# Programowanie aplikacji sieciowych zbiór zadań, część pierwsza

Katarzyna Mazur 30czerwca 2023

# Spis treści

1	Zadania wprowadzające	6
2	Analiza pakietów sieciowych	7
3	Gniazda klienckie	9
4	Gniazda serwerowe	12
5	Bezpiecznie gniazda	13
6	Protokoły pocztowe	14
7	Protokół HTTP	18
8	Odpowiedzi	19

# Gniazda w języku Python - moduł socket

Tworzenie gniazd klienckich/serwerowych TCP, IPv4

```
#!/usr/bin/env python3
import socket

if __name__ == '__main__':
    sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sockIPv4.close()
```

Tworzenie gniazd klienckich/serwerowych TCP, IPv6

```
#!/usr/bin/env python3
import socket

if __name__ == '__main__':
    sockIPv6 = socket.socket(socket.AF_INET6, socket.SOCK_STREAM)
    sockIPv6.close()
```

Tworzenie gniazd klienckich/serwerowych UDP, IPv4

```
#!/usr/bin/env python3
import socket

if __name__ == '__main__':

    sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sockIPv4.close()
```

Tworzenie gniazd klienckich/serwerowych UDP, IPv6

```
#!/usr/bin/env python3
import socket

if __name__ == '__main__':
    sockIPv6 = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM)
    sockIPv6.close()
```

#### Gniazdo klienckie TCP, IPv4: nawiązanie połączenia z serwerem

```
#!/usr/bin/env python3
import socket

if __name__ == "__main__":

    sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sockIPv4.settimeout(5)

try:
    sockIPv4.connect(("127.0.0.1", 80))
    except socket.error as exc:
        print(f"Wyjatek socket.error: {exc}")

    sockIPv4.close()
```

#### Gniazdo klienckie TCP, IPv6: nawiązanie połączenia z serwerem

```
#!/usr/bin/env python3
import socket

if __name__ == "__main__":

   address = socket.getaddrinfo("::1", 80, socket.AF_INET6)
   sockIPv6 = socket.socket(socket.AF_INET6, socket.SOCK_STREAM)
   sockIPv6.settimeout(5)

try:
       sockIPv6.connect(address[0][4])
   except socket.error as exc:
       print(f"Wyjatek socket.error: {exc}")
   sockIPv6.close()
```

#### Gniazdo klienckie TCP, IPv4: komunikacja z serwerem (wysyłanie i odbieranie danych)

```
#!/usr/bin/env python3
import socket

if __name__ == "__main__":
    sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sockIPv4.settimeout(5)

try:
    sockIPv4.connect(("127.0.0.1", 80))
    sockIPv4.sendall("Hello Server!".encode()) # wysylanie
    print(sockIPv4.recv(1024).decode()) # odbieranie
    except socket.error as exc:
        print(f"Wyjatek socket.error: {exc}")

sockIPv4.close()
```

#### Gniazdo klienckie TCP, IPv6: komunikacja z serwerem (wysyłanie i odbieranie danych)

```
#!/usr/bin/env python3
import socket

if __name__ == "__main__":

   address = socket.getaddrinfo("::1", 80, socket.AF_INET6)
   sockIPv6 = socket.socket(socket.AF_INET6, socket.SOCK_STREAM)
   sockIPv6.settimeout(5)

try:
        sockIPv6.connect(address[0][4])
        sockIPv6.sendall("Hello Server!".encode()) # wysylanie
        print(sockIPv6.recv(1024).decode()) # odbieranie
   except socket.error as exc:
        print(f"Wyjatek socket.error: {exc}")
```

#### Gniazdo klienckie UDP, IPv4: komunikacja z serwerem (wysyłanie i odbieranie danych)

```
#!/usr/bin/env python3
import socket

HOST = '127.0.0.1'
PORT = 80

sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_address = (HOST, PORT)

try:
    message = "Hello Server!"
    sent = sockIPv4.sendto(message.encode(), server_address) # wysylanie
    data, server = sockIPv4.recvfrom(4096) # odbieranie
    print(f"Received: {data.decode()}")

finally:
    sockIPv4.close()
```

#### Gniazdo klienckie UDP, IPv6: komunikacja z serwerem (wysyłanie i odbieranie danych)

```
#!/usr/bin/env python3
import socket

HOST = "::1"
PORT = 80

sockIPv6 = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM)

try:
    sockIPv6.sendto("Hello Server!".encode(), (HOST, PORT))  # wysylanie
    data, server = sockIPv6.recvfrom(4096)  # odbieranie
    print(f"Received: {data.decode()}")

finally:
    sockIPv6.close()
```

# 1 Zadania wprowadzające

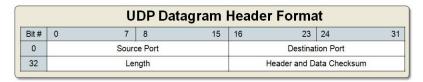
- 1.1 Napisz program, w którym pobierzesz od użytkownika nazwę pliku tekstowego w formacie \*.txt, a następnie skopiujesz go do pliku pod nazwą lab1zad1.txt. Zadbaj o prawidłową obsługę błędów.
- 1.2 Napisz program, w którym pobierzesz od użytkownika nazwę pliku graficznego w formacie \*.png, a następnie skopiujesz go do pliku pod nazwą lab1zad2.png. Zadbaj o prawidłową obsługę błędów.
- 1.3 Napisz program, w którym pobierzesz od użytkownika adres IPv4, a następnie sprawdzisz, czy jest on prawidłowym adresem. Zadbaj o prawidłową obsługę błędów. Podpowiedź: zadanie możesz rozwiązać przy pomocy wyrażeń regularnych
- **1.4** Napisz program, w którym pobierzesz od użytkownika adres IPv6, a następnie sprawdzisz, czy jest on prawidłowym adresem. Zadbaj o prawidłową obsługę błędów. *Podpowiedź: zadanie możesz rozwiązać przy pomocy wyrażeń regularnych*
- 1.5 Napisz program, który jako argument linii poleceń pobierze od użytkownika adres IPv4, a następnie wyświetli odpowiadającą mu nazwę hostname (nazwę domenową). Zadanie rozwiąż bez użycia dodatkowych bibliotek. Zadbaj o prawidłową obsługę błędów.

## 2 Analiza pakietów sieciowych

2.1 Poniżej znajduje się pełny zapis datagramu UDP w postaci szesnastkowej.

```
ed 74 0b 55 00 24 ef fd 70 72 6f 67 72 61 6d 6d 69 6e 67 20 69 6e 20 70 79 74 68 6f 6e 20 69 73 20 66 75 6e
```

Wiedząc, że w zapisie szesnastkowym jedna cyfra reprezentuje 4 bity, oraz znając strukturę datagramu UDP:



Napisz program, który z powyższego datagramu UDP wydobędzie:

- numer źródłowego portu
- numer docelowego portu
- dane (ile bajtów w tym pakiecie zajmują dane?)

