Київський національний університет

імені Т.Шевченка

**Звіт**

до лабораторної роботи №3

на тему:

*"Взаємодія розподілених процесів через механізм сокетів"*

***Студента другого курсу***

***Групи К-25***

***Факультету комп’ютерних наук***

***та кібернетики***

***Кримлова Данила Юрійовича***

***Київ***

***2021***

**Мета:**

Освоїти механізм (технологію) сокетів стеку протоколів TCP/IP, зокрема його реалізацію в MS Windows. Розробити дві програми (клієнт та сервер, які запускаються на різних станціях мережі), розробити протокол обміну даними між ними та продемонструвати роботи програм.

**Постановка задачі:**

1. Вивчити роботу із сокетами (принципи, функції підтримки роботи з WinSock).
2. Розробити протокол обміну даними для свого варіанту: сценарій обміну даними та його команди (через сокет передається просто повідомлення, а що з ним робити визначають команди).
3. Розробити дві програми, так звані програма-клієнт та програма-сервер. Ці дві програми, працюючи на різних робочих станціях (вузлах) мережі, мають обмінюватися командами, згідно розробленого автором протоколу, використовуючи механізм сокетів у профілі TCP/IP.
4. Має бути передбачена реакція на помилкові команди.
5. Номер порта призначачення вибрати так: №\_порта = 1025 + №\_варіанту.
6. Всі варіанти мають підтримувати команду із ім'ям *Who*, у відповідь на яку сервер дає інформацію про автора та номер і коротку назву варіанту.
7. Клієнт та сервер мають для кожного відісланого/прийнятого сокет-повідомлення робити запис у системний журнал (текстовий файл) про: час дії, команда(и) повідомлення.

**Протокол**

В даному протоколі описав структуру пакета обміну, команди, формати даних.

1. Щоб підключити між собою сервер з клієнтом ми спочатку створюємо сокети та підключаємо їх до локальної мережі: IP “127.0.0.1” PORT “1025+4(номер варіанта)”.

В сервері ми її створюємо

SOCKET sListen = socket(AF\_INET, SOCK\_STREAM, NULL);

bind(sListen, (SOCKADDR \*) &addr, sizeOfAddr);

listen(sListen, SOMAXCONN);

В клієнті підключаємось

Connection = socket(AF\_INET, SOCK\_STREAM, NULL);

if (connect(Connection, (SOCKADDR\*)&addr, sizeOfAddr) != 0){

cout << "Error connecting to the server" << endl;

return 1;

}else cout << "Connected succesfully" << endl;

Треба зазначити, що для підключення сокетів треба підключити бібліотеку <winsock2>;

2. Щоб фіксувати час відправки команд і передавати їх у файл (в мене — "serverlog, clientlog.txt") я використовую бібліотеку <ctime>;

Була створена функція getActualTime() яка повертає у вигляді строки дату, час, день тиждня в коректному форматі.

3. Для передачі рядків від клієнта до сервера використовуються функції:

1) Відправити рядок — send(connection, char[] msg, sizeof(char[]), NULL);

2) Прийняти рядок — recv(connection, char[] msg, sizeof(char[]), NULL);

Де connection – активне з’єднання.

сhar[] msg - повідомлення.

