מחסן אטומים

מטרתנו בחלק זה הינו לבנות מחסן של פחמן, חמצן ומימן.

אנו נקים מערכת Client-Server בסיסית, שבה לקוחות יכולים להוסיף בקשות להוסיף אטומים מסוגים שונים(CARBON,OXYGEN,HYDROGEN).

כעת, נקים את שרת המחסן האטומי עם תמיכה במספר לקוחות בו-זמנית(IO Multiplexing).

נממש שרת בשם atom_warehouse, המקבל התחברויות TCP , אשר יודע להאזין לחיבורים מלקוחות, לטפל במספר לקוחות בו זמנית בעזרת select(), לקבל פקודות להוספת אטומים למחסן.

בנוסף, שרת הTCP יודע לשלוח בחזרה ללקוח את הספירה המעודכנת, ולטפל במקרים חריגים של קלט שגוי.

נסביר בקצרה מה בנינו בקובץ השרת שלנו:

```
bool shutdown_requested = false;

void handle_shutdown(int) {
    shutdown_requested = true;
}
```

הגדרנו משתנה דגל , המציין האם נדרש לסגור את השרת שלנו (נראה בהמשך).

בנוסף, בנינו פונקציה שמופעלת אוטומטית כשמתקבל סיגנל SIGINT מהמשתמש.

כעת, ניכנס אל תוך פונקציית ה-main שלנו:

```
signal(SIGPIPE, SIG_IGN);
signal(SIGINT, handle_shutdown);
```

כעת, אפשר לראות כי אנו מפעילים פה 2 סיגנלים.

נזכיר כי **סיגנל** זהו דרך שמערכת ההפעלה שולחת אירועים אסינכרוניים לתהליך. בקוד שלנו, אנו משתמשים ב-2 סיגנלים עיקריים:

SIGPIPE← זהו כתיבה לחיבור סגור, כלומר: אם ננסה לשלוח(send) ל-socket שהצד השני סגור, נקבל ברירת מחדל של crash.

נאמר למערכת ההפעלה להתעלם מסיגנל הזה, (זאת דרך SIG_IGN), כדי שהשרת שלנו לא יתרסק אם לקוח יתנתק בזמן השליחה.

SIGINT← הפסקת התהליך , זאת דרך Ctrl+C, נגדיר שהמערכת תקרא לפונקציה שתסגור באופן מסודר את כל החיבורים , ונדפיס:

Server shut down.

כעת, נגדיר counterים עבור האטומים שהצטברו לפי סוג:

```
unsigned long long hydrogen_count = 0;
unsigned long long oxygen_count = 0;
unsigned long long carbon_count = 0;
```

נעשה בדיקה של קלט למשתמש, בכך שהשרת שלנו מקבל פורט כפרמטר(לפי הוראות השאלה, הפורט הינו הארגומנט היחיד שהשרת מקבל).

בנוסף, נבצע המרה של מחרוזת (הפורט מוצג כמחרוזת) למספר שלם:

```
// The server is only getting the port (for now):
if (argc != 2) {
    std::cerr << "Usage: " << argv[0] << " <port>\n";
    return 1;
}

// Convert the port from string to in   teger:
int port = std::stoi(argv[1]);
```

כעת ניצור את ה-socket מסוג ה-TCP:

נזכיר כי:

הsocket שלנו יכנס למשתנה מסוג int ששמו server_fd, המייצג את המזהה של הsocket שנוצר (File Descriptor),זהו המספר שבו מערכת ההפעלה מתייחסת לsocket.

AF_INET ← מייצג לנו כי הsocket ישתמש בפרוטוקול IPv4

SOCK_STREAM ← מייצג שזו תקשורת המבוססת TCP , כלומר חיבור רציף ואמין.

שמנו 0← זה מייצג את הפרוטוקול, כאן זה 0 כי אנחנו רוצים את ברירת המחדל עבור TCP.

```
// Create the socket TCP:
int server_fd = socket(AF_INET, SOCK_STREAM, 0);
if (server_fd < 0) {
    perror("Socket failed");
    return 1;
}
```

נגדיר את כתובת השרת:

ניצור משתנה בשם address מסוג sockaddr_in , זהו מבנה ייעודי לתיאור כתובת של socket ב-IPv4.

היא תכלול כתובת IP, מספר פורט, וסוג משפחת כתובות.

נפעיל את המתודה memset, אשר זוהי פעולה שמנקה את כל התוכן של המבנה address, היא ממלא אותו באפסים.

נועד לוודא שאין בו ערכים אקראיים בזיכרון, אשר עשויים לשבש את ההגדרה.