A następnie uzyskany wynik w postaci: zad2.1odp;src;X;dst;Y;data;Z gdzie:

- X to wydobyty z pakietu numer portu źródłowego
- Y to wydobyty z pakietu numer portu docelowego
- Z to wydobyte z pakietu dane

prześle do serwera UDP działającego na wskazanym porcie pod podanym adresem IPv4, w celu sprawdzenia, czy udało się prawidłowo odczytać wymagane pola. Serwer zwróci odpowiedź TAK lub NIE, a w przypadku błędnego sformatowania wiadomości, odeśle odpowiedź BAD\_SYNTAX. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

- 2.2 Zmodyfikuj program z zadania 2.1 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 2.3 Poniżej znajduje się pełny zapis segmentu TCP w postaci szesnastkowej (pole opcji ma 12 bajtów).

```
0b 54 89 8b 1f 9a 18 ec bb b1 64 f2 80 18
00 e3 67 71 00 00 01 01 08 0a 02 c1 a4 ee
00 1a 4c ee 68 65 6c 6c 6f 20 3a 29
```

 $Wiedząc, \dot{z}e \ w \ zapisie \ szesnastkowym \ jedna \ cyfra \ reprezentuje \ 4 \ bity, \ oraz \ znając \ strukturę \ segmentu \ TCP:$ 

TCP Segment Header Format										
Bit #	0	7	8	15	16	23	24	31		
0	Source Port				Destination Port					
32	Sequence Number									
64	Acknowledgment Number									
96	Data Offset Res Flags			Window Size						
128	Header and Data Checksum			Urgent Pointer						
160	Options									

Napisz program, który z powyższego segmentu TCP wydobędzie:

- numer źródłowego portu
- numer docelowego portu
- dane (ile bajtów w tym pakiecie zajmuja dane?)

A następnie uzyskany wynik w postaci: zad2.3odp;src;X;dst;Y;data;Z gdzie:

- $\bullet\,$  X to wydobyty z pakietu numer portu źródłowego
- Y to wydobyty z pakietu numer portu docelowego
- Z to wydobyte z pakietu dane

prześle do serwera TCP działającego na wskazanym porcie pod podanym adresem IPv4, w celu sprawdzenia, czy udało się prawidłowo odczytać wymagane pola. Serwer zwróci odpowiedź TAK lub NIE, a w przypadku błędnego sformatowania wiadomości, odeśle odpowiedź BAD\_SYNTAX. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

2.4 Zmodyfikuj program z zadania 2.3 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

#### 3 Gniazda klienckie

#### Gniazda TCP

- 3.1 Napisz program klienta, w którym połączysz się z serwerem na danym porcie przy użyciu protokołu TCP. Adres IPv4 serwera i numer portu pobierz jako argumenty linii poleceń. Wyświetl informację, czy udało się nawiązać połączenie. Program powinien akceptować adres w postaci adresu IPv4 jak i hostname. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.2 Zmodyfikuj program z zadania 3.1 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.3 Napisz program klienta (prosty skaner portów sieciowych), który dla danego serwera przy użyciu protokołu TCP będzie sprawdzał, jakie porty są otwarte, a jakie zamknięte. Adres IPv4 serwera pobierz jako argument linii poleceń. Program powinien akceptować adres w postaci adresu IPv4 jak i hostname. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.4 Zmodyfikuj program z zadania 3.3 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.5 Napisz program klienta, który z serwera o podanym adresie IPv4 i porcie pobierze aktualną datę i czas, a następnie wyświetli je na konsoli. Adres IPv4 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.6 Zmodyfikuj program z zadania 3.5 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.7 Napisz program klienta, który połączy się z serwerem TCP działającym pod podanym adresem IPv4 na podanym porcie, a następnie wyśle do niego wiadomość i odbierze odpowiedź. Adres IPv4 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.8 Zmodyfikuj program z zadania 3.7 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.9 Napisz program klienta, który połączy się z serwerem TCP działającym pod podanym adresem IPv4 na podanym porcie, a następnie będzie w pętli wysyłał do niego tekst wczytany od użytkownika (jako argument wywołania programu bądź jako dane podawane na konsoli), i odbierał odpowiedzi. Adres IPv4 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.10 Zmodyfikuj program z zadania 3.9 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- **3.11** Napisz program klienta (prosty skaner portów sieciowych), który dla danego serwera przy użyciu protokołu TCP będzie sprawdzał, jakie porty są otwarte, a jakie zamknięte. Oprócz informacji o otwartych /

zamkniętych portach, program powinien również wyświetlać informację o tym, jaka usługa jest uruchomiona na danym porcie (baner usługi). Adres IPv4 serwera pobierz jako argument linii poleceń. Program powinien akceptować adres w postaci adresu IPv4 jak i hostname. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

- 3.12 Zmodyfikuj program z zadania 3.11 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.13 Napisz program klienta, który połączy się z serwerem TCP działającym pod podanym adresem IPv4 na podanym porcie, a następnie wyśle do niego wiadomość i odbierze odpowiedź. Warunkiem zadania jest, aby klient wysłał i odebrał od serwera wiadomość o maksymalnej długości 20 znaków. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów. Uwzględnij sytuacje, gdy:
  - wiadomość do wysłania jest za krótka ma być wówczas uzupełniania do 20 znaków znakami spacji
  - wiadomość do wysłania jest za długa ma być przycięta do 20 znaków (lub wysłana w całości sprawdź, co się wówczas stanie)
- 3.14 Zmodyfikuj program z zadania 3.13 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.15 Dostępne dla gniazd funkcje recv i send nie gwarantują wysłania / odbioru wszystkich danych. Rozważmy funkcję recv. Przykładowo, 100 bajtów może zostać wysłane jako grupa po 10 bajtów, albo od razu w całości. Oznacza to, iż jeśli używamy gniazd TCP, musimy odbierać dane, dopóki nie mamy pewności, że odebraliśmy odpowiednią ich ilość. Napisz program klienta, który połączy się z serwerem TCP działającym pod podanym adresem IPv4 na podanym porcie, a następnie wyśle do niego wiadomość i odbierze odpowiedź. Dane odbieraj / wysyłaj w ten sposób, aby mieć pewność, że klient w rzeczywistości odebrał / wysłał wiadomość o wymaganej długości. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- **3.16** Zmodyfikuj program z zadania **3.15** w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

#### Gniazda UDP

- 3.17 Napisz program klienta, który połączy się z serwerem UDP działającym pod podanym adresem IPv4 na podanym porcie, a następnie wyśle do niego wiadomość i odbierze odpowiedź. Adres IPv4 serwera oraz numer portu pobierz jako argumenty linii poleceń. Program powinien akceptować adres w postaci adresu IPv4 jak i hostname. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.18 Zmodyfikuj program z zadania 3.17 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.19 Napisz program klienta, który połączy się z serwerem UDP działającym pod podanym adresem IPv4 na podanym porcie, a następnie będzie w pętli wysyłał do niego tekst wczytany od użytkownika, i odbierał odpowiedzi. Adres IPv4 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.20 Zmodyfikuj program z zadania 3.19 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

- 3.21 Napisz program klienta, który połączy się z serwerem UDP działającym pod podanym adresem IPv4 na podanym porcie, a następnie prześle do serwera liczbę, operator, liczbę (pobrane od użytkownika) i odbierze odpowiedź. Adres IPv4 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.22 Zmodyfikuj program z zadania 3.21 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.23 Napisz program klienta, który połączy się z serwerem UDP działającym pod podanym adresem IPv4 na podanym porcie, a następnie prześle do serwera pobrany z linii poleceń adres IP, i odbierze odpowiadającą mu nazwę hostname. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- **3.24** Zmodyfikuj program z zadania **3.23** w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.25 Napisz program klienta, który połączy się z serwerem UDP działającym pod podanym adresem IPv4 na podanym porcie, a następnie prześle do serwera nazwę hostname pobraną z linii poleceń, i odbierze odpowiadający mu adres IP. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 3.26 Zmodyfikuj program z zadania 3.25 w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Adres IPv6 serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

