C語言Socket網絡文件傳輸（可循環發送多個文件）

C語言加 每日

次文件傳輸的實現主要是通過客戶端向服務器發送下載，然後在服務器中找到對應的文件並打開文件，再繼續向客戶端發送文件，而客戶端就在不停的請求接收。文件可能比較大，一個緩衝陣列只能保存重要文件內容，因此可以不斷從文件中讀取內容並保存客戶端，而客戶端得不停的循環接收。

記得一定要先運行服務器，在運行客戶端。這裡是本地回環測試，如果有公網服務器或局域網測試，請吧127.0.0.1改成對應的服務器所在計算機的ip地址。

測試過7G的文件傳輸，總用時02m03s

- 文件大小 讀取35s
- 文件傳輸0m27s

文件結構：

- tcpSocket.h 簡單封裝的Tcp socket接口頭文件
- tcpSocket.c 簡單封裝的Tcp socket接口源文件
- server.c 文件傳輸服務器
- client.c 文件傳輸客戶端

tcpSocket.h

```
1 #ifndef _TCPSOCKET_H_
```

```
#define _TCPSOCKET_H_
#include<stdbool.h>
#include<stdio.h>
#include<WinSock2.h>//头文件
#pragma comment(lib,"ws2_32.lib")//库文件

#define err(errMsg) printf("[error] %s failed,code %d \
line:%d\n",errMsg, WSAGetLastError(),__LINE__);

#define PORT 8888//0~1024 是系统保留，我们一般不用
#define SendSize((size_t)(100 * 1024 * 1024))//一次性发送的数据大小
//初始化网络库
bool init_Socket();
//关闭网络库
bool close_Socket();
//服务器：创建服务器socket
SOCKET create_serverSocket();
//客户端：创建客户端socket
SOCKET create_clientSocket(const char* ip);

#endif // !_TCPSOCKET_H_
```

tcpSocket.c

```
#include"tcpSocket.h"

```

```
bool init_Socket()
{
  WSADATA wsadata;
  if (0 != WSAStartup(MAKEWORD(2, 2), &wsadata))    //wsa  windows socket ansyc  windows异步套接字
  {
    err("WSAStartup");
    return false;
  }
  return true;
}

bool close_Socket()
{
  if (0 != WSACleanup())
  {
    err("WSACleanup");
    return false;
  }
  return true;
}

SOCKET create_serverSocket()
{
  //1，创建一个空的socket
  SOCKET fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (INVALID_SOCKET == fd)
  {
```

```
30      err("socket");
31      return INVALID_SOCKET;
32    }
33    //~0  对于有符号来说是-1  对于无符号来说是最大值
34
35    //2，给socket绑定本地ip地址和端口号
36    struct sockaddr_in addr;
37    addr.sin_family = AF_INET;
38    addr.sin_port = htons(PORT);      //把本地字节序转为网络字节序，  大端存储和小端存储
39    addr.sin_addr.S_un.S_addr = ADDR_ANY;  //绑定本地任意ip
40    if (SOCKET_ERROR == bind(fd, (struct sockaddr*)&addr, sizeof(addr)))
41    {
42      err("bind");
43      return INVALID_SOCKET;
44    }
45
46    //2，开始监听
47    listen(fd, 10);
48
49    return fd;
50  }
51
52  SOCKET create_clientSocket(const char* ip)
53  {
54    //1，创建一个空的socket
55    SOCKET fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
56    if (INVALID_SOCKET == fd)
```

```
  {
    err("socket");
    return INVALID_SOCKET;
  }
  //~0  对于有符号来说是-1  对于无符号来说是最大值

  //2，给socket绑定服务端的ip地址和端口号
  //struct sockaddr;
  struct sockaddr_in addr;
  addr.sin_family = AF_INET;
  addr.sin_port = htons(PORT);      //把本地字节序转为网络字节序，  大端存储和小端存储
  addr.sin_addr.S_un.S_addr = inet_addr(ip);      //绑定服务器ip
  if (INVALID_SOCKET == connect(fd, &addr, sizeof(addr)))
  {
    err("connet");
    return INVALID_SOCKET;
  }
  return fd;
}
```

服務器.c

```
//获取文件大小(Byte)
size_t fileSize(const char* fileName);
```

```
bool readAndSendFile(SOCKET s, const char* fileName);
#include"tcpSocket.h"
int main()
{
  init_Socket();

  SOCKET serfd = create_serverSocket();
  printf("server create successed ,wait client connet...\n");
  //等待客户端连接
  SOCKET clifd = accept(serfd, NULL, NULL);
  if (clifd == INVALID_SOCKET)
  {
    err("accept");
  }
  while (1)
  {

    //可以和客户端进行通信了
    printf("等待客户端，请求数据...\n");
    //接受客户端请求的文件名
    char clientFileNames[100] = "";
    int ret = recv(clifd, clientFileNames, 100, 0);
    if (ret <= 0)
    {
      closesocket(clifd);
      break;
    }
```

```
    readAndSendFile(clifd, clientFileNames);

  }



  closesocket(serfd);
  close_Socket();
  system("pause");
  return 0;
}
//获取文件大小(Byte)
size_t fileSize(const char* fileName)
{
  FILE* fp = fopen(fileName, "rb");
  if (fp == NULL)
  {
    perror("file open failed:");
    return -1;
  }
  char* buf = calloc(SendSize, sizeof(char));
  if (!buf)
  {
    printf("内存申请失败\n");
    return -1;
  }