דרך שימוש ב-INADDR_ANY, שמביא למצב כי השרת יקשיב לכל כתובת ה-IP שיש לו. אם לשרת יש כמה כתובות, הוא יאזין לכולן.

כעת, נציב את המספר הפורט שהשרת יאזין עליו, זאת נעשה דרך htons , אשר ממיר את מספר הפורט מפורט מקומח לפורמט רשת:

```
// Define the address's structure of the server:
sockaddr_in address;
std::memset(&address, 0, sizeof(address));
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons(port);
```

כעת נבצע את פעולת הbind, אשר מחברת את השרת לכתובת ולפורט, כך בעצם השרת מאזין על פורט מסוים במכונה:

```
// bind to the port:
int bind_result = bind(server_fd, (sockaddr *)&address, sizeof(address));
if (bind_result < 0){
    perror("bind failed");
    close(server_fd);
    return 1;
}
```

כעת, נתחיל בהאזנה ← נמתין לחיבורים נכנסים, נגדיר ב-5 את גודל התור שלנו, משמע← כמה חיבורים ממתינים אפשר לשים בתור לפני שהשרת דוחה אותם:

```
// Starts the listen action:
int listen_result = listen(server_fd, 5);
if (listen_result < 0){
    perror("Listen failed");
    close(server_fd);
    return 1;
}
std::cout << "Server is listening on port " << port << "...\n";
```

כעת, נבצע לולאה אינסופית עם IO Multiplexing , אשר תמשיך לפעול עד שתקבל אות לסגירה (SIGINT או SHUTDOWN מהלקוח).

נאפס את מערך הקבצים שה-select יבדוק, ונוסיף את הsocket הראשי לבדיקה.

נוסיף את כל הsocketים של הלקוחות לרשימת הקבצים, אשר הselect בודק:

```
// Infinite loop-for IO MUX:
while (!shutdown_requested) {
    fd_set readfds; // Struct that includes the sockets's list
    FD_ZERO(&readfds); // Starting from empty list
    FD_SET(server_fd, &readfds); // Adding the listening socket to the readfds
    int max_fd = server_fd;

    // Adding all the clients:
    for (int client_fd : clients) {
        FD_SET(client_fd, &readfds);
        if (client_fd > max_fd){
            max_fd = client_fd;
        }
    }
```

פעולת ה-**select** הינה פעולה המאפשרת לבדוק מספר socketים בו-זמנית, ולדעת איזה מהם מוכן לכתיבה, קריאה, או קיבל שגיאה- וכל זה בלי לבזבז זמן על בדיקה חוזרת.

נניח ויש לי מספר לקוחות כלשהם, נקרא **לselect**, והיא תעצור את התוכנית עד שמישהו שולח פקודה/מישהו מתחבר/מישהו מתנתק, ברגע אחד מהדברים האלו קורה, אזי הselect מחזירה בדיוק מי מהsocketים מוכן לפעולה.

אם הselect שלנו מחזיר 1-, וגם לא נלחץ Ctrl+C ← משמע שקראה שגיאה:

```
// Active waiting until one of the sockets will be ready for reading:
int activity = select(max_fd + 1, &readfds, nullptr, nullptr, nullptr);
if (activity < 0 && !shutdown_requested) {
    perror("Select error");
    break;
}
```

כעת, נבצע טיפול בחיבורים חדשים, אם יש חיבור חדש מהלקוח, אזי נקרא לaccept כדי לקבל את החיבור, ונשמור את מזהה הsocket ברשימת הלקוחות:

```
// If a new connection is coming:
if (FD_ISSET(server_fd, &readfds)) {
    sockaddr_in client_addr;
    socklen_t client_len = sizeof(client_addr);
    int new_client = accept(server_fd, (sockaddr*)&client_addr, &client_len);
    if (new_client < 0) {
        perror("Accept failed");
        continue;
    }
    clients.insert(new_client);
    std::cout << "[INFO] New client connected: " << inet_ntoa(client_addr.sin_addr) << "\n";
}
```

כעת, נבצע טיפול בלקוחות קיימים:

נעבור על כולם, אם אחד מהם שלח מידע← נקרא אותם עם recv, נקרא את הפקודה שנשלחה מהלקוח, ונכניס אותה למשתנה buffer:

אם הלקוח הינו התנתק:

כלומר אם לא התקבלו בתים← משמע שכנראה שהלקוח שלנו סגר את החיבור← ולכן סוגרים את החיבור בצד השרת:

```
for (int client_fd : clients) {
    if (FD_ISSET(client_fd, &readfds)) {
        char buffer[1024] = {0};
        int bytes_received = recv(client_fd, buffer, sizeof(buffer) - 1, 0);

        if (bytes_received <= 0) {
            std::cout << "[INFO] Client disconnected.\n";
            close(client_fd);
            disconnected.insert(client_fd);
            continue;
        }
```