# 4 Gniazda serwerowe

Gniazda TCP

Gniazda UDP

_	-	•	•	•	1
5	Bezp	ieczn	ne	gniazo	a

### 6 Protokoły pocztowe

#### Protokół SMTP

- 6.1 Pod podanym adresem i numerem portu działa serwer obsługujący protokół ESMTP/SMTP. Wykorzystując gotowe oprogramowanie klienckie połącz się z serwerem działajacym pod podanym adresem i numerem portu, a następnie wyślij wiadomość e-mail korzytając z komend protokołu ESMTP. (Zadanie nie jest zadaniem programistycznym, wymaga jedynie użycia gotowego klienta protokołu SMTP w celu zapoznania się z podstawowymi poleceniami protokołu.)
- 6.2 Pod podanym adresem i numerem portu działa serwer obsługujący protokół ESMTP/SMTP. Wykorzystując gotowe oprogramowanie klienckie połącz się z serwerem działajacym pod podanym adresem i numerem portu, a następnie wyślij kilka wiadomości e-mail do kilku odbiorców korzytając z komend protokołu ESMTP. (Zadanie nie jest zadaniem programistycznym, wymaga jedynie użycia gotowego klienta protokołu SMTP w celu zapoznania się z podstawowymi poleceniami protokołu.)
- **6.3** Pod podanym adresem i numerem portu działa serwer obsługujący protokół ESMTP/SMTP. Wykorzystując gotowe oprogramowanie klienckie połącz się z serwerem działajacym pod podanym adresem i numerem portu, a następnie wyślij wiadomość spoofed e-mail (z podmienionym adresem nadawcy) korzytając z komend protokołu ESMTP. (Zadanie nie jest zadaniem programistycznym, wymaga jedynie użycia gotowego klienta protokołu SMTP w celu zapoznania się z podstawowymi poleceniami protokołu.)
- 6.4 Pod podanym adresem i numerem portu działa serwer obsługujący protokół ESMTP/SMTP. Wykorzystując gotowe oprogramowanie klienckie połącz się z serwerem działajacym pod podanym adresem i numerem portu, a następnie wyślij wiadomość e-mail korzytając z komend protokołu ESMTP. Do wiadomości dodaj załącznik dowolny plik tekstowy (sprawdź format MIME: Multipart i Content-Type). Możesz wykorzystać openssł do przekonwertowania pliku: cat plik.txt | openssł base64. (Zadanie nie jest zadaniem programistycznym, wymaga jedynie użycia gotowego klienta protokołu SMTP w celu zapoznania się z podstawowymi poleceniami protokołu.)
- 6.5 Pod podanym adresem i numerem portu działa serwer obsługujący protokół ESMTP/SMTP. Wykorzystując gotowe oprogramowanie klienckie połącz się z serwerem działajacym pod podanym adresem i numerem portu, a następnie wyślij wiadomość e-mail korzytając z komend protokołu ESMTP. Do wiadomości dodaj załącznik dowolny obrazek (sprawdź format MIME: Multipart i Content-Type). Możesz wykorzystać openssl do przekonwertowania obrazka: cat obrazek |openssl base64. (Zadanie nie jest zadaniem programistycznym, wymaga jedynie użycia gotowego klienta protokołu SMTP w celu zapoznania się z podstawowymi poleceniami protokołu.)
- 6.6 Napisz program klienta, który połączy się z serwerem obsługującym protokół ESMTP/SMTP, działającym na podanym porcie, a następnie wyśle wiadomość e-mail używając komend protokołu ESMTP. O adres nadawcy, odbiorcy (odbiorców), temat wiadomości i jej treść zapytaj użytkownika. Adres serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 6.7 Napisz program klienta, który połączy się z serwerem obsługującym protokół ESMTP/SMTP, działającym na podanym porcie, a następnie wyśle wiadomość e-mail używając komend protokołu ESMTP. Do wiadomości dodaj załącznik dowolny plik tekstowy (sprawdź format MIME: Multipart i Content-Type). O adres nadawcy, odbiorcy (odbiorców), temat wiadomości i jej treść zapytaj użytkownika. Adres serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 6.8 Napisz program klienta, który połączy się z serwerem obsługującym protokół ESMTP/SMTP, działającym na podanym porcie, a następnie wyśle wiadomość e-mail używając komend protokołu ESMTP. Do wiadomości dodaj załącznik dowolny obrazek (sprawdź format MIME: Multipart i Content-Type). O adres nadawcy, odbiorcy (odbiorców), temat wiadomości i jej treść zapytaj użytkownika. Adres serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

- 6.9 Napisz program klienta, który połączy się z serwerem obsługującym protokół ESMTP/SMTP, działającym na podanym porcie, a następnie wyśle wiadomość e-mail używając komend protokołu ESMTP. Do wiadomości dodaj załącznik dowolny plik dźwiękowy. (sprawdź format MIME: Multipart i Content-Type). O adres nadawcy, odbiorcy (odbiorców), temat wiadomości i jej treść zapytaj użytkownika. Adres serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 6.10 Napisz program klienta, który połączy się z serwerem obsługującym protokół ESMTP/SMTP, działającym na podanym porcie, a następnie wyśle wiadomość e-mail używając komend protokołu ESMTP. Treść wiadomości powinna zostać sformatowana za pomocą tagów HTML, przykładowo: <bpogrubienie</bp>
  <i>pochylenie</i>, <u>podkreślenie</u> i innych wybranych. O adres nadawcy, odbiorcy (odbiorców), temat wiadomości i jej treść zapytaj użytkownika. Adres serwera i numer portu pobierz jako argumenty linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 6.11 Napisz program serwera, który działając pod adresem IPv4 127.0.0.1 oraz na określonym porcie TCP, będzie serwerem poczty, obsługującym protokół SMTP. Nie realizuj faktycznego wysyłania e-maila, tylko zasymuluj jego działanie tak, żeby napisany wcześniej klient SMTP myślał, że wiadomość została wysłana. Pamiętaj o obsłudze przypadku, gdy klient poda nie zaimplementowaną przez serwer komendę. Numer portu pobierz jako argument linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- **6.12** Zmodyfikuj program z zadania **6.11** w taki sposób, aby łączył serwer działał pod adresem IPv6 ::1. Numer portu pobierz jako argument linii poleceń. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

# Protokół POP3

# Protokół IMAP

#### 7 Protokół HTTP

#### Protokół HTTP w wersji 1.1

- 7.1 Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/pas-book-p1-ch7-ex1:latest), uruchom serwer HTTP. Po uruchomieniu obrazu, pod adresem IPv4 127.0.0.1 na porcie TCP o numerze 7001 działa serwer obsługujący protokół HTTP w wersji 1.1. Pod odnośnikiem /html udostępnia prostą stronę HTML. Napisz program klienta, który połączy się z serwerem, a następnie pobierze treść strony i zapisze ją na dysku jako plik z rozszerzeniem \*.html. Spreparuj żądanie HTTP tak, aby serwer myślał, że żądanie przyszło od przeglądarki Safari 7.0.3. Jakich nagłówków HTTP należy użyć? Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 7.2 Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/pas-book-p1-ch7-ex2:latest), uruchom serwer HTTP. Po uruchomieniu obrazu, pod adresem IPv4 127.0.0.1 na porcie TCP o numerze 7002 działa serwer obsługujący protokół HTTP w wersji 1.1. Pod odnośnikiem /image/png udostępnia obrazek. Napisz program klienta, który połączy się z serwerem, a następnie pobierze obrazek i zapisze go na dysku. Jakich nagłówków HTTP należy użyć? Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 7.3 Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/pas-book-p1-ch7-ex3:latest), uruchom serwer HTTP. Po uruchomieniu obrazu, pod adresem IPv4 127.0.0.1 na porcie TCP o numerze 7003 działa serwer obsługujący protokół HTTP w wersji 1.1. Pod odnośnikiem /image.jpg udostępnia obrazek. Napisz program klienta, który połączy się z serwerem, a następnie pobierze z serwera obrazek w 3 częściach i po odebraniu wszystkich części złoży go w całość i zapisze na dysku. Jakich nagłówków HTTP należy użyć? Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.
- 7.4 Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/pas-book-p1-ch7-ex4:latest), uruchom serwer HTTP. Po uruchomieniu obrazu, pod adresem IPv4 127.0.0.1 na porcie TCP o numerze 7004 działa serwer obsługujący protokół HTTP w wersji 1.1. Pod odnośnikiem /post udostępnia formularz z polami do wypełnienia. Napisz program klienta, który połączy się z serwerem, a następnie uzupełni formularz danymi pobranymi od użytkownika, a następnie prześle go do serwera i odbierze odpowiedź.