```

```
  size_t size = 0;
  while (!feof(fp))
  {
    int ret = fread(buf, sizeof(char), SendSize, fp);
    size += ret;
  }

  fclose(fp);
  return size;
}

bool readAndSendFile(SOCKET s, const char* fileName)
{
  //获取文件大小
  size_t fileSzie = fileSize(fileName);
  if (fileSzie == (size_t)-1)
  {
    printf("get file size failed\n");
    return false;
  }


  //发送文件大小
  send(s, &fileSzie, sizeof(size_t), 0);


  FILE* read = fopen(fileName, "rb");
```

```
84    if (!read)
85    {
86      perror("file open failed");
87      return false;
88    }
89
90    size_t nblock = fileSzie / SendSize;        //计算分成100M的包，可以分多少块
91    size_t remain = fileSzie - nblock * SendSize;    //看有没有剩下的字节
92    printf("nblock:%llu  remain:%llu\n", nblock, remain);
93
94    //分配内存
95    char*  fileBuf = calloc(SendSize, sizeof(char));
96    if (!fileBuf)
97    {
98      printf("[error line:%d] memeory alloc failed\n", __LINE__);
99      return false;
100   }
101
102   for (int i = 0; i < nblock; i++)
103   {
104     //把文件读到内存中来
105     int len =  fread(fileBuf, sizeof(char), SendSize, read);
106     //发送
107     int ret = send(s, fileBuf, len, 0);
108     if (ret == SOCKET_ERROR)
109     {
110       err("Send");
```

```
111         return false;
112       }
113
114       printf("第%02d块发送成功,共(%d)Byte\n",i,ret);
115     }
116
117     //发送多余的数据
118     if (remain != 0)
119     {
120       //把文件读到内存中来
121       fread(fileBuf, sizeof(char), remain, read);
122       //发送
123       int ret = send(s, fileBuf, remain, 0);
124       if (ret == SOCKET_ERROR)
125       {
126         err("Send");
127         return false;
128       }
129       printf("最后一块发送成功,共(%d)Byte\n", ret);
130
131     }
132     printf("\nAll file send successfully · totoa %llu Byte\n", nblock * SendSize + remain);
133
134     free(fileBuf);
135
136     fclose(read);
137     return true;
```

```
}
```

客戶端

```
#include"tcpSocket.h"
const char* getFileName(const char* fullPath);
bool recvAndSaveFile(SOCKET s, const char* fileName);
int main()
{
  init_Socket();

  SOCKET fd = create_clientSocket("127.0.0.1");
  printf("connet server successed..\n");

  while (true)
  {
    //向服务器发送需要获取的文件名
    char filePath[100] = "";
    printf("请输入要downLoad的文件名:");
    gets_s(filePath, 100);
    send(fd, filePath, strlen(filePath), 0);


    const char * saveFileName = getFileName(filePath);
    recvAndSaveFile(fd, saveFileName);
```

```
22
23    }
24
25    closesocket(fd);
26    close_Socket();
27    system("pause");
28    return 0;
29  }
30  //从完整路径中获取文件名
31  const char* getFileName(const char* fullPath)
32  {
33    char* resStr = NULL;
34    if ((resStr = strrchr(fullPath, '/')) == NULL)
35    {
36      resStr = strrchr(fullPath, '\\');
37    }
38    return resStr + 1;
39  }
40  //从服务器接受并保存文件
41  bool recvAndSaveFile(SOCKET s, const char* fileName)
42  {
43    //总文件大小
44    size_t fileSize = -1;
45    //接收文件大小
46    recv(s, &fileSize, sizeof(size_t), 0);
47    printf("recv filesize:%llu\n", fileSize);
48
```

```
49    FILE* fp = fopen(fileName, "wb+");
50    if (fp == NULL)
51    {
52      perror("file open failed:");
53      return false;
54    }
55
56    char* fileBuf = calloc(SendSize, sizeof(char));
57    if (!fileBuf)
58    {
59      printf("[error line:%d] memeory alloc failed\n", __LINE__);
60      return false;
61    }
62
63    size_t recvSize = 0;         //当前接受到的文件总大小
64    while (recvSize < fileSize)  //如果没有接收完整，则继续接收
65    {
66      int ret = recv(s, fileBuf, SendSize, 0);
67      if (ret > 0)
68      {
69        //实际接收到多少数据，就写入多少数据
70        fwrite(fileBuf, sizeof(char), ret, fp);
71      }
72      else if (ret <= 0)
73      {
74        printf("服务器下线...\n");
75        break;
```

```
76      }
77      recvSize += ret;
78      printf("%lfM/%lfM\n", (double)recvSize/1024/1024, (double)fileSize /1024/1024);
79      //printf("当前接受到(%.2lf)MB的数据，剩余(%.2lf)MB未接收\n", (double)ret / 1024 / 1024, (double)(g_fileSzie - re
80      //printf("当前接受到(%d)Byte的数据，剩余(%llu)Byte未接收\n", ret,g_fileSzie-recvSize);
81    }
82    fclose(fp);
83    printf("总共接受到 %llu Byte的数据\n", recvSize);
84    free(fileBuf);
85
86    return false;
87  }
```