
[0...99]	char [100]	0x1254c
[100...199]	char [100]	0x125b0
[200...299]	char [100]	0x12614

图 19 变量观察窗口

5、停止及重启

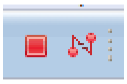


<terminated>step2_serialtest Debug [C/C++ Remote Application]

<terminated, exit value: 0>gdb

```
        return -1;
    }

    return status;
}

CSerial::CSerial()
{
    m_DatLen = 0;
    DatLen = 0;
    m_ExitThreadFlag = 0;
}
```

图 20 停止和重启调试程序

点击 ，停止 gdb 调试，目标板将退出 gdbserver。**再次调试，需要在目标板上重新运行 gdbserver**，然后 Eclipse 中可以选中 Debug 页面中的调试项目，点击鼠标右

键，选择 Relaunch 既可以重新启动调试，如图 20。