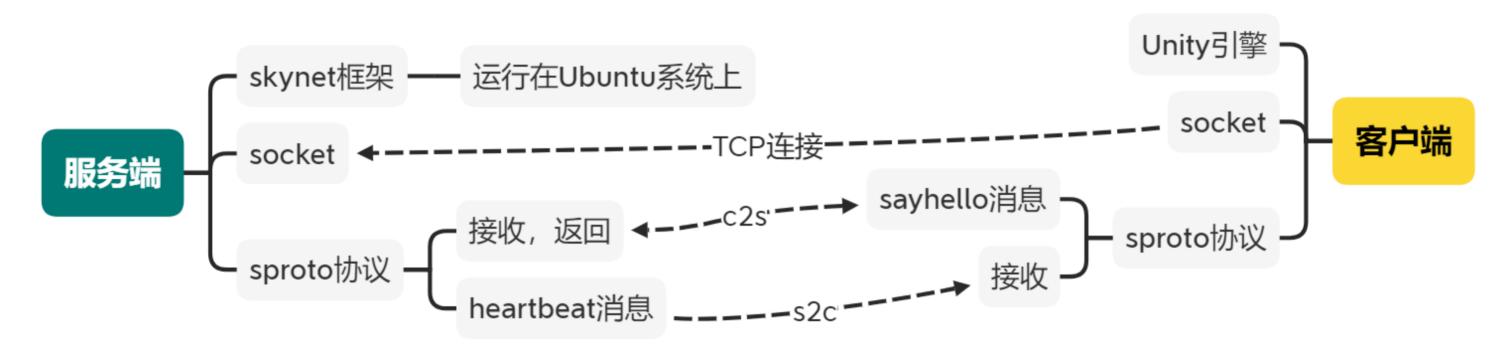
文章目录

- 一、前言
- 二、搭建Skynet服务端
- 三、Unity客户端
 - 1、创建Unity工程
 - 2、导入开源项目
 - 2.1、sproto-Csharp开源项目
 - 2.2、sprotodump开源项目
 - 2.3、sproto-Unity开源项目
 - 3、编写.sproto协议文件
 - 3.1、服务端协议文件: proto.lua
 - 3.2、客户端协议文件: game.sproto
 - 4、客户端.sproto文件转C#脚本
 - 4.1、安装lua
 - 4.2、sprotodump工具: sproto文件生成C#脚本
 - 5、客户端连接服务端
- 四、客户端与服务端通信
 - 1、客户端发消息给服务端: c2s
 - 1.1、客户端部分
 - 1.2、服务端部分
 - 1.3、运行测试
 - 2、服务端发消息给客户端: s2c
 - 2.1、服务端部分
 - 2.2、客户端部分
 - 2.3、运行测试
- 五、工程源码
- 五、完毕

一、前言

嗨,大家好,我是新发。

最近在搞服务端 Skynet 框架,今天我想写一下 Unity 通过 sproto 协议与 Skynet 服务端通信的流程,画成图是这样子:



1.html

话不多说,我们开始吧~

二、搭建Skynet服务端

关于搭建 Skynet 服务端,我前两篇文章写了教程,建议先阅读我之前这两篇文章:

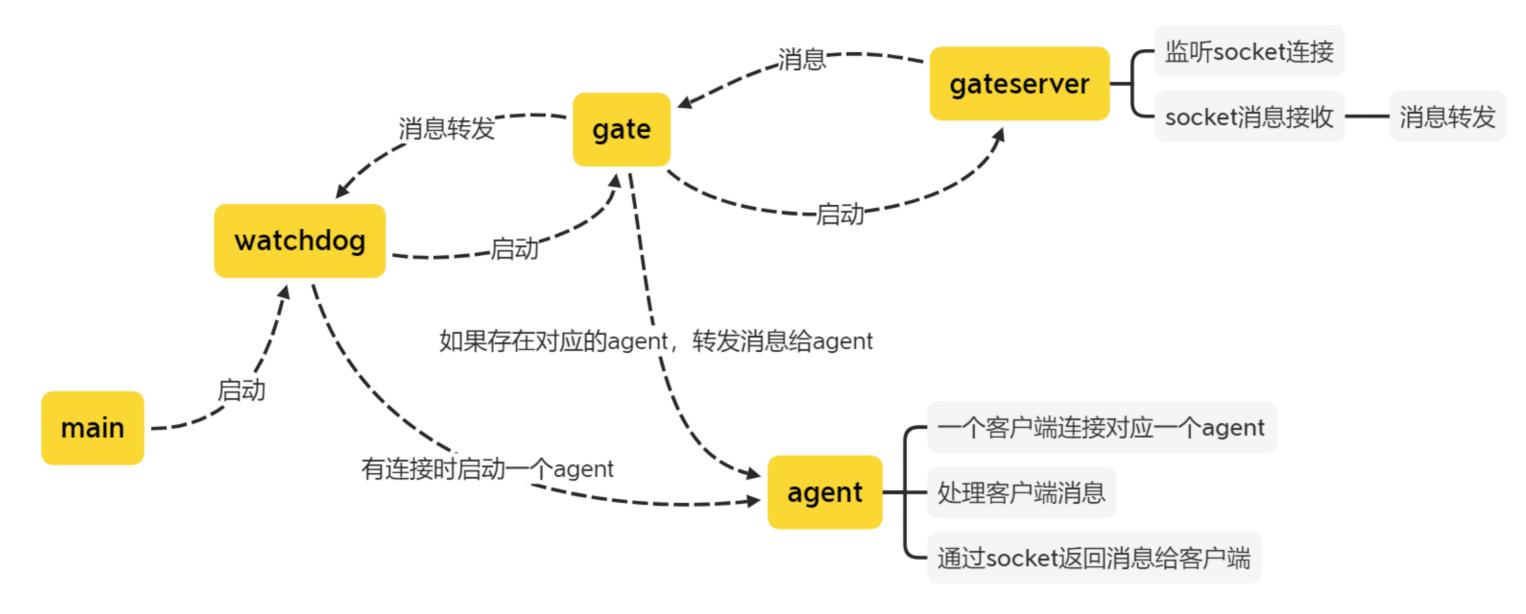
【游戏开发实战】手把手教你从零跑一个Skynet,详细教程,含案例讲解(服务端 | Skynet | Ubuntu)

【游戏开发实战】手把手教你在Windows上通过WSL运行Skynet,不用安装虚拟机,方便快捷(WSL | Linux | Ubuntu | Skynet | VSCode)