נבצע הסרה של תווים מיותרים:

```
std::string command(buffer);
command.erase(std::remove(command.begin(), command.end(), '\n'), command.end());
command.erase(std::remove(command.begin(), command.end(), '\r'), command.end());
```

אם הלקוח שלנו שלח EXIT ← סוגרים רק את החיבור שלו, אם הלקוח שלח SHUTDOWN ← נסגור את כל הרשת:

```
// Checking if the client request to exit;
if (command == "EXIT") {
    std::cout << "[INFO] Client requested EXIT.\n";
    close(client_fd);
    disconnected.insert(client_fd);
    continue;
}

// Checking if the client request to close all of th network:
if (command == "SHUTDOWN") {
    std::cout << "[INFO] Shutdown command received from client.\n";
    shutdown_requested = true;
    close(client_fd);
    disconnected.insert(client_fd);
    break;
}
```

נפצל את המחרוזת שהתקבלה לפעולה, סוג האטום, וכמות.

נבדוק את התקינות לפקודה: נבדוק שה-action הינו ADD, נבדוק כי הכמות גדולה מאפס, ונבדוק שסוגי האטומים הינם:
HYDROGEN,OXYGEN, CARBON בלבד.

אם ישנה בעיה, נשלח ללקוח הודעת שגיאה

```
std::istringstream iss(command);
std::string action, atom_type;
long long amount = -1;

if (!(iss >> action >> atom_type >> amount)) {
    std::string error_msg = "ERROR: Usage: ADD <ATOM_TYPE> <POSITIVE_AMOUNT>\n";
    send(client_fd, error_msg.c_str(), error_msg.size(), 0);
    continue;
}

if (action != "ADD" || amount <= 0) {
    std::string error_msg = "ERROR: Invalid command or non-positive amount.\n";
    send(client_fd, error_msg.c_str(), error_msg.size(), 0);
    continue;
}

unsigned long long* target = nullptr;
if (atom_type == "HYDROGEN") {
    target = &hydrogen_count;
}
else if (atom_type == "OXYGEN") {
    target = &oxygen_count;
}
else if (atom_type == "CARBON") {
    target = &carbon_count;
}
else {
    std::string error_msg = "ERROR: Unknown atom type. Use HYDROGEN, OXYGEN, or CARBON.\n";
    send(client_fd, error_msg.c_str(), error_msg.size(), 0);
    continue;
}
```

כעת, נעדכן את המלאי ונשלח תשובה:

נוסיף את כמות האטומים למונה המתאים, נשלח ללקוח את הסכום החדש:

```
*target += static_cast<unsigned long long>(amount);
std::cout << "[INFO] Atoms: H=" << hydrogen_count
          << ", O=" << oxygen_count
          << ", C=" << carbon_count << "\n";

unsigned int response = static_cast<unsigned int>(*target);
send(client_fd, &response, sizeof(response), 0);
```

כעת, ננקה לקוחות מנותקים, ונסגור את כל הsocketים עם סיום:

```
    for (int fd : disconnected) {
        clients.erase(fd);
    }
}

for (int fd : clients){
    close(fd);
}
close(server_fd);
std::cout << "[INFO] Server shut down.\n";
return 0;
```

לאחר שסיימנו לבנות ולהסביר על קובץ השרת, כעת ננתח לעומק את הקובץ של הלקוח atom_supplier, הלקוח מתחבר לשרת ושולח בקשות על פי ממשק משתמש לבחירתנו.

כעת, הלקוח מעביר שני ארגומנטים← כתובת השרת והפורט:

```
// With the address of the server and our port:
if (argc != 3) {
    std::cerr << "Usage: " << argv[0] << " <host> <port>\n";
    return 1;
}
```

נשמור את כתובת השרת והפורט כפרמטרים נפרדים.