Aby sprawdzić, jak wygląda żądanie HTTP potrzebne do wypełnienia i wysłania formularza:

- $\bullet$  jakie nagłówki~HTTP są wykorzystywane,
- jak wygląda ciało zapytania,

podsłuchaj komunikację z serwerem za pomocą Wiresharka, tj. uruchom przeglądarkę oraz Wiresharka; uzupełnij i zatwierdź formularz ręcznie za pomocą przeglądarki, a następnie sprawdź pakiety podsłuchane podczas komunikacji z serwerem działającym pod adresem http://127.0.0.1:7004. Możesz użyć filtrów Wiresharka: http.request oraz http.response (http.request || http.response). Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

# 8 Odpowiedzi

#### Gniazda klienckie TCP

3.1 Aby przetestować poprawność swojego rozwiązania, możesz przeskanować swój własny komputer, żeby sprawdzić, czy skanowanie da taki sam wynik, jak Twój program. Do tego celu możesz wykorzystać np. narzędzie nmap. Zainstaluj nmap: sudo apt install nmap, a następnie wydaj polecenie: nmap -p 22 localhost (lub nmap -p 22 127.0.0.1) aby sprawdzić, czy port 22 jest otwarty na Twoim komputerze. Porównaj wynik z działaniem Twojego programu dla tego samego portu.

Rysunek 1: Wynik działania programu nmap dla lokalnej maszyny i portu 22

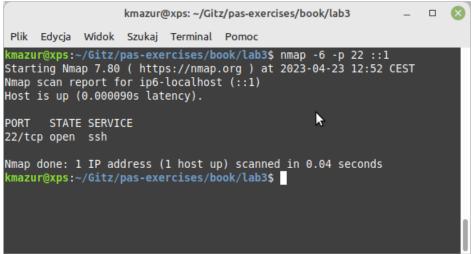
```
kmazur@xps: ~/Gitz/pas-exercises/book/lab3 — 
Plik Edycja Widok Szukaj Terminal Pomoc

kmazur@xps:~/Gitz/pas-exercises/book/lab3$ nmap -p 22 localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-23 11:51 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000095s latency).

PORT STATE SERVICE
22/tcp open ssh

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
kmazur@xps:~/Gitz/pas-exercises/book/lab3$
```

Rysunek 2: Wynik działania programu nmap dla lokalnej maszyny i portu 22



3.3 Aby przetestować poprawność swojego rozwiązania, możesz przeskanować swój własny komputer, żeby sprawdzić, czy skanowanie da taki sam wynik, jak Twój program. Do tego celu możesz wykorzystać np. narzędzie nmap. Zainstaluj nmap: sudo apt install nmap, a następnie wydaj polecenie: nmap - p1-65535 localhost (lub nmap -p1-65535 127.0.0.1) aby sprawdzić, jakie porty są otwarte na Twoim komputerze. Porównaj wynik z działaniem Twojego programu dla tego samego portu.

Rysunek 3: Wynik działania programu nmap dla lokalnej maszyny i wszystkich jej portów

```
kmazur@xps: ~/Gitz/pas-exercises/book/lab3
Plik Edycja Widok Szukaj Terminal Pomoc
kmazur@xps:~/Gitz/pas-exercises/book/lab3$ nmap -p1-65535 localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-23 12:36 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00012s latency).
Not shown: 65529 closed ports
PORT
          STATE SERVICE
22/tcp
          open ssh
                                                   k
80/tcp
          open http
443/tcp
          open https
631/tcp
          open
               ipp
3306/tcp
         open
                mysql
33060/tcp open mysqlx
Nmap done: 1 IP address (1 host up) scanned in 2.68 seconds
kmazur@xps:~/Gitz/pas-exercises/book/lab3$
```

3.4 Aby przetestować poprawność swojego rozwiązania, możesz przeskanować swój własny komputer, żeby sprawdzić, czy skanowanie da taki sam wynik, jak Twój program. Do tego celu możesz wykorzystać np. narzędzie nmap. Zainstaluj nmap: sudo apt install nmap, a następnie wydaj polecenie: nmap -6 -p1-65535 ip6-localhost (lub nmap -6 -p1-65535 ::1) aby sprawdzić, jakie porty są otwarte na Twoim komputerze. Porównaj wynik z działaniem Twojego programu dla tego samego portu.

Rysunek 4: Wynik działania programu nmap dla lokalnej maszyny i wszystkich jej portów

```
Plik Edycja Widok Szukaj Terminal Pom
kasiula@dell:~/Gitz/pas-exercises/book$ nmap -6 -p1-65535 ::1
Starting Nmap 7.80 ( https://nmap.org ) at 2023-05-19 20:53 CEST
Nmap scan report for ip6-localhost (::1)
Host is up (0.000087s latency).
Not shown: 65527 closed ports
           STATE SERVICE
PORT 
22/tcp
           open ssh
111/tcp
           open
                   rpcbind
631/tcp
           open
                  ipp
2049/tcp open
                   nfs
35855/tcp open
                   unknown
40071/tcp open
                   unknown
40143/tcp open
                  unknown
51369/tcp open
                  unknown
Nmap done: 1 IP address (1 host up) scanned in 2.20 seconds
kasiula@dell:~/Gitz/pas-exercises/book$
```

- **3.5** Aby przetestować poprawność swojego rozwiązania, możesz skorzystać z gotowego serwera, do którego może połączyć się Twój klient, aby pobrać aktualną datę i czas. Aby uruchomić serwer, zainstaluj Dockera, a następnie w konsoli wydaj polecenia:
  - Pobierz obraz serwera: docker pull mazurkatarzyna/pas-book-p1-ch3-ex5-server:latest
  - Uruchom serwer za pomocą Dockera: docker run -dp 3005:3005 mazurkatarzyna/pas-book-p1-ch3-ex5-server

Tak uruchomiony serwer działa na Twoim komputerze, jego adres IPv4 to 127.0.0.1 (localhost), numer portu to 3005.

- 3.6 Aby przetestować poprawność swojego rozwiązania, możesz skorzystać z gotowego serwera, do którego może połączyć się Twój klient, aby pobrać aktualną datę i czas. Aby uruchomić serwer, zainstaluj Dockera, a następnie w konsoli wydaj polecenia:
  - Pobierz obraz serwera: docker pull mazurkatarzyna/pas-book-p1-ch3-ex6-server:latest
  - Uruchom serwer za pomocą Dockera: docker run -dp 3006:3006 mazurkatarzyna/pas-book-p1-ch3-ex6-server

Tak uruchomiony serwer działa na Twoim komputerze, jego adres IPv6 to ::1 (localhost), numer portu to 3006.

- 3.7 Aby przetestować poprawność swojego rozwiązania, możesz skorzystać z gotowego serwera, do którego może połączyć się Twój klient, aby wysłać wiadomość. Aby uruchomić serwer, zainstaluj Dockera, a następnie w konsoli wydaj polecenia:
  - Pobierz obraz serwera: docker pull mazurkatarzyna/pas-book-p1-ch3-ex7-server:latest
  - Uruchom serwer za pomocą Dockera: docker run -dp 3007:3007 mazurkatarzyna/pas-book-p1-ch3-ex7-server

Tak uruchomiony serwer działa na Twoim komputerze, jego adres IPv4 to 127.0.0.1 (localhost), numer portu to 3007.

