***Сетевое программирование сокетов: понятие, работа с протоколами SMTP и FTP.***

***Разработать приложение на языке С++ с функцией генерации N паролей. Требования к паролю задается пользователем в виде параметров: L-длина пароля, S-строка с обязательными символами, К-количество обязательных символов.***

Сетевое программирование сокетов — это способ подключения двух узлов в сети для связи друг с другом. Сокеты представляют собой программный интерфейс для создания каналов взаимодействия между устройствами с целью обмена информацией.

Работа с протоколами SMTP и FTP на C++ возможна с использованием библиотеки Winsocket. Для работы с почтовыми службами (SMTP и POP3) вызов функции создания нового сокета будет выглядеть так: Socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP). Локальный адрес потребуется указать лишь при работе с некоторыми протоколами высокого уровня, например с SMTP (протокол передачи сообщений).

Для работы с FTP на C++ можно использовать библиотеку FTP Client Engine for C/C++ (FCE4C). Она позволяет подключаться к любому FTP-серверу, перемещаться по его структуре каталогов, перечислять файлы, загружать, удалять, добавлять и загружать файлы с использованием протокола FTP.

В C# для сетевого программирования сокетов используется класс Socket из пространства имён System.NET.Sockets, который предоставляет низкоуровневый интерфейс для приёма и отправки сообщений по сети.

**Сетевое программирование с использованием сокетов и протоколов SMTP** (Simple Mail Transfer Protocol) и FTP (File Transfer Protocol) позволяет создавать приложения для отправки электронной почты и передачи файлов. Рассмотрим, как работать с этими протоколами в C# и C++.

Понятие протоколов SMTP и FTP

**SMTP** (Simple Mail Transfer Protocol): Протокол прикладного уровня для отправки электронной почты. Клиент (например, почтовый клиент) отправляет электронное письмо SMTP-серверу, который пересылает его другим SMTP-серверам, пока письмо не достигнет получателя. SMTP обычно использует порт 25 (и другие, например, 587 для Submission).

**FTP** (File Transfer Protocol): Протокол прикладного уровня для передачи файлов между клиентом и сервером. FTP использует два TCP-соединения: одно для команд (обычно порт 21) и другое для передачи данных (обычно порт 20 для активного режима, для пассивного используются случайные порты).

**Работа с SMTP в C#**

C# предоставляет удобные классы для работы с SMTP в пространстве имен System.Net.Mail.

Пример отправки письма через SMTP:

using System;

using System.Net;

using System.Net.Mail;

using System.Threading.Tasks;

public class SmtpClientExample

{

public static async Task Main(string[] args)

{

string smtpServer = "smtp.example.com"; // Замените на адрес вашего SMTP-сервера

int smtpPort = 587; // Порт SMTP (может быть 25 или 465)

string smtpUser = "your\_email@example.com"; // Замените на ваш email

string smtpPassword = "your\_password"; // Замените на ваш пароль

string senderEmail = "your\_email@example.com";

string recipientEmail = "recipient@example.com"; // Замените на email получателя

string subject = "Test Email from C#";

string body = "This is a test email sent from C# using SMTP.";

using (SmtpClient smtp = new SmtpClient(smtpServer, smtpPort))

{

smtp.EnableSsl = true; // Включить SSL/TLS

smtp.Credentials = new NetworkCredential(smtpUser, smtpPassword);

using (MailMessage mail = new MailMessage(senderEmail, recipientEmail, subject, body))

{

try

{

await smtp.SendMailAsync(mail);

Console.WriteLine("Email sent successfully.");

}

catch (Exception ex)

{

Console.WriteLine($"Failed to send email: {ex.Message}");

if(ex.InnerException!=null)

Console.WriteLine($"Inner exception: {ex.InnerException.Message}");

}

}

}

Console.ReadKey();

}

}

**Работа с FTP в C#**

Для работы с FTP в C# можно использовать класс FtpWebRequest.

Пример загрузки файла на FTP-сервер:

using System;

using System.IO;

using System.Net;

using System.Threading.Tasks;

public class FtpClientExample

{

public static async Task Main(string[] args)

{

string ftpServer = "<ftp://ftp.example.com/path/to/upload/>"; // Замените на ваш FTP-сервер

string ftpUser = "your\_ftp\_user"; // Замените на вашего FTP-пользователя

string ftpPassword = "your\_ftp\_password"; // Замените на ваш FTP-пароль

string sourceFilePath = "test.txt"; // Имя файла для загрузки

string targetFile = "uploaded\_test.txt"; // Имя файла на сервере

var fullUri = new Uri(ftpServer + targetFile);

Console.WriteLine(fullUri.ToString());

FtpWebRequest request = (FtpWebRequest)WebRequest.Create(fullUri);

request.Method = WebRequestMethods.[Ftp.UploadFile](file:///C:\Users\King%20Night\Documents\GitHub%20Repository\UP_sys\UltimateProject\Билет%2013\Ftp.UploadFile);

request.Credentials = new NetworkCredential(ftpUser, ftpPassword);

request.UsePassive = true;

try

{

using (Stream fileStream = File.OpenRead(sourceFilePath))

{

using (Stream requestStream = await request.GetRequestStreamAsync())

{

await fileStream.CopyToAsync(requestStream);

}

using (FtpWebResponse response = (FtpWebResponse)await request.GetResponseAsync())

{

Console.WriteLine($"Upload successful. Status code: {response.StatusCode}");

response.Close();

}

}

}

catch (Exception ex)

{

Console.WriteLine($"FTP upload failed: {ex.Message}");

}

Console.ReadKey();

}

}

**Работа с SMTP в C++**

Для работы с SMTP в C++ можно использовать сторонние библиотеки, например, libcurl или “чистые” сокеты. Использование библиотек проще, так как они обрабатывают низкоуровневую работу с сокетами.