כעת, מימוש כתובת השרת יבצע שימוש ב-**getaddrinfo**, זאת על מנת לתמוך בשרת מארח , ולא רק בכתובות IP מספריות , הוא מחזיר addrinfo מוכן ל-connect , יצירת ה-socket שלנו נוצרת בסגנון שונה, מבצעת שימוש ישירות בפרמטרים מתוך ה← getaddrinfo, כך שהכל מותאם בצורה נכונה, לבסוף יש ניקוי של מבנה הaddrinfo שהוקצה דינאמית, זאת על מנת למנוע זליגת זיכרון:

```
// Define the server's address using getaddrinfo:
struct addrinfo hints{}, *res;
hints.ai_family = AF_INET; // IPv4 only
hints.ai_socktype = SOCK_STREAM; // TCP

int status = getaddrinfo(server_ip, port_str, &hints, &res);
if (status != 0) {
    std::cerr << "getaddrinfo error: " << gai_strerror(status) << "\n";
    return 1;
}

// Creating the socket TCP:
int sock = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
if (sock < 0) {
    perror("socket failed");
    freeaddrinfo(res);
    return 1;
}

if (connect(sock, res->ai_addr, res->ai_addrlen) < 0) {
    perror("Connection failed");
    freeaddrinfo(res);
    return 1;
}

std::cout << "Connected to server at " << server_ip << ":" << port_str << "\n";
freeaddrinfo(res); // We no longer need the addrinfo result
```

נקבל פקודה מהמשתמש (כמו ADD HYDROGEN 5 , או EXIT):

```
while (true) {
    // Request a command from the user:
    std::cout << "Enter command (e.g., ADD HYDROGEN 3 or EXIT): ";
    std::string command;
    std::getline(std::cin, command);

    // EXIT --> Terminates the program:
    if (command == "EXIT") {
        std::cout << "Closing connection. Bye!\n";
        break;
    }
```

אם המשתמש כותב EXIT , אז סוגרים את התקשורת ויוצאים מהתוכנית.

נשלח תגובה לשרת:

```
// Sending command to the server:
ssize_t bytes_sent = send(sock, command.c_str(), command.size(), 0);
if (bytes_sent < 0) {
    perror("Send failed");
    break;
}
```

כעת, נקבל תגובה מהשרת, ונאזין לתשובת השרת:

```
// Getting response from the server:
char response_buffer[1024] = {0};
ssize_t bytes_received = recv(sock, response_buffer, sizeof(response_buffer) - 1, 0);
if (bytes_received <= 0) {
    std::cerr << "Failed to receive response from server\n";
    break;
}
```

ננסה להבין את התשובה:

אם התשובה מספרית, אז נדפיס את כמות האטומים, אם זו הודעת שגיאה, או טקסט אחר מהשרת , אזי נדפיס אותה ישירות:

```
// Try to parse it as a number
try {
    unsigned int count = std::stoul(response_buffer);
    std::cout << "Server returned count: " << count << "\n";
} catch (...) {
    std::cout << "[Server Message] " << response_buffer << "\n";
}
```

ולבסוף, נסגור את החיבור בצורה מסודרת:

```
close(sock);
return 0;
```

כעת, מה שנראה הינו הרצה מסודרת כמו שצריך בין השרת לבין הלקוח.

ראשית, נבנה **makefile** מסודר לכך:

```
1
2   CXX = g++
3   CXXFLAGS = -std=c++17 -Wall -Wextra
4   TARGETS = atom_warehouse atom_supplier
5   OBJECTS =
6
7   all: $(TARGETS)
8
9   atom_warehouse: atom_warehouse.cpp
10      $(CXX) $(CXXFLAGS) -o atom_warehouse atom_warehouse.cpp
11
12  atom_supplier: atom_supplier.cpp
13      $(CXX) $(CXXFLAGS) -o atom_supplier atom_supplier.cpp
14
15  server: atom_warehouse
16
17  client: atom_supplier
18
19  clean:
20      rm -f $(TARGETS) $(OBJECTS)
21
```

כעת, נתחיל בהרצה:

נפתח בשביל זה 2 טרמינלים להרצה:

עבור הטרמינל של השרת:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ make
g++ -std=c++17 -Wall -Wextra -o atom_warehouse atom_warehouse.cpp
g++ -std=c++17 -Wall -Wextra -o atom_supplier atom_supplier.cpp
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_warehouse 12345
Server is listening on port 12345...
```

כעת, עבור הטרמינל של הלקוח:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT):
```

כעת, נכתוב משהו תקין כפקודה בלקוח:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD OXYGEN 230
[Server Message]
Enter command (e.g., ADD HYDROGEN 3 or EXIT):
```

כעת, ניכנס לחלון טרמינל של השרת:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ make
g++ -std=c++17 -Wall -Wextra -o atom_warehouse atom_warehouse.cpp
g++ -std=c++17 -Wall -Wextra -o atom_supplier atom_supplier.cpp
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_warehouse 12345
Server is listening on port 12345...
[INFO] New client connected: 127.0.0.1
[INFO] Atoms: H=0, O=230, C=0
```