本文我就不过多赘述,搭建好环境后,运行 Skynet ,效果如下:

```
linxinfa@linxinfapc:/mnt/e/wsl/skynet$ . start.sh
[:01000002] LAUNCH snlua bootstrap
[:01000003] LAUNCH snlua launcher
[:01000004] LAUNCH snlua cmaster
 :01000004] master listen socket 0.0.0.0:2013
 :01000005] LAUNCH snlua cslave
 :01000005] slave connect to master 127.0.0.1:2013
 :01000006] LAUNCH harbor 1 16777221
 :01000004] connect from 127.0.0.1:16306 4
 :01000004] Harbor 1 (fd=4) report 127.0.0.1:2526
 :01000005] Waiting for 0 harbors
 :01000005] Shakehand ready
 :01000007] LAUNCH snlua datacenterd
 :01000008] LAUNCH snlua service_mgr
 :01000009] LAUNCH snlua main
 :01000009] Server start
 :01000009] hello skynet
 :0100000a] LAUNCH snlua protoloader
 :0100000b] LAUNCH snlua console
 :0100000c] LAUNCH snlua debug_console 8000
 :0100000c] Start debug console at 127.0.0.1:8000
 :0100000d] LAUNCH snlua simpledb
 :0100000e] LAUNCH snlua watchdog
 :0100000f] LAUNCH snlua gate
[:0100000f] Listen on 0.0.0.0:8888
 :01000009] Watchdog listen on 8888
:01000009] KILL self
 :01000002] KILL self
```

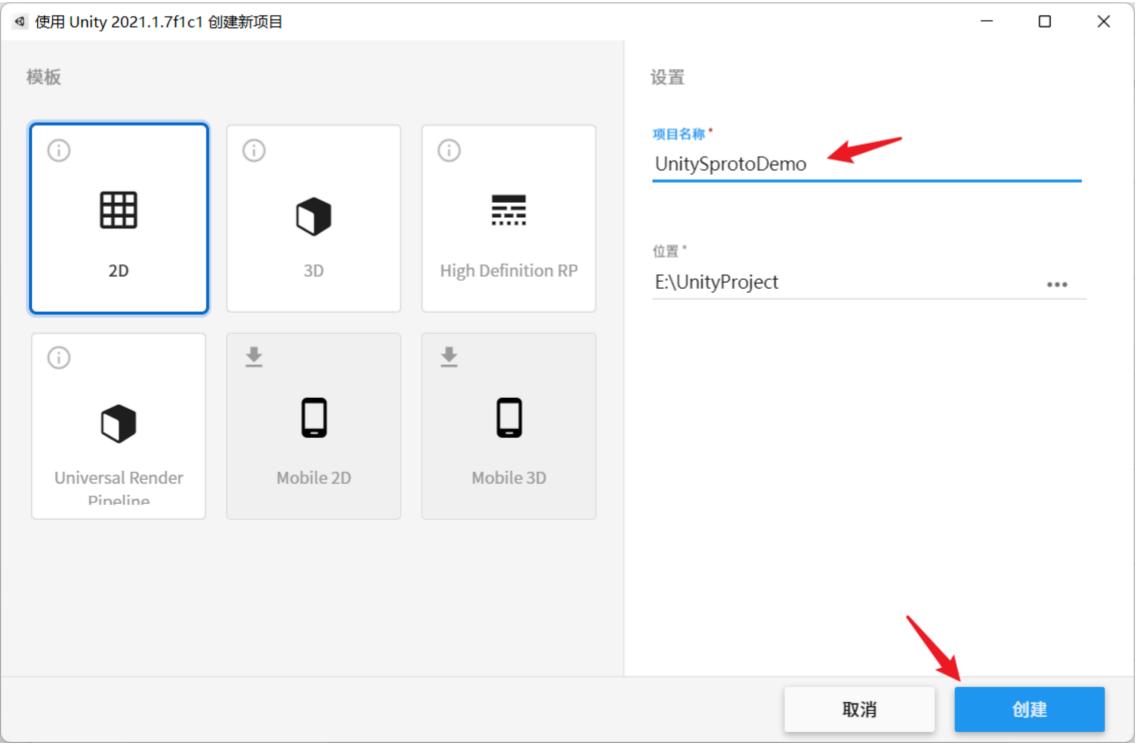
服务端模块架构如下:



三、Unity客户端

1、创建Unity工程

我使用的 Unity 版本为 Unity 2021.1.7f1c1 ,因为这里我只演示客户端与服务端通过 sproto 协议通信的流程,不需要使用 3D 相关的功能,所以我创建的是一个 2D 模板的工程,工程名叫 UnitySprotoDemo ,如下:

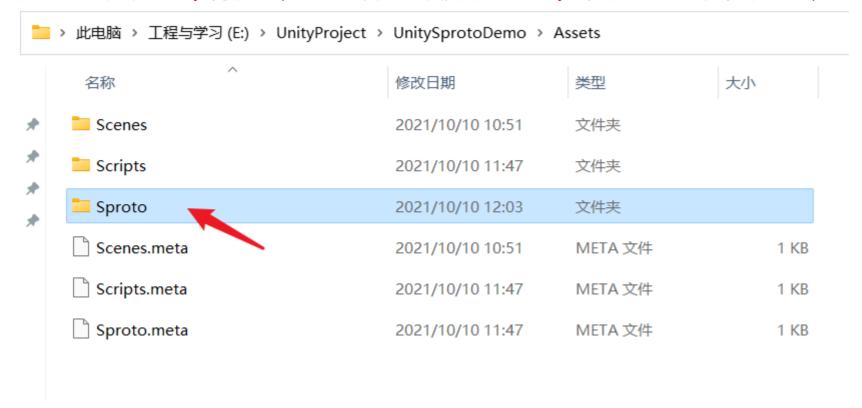


2、导入开源项目

2/13

我们要在 Unity 中使用 sproto 协议进行通信,那就需要一套 sproto 协议的 C# 实现和工具。这种嘛,首选在 GitHub 中搜索相关的开源项目,避免不必要的重复造轮子。 关于 GitHub 的使用,我之前写过一篇教程,感兴趣的同学可以看看:《GitHub使用教程与常见问题解决——上传本地工程到GitHub仓库》

我找到了一套可以在 Unity 中使用的 sproto 的 C# 实现与工具,我们先在 Unity 工程的 Assets 文件夹中新建一个 sproto 文件夹,用于存放从 GitHub 中下载下来的 sproto 开源项目,

