

# Sprawozdanie – laboratorium 2 PSI

Przedmiot: Programowanie Sieciowe

Autorzy: Marek Dziecioł, Jakub Mieczkowski, Tomasz Nejman-lider

Data sporządzenia: 27.11.2025

Wersja: 1.0

## Treść zadania:

Napisz zestaw dwóch programów – klienta i serwera komunikujących się poprzez TCP. Klient oraz serwer musi być napisany w konfiguracji C + Python (do wyboru co w czym).

Klient wysyła do serwera działanie, które serwer ma obliczyć i odesłać odpowiedź do klienta. Dla uproszczenia proszę przyjąć, że klient wysyła 3 wiadomości w ramach działania: liczba1, znak „\*”, liczba2. Po otrzymaniu tych 3 wiadomości serwer oblicza „liczba1 \* liczba2” i odsyła wynik do klienta. Klient i serwer wypisują działanie oraz wynik w konsoli. Program klienta ma być interaktywny, tzn. po uruchomieniu to użytkownik wpisuje działanie na zasadzie: liczba1, enter, „\*”, enter, liczba2, enter → powinien pokazać się wynik. Proszę założyć, że użytkownik zawsze wpisze dobre liczby i znak działania (nie tracić czasu na walidację).

## Opis rozwiązania problemu:

Utworzyliśmy dwa programy:

- Klient tcp w C
- Serwer tcp w Pythonie

Serwer nasłuchuje na danym porcie TCP (domyślnie 8888), używając funkcji `recv()`. Po odebraniu danych, dekoduje je na liczbę pierwszą, operator i liczbę drugą. Następnie przeprowadza podaną operację arytmetyczną na liczbach i odsyła rezultat w postaci tekstowej.

Klient wysyła 3 pakiety TCP korzystając z funkcji `send()`:

- Liczbę pierwszą, operator i liczbę drugą.
- Dane wczytywane są kolejno do bufora o rozmiarze `BSIZE=1024`
- Przed wysłaniem dla pewności na końcu bufora dodawany jest terminator tekstu `\0`
- Czeka na odpowiedź, używając `recv()`
- Następnie odpowiedź jest wyświetlana jako zwykły tekst

## Napotkane problemy

1. Po stronie klienta nie wyświetlały nam się żadne dane. Przyczyną okazał się być docker i standardowy strumień wyjścia stdout. Aby to obejść skorzystaliśmy ze znaku końca linii `\n`, `scanf` i trybu interaktywnego (`-ai`).
2. Mieliśmy wątpliwości czy TCP nie scali naszych danych w jeden duży pakiet (algorytm Nagla). Jednakże w trakcie testów, okazało się że do serwera nasz kod wysyła 3 różne pakiety.

## Opis konfiguracji

Serwer/środowisko: `bigubu.ii.pw.edu.pl`

Port domyślny: 8888

Sieć: `z38_network`

Adres sieci: `172.21.38.0/24`, `fd00:1032:ac21:38::/64`

Nazwy kontenerów:

- Serwer: `z38_server`
- Klient: `z38_client`

## Opis testowania i wyniki testów

Przykładowa poprawna komunikacja klienta z serwerem:

Connecting...

Connected.

Input operand no. 1: 2

Input operator: \*

Input operand no. 2: 3

Performed operation:  $2 * 3$

Return code: 3

Server response: 6.0

Input operand no. 1: 10

Input operator: +

Input operand no. 2: 11

Performed operation:  $10 + 11$

Return code: 4

Server response: 21.0

Input operand no. 1: 3

Input operator: -

Input operand no. 2: 5

Performed operation:  $3 - 5$

Return code: 4

Server response: -2.0

Input operand no. 1: 123

Input operator: +

Input operand no. 2: 321

Performed operation:  $123 + 321$

Return code: 5

Server response: 444.0

Input operand no. 1: 42

Input operator: -

Input operand no. 2: 24

Performed operation: 42 - 24

Return code: 4

Server response: 18.0

Input operand no. 1: 5

Input operator: -

Input operand no. 2: 5

Performed operation: 5 - 5

Return code: 3

Server response: 0.

Przykładowa komunikacja z błędami:

Input operand no. 1:

1

Input operator: 2

Input operand no. 2: 2

Performed operation: 1 2 2

Return code: 23

Server response: ERROR: unknown operator

Input operand no. 1: 1

Input operator: &

Input operand no. 2: 3

Performed operation: 1 & 3

Return code: 23

Server response: ERROR: unknown operator

Input operand no. 1: abc

Input operator: +

Input operand no. 2: cde

Performed operation: abc + cde

Return code: 62

Server response: ERROR: invalid data (could not convert  
string to float: 'abc')

## Wnioski końcowe

Udało nam się poprawnie zaimplementować zarówno program kliencki jak i serwer dla powyższego zadania. Przeprowadziliśmy wiele udanych testów. Być może sam protokół oprzeć na strumieniu danych niż na wysyłaniu pakietów TCP. Jednakże nasze podejście w praktyce okazało się działać poprawnie.