服务器

代码结构介绍

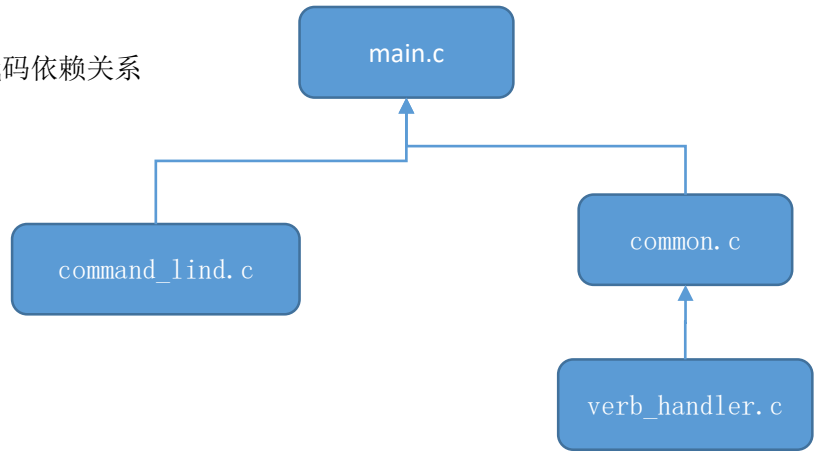
server.c: 程序入口，控制程序运行的整体逻辑。包含函数：main()

command_line.c: 解析命令行（提取 root port 参数）

common.c: 实现了通信层（下面将会说明）的读写函数，为上层提供了接口，实现了 CreateAndListen()、ListenAndReply()函数，封装了网络通信与 FTP 服务器的基本逻辑流程，为 main()函数提供了调用的接口。此外，实现了 keyFromString()、getIpAddress()两个查询、映射函数。

verb_handler.c: 实现了解析、处理指令的逻辑细节。

代码依赖关系



功能特色

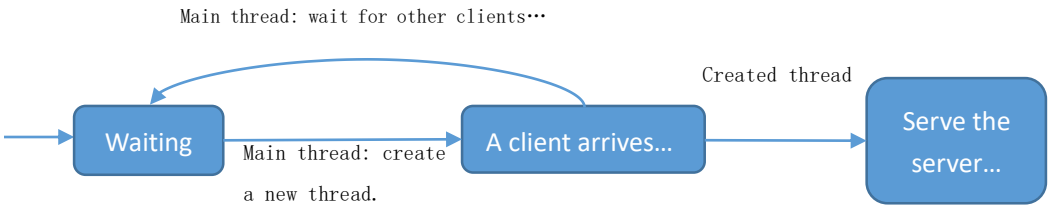
基本要求：支持多客户连接。

方法：

使用多线程的方式。主线程在 main 函数中持续监听控制端口（默认 21），当有客户请求连接时，单开线程进行服务。具体流程如下图所示。

代码位置：serve.c while 块

程序整体执行流程



特色：多层次的代码结构。

描述：

代码分为三个层次：翻译层、执行层与通信层。翻译层负责将客户端发送的数据按照指令、参数作初步分割；执行层根据指令决定执行对应的操作；通信层负责实现具体的网络传输细节。三个层次相互独立，接口清晰，便于调试与维护。

代码位置：翻译层（verb_handler.c: commandParser()）、执行层（verb_handler.c: Execute()）、通信层（verb_handler.c: readFromSocket()）。

特色：具有较高的鲁棒性。
描述：

该 ftp 服务器具有较高的鲁棒性，具体体现在指令解析、执行两个方面。

该 ftp 服务器中保存有 ftp 协议中涉及到的全部指令（“全部”的范围参考：http://cr.yp.to/ftp.html，位置：common.h），如果解析得到的指令没有被实现，该服务器将会根据 RFC959 协议返回："202 Verb not implemented.\n"（代码位置：verb_handler.c: commandParser()最后一个 else 块），如果出现了未知指令，将会返回"500 Undefined verb.\n"（同一函数倒数第二个 else 块），总之，该 ftp 对于错误的指令具有一定鲁棒性。

执行方面，该 ftp 对各种异常输入作了较为完善的异常处理。例如：处理 LIST 指令时，该 ftp 能够预先检测数据连接（异常返回 425）、文件夹的存在性（异常返回 550），虽然没有 RFC959 中定义的那样细致全面，但是基本能够做到不会因为错误的输入而导致程序崩溃。此外，其他存取相关的指令在实现时也考虑到了各种异常情况，在此不一一列举。（代码位置：verb_handler.c … case LIST: …）