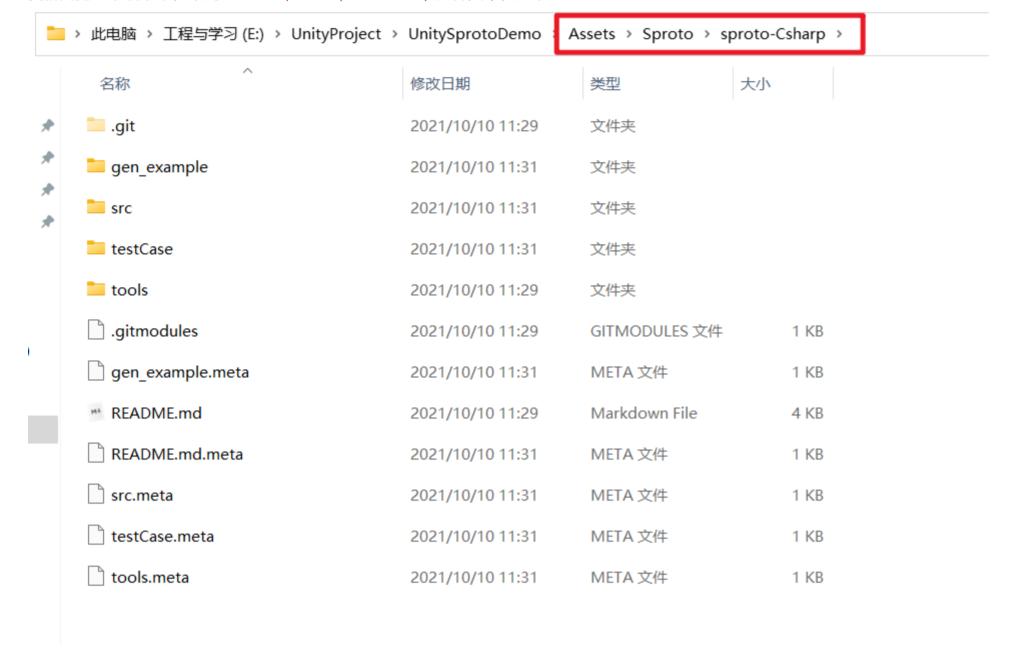


2.1、sproto-Csharp开源项目

GitHub 地址: https://github.com/lvzixun/sproto-Csharp

sproto-Csharp 是 sproto 的纯 C# 实现。

我们先把它下载下来,放到 Assets/Sproto/sproto-Csharp 文件夹中,如下:



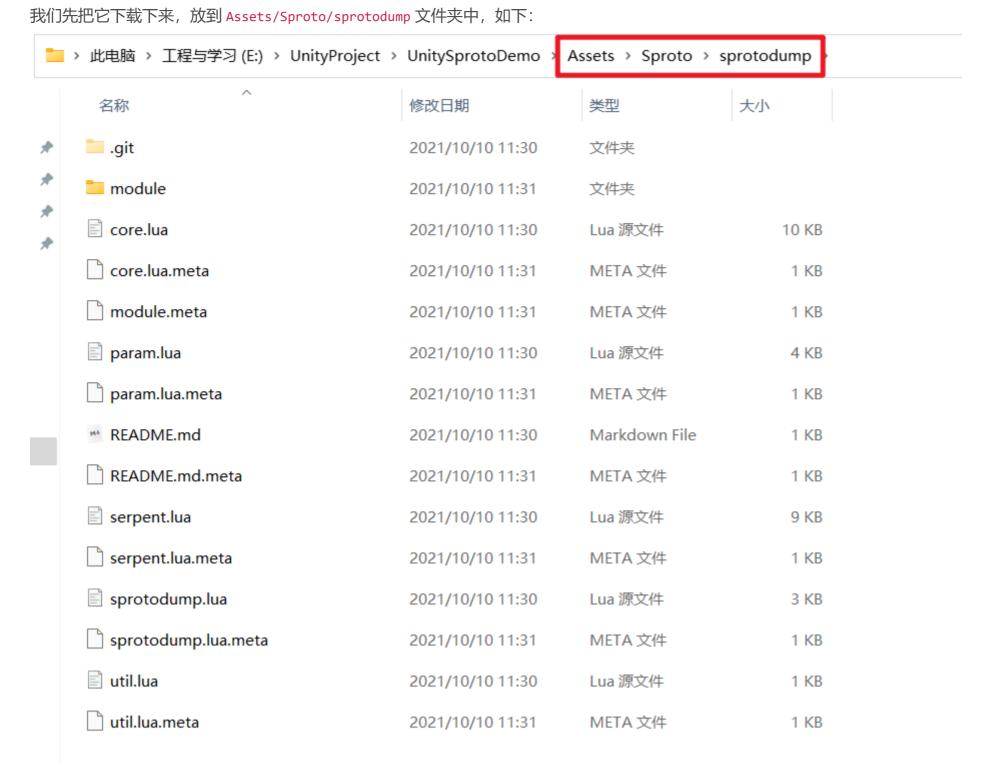
有一些测试用的代码,我们可以把它删掉,如下:

>	此电脑 > 工程与学习 (E:) > UnityProject >	UnitySprotoDemo > A	ssets > Sproto > sp	proto-Csharp
	名称	修改日期	类型	大小
•	git	2021/10/10 11:29	文件夹	
	gen_example	2021/10/10 11:31	文件夹	
	src	2021/10/10 11:31	文件夹	
	testCase	2021/10/10 11:31	文件夹	
	tools	2021/10/10 11:29	文件夹	
	gitmodules	2021/10/10 11:29	GITMODULES 文件	1 KB
	🗋 gen_example.meta 删除这	内介又件 买	META 文件	1 KB
	** README.md	2021/10/10 11:29	Markdown File	4 KB
	README.md.meta	2021/10/10 11:31	META 文件	1 KB
	src.meta	2021/10/10 11:31	META 文件	1 KB
	testCase.meta	2021/10/10 11:31	META 文件	1 KB
	tools.meta	2021/10/10 11:31	META 文件	1 KB

2.2、sprotodump开源项目

GitHub 地址: https://hub.fastgit.org/lvzixun/sprotodump

sprotodump 是将 .sproto 文件转为 .cs 、 .spb 、 .spb 、 .go 、 .md 、 .lua 等文件的工具,下文我会讲如何使用这个工具。

