# 实验课：基于 tRPC-Cpp 的流式文件传输 (本地编译版)

课程时长： 90分钟

核心目标：

1. **实践 Protobuf：** 亲手编写 .proto 文件来定义服务和消息。
2. **掌握流式 RPC：** 理解并实现 tRPC-Cpp 的客户端流（Client Streaming）模式。
3. **熟悉本地编译与运行：** 掌握 CMake 构建流程，并在多终端环境中进行服务联调。

## 第一部分：环境准备 (15分钟)

在开始编码前，我们需要确保 Ubuntu 虚拟机里已经安装好了所有必要的工具和 tRPC-Cpp 框架。

1. 安装基础构建工具

打开一个终端，运行以下命令：

sudo apt-get update  
sudo apt-get install -y build-essential cmake git libnuma-dev

2. 安装 tRPC-Cpp 框架 (约5-10分钟) - [更正后的步骤]

我们将从源码编译并安装 tRPC-Cpp。这是当前版本的正确流程。

# 克隆源码  
git clone [https://github.com/trpc-group/trpc-cpp.git](https://github.com/trpc-group/trpc-cpp.git)  
cd trpc-cpp  
  
# 切换到一个稳定的版本  
git checkout v1.2.0  
  
# 1. 配置项目：创建一个 build 目录并生成构建系统文件  
cmake -B build -S .  
  
# 2. 编译项目：使用所有CPU核心进行编译，这会花费几分钟时间  
cmake --build build -j$(nproc)  
  
# 3. 安装项目：使用 sudo 将编译好的库和头文件安装到系统目录  
# 这是代替旧版本 install.sh 的标准方法  
sudo cmake --install build  
  
# 返回到你的项目目录  
cd ..

安装成功后，tRPC-Cpp 的库和头文件会被安装到系统的 /usr/local/trpc 目录下，这样我们的实验项目才能通过 find\_package 命令找到它。

## 第二部分：代码实现 (45分钟)

这部分与之前的指导完全相同。你需要按照理论指导，完成 proto 定义的理解，并填充客户端和服务端的核心业务逻辑。

### 步骤一：理解“合同” (.proto 文件) (10分钟)

请在 proto/ 目录下找到 file\_transfer.proto 文件，并仔细阅读。思考 Protobuf message, service, stream 关键字的作用。

### 步骤二：实现服务端 (15分钟)

打开 server/file\_transfer\_service.cc 文件。你需要在这里补全 UploadFile 方法的核心逻辑。文件中已经为你准备好了框架，你需要关注 // TO-DO: 部分。

**核心逻辑：** 创建文件流 -> 循环读取网络流 -> 将数据块写入文件 -> 关闭文件 -> 设置成功响应。

### 步骤三：实现客户端 (20分钟)

打开 client/main.cc 文件。你需要在这里补全文件发送的逻辑。

**核心逻辑：** 打开本地文件 -> 发起 RPC 调用获取 stream\_writer -> 循环读取本地文件块 -> 将文件块写入流 -> 发送结束信号 (WritesDone) -> 等待最终确认 (Finish)。

## 第三部分：编译与运行 (30分钟)

当你完成了客户端和服务端的代码后，我们就可以在本地编译并运行它们了！

1. 编译项目 (约1分钟)

在项目根目录下，执行以下命令：

# 创建一个构建目录  
cmake -B build -S .  
  
# 使用所有CPU核心进行编译  
cmake --build build -j$(nproc)

编译成功后，所有生成的文件（包括可执行文件和配置文件）都会被准备好。

2. 安装程序

我们将编译好的程序和配置文件“安装”到一个独立的 install 目录中。

# 将 build 目录中的结果安装到当前目录下的 install 文件夹  
cmake --install build --prefix ./install

执行完毕后，你会看到一个 install/bin 目录，里面包含了我们运行所需的一切。

3. 运行与测试

现在，我们需要两个终端窗口来模拟客户端和服务端的交互。

**打开第一个终端（作为服务端）：**

# 进入程序所在的目录  
cd install/bin/  
  
# 启动服务端程序  
./file\_transfer\_server

如果一切正常，你会看到服务端成功启动并开始监听的日志。此终端会保持运行状态。

**打开第二个终端（作为客户端）：**

# 同样进入程序所在的目录  
cd install/bin/  
  
# 运行客户端程序  
./file\_transfer\_client

客户端会自动连接服务端，发送 sample.txt 文件，然后打印服务端的响应并退出。

4. 验证结果

回到第一个终端（服务端），按 Ctrl + C 停止服务。然后，使用 ls 命令查看当前目录，你会发现多了一个 received\_file.txt 文件。

使用 diff 命令来验证两个文件的内容是否完全一致：

diff sample.txt received\_file.txt

如果这个命令没有任何输出，那就说明两个文件一模一样，实验圆满成功！

恭喜你完成了本次实验！