Пример отправки письма с использованием libcurl:

#include <iostream>

#include <curl/curl.h>

#include <string>

#include <vector>

// Структура для представления почтового адреса

struct MailAddress

{

std::string name;

std::string email;

MailAddress(std::string name, std::string email) : name(name), email(email) {}

};

// Callback для чтения данных из сообщения

size\_t read\_callback(char \*buffer, size\_t size, size\_t nitems, void \*userdata)

{

const std::string \*message\_ptr = static\_cast<const std::string \*>(userdata);

const std::string &message = \*message\_ptr;

static int currentPos = 0; // Позиция чтения

size\_t maxLength = size \* nitems;

size\_t bytesLeft = message.length() - currentPos;

if (bytesLeft <= 0) return 0;

size\_t bytesToCopy = std::min((size\_t)bytesLeft, maxLength);

memcpy(buffer, message.c\_str() + currentPos, bytesToCopy);

currentPos += bytesToCopy;

return bytesToCopy;

}

int main()

{

CURL \*curl;

CURLcode res = CURLE\_OK;

curl\_global\_init(CURL\_GLOBAL\_ALL);

curl = curl\_easy\_init();

if (curl)

{

std::string smtpServer = "smtp.example.com:587"; // SMTP-сервер

std::string smtpUser = "your\_email@example.com"; // SMTP-пользователь

std::string smtpPassword = "your\_password"; // SMTP-пароль

MailAddress sender("Sender Name", "your\_email@example.com");

MailAddress recipient("Recipient Name", "recipient@example.com"); // Получатель

// Адрес отправки

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_FROM, sender.email.c\_str());

curl\_easy\_setopt(curl, CURLOPT\_USERNAME, smtpUser.c\_str());

curl\_easy\_setopt(curl, CURLOPT\_PASSWORD, smtpPassword.c\_str());

//Адреса получателей

struct curl\_slist \*recipients = NULL;

recipients = curl\_slist\_append(recipients, recipient.email.c\_str());

curl\_easy\_setopt(curl, CURLOPT\_MAIL\_RCPT, recipients);

// Текст сообщения

std::string message = "Date: Fri, 28 Aug 2024 12:00:00 +0100\r\n"

"To: " + recipient.name + "<" + recipient.email + ">\r\n"

"From: " + sender.name + "<" + sender.email + ">\r\n"

"Subject: Test Email from C++\r\n"

"\r\n"

"This is a test email from C++ using SMTP with libcurl.\r\n";

curl\_easy\_setopt(curl, CURLOPT\_READFUNCTION, read\_callback);

curl\_easy\_setopt(curl, CURLOPT\_READDATA, &message);

curl\_easy\_setopt(curl, CURLOPT\_UPLOAD, 1L);

curl\_easy\_setopt(curl, CURLOPT\_INFILESIZE, (long)message.length());

curl\_easy\_setopt(curl, CURLOPT\_URL, smtpServer.c\_str());

res = curl\_easy\_perform(curl);

if(res != CURLE\_OK)

fprintf(stderr, "curl\_easy\_perform() failed: %s\n",

curl\_easy\_strerror(res));

else

fprintf(stdout, "Mail sent succesfully!\n");

curl\_easy\_cleanup(curl);

curl\_slist\_free\_all(recipients);

}

curl\_global\_cleanup();

return 0;

}

**Работа с FTP в C++**

Для работы с FTP в C++ можно использовать библиотеку libcurl.

Пример загрузки файла на FTP-сервер (с использованием libcurl):

#include <iostream>

#include <curl/curl.h>

#include <fstream>

#include <string>

size\_t read\_callback(char \*buffer, size\_t size, size\_t nitems, void \*userdata)

{

std::ifstream \*file = static\_cast<std::ifstream\*>(userdata);

size\_t read\_size = size \* nitems;

file->read(buffer, read\_size);

return file->gcount();

}

int main() {

CURL \*curl;

CURLcode res;

curl\_global\_init(CURL\_GLOBAL\_ALL);

curl = curl\_easy\_init();

if(curl)

{

std::string ftpServer = "<ftp://ftp.example.com/path/to/upload/uploaded_test.txt>"; // FTP-сервер

std::string ftpUser = "your\_ftp\_user"; // FTP-пользователь

std::string ftpPassword = "your\_ftp\_password"; // FTP-пароль

std::string sourceFilePath = "test.txt"; // Путь к локальному файлу

std::ifstream file(sourceFilePath, std::ios::binary);

if(!file.is\_open())

{

std::cerr<<"Unable to open file"<<std::endl;

return 1;

}

curl\_easy\_setopt(curl, CURLOPT\_URL, ftpServer.c\_str());

curl\_easy\_setopt(curl, CURLOPT\_UPLOAD, 1L);

curl\_easy\_setopt(curl, CURLOPT\_USERNAME, ftpUser.c\_str());

curl\_easy\_setopt(curl, CURLOPT\_PASSWORD, ftpPassword.c\_str());

curl\_easy\_setopt(curl, CURLOPT\_READFUNCTION, read\_callback);

curl\_easy\_setopt(curl, CURLOPT\_READDATA, &file);

file.seekg(0, std::ios::end);

long fileSize = file.tellg();

file.seekg(0, std::ios::beg);

curl\_easy\_setopt(curl, CURLOPT\_INFILESIZE, fileSize);

res = curl\_easy\_perform(curl);

if(res != CURLE\_OK)

std::cerr << "curl\_easy\_perform() failed: " << curl\_easy\_strerror(res) << std::endl;

else

std::cout << "FTP upload successful" << std::endl;

curl\_easy\_cleanup(curl);

}

curl\_global\_cleanup();

return 0;

}