- 3.8 Aby przetestować poprawność swojego rozwiązania, możesz skorzystać z gotowego serwera, do którego może połączyć się Twój klient, aby wysłać wiadomość. Aby uruchomić serwer, zainstaluj Dockera, a następnie w konsoli wydaj polecenia:
  - Pobierz obraz serwera: docker pull mazurkatarzyna/pas-book-p1-ch3-ex8-server:latest
  - Uruchom serwer za pomocą Dockera: docker run -dp 3008:3008 mazurkatarzyna/pas-book-p1-ch3-ex8-server

Tak uruchomiony serwer działa na Twoim komputerze, jego adres IPv6 to ::1 (localhost), numer portu to 3008.

- 3.9 Aby przetestować poprawność swojego rozwiązania, możesz skorzystać z gotowego serwera, do którego może połączyć się Twój klient, aby wysłać wiadomość. Aby uruchomić serwer, zainstaluj Dockera, a następnie w konsoli wydaj polecenia:
  - Pobierz obraz serwera: docker pull mazurkatarzyna/pas-book-p1-ch3-ex9-server:latest
  - Uruchom serwer za pomocą Dockera: docker run -dp 3009:3009 mazurkatarzyna/pas-book-p1-ch3-ex9-server

Tak uruchomiony serwer działa na Twoim komputerze, jego adres IPv4 to 127.0.0.1 (localhost), numer portu to 3009.

- **3.10** Aby przetestować poprawność swojego rozwiązania, możesz skorzystać z gotowego serwera, do którego może połączyć się Twój klient, aby wysłać wiadomość. Aby uruchomić serwer, zainstaluj Dockera, a następnie w konsoli wydaj polecenia:
  - Pobierz obraz serwera: docker pull mazurkatarzyna/pas-book-p1-ch3-ex10-server:latest
  - Uruchom serwer za pomocą Dockera: docker run -dp 3010:3010 mazurkatarzyna/pas-book-p1-ch3-ex10-server

Tak uruchomiony serwer działa na Twoim komputerze, jego adres IPv6 to ::1 (localhost), numer portu to 3010.

3.11 Aby przetestować poprawność swojego rozwiązania, możesz przeskanować swój własny komputer, żeby sprawdzić, czy skanowanie da taki sam wynik, jak Twój program. Do tego celu możesz wykorzystać np. narzędzie nmap z parametrem --script=banner. Zainstaluj nmap: sudo apt install nmap, a następnie wydaj polecenie: nmap -p 22 --script=banner localhost (lub nmap -p 22 --script=banner 127.0.0.1) aby sprawdzić, czy port 22 jest otwarty na Twoim komputerze, i jaka usługa jest uruchomiona na tym porcie. Porównaj wynik z działaniem Twojego programu dla tego samego portu.

Rysunek 5: Wynik działania programu nmap dla lokalnej maszyny i portu 22

```
Pik Edycja Widok Szukaj Terminal Pomoc

→ nmap -p 22 --script=banner 127.0.0.1

Starting Nmap 7.80 ( https://nmap.org ) at 2023-06-02 21:43 CE ST

Nmap scan report for localhost (127.0.0.1)

Host is up (0.00010s latency).

PORT STATE SERVICE

22/tcp open ssh
|_banner: SSH-2.0-0penSSH_8.2p1 Ubuntu-4ubuntu0.7

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds

→ ~
```

3.12 Aby przetestować poprawność swojego rozwiązania, możesz przeskanować swój własny komputer, żeby sprawdzić, czy skanowanie da taki sam wynik, jak Twój program. Do tego celu możesz wykorzystać np. narzędzie nmap z parametrem --script=banner. Zainstaluj nmap: sudo apt install nmap, a następnie wydaj polecenie: nmap -p 22 -6 --script=banner ip6-localhost (lub nmap -p 22 -6 --script=banner ::1) aby sprawdzić, czy port 22 jest otwarty na Twoim komputerze, i jaka usługa jest uruchomiona na tym porcie. Porównaj wynik z działaniem Twojego programu dla tego samego portu.

Rysunek 6: Wynik działania programu nmap dla lokalnej maszyny i portu 22

```
Pik Edycja Widok Szukaj Terminal Pomoc

→ nmap -p 22 -6 --script=banner ::1

Starting Nmap 7.80 ( https://nmap.org ) at 2023-06-02 21:49 CE

ST

Nmap scan report for ip6-localhost (::1)

Host is up (0.000085s latency).

PORT STATE SERVICE

22/tcp open ssh

|_banner: SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.7

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
```

#### Protokoły pocztowe - protokół SMTP

- **6.1** Do wykonania zadania możesz użyć:
  - Klienta telnet, jeśli serwer nie wymaga szyfrowania, polecenie do nawiązania połączenia z serwerem: telnet server\_ip port
  - Klienta OpenSSL, o nazwie s\_client jeśli serwer wymaga szyfrowania, polecenie do nawiązania połączenia z serwerem: openssl s\_client -crlf -connect server\_ip:port

Jako serwera SMTP możesz użyć:

- iRedMail, który dostępny jest jako kontener Dockerowy,
- Serwera pocztowego udostępnianego np. przez interia.pl, gdzie poczta.interia.pl jest adresem serwera SMTP, 465 jest numerem portu, na którym działa serwer. Potrzebujesz również konta na serwerze.

Poniżej przykład połączenia z serwerem poczta.interia.pl:

```
PIRE Edycla Widok Sculag Terminal Pomoc

- openssl s_client -crlf -connect poczta.interia.pl:465

CONNECTED(00000003)

depth=2 C = US, 0 = DigiCert Inc, 0U = www.digicert.com, CN = DigiCert Global Ro ot CA

verify return:1

depth=1 C = US, 0 = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1

verify return:1

depth=0 C = PL, L = Krakow, 0 = Grupa INTERIA.PL sp. z o.o. sp. k., CN = *.interia.pl

verify return:1

---

Certificate chain
0 s:C = PL, L = Krakow, 0 = Grupa INTERIA.PL sp. z o.o. sp. k., CN = *.interia.pl

i:C = US, 0 = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1

1 s:C = US, 0 = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1

1 s:C = US, 0 = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1

i:C = US, 0 = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root

CA

---

Server certificate

----BEGIN CERTIFICATE----

MIIFYZCCBOigAwIBAGIQD/Ztrk8kirkvn+QyffFMbjAKBggqhkjOPQQDAzBWMQsw
CQYDVQQGEwJVUZEWMBMGAIUEChMMRGlnaUNIcnQgSW5jMTAwLgYDVQQDEydEaWdp
Q2VydGBUTFMgSHlicmlkIEVDQyBTSEEzODQgMjAyMCBDQTEwHhcNMjTwNzI3MDAw
MDAwWhcNNjmMwL33MjM10TUSWjBiMQswCQYDVQQGEwJQTDEPMA0GAIUEBxMGS3Jh
a293MSswKQYDVQQKEyJHcnYwYSBJTNRFUKIBLIBMIHNWLIBGIGBUby4gc3AUIGsu
MRUwEwYDVQQDDAwqLmludGVyaWEucGwwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC
```