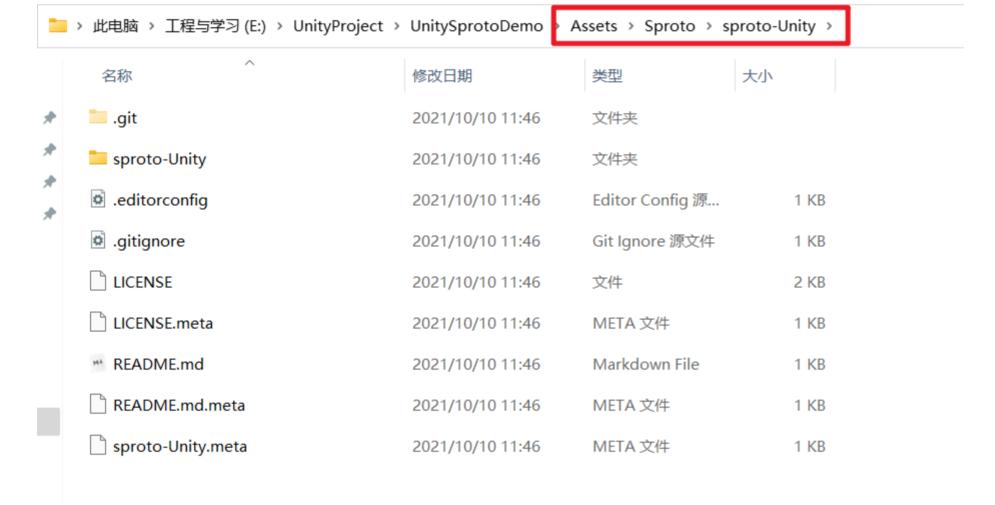


2.3、sproto-Unity开源项目

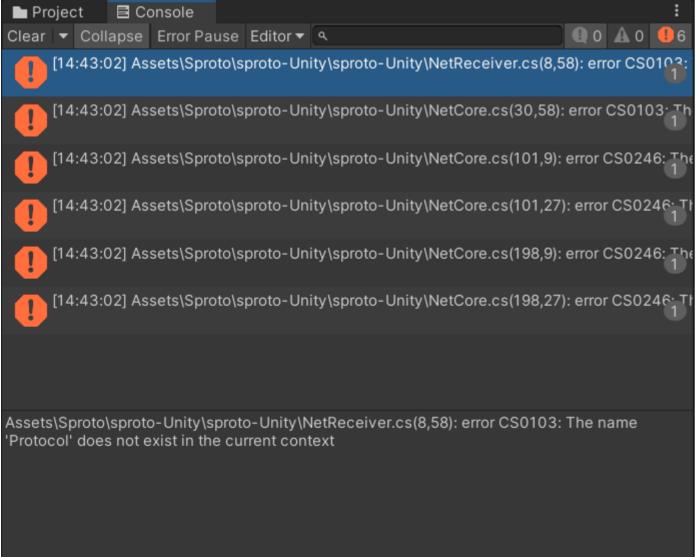
GitHub 地址: https://github.com/m2q1n9/sproto-Unity

sproto-Unity 封装了三个类: NetCore 、 NetSender 、 NetReceiver ,下文我会讲下如何使用。

我们先把它下载下来,放到 Assets/Sproto/sproto-Unity 文件夹中,如下:



此时工程会报错,



这是因为我们刚刚删除了 sproto-Csharp 的一些代码导致的,不用担心,我们等下执行 .sproto 生成 cs 就自动解决了。

3、编写.sproto协议文件

现在我们来写协议文件,包括服务端和客户端。

3.1、服务端协议文件: proto.lua

skynet 框架中的 examples 里,已经为我们准备好了一个协议文件: proto.lua ,

```
💕 proto.lua 🛛 🗡
 EXPLORER
                                                      skynet > examples > 💣 proto.lua
                                       中の甘む
✓ SERVER
                                                              local sprotoparser = require "sprotoparser"
  cluster1.lua
  cluster2.lua
                                                              local proto = {}
  clustername.lua
                                                             proto.c2s = sprotoparser.parse [[
   ≡ config
                                                              .package {
   ≡ config_log
                                                                  type 0 : integer
   ≡ config.c1
                                                                  session 1 : integer
   ≡ config.c2
   ≡ config.login
   ≡ config.mc
                                                             handshake 1 {
                                                        11
                                                                  response {
   ≡ config.mongodb
                                                        12
                                                                      msg 0 : string
                                                        13
   14
   ≡ config.path
                                                        15
   ≡ config.userlog
  globallog.lua
                                                        17
                                                             get 2 {
  injectlaunch.lua
                                                                  request {
                                                        18
  main_log.lua
                                                                      what 0 : string
                                                        19
  main_mongodb.lua
                                                        21
                                                                  response {
  main_mysql.lua
                                                                      result 0 : string
                                                        22
  💕 main.lua
  greload.lua
  proto.lua
                                                        25
  protoloader.lua
                                                              set 3 {
  share.lua
                                                                  request {
                                                                      what 0 : string
  simpledb.lua
                                                                      value 1 : string
                                                        29
  simplemonitor.lua
                                                        30
  simpleweb.lua
                                                        31
  simplewebsocket.lua
                                                        32
```

1.html

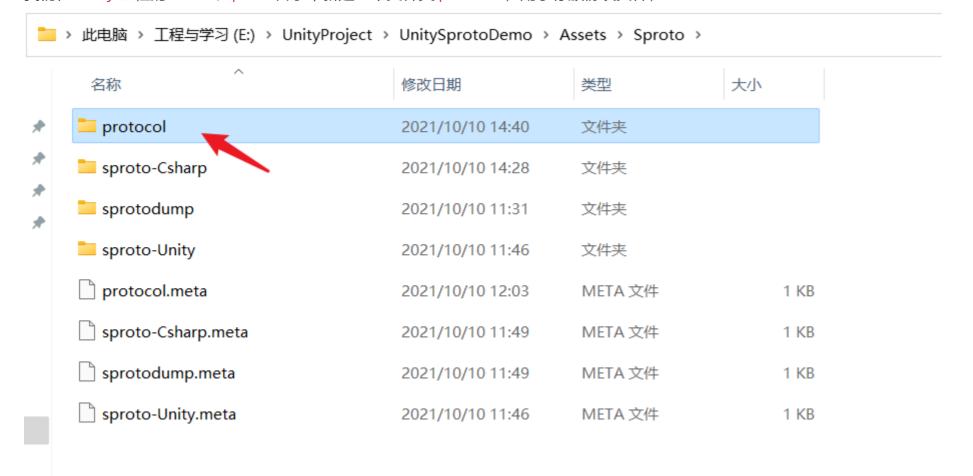
我们改一下, 把不需要的协议删掉, 最终如下:

```
1 | local sprotoparser = require "sprotoparser"
    local proto = {}
    proto.c2s = sprotoparser.parse [[
    .package {
        type 0 : integer
        session 1 : integer
 8
 9
    sayhello 1 {
10
        request {
11
          what 0 : string
12
13
        response {
          error_code 0 : integer
14
          msg 1 : string
15
16
17
18
19
    proto.s2c = sprotoparser.parse [[
20
     .package {
21
        type 0 : integer
        session 1 : integer
22
23
24
    heartbeat 2 {}
    ]]
25
26
    return proto
27
28
29
30
31
```