נוסיף עוד משהו אחד לטובת ההבנה וה-counterים:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD OXYGEN 230
[Server Message]
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD HYDROGEN 34
[Server Message] "
Enter command (e.g., ADD HYDROGEN 3 or EXIT):
```

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ make
g++ -std=c++17 -Wall -Wextra -o atom_warehouse atom_warehouse.cpp
g++ -std=c++17 -Wall -Wextra -o atom_supplier atom_supplier.cpp
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_warehouse 12345
Server is listening on port 12345...
[INFO] New client connected: 127.0.0.1
[INFO] Atoms: H=0, O=230, C=0
[INFO] Atoms: H=34, O=230, C=0
```

כעת, ננסה לכתוב הוספה של מספר שלילי(מספר לא תקין):

קיבלנו שגיאה:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD OXYGEN 230
[Server Message] ♦
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD HYDROGEN 34
[Server Message] "
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD CARBON -7
[Server Message] ERROR: Invalid command or non-positive amount.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): 
```

ובשרת שום דבר לא השתנה:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ make
g++ -std=c++17 -Wall -Wextra -o atom_warehouse atom_warehouse.cpp
g++ -std=c++17 -Wall -Wextra -o atom_supplier atom_supplier.cpp
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_warehouse 12345
Server is listening on port 12345...
[INFO] New client connected: 127.0.0.1
[INFO] Atoms: H=0, O=230, C=0
[INFO] Atoms: H=34, O=230, C=0
```

כעת, ננסה סתם לכתוב משהו שאינו בפורמט:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD OXYGEN 230
[Server Message] ♦
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD HYDROGEN 34
[Server Message] "
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD CARBON -7
[Server Message] ERROR: Invalid command or non-positive amount.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): My name is Yom Tov Biton
[Server Message] ERROR: Usage: ADD <ATOM_TYPE> <POSITIVE_AMOUNT>

Enter command (e.g., ADD HYDROGEN 3 or EXIT): 
```

כעת, הלקוח יבצע EXIT:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD OXYGEN 230
[Server Message] ♦
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD HYDROGEN 34
[Server Message] "
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD CARBON -7
[Server Message] ERROR: Invalid command or non-positive amount.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): My name is Yom Tov Biton
[Server Message] ERROR: Usage: ADD <ATOM_TYPE> <POSITIVE_AMOUNT>

Enter command (e.g., ADD HYDROGEN 3 or EXIT): EXIT
Closing connection. Bye!
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ 
```

אפשר לראות שמצד הלקוח נסגר לנו החיבור, אך כאשר נסתכל על החלון של השרת:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd AtomicWarehouse
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ make
g++ -std=c++17 -Wall -Wextra -o atom_warehouse atom_warehouse.cpp
g++ -std=c++17 -Wall -Wextra -o atom_supplier atom_supplier.cpp
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_warehouse 12345
Server is listening on port 12345...
[INFO] New client connected: 127.0.0.1
[INFO] Atoms: H=0, O=230, C=0
[INFO] Atoms: H=34, O=230, C=0
[INFO] Client disconnected.
```

עדיין צד השרת אינו נסגר, ולכן כדי שזה ייסגר:

בצד הלקוח נכתוב את הSHUTDOWN:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT): SHUTDOWN
Failed to receive response from server
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$
```

וכאשר ניכנס לצד השרת:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ ./atom_warehouse 12345
Server is listening on port 12345...
[INFO] New client connected: 127.0.0.1
[INFO] Atoms: H=0, O=230, C=0
[INFO] Atoms: H=34, O=230, C=0
[INFO] Client disconnected.
SHUTDOWN
[INFO] New client connected: 127.0.0.1
[INFO] Shutdown command received from client.
[INFO] Server shut down.
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/AtomicWarehouse$ SHUTDOWN
```

צד השרת שלנו נסגר.

כעת , לאחר שהראנו הרצה מסודרת, נצטרך לבצע כיסוי קוד (**gcov**):

נערוך את הmakefile שלנו:

AtomicWarehouse > M makefile

```
CXX = g++
CXXFLAGS = -std=c++17 -Wall -Wextra -fprofile-arcs -ftest-coverage
LDFLAGS = -lgcov

all: atom_warehouse atom_supplier

atom_warehouse: atom_warehouse.cpp
	$(CXX) $(CXXFLAGS) -o atom_warehouse atom_warehouse.cpp $(LDFLAGS)

atom_supplier: atom_supplier.cpp
	$(CXX) $(CXXFLAGS) -o atom_supplier atom_supplier.cpp $(LDFLAGS)

clean:
	rm -f atom_warehouse atom_supplier *.gcno *.gcda *.gcov
```