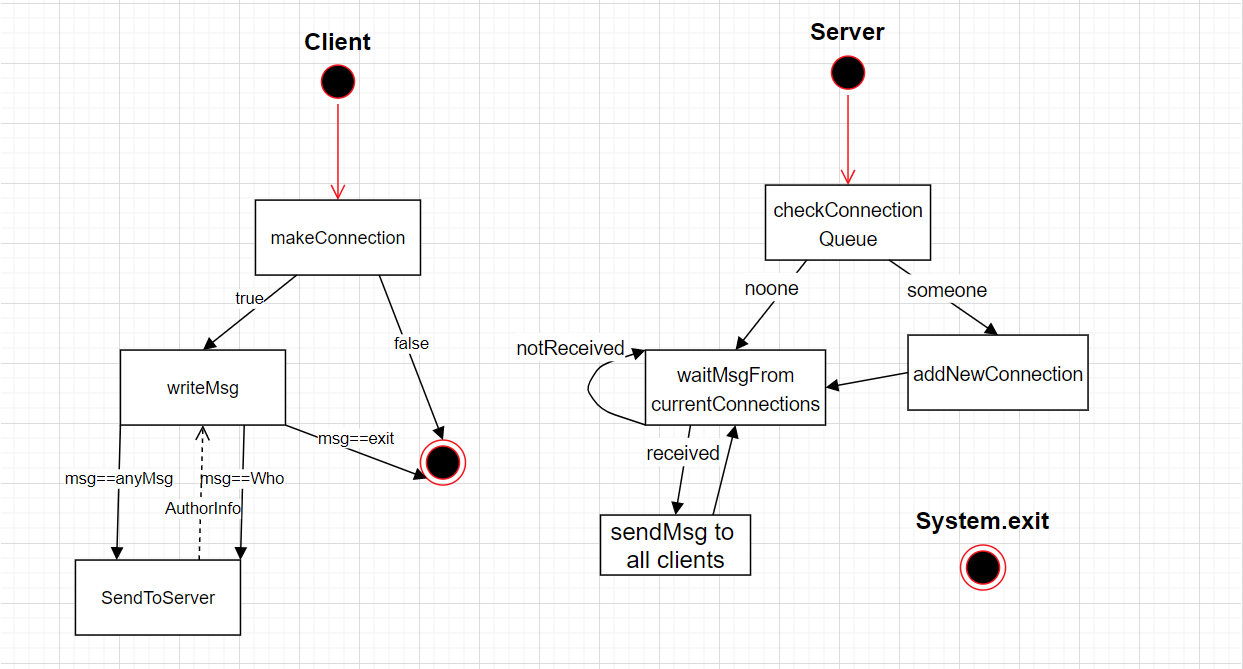
sizeof(msg) - обсяг повідомлення.

NULL - пустий останній параметр(не потрібен)

4. Клієнти комунікують через сервер повідомленнями. Клієнт вводить повідомлення вручну через потік cin>>. Повідомлення йде до сервера, а від сервера відправляється всім під’єднаним клієнтам крім відправника. Відправка і прийняття повідомлень відбуваються в багатопоточному режимі, отже немає якоїсь певної послідовності відправки, дії можуть відбуватись одночасно;

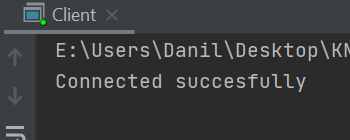
5. Передбачені команди: exit, Who. Якщо повідомлення є однією з команд, то виконується відповідний функціонал, і в log file буде записано що була викликана команда. В іншому випадку - повідомлення. Команди викликані з клієнта інші під’єднані клієнти не бачать, вони отримують лише повідомлення.

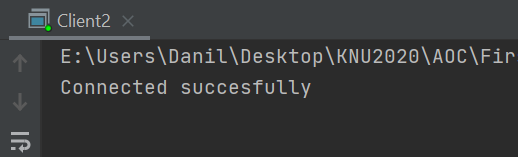
6. Файл протоколу заповнюється переданими повідомленнями з відповідними характеристиками: хто відправив, коли, чи це команда чи повідомлення, текст повідомлення.

****

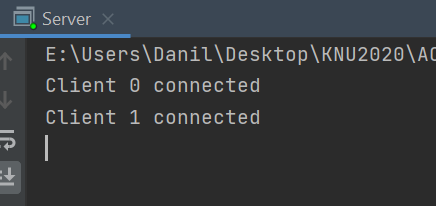
**Робота програм**

Зкомпілюємо сервер, клієнт 1 та клієнт 2. Запустимо їх. При успішому підключенню клієнта до сервера в консолі клієнта буде відповідний напис:



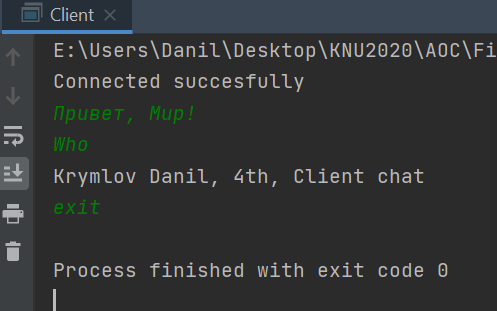


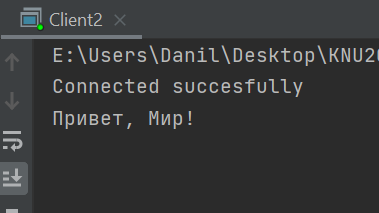
і в консолі сервера:



0,1 - індекс підключенного сервера.(максимум 100 одночасних підключень)

Після цього на будь якому клієнті ми можемо відправити повідомлення, і отримати його на клієнті 2 і навпаки. Також продемострую роботу команди Who та exit(їх використання не повинно бути видимим іншим клієнтам, це ми побачимо пізніше при демонстрації файлів протоколу):





Ми можемо знову запустити тільки що закритий клієнт, і продовжити спілкування, до моменту поки сервер працює.

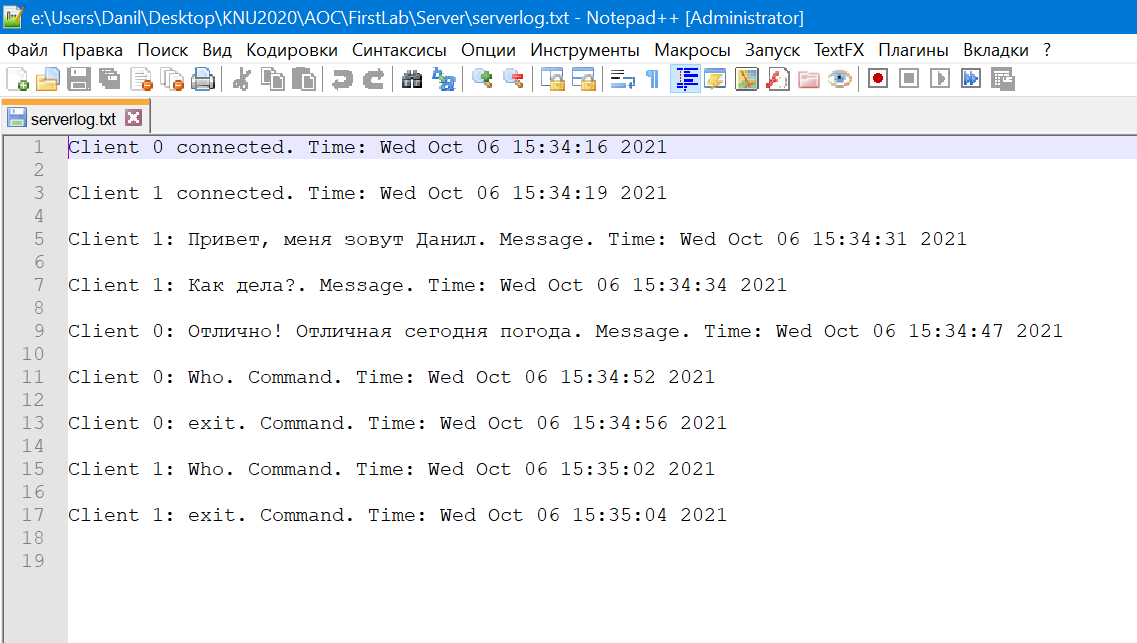
Програма генерує файл протоколу одразу після завершення роботи, вони знаходяться за заздалегідь зазначеним шляхом. Для клієнта clientlog.txt, для сервера serverlog.txt(демонстрація цих файлів далі).