注:由于客户端的 sproto 工具的不能支持 c2s 和 s2c 的协议使用相同的 tag ,比如 sayhello 消息的 tag 是 1 ,那么 heartbeat 消息就不可以使用 1 作为 tag ,这里我是使用 2 作为 heartbeat 的 tag 。

3.2、客户端协议文件: game.sproto

我们在 Unity 工程的 Assets/Sproto 目录中新建一个文件夹 protocol ,用于存放协议文件,



在 protocol 文件夹中新建一个 game.proto 文件,内容如下:

```
.package {
      type ₀ : integer
      session 1 : integer
 4
 5
    sayhello 1 {
      request {
        what ₀ : string
 8
 9
      response {
10
        error_code 0 : integer
11
        msg 1 : string
12
13
14
15
    heartbeat 2 {}
16
```

4、客户端.sproto文件转C#脚本

接下来,我们要使用 sprotodump 将客户端的 game.proto 文件转成 C# 脚本。因为 sprotodump 需要是用 lua 来执行,所以这里我们需要先安装 lua 环境。

4.1、安装lua

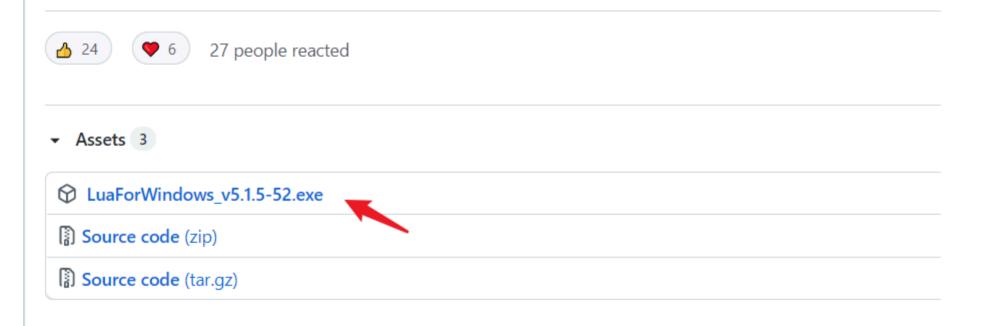
lua 官网: http://www.lua.org/

lua Windows版: https://github.com/rjpcomputing/luaforwindows/releases

Lua for Windows v5.1.5-52 Released

absjw released this on 31 Jan 2018

Preventing download of vcredist to solve installation issues.



我们下载下来然后安装即可,完整完毕后,打开终端,执行 lua -v , 如果能输出版本号,则说明 lua 环境弄好了。

```
Windows PowerShell

Windows PowerShell

版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell, 了解新功能和改进! https://aka.ms/PSWindows

PS E:\UnityProject\UnitySprotoDemo\Assets\Sproto\protocol> lua -v

Lua 5.1.5 Copyright (C) 1994-2012 Lua.org, PUC-Rio

PS E:\UnityProject\UnitySprotoDemo\Assets\Sproto\protocol> |
```

4.2、sprotodump工具: sproto文件生成C#脚本

sprotodump 是将 .sproto 文件转为 .cs 、 .spb 、 .spb 、 .go 、 .md 、 .lua 等文件的工具,我们这里是要把 game .proto 文件转成 C# 脚本。 sprotodump 的使用方法如下:

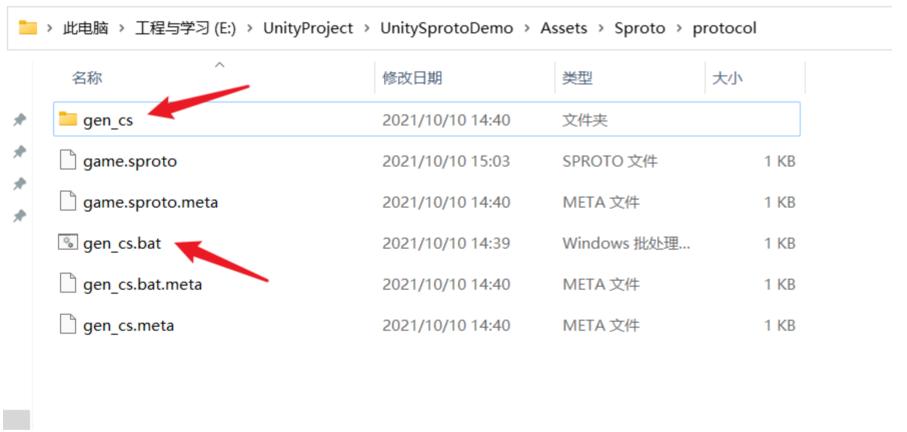
```
usage: lua sprotodump.lua <option> <sproto_file1 sproto_file2 ...> [[<out_option> <outfile>] ...] [namespace_option]
2
 3
        option:
                             dump to cSharp code file
                            dump to binary spb file
            -spb
                             dump to go code file
            -go
                            dump to markdown file
            -md
                             dump to lua table
            -lua
 9
10
        out_option:
            -d <dircetory>
                                         dump to speciffic dircetory
11
12
            -o <file>
                                         dump to speciffic file
13
                                         set package name(only cSharp code use)
            -p <package name>
14
15
        namespace_option:
16
            -namespace
                             add namespace to type and protocol
```

例:

1 | lua lua sprotodump.lua -cs game.sproto -o gamesproto.cs

我们把命令写到 bat 脚本中,双击执行即可,提高工作效率。

在 Assets/Sproto/protocol 目录中创建 gen_cs.bat 文件,再创建一个 gen_cs 文件夹用于存放生成的 C# 脚本。



gen_cs.bat 脚本内容如下,我写了详细注释,大家应该能看懂,

pause

11

```
1 echo off
2 :: 当前路径
3 set curdir=%~dp0
4 :: 进入sprotodump目录
5 cd /d %curdir%/../sprotodump
6 :: 将.sproto文件转为C#脚本,存放在gen_cs文件夹中
7 lua ./sprotodump.lua -cs %curdir%/game.sproto -o %curdir%/gen_cs/gamesproto.cs
8 :: 输出完成
9 echo sproto to cs, done
10 :: 按任意键退出
```

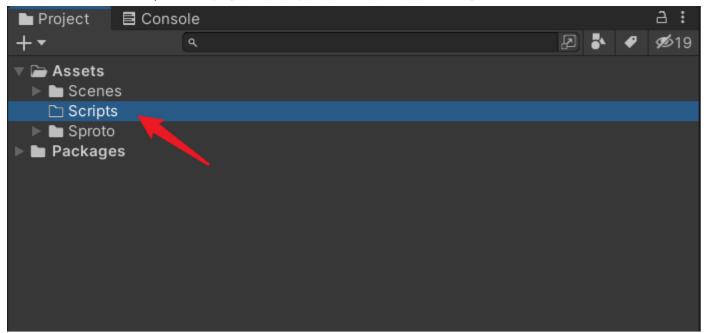
我们执行 gen cs.bat 脚本时,可能会报错:

```
PS E:\UnityProject\UnitySprotoDemo\Assets\Sproto\protocol> .\gen_cs.bat
 E:\UnityProject\UnitySprotoDemo\Assets\Sproto\protocol>echo off
 lua: ./sprotodump.lua:9: nesting of [[...]] is deprecated near '['
 sproto to cs, done
 请按任意键继续...
我们打开 sprotodump.lua 脚本,把 README 后面的 [[和]]改成 [=[和]=]即可,
       local README
       sprotodump is a simple tool to convert sproto file to spb binary.
       usage: lua sprotodump.lua <option> <sproto_file1 sproto_file2 ...> [[<out_option> <outfile>] ...] [namespace_option]
  10
  11
          option:
                              dump to cSharp code file
  12
               -cs
                              dump to binary spb file
  13
               -spb
                              dump to go code file
  14
               -go
               -md
                              dump to markdown file
  15
              -lua
                              dump to lua table
  17
  18
          out_option:
                                          dump to speciffic directory
              -d <dircetory>
  19
               -o <file>
                                          dump to speciffic file
              -p <package name>
                                          set package name(only cSharp code use)
  21
  22
          namespace option:
  23
  24
               -namespace
                              add namespace to type and protocol
        ]]
  25
  27
如下:
      local README = [=[
  6
      sprotodump is a simple tool to convert sproto file to spb binary.
      usage: lua sprotodump.lua <option> <sproto_file1 sproto_file2 ...> [[<out_option> <outfile>] ...] [namespace_option]
          option:
 11
                              dump to cSharp code file
 12
              -cs
                              dump to binary spb file
 13
              -spb
                              dump to go code file
 14
              -go
                              dump to markdown file
 15
              -md
              -lua
                              dump to lua table
 16
          out_option:
 18
                                         dump to speciffic dircetory
              -d <dircetory>
 19
              -o ⟨file⟩
                                         dump to speciffic file
                                         set package name(only cSharp code use)
              -p <package name>
 21
 22
 23
          namespace option:
              -namespace
                              add namespace to type and protocol
 24
 25
 26
重新执行 gen_cs.bat 脚本,
 PS E:\UnityProject\UnitySprotoDemo\Assets\Sproto\protocol> .\gen_cs.bat
 E:\UnityProject\UnitySprotoDemo\Assets\Sproto\protocol>echo off
 dump to E:\UnityProject\UnitySprotoDemo\Assets\Sproto\protocol\/gen_cs/gamesproto.cs file
 sproto to cs, done
 请按任意键继续...
生成成功,可以看到 gen_cs 文件夹中生成了 gamesproto.cs 脚本,
  🚞 > 此电脑 > 工程与学习 (E:) > UnityProject > UnitySprotoDemo > Assets > Sproto > protocol > gen_cs
       名称
                                     修改日期
                                                                     大小
      gamesproto.cs
                                     2021/10/10 15:23
                                                       C# 源文件
                                                                          5 KB
```

1.html

5、客户端连接服务端

我们先新建一个 Scripts 文件夹,用于存放我们写的游戏逻辑脚本,



在 Scripts 文件夹中创建一个 Main.cs 脚本,作为入口脚本,在 Start 函数中做一些初始化操作,并在 Update 中驱动 NetCore 的消息分发,

```
// Main.cs
    using UnityEngine;
    public class Main : MonoBehaviour
        void Start()
 8
           // 初始化
 9
           NetCore.Init();
10
            NetSender.Init();
11
            NetReceiver.Init();
12
            NetCore.enabled = true;
13
14
15
        void Update()
16
17
            // 驱动消息分发
18
            NetCore.Dispatch();
19
20
21
```