Na niebiesko zaznaczono polecenia / komendy, których musisz użyć. Kolorem czarnym oznaczono odpowiedzi serwera:

```
openssl s_client -crlf -connect poczta.interia.pl:465
CONNECTED (00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
verify return:1
depth=0 C = PL, L = Krakow, O = Grupa INTERIA.PL sp. z o.o. sp. k., CN = *.interia.pl
verify return:1
Certificate chain
  O s:C = PL, L = Krakow, O = Grupa INTERIA.PL sp. z o.o. sp. k., CN = *.interia.pl
  i:C = US, O = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
1 s:C = US, O = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
     i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
Server certificate
      ---BEGIN CERTIFICATE----
MIIFYzCCBOigAwIBAgIQD/Ztrk8kirkvn+QyffFMbjAKBggqhkjOPQQDAzBWMQsw
CQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMTAwLgYDVQQDEydEaWdp
\tt Q2VydCBUTFMgSHlicmlkIEVDQyBTSEEzODQgMjAyMCBDQTEwHhcNMjIwNzI3MDAward and the constraint of the cons
MDAwWhcnMjMwNzI3MjM10TUSWjBiMQswCQYDVQQGEwJQTDEPMAOGA1UEBxMGS3Jh
a293MSswKQYDVQQKEyJHcnVwYSBJT1RFUklBLlBMIHnwLiB61G8uby4gc3AuIGsu
MRUWEWYDVQQDDAwqLmludGVyaWEucGwwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC
AATj14S/K9d1aInTO/N6wXhyj7/OYxfJlR7jOxE8C5JiUZpaip8/DDL7syoNB3xS
LtJIpG1Ygqy9kRHr8wfIVOCzo4IDijCCA4YwHwYDVROjBBgwFoAUCrwIKReMpTlt
eg70M8cus+37w3owHQYDVROOBBYEFEzd3VGbvBiH2K8Z1aE6Py9COHHNMCMGA1Ud
EQQcMBqCDCouaW50ZXJpYS5wbIIKaW50ZXJpYS5wbDA0BgNVHQ8BAf8EBAMCB4Aw
HQYDVRO1BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMIGbBgNVHR8EgZMwgZAwRqBE
oEKGQGhOdHA6Ly9jcmwzLmRpZ21jZXJOLmNvbS9EaWdpQ2VydFRMUOh5YnJpZEVD
Q1NIQTM4NDIwMjBDQTEtMS5jcmwwRqBEoEKGQGhOdHA6Ly9jcmwOLmRpZ2ljZXJO
VROgBDcwNTAzBgZngQwBAgIwKTAnBggrBgEFBQcCARYbaHROcDovL3d3dy5kaWdparfine Argument and Argument a
Y2VydC5jb2OvQ1BTMIGFBggrBgEFBQcBbQR5HHcwJAYIKwYBBQUHMAGGGGhOdHA6
Ly9vY3NwLmRpZ21jZXJOLmNvbTBPBggrBgEFBQcwAoZDaHROcDovL2NhY2VydHMu
ZGlnaWNlcnQuY29tLORpZ21DZXJOVExTSHlicmlkRUNDUOhBMzgOMjAyMENBMSOx
LmNydDAJBgNVHRMEAjAAMIIBfQYKKwYBBAHWeQIEAgSCAWOEggFpAWcAdQCt9776
fP8QyIudPZwePhhqtGcpXc+xDCTKhYY069yCigAAAYJBhgsMAAAEAwBGMEQCIEL1
g7IxBfCpgigx/rDU7kUvWaYWvMxfOtRkTL2UHbSVAiBIGwYMqfuFuoThbrnyOstk
{\tt O4hwJoak4J69MvZ+HasnAQB2ADXPGRu/sWxXvw+tTG1Cy7u2JyAmUeo/4SrvqAPD}
O9ZMAAABgkGGCtYAAAQDAEcwRQIhAPFjNAwBAB5vOGOQwOuaDWMF7n3n/7s2t7gZ
Nlhy6Z8sAiBcLcCvAP/llzyfk/5olh01c85cWB6+HD3if6BUoUNMlgB2ALNzdwfh
hFD4Y4bWBancEQ1KeS2xZwwLh9zwAw55NqWaAAABgkGGCwEAAAQDAEcwRQIgSsPk
\hbox{\tt X+yBkecqv0CuIXmWg9V2e/s4LYyYvsuIJU7ksxgCIQDkrabch4m6Sg9U6AaqeW8M}
\verb|vPtFrIjAzwSymXt21NyKTDAKBggqhkjOPQQDAwNpADBmAjEAv+cUUU3e9sdRaOFs||
ntdpEVKb4S71HWCsVRZcS2MGIMOZRJ5LVEhhXHK1+EIaJ257AjEA8rSvrwL4MIRz
\tt fCA70evopfs2fJcQVU4TLEbSFBwkAdhOL1HAnXH+NELH1ttjQtfh
       --END CERTIFICATE-
subject=C = PL, L = Krakow, O = Grupa INTERIA.PL sp. z o.o. sp. k., CN = *.interia.pl
issuer=C = US, O = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: ECDSA
Server Temp Key: X25519, 253 bits
SSL handshake has read 2811 bytes and written 389 bytes
Verification: OK
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 256 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
Post-Handshake New Session Ticket arrived:
SSL-Session:
         Protocol : TLSv1.3
                               : TLS_AES_256_GCM_SHA384
         Cipher
         Session-ID: FA72D8E642670674F87A2BF7DE7D8716408D48DB0AF3F21F374FC989B6944C41
         Session-ID-ctx:
         Resumption PSK:
         PSK identity: None
         PSK identity hint: None
         SRP username: None
         TLS session ticket lifetime hint: 7200 (seconds)
         TLS session ticket:
         0000 - d6 18 a0 85 49 b4 4b 4d-46 6d 71 b6 5f 38 ce 17 0010 - be e4 36 99 4d 17 4c ce-8b 13 6f 61 0b 3d e6 17
                                                                                                                                        ....I.KMFmq._8..
                                                                                                                                        ..6.M.L...oa.=.
         0020 - 51 53 d6 d3 bf fc da 87-0f 38 eb 9c f0 fd 0e 5d QS.......
```

```
0030 - c0 9d 2f 95 e4 b3 78 24-98 b8 cd b4 83 50 08 43
    0040 - 1b e8 2b eb cd ee 3d 20-ef d6 4e 89 9d a6 21 92
    0050 - a7 0e 9b 04 e1 1e 9f 36-3f 6d 7b a5 a9 37 7d af
                                                                 .....6?m{..7}.
    0060 - b4 3a 52 dc 68 7a 35 3b-89 74 89 ea ae 34 ba 46
                                                                 .:R.hz5;.t...4.F
    0070 - fb e6 c6 da c3 f9 c1 89-f4 le 89 28 50 32 12 74
    0080 - 07 55 d0 7d 70 2f 1d e0-02 ca 6a e7 41 25 8f 53
                                                                 .U.}p/....j.A%.S
                                                                 ^g.%....M.xFO
    0090 - 5e 67 1d 25 f5 9f ab f9-e5 ce 0d 4d 05 78 46 4f
                                                                 ...;..&q(=.....
.w....8Q.....~.
    00a0 \ - \ 11 \ d4 \ bb \ 3b \ 84 \ d2 \ 26 \ 71-28 \ 3d \ 0b \ 13 \ a9 \ aa \ 85 \ 95
    00b0 - 8d 77 c9 d0 be e5 d3 38-51 b1 1b f2 8b 9f 7e 8c
    00c0 - bb \ e9 \ 5a \ f4 \ f9 \ 71 \ 40 \ 3d-a1 \ 1c \ 5b \ fe \ 35 \ c3 \ 74 \ bf
                                                                 ..Z..q@=..[.5.t.
    Start Time: 1685969504
    Timeout : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
    Max Early Data: 0
read R BLOCK
220 ESMTP INTERIA.PL
ehlo student
250-poczta.interia.pl
250\text{-PIPELINING}
250-SIZE 157286400
250-AUTH PLAIN LOGIN PLAIN LOGIN PLAIN LOGIN
250-AUTH=PLAIN LOGIN PLAIN LOGIN PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-SMTPUTF8
250 CHUNKING
auth login
334 VXN1cm5hbWU6
a29jaGFtLnVtY3NAaW50ZXJpYS5wbA==
334~UGFzc3dvcmQ6\\
a29jaGFtLnVtY3MyMDIz
235 2.7.0 Authentication successful
mail from: <kocham.umcs@interia.pl>
250 2.1.0 Ok
rcpt to: <pas.umcs@poczta.fm>
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
to: <pas.umcs@poczta.fm>
from: <kocham.umcs@interia.pl>
subject: test email
Hello!
250 OK. ID: 1054c830c3886b54
221 2.0.0 Bye
closed
```

