report

支持命令

用户

USER PASS：用户身份验证

传输模式

PORT：客户端指定地址，服务器监听

PASV：服务器指定地址，客户端监听

文件传输

RETR：下载服务器当前目录下的特定文件

STOR：上传客户端当前目录下的特定文件

LIST：获取服务器当前目录下的文件列表

文件操作

MKD：创建目录

CWD：切换工作路径

PWD：输出当前工作路径

RMD：删除文件

RNFR：重命名文件原名

RNTO：重命名文件新名

多进程解决多客户端问题

为了让服务器能够同时服务多个客户端，我使用了C语言的多进程

核心代码如下

```
// 监听循环
while (1) {
```

```
    // 用于通信的socket 等待client的连接 阻塞
    sock_cmd = accept(sock_listen, (struct sockaddr*)&addr_client, &len);
    if (sock_cmd == -1) ERR_CONTINUE;

    int pid = fork();
    if (pid < 0) ERR_RETURN;
    if (pid > 0){
      close(sock_cmd);
      continue;
    }

    // 连接成功
    // ...
```

在监听到一个客户端的连接时，我使用 `fork` ，让一个进程继续执行连接成功之后的操作，另一个进程继续执行监听循环，等待下一个客户端的连接。

宏定义简化代码

我使用宏定义简化错误处理部分的代码，提高了代码的美观性和开发的便捷性

```
#define ERR_RETURN { \
  printf("%s(%d)\n", strerror(errno), errno); \
  return 1; \
}

#define ERR_CONTINUE { \
  printf("%s(%d)\n", strerror(errno), errno); \
  continue; \
}

#define ERR_BREAK { \
  printf("%s(%d)\n", strerror(errno), errno); \
  break; \
}
```

适配Linux和WSL环境

我使用WSL环境开发程序。由于WSL的ip地址是Windows分配的局域网ip，而不是公网ip，所以我需要对Linux进行适配

我在程序选项中添加了 `-local` 选项。在WSL中，只能使用 `127.0.0.1` 作为PORT和PASV模式中传递的ip。在Linux中，可以通过 `-local r` 来让程序获取自身公网ip，然后传递。具体来说，程序使用 `ip a` 命令来获取自身ip.