接着,我们就可以连接服务端了,由于是本地连接,所以 IP 地址使用 127.0.0.1 即可,端口的话,我们可以看到服务端监听的端口是 8888,

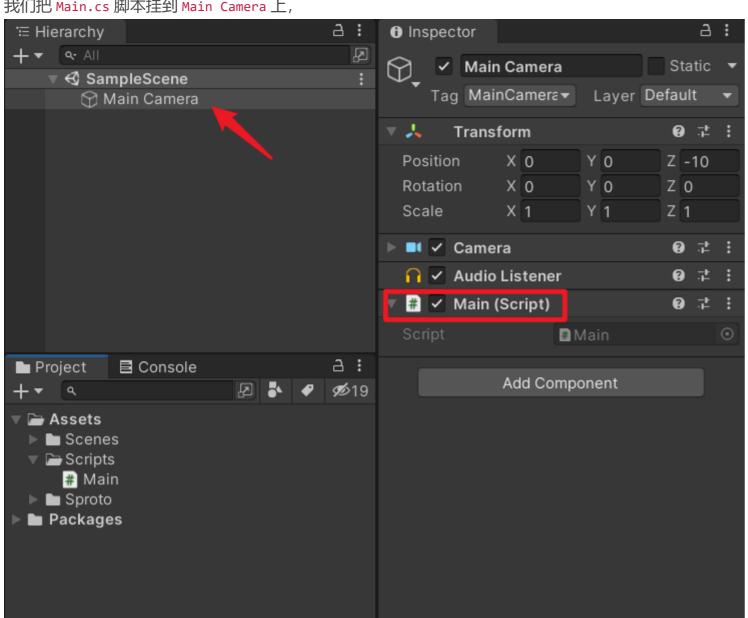
```
TERMINAL
linxinfa@LAPTOP-KD7VD0IS:~$ . cd_skynet.sh
linxinfa@LAPTOP-KD7VD0IS:/mnt/e/UnityProject/UnitySprotoDemo/Server/skynet$ . start.sh
[:01000002] LAUNCH snlua bootstrap
 :01000003] LAUNCH snlua launcher
 :01000004] LAUNCH snlua cmaster
 :01000004] master listen socket 0.0.0.0:2013
 :01000005] LAUNCH snlua cslave
 :01000005] slave connect to master 127.0.0.1:2013
 :01000004] connect from 127.0.0.1:56848 4
 :01000006] LAUNCH harbor 1 16777221
 :01000004] Harbor 1 (fd=4) report 127.0.0.1:2526
 :01000005] Waiting for 0 harbors
 :01000005] Shakehand ready
 :01000007] LAUNCH snlua datacenterd
 :01000008] LAUNCH snlua service mgr
 :01000009] LAUNCH snlua main
 :01000009] Server start
 :0100000a] LAUNCH snlua protoloader
 :0100000b] LAUNCH snlua console
:0100000c] LAUNCH snlua debug_console 8000
 :0100000c] Start debug console at 127.0.0.1:8000
 :0100000d] LAUNCH snlua simpledb
:0100000e] LAUNCH snlua watchdog
:0100000f] LAUNCH snlua gate
 :0100000f] Listen on 0.0.0.0:8888
 :01000009] Watchdog listen on 8888
:01000009] KILL self
:01000002] KILL self
```

客户端连接服务端代码如下:

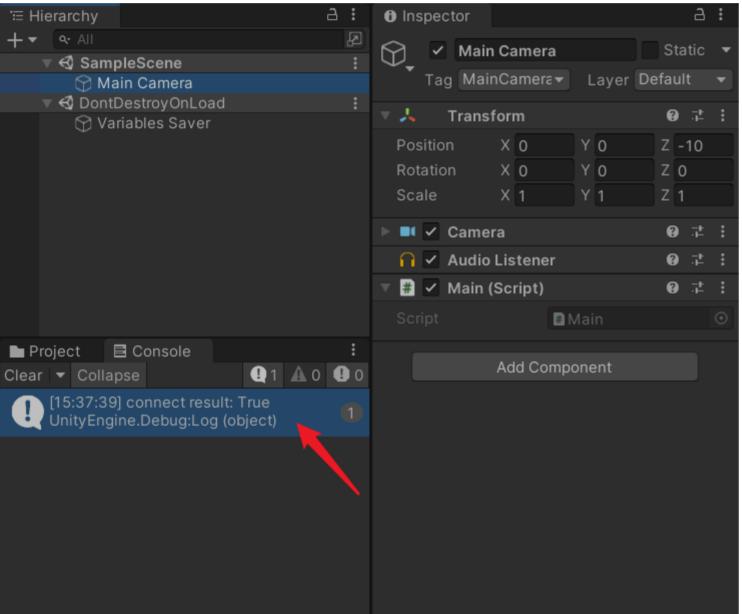
```
1 // Main.cs
2
    void Start()
4
       // ...
6
       // 连接服务端
7
       NetCore.Connect("127.0.0.1", 8888, () =>
8
9
           // 连接结果
10
           Debug.Log("connect result: " + NetCore.connected);
11
       });
12
13
```

1.html

我们把 Main.cs 脚本挂到 Main Camera 上,



运行 Unity,可以看到输出了 connect result: True,说明连接服务器成功了,



此时服务端也输出了相关日志,

```
:01000009] LAUNCH snlua main
:01000009] Server start
:0100000a] LAUNCH snlua protoloader
:0100000b] LAUNCH snlua console
:0100000c] LAUNCH snlua debug console 8000
:0100000c] Start debug console at 127.0.0.1:8000
:0100000d] LAUNCH snlua simpledb
:0100000e] LAUNCH snlua watchdog
:0100000f] LAUNCH snlua gate
:0100000f] Listen on 0.0.0.0:8888
:01000009] Watchdog listen on 8888
:01000009] KILL self
                                                         收到客户端的连接
:01000002] KILL self
:0100000e] New client from : 127.0.0.1:40340
:01000010] LAUNCH snlua agent
                                                     Pagent
:01000010] TRACE nil
:01000010] <TRACE :01000010-1> 9345113416500 trace
:01000010] <TRACE :01000010-1> 9345113425700 call : @./examples/agent.lua:87 @./examples/agent.lua:99 @.
:0100000f] <TRACE :01000010-1> 9345113558900 request
:0100000f] <TRACE :01000010-1> 9345113583900 response
:0100000f] <TRACE :01000010-1> 9345113586600 end
:01000010] <TRACE :01000010-1> 9345113611500 resume
:01000010] <TRACE :01000010-1> 9345113614200 response
:01000010] <TRACE :01000010-1> 9345113615500 end
```

四、客户端与服务端通信

1、客户端发消息给服务端: c2s

1.1、客户端部分

现在,我们让客户端给服务端发一条 sayhello 消息,

```
1 // Main.cs
   void Start()
       // ...
5
       NetCore.Connect("127.0.0.1", 8888, () =>
```

```
2024/5/16 11:41
                                                                                                      1.html
    8
               Debug.Log("connect result: " + NetCore.connected);
    9
               if (NetCore.connected)
   10
   11
                   // 给服务端发送一条sayhello消息
   12
                   SendSayHello();
   13
   14
           });
   15
   16
       void SendSayHello()
   17
   18
           var req = new SprotoType.sayhello.request();
   19
           req.what = "Hi, I am Unity!";
   20
           Debug.Log("发送sayhello消息给服务端");
   21
           NetSender.Send<Protocol.sayhello>(req, (data) =>
   22
   23
               var rsp = data as SprotoType.sayhello.response;
   24
               Debug.LogFormat("服务端sayhello返回, error_code: {0}, msg: {1}", rsp.error_code, rsp.msg);
   25
           });
   26
   27
```

1.2、服务端部分

我们给服务端的 agent.lua 脚本添加 sayhello 的响应,

```
function REQUEST:sayhello()
print("recv client sayhello: ", self.what)
return { error_code = 0, msg = "Hi, I am Skynet!" }
end
```