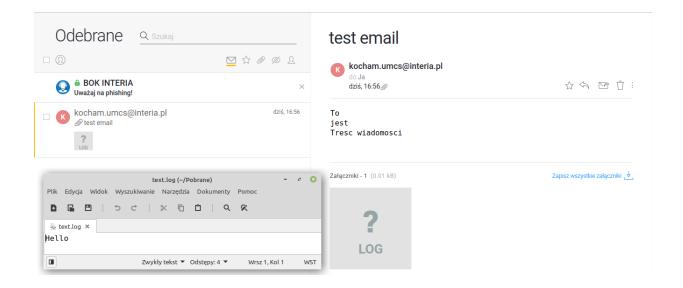
- 6.2 Możesz skorzystać z jednego z serwerów SMTP zaproponowanego w rozwiązaniu do zadania 6.1. Jaka komenda protokołu SMTP pozwala na określenie odbiorcy wiadomości?
- **6.3** Możesz skorzystać z jednego z serwerów SMTP zaproponowanego w rozwiązaniu do zadania **6.1**. Jaka komenda protokołu SMTP pozwala na określenie nadawcy wiadomości?
- 6.4 Możesz skorzystać z jednego z serwerów SMTP zaproponowanego w rozwiązaniu do zadania 6.1. Pamiętaj, że każda nowa linia ma znaczenie. Jeśli nie zachowasz odpowiedniej składni, serwer może "nie zrozumieć" wiadomości. Poniżej przykładowe rozwiązanie.

```
openssl s_client -crlf -connect poczta.interia.pl:465
CONNECTED(00000003)
depth=2 C = US, 0 = DigiCert Inc, 0U = www.digicert.com, CN = DigiCert Global Root CA
verify return:1
depth=1 C = US, 0 = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
verify return:1
depth=0 C = PL, L = Krakow, 0 = Grupa INTERIA.PL sp. z o.o. sp. k., CN = *.interia.pl
verify return:1
---
Certificate chain
0 s:C = PL, L = Krakow, 0 = Grupa INTERIA.PL sp. z o.o. sp. k., CN = *.interia.pl
i:C = US, 0 = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
1 s:C = US, 0 = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
1 s:C = US, 0 = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
---
Server certificate
----BEGIN CERTIFICATE----
MIIFYzCCBDigAwIBAgIQD/Ztrk8kirkvn+QyffFNbjAKBggqhkjDPQQDAzBWMQsw
CQYDVQQGEwJVUzEVMBMGAIUEChMMRGlnaUNlcnQgSW5jMTAwLgYDVQQDEydEaWdp
QVyydCBUTFNgSHlicmlkIEVDQyBTSEEzODQgMjAyMCBDQTEwHcNMjIwnzI3MDAw
MDAwWhcNMjNwnzI3MjM10TUSWjBiMQswCQYDVQQGewJQTDEPMAOGAIUEBxMGS3Jh
a293MSswKQYDVQQKEyJHcnVwYSBJT1RFUklBL1BMIHNwLiB6IGSuby4gc3AuIGsu
```

```
AATj14S/K9d1aInTO/N6wXhyj7/OYxfJlR7j0xE8C5JiUZpaip8/DDL7syoNB3xS
\texttt{LtJIpG1Ygqy9kRHr8wfIVOCzo4IDijCCA4YwHwYDVROjBBgwFoAUCrwIKReMpTlt}
 eg70M8cus+37w3owHQYDVR00BBYEFEzd3VGbvBiH2K8Z1aE6Py9C0HHNMCMGA1Ud
EQQcMBqCDCouaW5OZXJpYS5wbIIKaW5OZXJpYS5wbDAOBgNVHQ8BAf8EBAMCB4Aw
HQYDVRO1BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMIGbBgNVHR8EgZMwgZAwRqBE
oEKGQGhOdHA6Ly9jcmwzLmRpZ21jZXJOLmNvbS9EaWdpQ2VydFRMUOh5YnJpZEVD
Q1NIQTM4NDIwMjBDQTEtMS5jcmwwRqBEoEKGQGhOdHA6Ly9jcmwOLmRpZ21jZXJO
VROgBDcwNTAzBgZngQwBAgIwKTAnBggrBgEFBQcCARYbaHROcDovL3d3dy5kaWdp
Y2VydC5jb2ovQ1BTMIGFBggrBgEFBQcBAQR5MHcwJAYIKwYBBQUHMAGGGGhOdHA6
Ly9vY3NwLmRpZ21jZXJOLmNvbTBPBggrBgEFBQcwAoZDaHROcDovL2NhY2VydHMu
ZGlnaWNlcnQuY29tLORpZ2lDZXJOVExTSHlicmlkRUNDUOhBMzgOMjAyMENBMSOx
LmNydDAJBgNVHRMEAjAAMIIBfQYKKwYBBAHWeQIEAgSCAWOEggFpAWcAdQCt9776
TPRQJYLIAPZwePhhqtGcpXc+xDCTKhYYO609vCigAAAYJBhgsMAAAEAwBGMEQCTEL1
g7IxBfCpgigx/rDU7kUvWaYWvMxfOtRkTL2UHbSVAiBIGwYMqfuFuoThbrnyOstk
U4hwJoak4J69MvZ+HasnAQB2ADXPGRu/sWxXvw+tTG1Cy7u2JyAmUeo/4SrvqAPD
U9ZMAAABgkGGCtYAAAQDAEcwRQIhAPFjNAwBAB5vOGOQwOuaDWMF7n3n/7s2t7gZ
Nlhy6Z8sAiBcLcCvAP/llzyfk/5olh01c85cWB6+HD3if6BUoUNMlgB2ALNzdwfh
hFD4Y4bWBancEQ1KeS2xZwwLh9zwAw55NqWaAAABgkGGCwEAAAQDAEcwRQIgSsPk
\tt X+yBkecqv0CuIXmWg9V2e/s4LYyYvsuIJU7ksxgCIQDkrabch4m6Sg9U6AaqeW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW8MageW
vPtFrIjAzwSymXt2INyKTDAKBggqhkjOPQQDAwNpADBmAjEAv+cUUU3e9sdRaOFs
ntdpEVKb4S71HWCsVRZcS2MGIMOZRJ5LVEhhXHK1+EIaJ257AjEA8rSvrwL4MIRz
fCA7Oevopfs2fJcQVU4TLEbSFBwkAdhOLlHAnXH+NELH1ttjQtfh
 ----END CERTIFICATE--
subject=C = PL, L = Krakow, O = Grupa INTERIA.PL sp. z o.o. sp. k., CN = *.interia.pl
issuer=C = US, O = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: ECDSA
Server Temp Key: X25519, 253 bits
SSL handshake has read 2810 bytes and written 389 bytes
Verification: OK
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 256 bit
Secure Renegotiation IS {\tt NOT} supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
Post-Handshake New Session Ticket arrived:
SSL-Session:
       Protocol : TLSv1.3
                        : TLS_AES_256_GCM_SHA384
       Session-ID: F75D463B69633B1F1958D6CE033697EAC3AC78E1ABAA461599D03B4822ACCEE6
       Session-ID-ctx:
       Resumption PSK: 74B14972C5CC86292F1A2869821072E8A55177841455905EE18CA32CC871922F62A95CCA9B68450180CCD5D937D57981
       PSK identity: None
       PSK identity hint: None
       SRP username: None
       TLS session ticket lifetime hint: 7200 (seconds)
      TLS session ticket: 0000 - a9 09 45 e3 8a 1e 19 3c-8a f7 6c ae 0e 95 f1 62
                                                                                                          ..E....<..1...b
       0010 - ac 28 8e 1c aa 29 02 42-86 75 c5 88 5e 2f f3 c6
                                                                                                            .(...).B.u..^/..
       0020 - 44 87 f9 04 91 ce 7e 6c-1d 0f 18 9a be 7d 11 56
0030 - 73 71 be 37 61 bd 3e 8c-64 d2 7f d4 3c 7b 90 ab
                                                                                                           sq.7a.>.d...<{..
                                                                                                           .c...;...5.B...
.@[...@W.=^1...6
       0040 - fd 63 f7 c6 dd d5 3b a6-d6 cf 08 35 d8 42 88 a8
       0050 \ - \ be \ 40 \ 5b \ fa \ a9 \ a3 \ 40 \ 57 - 85 \ 3d \ 5e \ 6c \ b6 \ d7 \ 97 \ 36
       0060 - 4c 17 41 f1 d1 da e8 79-49 81 36 bc 95 5e 09 c4 0070 - 88 78 16 99 b0 28 e5 12-30 aa 5f f9 35 ac 2a c8
                                                                                                           L.A...yI.6..
                                                                                                            .x...(..0._.5.*.
       0080 - 79 68 51 93 0d 5d 5c 50-1a 5a ed 84 7f ee c1 7a
                                                                                                           yhQ..]¶.Z....z
       0090 - 10 d9 50 81 4c 9c 7d 75-9b f5 33 d7 09 85 43 1c \,
                                                                                                            ..P.L.}u..3...C
       00a0 - e0 54 ca e0 f9 bc 39 a3-55 c1 82 e2 82 96 a8 ef
                                                                                                            .T....9.U.....
       00b0 - a3 99 6e 61 e7 a5 e0 7f-c8 e9 77 16 14 ad 08 81
       00c0 - dd \ 93 \ df \ 4f \ db \ 13 \ d5 \ d0-5d \ 74 \ 7b \ 71 \ aa \ 12 \ 26 \ a2
                                                                                                           ...O....]t{q..&.
       Start Time: 1685976897
                       : 7200 (sec)
       Timeout
       Verify return code: 0 (ok)
       Extended master secret: no
       Max Early Data: 0
read R BLOCK
220 ESMTP INTERIA.PL
ehlo student
250-poczta.interia.pl
250 - \hbox{PIPELINING}
250-SIZE 157286400
250-AUTH PLAIN LOGIN PLAIN LOGIN PLAIN LOGIN
250-AUTH=PLAIN LOGIN PLAIN LOGIN PLAIN LOGIN
250-ENH AN CEDST ATUS CODES
250-8BITMIME
250-SMTPUTF8
```

```
250 CHUNKING
auth login
334 VXNIcm5hbWU6
a29jaGFtLnVtY3NAaW50ZXJpYS5wbA==
334 UGFzc3dvcmQ6
a29jaGFtLnVtY3MyMDIz
235 2.7.0 Authentication successful
mail from: <kocham.umcs@interia.pl>
250 2.1.0 Ok
rcpt to: <pas.umcs@poczta.fm>
250 2.1.5 0k
354 End data with <CR><LF>.<CR><LF>
to: <pas.umcs@poczta.fm>
from: <kocham.umcs@interia.pl>
subject: test email
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=sep
То
jest
Tresc wiadomosci
--sep
Content-Type: text/x-log; name=text.log
Content-Disposition: attachment; filename=text.log
Content-Transfer-Encoding: base64
SGVsbG8K
--sep
250 OK. ID: bb63e06c9c318562
quit
221 2.0.0 Bye
closed
```

