**Написання програми-сервера**

Протокол роботи клієнтської та серверної частини

*Сервер:*

1) Створюємо сокет

2) Прив'язуємо сокет до певної IP-адреси та порту.

3) Встановлюємо сокет в "слухає" режим.

4) Чекаємо на запит на з'єднання.

5) Обробляємо вхідний запит (якщо він надійшов у п. 4), а саме приймаємо від клієнта одне повідомлення та виводимо його на екран.

6) Закриваємо з'єднання з цим клієнтом

7) Якщо максимальний ліміт вхідних з'єднань для даної програми не досягнуто (встановлюється при виклику listen), то йдемо в пункт 4 (і чекаємо на наступне з'єднання), інакше завершуємо роботу.

Сервер приймає з'єднання до тих пір, поки не буде досягнуто ліміт, встановлений при виклику listen, або ви не закрили програму-сервер. При відкритті чергового з'єднання сервер отримує повідомлення від клієнта, виводить його на екран і закриває з'єднання.

*Клієнт:*

1) Створюємо сокет

2) Виконуємо з'єднання із сервером

3) Надсилаємо повідомлення

4) Закриваємо з'єднання із сервером

5) Завершуємо роботу.

При роботі з сокетом більшість програмістів використовують стандартний набір подій: для надсилання даних, прийому даних, з'єднання з іншим сокетом, для встановлення каналу передачі даних при вхідному запиті та для закриття сокету. Можливо, є й інші події для сокетів, але в цій статті ми зосередимося на основних. Коли одне з цих подій, асоційованих із сокетом, відбувається, ми отримуємо сигнал і виробляємо необхідні дії обробки цієї події.

Ось основні константи, що описують мережеві події:

FD\_ACCEPT

FD\_READ

FD\_WRITE

FD\_CLOSE

FD\_CONNECT

Отже, розглянемо весь процес створення, відстеження та обробки мережевих подій.

По-перше, нам доведеться ініціалізувати бібліотеку winsock2.

Ініціалізувати можна будь-де програми, але обов'язково до виклику будь-яких функцій winsock. Наступний важливий момент - це створення події, яку хочемо відслідковувати цьому сокеті. Для цього використовуватимемо виклик Winsock2 API ::WSACreateEvent(); Після того, як подія створена, її потрібно пов'язати із сокетом, події якого ми хочемо контролювати та обробляти. Це робиться функцією WSAEventSelect(...). Далі нам потрібно відстежувати події, що сигналізують про те, що у сокет прийшов запит встановлення каналу зв'язку. Зазвичай таку операцію можна робити на сокеті, що прослуховує (listening).

Обробка повідомлень про події

Тепер, коли події задані, нам необхідно чекати на них і, відповідно, обробляти. Для очікування подій можна використовувати функцію WSAWaitForMultipleEvents(...). Ця функція буде працювати як потік у сплячому режимі, доки не відбудеться подія, на яку ми хотіли б відреагувати.

Перший параметр – це кількість подій, які ми хочемо очікувати. Другий параметр – це покажчик на масив подій, на які ми хочемо очікувати. Третій параметр має значення BOOL, яке визначає - чи буде функція залишатися в сплячому режимі, доки не спрацюють всі події. Зазвичай цей параметр задається як false, але можливо, Вам може знадобитися очікувати настання всіх подій. Четвертий параметр визначає - як довго чекати настання події. Зазвичай я запускаю окремий потік та лишаю його як infinite. Але, якщо Ви запускатимете функцію в основному потоці, то може знадобитися поставити обмеження в 5 (або більше) секунд, щоб дати можливість застосуванню обробляти інші події. П'ятий параметр вказує на те, хочемо ми чи ні отримувати алерти.

Далі необхідно подбати про обробників кожної події. Насамперед нам необхідно отримати достовірну інформацію про те, яка подія виникла. І тому існує функція ::WSAEnumNetworkEvents(...).

**Опис створення клієнт/серверу на Сі++**

Перед початком роботи необхідно підключити бібліотеку winsock2.h для створення та роботи з сокетами а також iostream для роботи з функціями мови С++.

*Сервер.*

Створюється сокет для передачі інформацією, а також функція, яка буде зчитувати усі рядки(за допомогою recv), які їх будуть надсилати клієнти та розсилати ці рядки іншим клієнтам(за допомогою send). Кінцевий одержувач повідомлення визначається унікальним іменем, яке вказав автор повідомлення з консолі. Ця функція називається ClientHandler.

В головній програмі якщо нема виключних ситуацій то створюється сокет та визначаються усі необхідні поля для роботи з ним.

Далі проводиться з’єднання з клієнтами, та їм повідомляється про те, як користуватись клієнтно-серверним середовищем. Далі записується ім’я користувача та виводиться інформація про те, що користувач під’єднався до чату та його унікальний ідентифікатор, за яким до нього зможуть звертатись сервер, а за допомго сервера й інші користувачі.

*Клієнт.*

У клієнта також присутня функція ClientHandler, хоча функціонал вона має набагато менший ніж функція з цією назвою у сервері. В клієнті ця функція буде приймати інформацію, яка їй буде надходити від сервера за допомогою створеного зв’язку через сокети.

В головній програмі початок аналогічний до серверної програми, а саме створення сокету за умови відсутності виключних ситуацій.

Однак далі у зміну msg1 записується з консолі повідомлення користувача до серверу, адресоване іншому користувачу, воно проходить перевірку сервером на правильність звертання і тоді вже сервер пересилає це повідомлення вибраному користувачу.

Підсумок.

Програма дозволяє створювати з’єднання між багатьма користувачами за допомогою створення клієнтно-серверного середовища, в якому сервер отримує від користувачів та відправляє їм повідомлення. Повідомлення можуть бути приватними та публічними, в залежності від формату його оформлення з консолі користувача.