如下:

```
    ■ agent.lua M X
 EXPLORER
                                          skynet > examples > © agent.lua
✓ SERVER
                                                 local skynet = require "skynet"
 skynet
                                                  local socket = require "skynet.socket"
  > 3rd
                                                  local sproto = require "sproto"
  > cservice
                                                  local sprotoloader = require "sprotoloader"
  examples
   > login
                                                  local WATCHDOG
   abort.lua
                                                  local host
                                                 local send_request
   g agent.lua
                                    М
   checkdeadloop.lua
                                                  local CMD = \{\}
   client.lua
                                            11
                                                  local REQUEST = {}
   cluster1.lua
                                                 local client_fd
                                            12
   cluster2.lua
                                            13
   clustername.lua
                                                 function REQUEST:sayhello()
                                            14
                                                      print("recv client sayhello: ", self.what)
   ≡ config
                                            15
                                                      return { error_code = 0, msg = "Hi, I am Skynet!" }
                                            16
   ≡ config_log
                                            17
   ≡ config.c1
                                            18
   ≡ config.c2
                                                 local function request(name, args, response)
                                            19
   ≡ config.login
                                                      local f = assert(REQUEST[name])
   ≡ config.mc
                                                      local r = f(args)
                                            21
                                                      if response then
   ≡ config.mongodb
                                            22
                                                          return response(r)
                                            23
   ≡ config.mysql
   ≡ config.path
```

1.3、运行测试

我们重新启动服务端,然后运行客户端,可以看到客户端发送了 sayhello 消息,并受到了服务端的返回,

我们看服务端的日志,也输出了相关日志,说明正常接收到了客户端的消息了,

```
:0100000d] LAUNCH snlua simpledb
 :0100000e] LAUNCH snlua watchdog
[:0100000f] LAUNCH snlua gate
:0100000f] Listen on 0.0.0.0:8888
:01000009] Watchdog listen on 8888
:01000009] KILL self
:01000002] KILL self
 :0100000e] New client from : 127.0.0.1:40352
:01000010] LAUNCH snlua agent
:01000010] TRACE nil
:01000010] <TRACE :01000010-1> 10396761805500 trace
:01000010] <TRACE :01000010-1> 10396761818200 call : @./examples/agent.lua:73 @./ex
:0100000f] <TRACE :01000010-1> 10396761915700 request
:0100000f] <TRACE :01000010-1> 10396761947700 response
recv client sayhello: Hi, I am Unity!
[:0100000f] <TRACE :01000010-1> 10396761951200 end
:01000010] <TRACE :01000010-1> 10396761982600 resume
:01000010] <TRACE :01000010-1> 10396761985800 response
:01000010] <TRACE :01000010-1> 10396761987700 end
:01000010] TRACE nil
:01000010] <TRACE :01000010-2> 10396762072700 trace
 :01000010] <TRACE :01000010-2> 10396762154400 end
```

1.html

2、服务端发消息给客户端: s2c

服务端每隔 5秒 给客户端发送一条 heartbeat 消息,消息定义如下:

```
1 heartbeat 2 {}
```

2.1、服务端部分

如下,服务端受到客户端连接后,会循环每隔 5秒 给客户端发送一条 heartbeat 消息。

```
function CMD.start(conf)
         local fd = conf.client
        local gate = conf.gate
60
        WATCHDOG = conf.watchdog
62
         host = sprotoloader.load(1):host "package"
         send_request = host:attach(sprotoloader.load(2))
64
         skynet.fork(function()
             while true do
66
                 send_package(send_request("heartbeat"))
67
                 skynet.sleep(500)
         end)
71
         client_fd = fd
72
         skynet.call(gate, "lua", "forward", fd)
```

注意:因为 heartbeat 消息没有参数,所以这里不用传其他参数,如果 heartbeat 的消息定义中有参数,比如这样

```
1 heartbeat 2 {
       request {
3
         cnt 0 : integer
4
5 }
```

那么发消息时传 cnt 参数是这样子的:

```
1 | send_package(send_request("heartbeat", { cnt = 666 }))
```

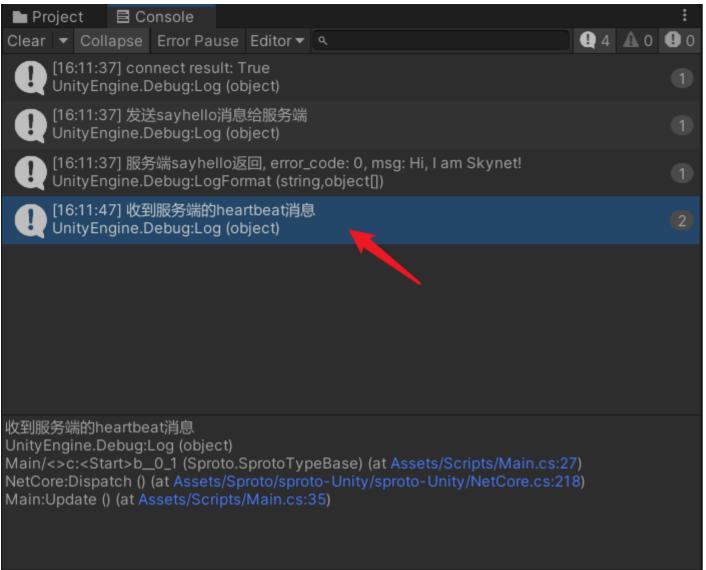
2.2、客户端部分

客户端部分需要通过 NetReceiver 注册消息的响应函数,如下:

```
1 NetReceiver.AddHandler<Protocol.heartbeat>((data) =>
2
      Debug.Log("收到服务端的heartbeat消息");
3
       return null;
  });
```

2.3、运行测试

我们重新启动服务端,然后运行客户端,可以看到客户端收到了服务端发送的 heartbeat 消息了,



五、工程源码

本文工程源码,我已上传到 CODE CHINA ,感兴趣的同学可自行下载学习,地址: https://codechina.csdn.net/linxinfa/UnitySprotoDemo

林新发 / UnitySprotoDemo



五、完毕

好了,就先写这么多吧,关于 Skynet 、 Sproto 还有很多很多内容,本文只是一个入门,希望可以帮助到新手同学~

我是林新发: https://blog.csdn.net/linxinfa原创不易,若转载请注明出处,感谢大家~

喜欢我的可以点赞、关注、收藏,如果有什么技术上的疑问,欢迎留言或私信~