特色：在目录操作上具有较高的灵活性以及安全性
描述：

灵活性：支持(cd) .. 返回上一级目录。在 CWD、MKD、RMD、等指令的实现中，对于各种输入方式具有较高的容错性（但是为了安全起见不决定支持绝对路径访问），/tmp　tmp/　/tmp/　tmp 四种输入方式都可以被正确解析。（verb_handler.c 搜索’/’可找到相关代码）。

安全性：不支持绝对路径访问，并且在客户端要求回退上一级目录时，会预先判断是否超出 server 启动时输入的目录（默认/tmp），超出时拒绝。（代码：server.c 与 verb_handler.c 中搜索 level 变量）

特色：支持用户名-密码的匹配。
描述：

该 ftp 服务器除了支持作业说明中要求的 anonymous 匿名登陆外，还支持用户名密码的匹配登陆。如果希望启用该功能，请在 server 应用目录下添加 userdata 文件，按行输入用户名和密码，中间用空格隔开。如果想禁用该功能（即允许所有用户登陆），删除 userdata 文件即可。此外，为防止用户恶意破解密码，如果用户名、密码不匹配，服务器 5 秒后才会有回应。（代码：verb_handler.c 搜索：case PASS: ）

Stage2

功能 1：断点续传
实现方法：

FTP 协议中定义了 REST 指令，用于设置传输起始位置。当传输中断时，客户端已经储存了一部分数据，因此只需要利用 REST 指令向服务器发送请求，即可在原文件基础上追加传输。

功能 2：传输大文件时防止阻塞服务器

实现方法：

当服务器接收到 RETR 指令时，单开一个工作线程用于传输即可。

功能 3：支持多用户同时连接

实现方法：

如前所述。

客户端

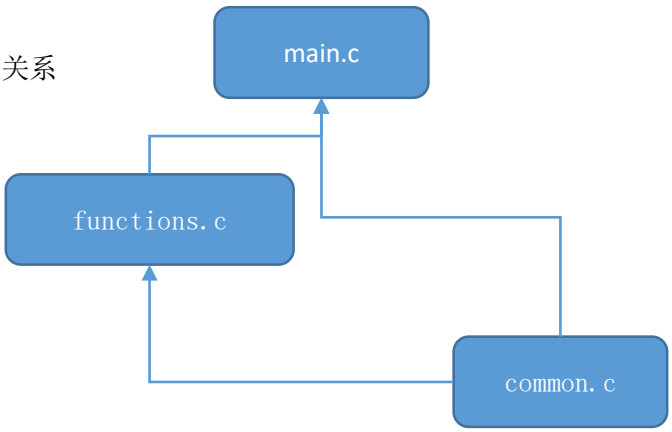
代码结构介绍

client.c: 实现网络连接，控制程序执行的基本流程。

common.c: 实现网络端口的读写，指令解析。

functions.c: 实现一些指令对应的具体功能，包括登陆、下载、发送、传输文件目录等功能。

代码依赖关系



遇到的一些困难及感想

1、 TCP 例程输入端口号的时候并没有使用 htons 函数，因此我认为自己绑定了 21 号端口的时候实际上一直绑定的是 5376 端口…所以 Ubuntu 自带的客户端一直都连不上默认端口…调了好久没想明白问题是出在哪里，甚至怀疑是不是 Ubuntu 虚拟机启动了 NAT 端口重定向功能，装 Ubuntu 双系统结果又遇到了其他各种诡异的问题，血泪史就不展开讲了…直到后来的某一天突然反应过来，原来 5376 是 21 的 256 倍，然后恍然大悟…强烈建议把这个例程改一改。

2、 Project Guide 里并没有给出 autograde.py 的使用方法，我觉得应该简单说明一下，避免学生在不重要的地方走很多弯路。

3、 实际上在这次作业 stage1 的 DDL 前，网络部分基本还没有深入去讲，所以在实际完成作业的过程中，我所遇到的大多数问题都是自己查资料解决的，走了不少弯路，如果适当调整一下授课以及留作业的顺序，是不是这个大作业更能发挥好巩固课堂内容的作用？

4、 感谢助教的耐心指导，真的感谢。