Po sprawdzeniu swojej skrzynki pocztowej, powinieneś zobaczyć wysłaną wiadomość wraz z załącznikiem:



6.5 Możesz skorzystać z jednego z serwerów SMTP zaproponowanego w rozwiązaniu do zadania 6.1. Pamiętaj, że każda nowa linia ma znaczenie. Jeśli nie zachowasz odpowiedniej składni, serwer może "nie zrozumieć" wiadomości. Poniżej przykładowe rozwiązanie.

```
telnet interia.pl 587
Trying 217.74.65.23...
Connected to interia.pl.
Escape character is '-]'.
220 ESMTP INTERIA.PL
HELO student
AUTH LOGIN
cGFzMjAxNOBpbnRlcmlhLnBs
UDRTSW5mMjAxNw==
MAIL FROM: <pas2017@interia.pl>
RCPT TO: <pas2017@interia.pl>
RCPT TO: <kasiula.mazur@gmail.com>
DATA

From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
```

**6.6** Możesz skorzystać z jednego z serwerów SMTP zaproponowanego w rozwiązaniu do zadania **6.1**. Poniżej przykład nawiązania połączenia z serwerem (połączenie szyfrowane), i odebrania od serwera pierwszej wiadomości po połączeniu:

```
#!/bin/usr/env python3
import socket, ssl
def recv_all_until(secure_sock, crlf):
    data = ""
    while not data.endswith(crlf):
        data = data + secure sock.read(1).decode()
    return data
if __name__ == '__main__':
    HOST = 'poczta.interia.pl'
    PORT = 465
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((HOST, PORT))
    context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
    secure_sock = context.wrap_socket(sock)
    print(recv_all_until(secure_sock, '\r\n'))
    secure_sock.close()
    sock.close()
```

6.7 Możesz skorzystać z jednego z serwerów SMTP zaproponowanego w rozwiązaniu do zadania 6.1.

#### Protokół HTTP

7.1 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/pas-book-p1-ch7-ex1:latest
docker run -dp 7001:80 mazurkatarzyna/pas-book-p1-ch7-ex1
```

Działanie serwera możesz przetestować za pomocą przeglądarki. W swojej przeglądarce otwórz stronę: http://127.0.0.1:7001/html

7.2 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/pas-book-p1-ch7-ex2:latest
docker run -dp 7002:80 mazurkatarzyna/pas-book-p1-ch7-ex2
```

Działanie serwera możesz przetestować za pomocą przeglądarki. W swojej przeglądarce otwórz stronę: http://127.0.0.1:7002/image/png

7.3 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/pas-book-p1-ch7-ex3:latest
docker run -dp 7003:80 mazurkatarzyna/pas-book-p1-ch7-ex3
```

Działanie serwera możesz przetestować za pomocą przeglądarki. W swojej przeglądarce otwórz stronę: http://127.0.0.1:7003/image.jpg

7.4 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/pas-book-p1-ch7-ex4:latest docker run -dp 7004:80 mazurkatarzyna/pas-book-p1-ch7-ex4
```

Działanie serwera możesz przetestować za pomocą przeglądarki. W swojej przeglądarce otwórz stronę: http://127.0.0.1:7004/post