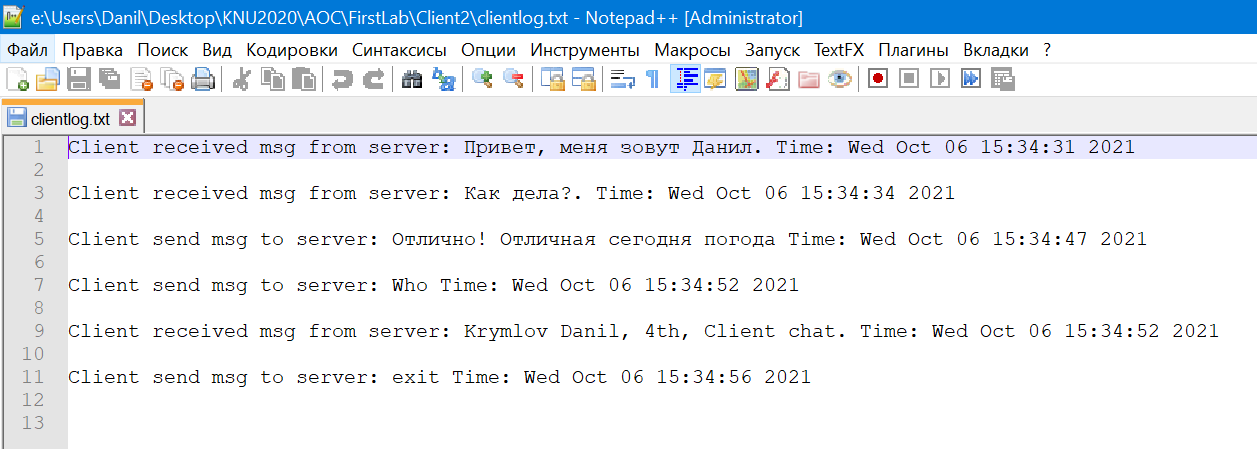
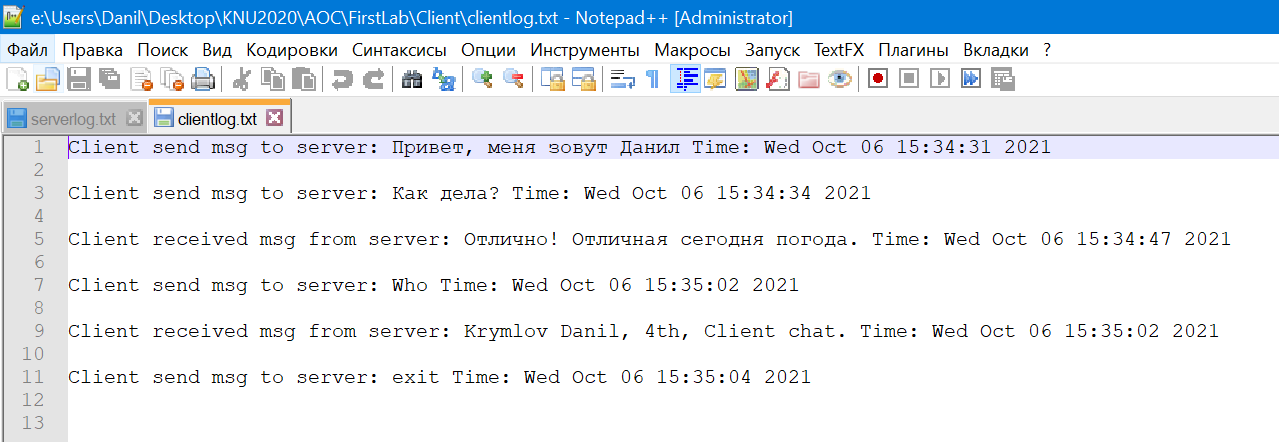
**Файл протоколу**

Після спілкування і завершення роботи кожного з клієнтів та сервера, вони створюють файли .txt в папках проектів.

Імітуємо живе спілкування: обміняємось декількома повідомленнями та виконаємо декілька команд. Після розмови вимкнемо сервер та клієнти, отримаємо файли.

Приклади файлів клієнтів та сервера:



****

**Код програм**

<https://pastebin.com/T4pntP9S> Addr.h

<https://pastebin.com/7YavwhM1> utils.h

<https://pastebin.com/4MKXwuAv> main.cpp(server)

<https://pastebin.com/pkXPuV0D> main.cpp(clients)

***Дякую за увагу!***