Programowanie aplikacji sieciowych

Zbiór zadań, część pierwsza

Katarzyna Mazur

Spis treści

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*

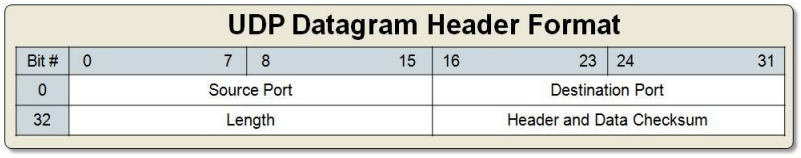
1.5 Napisz program, który jako argument linii poleceń pobierze od użytkownika adres `IPv4`, a następnie wyświetli odpowiadającą mu nazwę `hostname` (nazwę domenową). Zadanie rozwiąż bez użycia dodatkowych bibliotek. Zadbaj o prawidłową obsługę błędów.

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

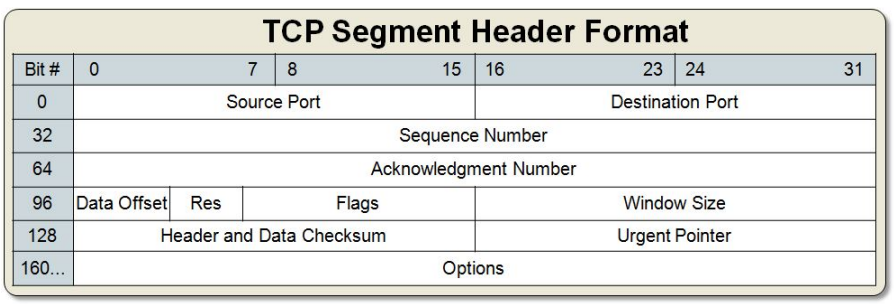
prześle do serwera UDP działającego na wskazanym porcie pod podanym adresem IPv4, w celu sprawdzenia, czy udało się prawidłowo odczytać wymagane pola. Serwer zwróci odpowiedź `TAK` lub `NIE`, a w przypadku błędnego sformatowania wiadomości, odeśle odpowiedź `BAD_SYNTAX`. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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, że w zapisie szesnastkowym jedna cyfra reprezentuje 4 bity, oraz znając strukturę segmentu TCP:



Napisz program, który z powyższego segmentu TCP wydobędzie:

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

A następnie uzyskany wynik w postaci: `zad2.2odp;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 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`. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). Zadbaj o prawidłową obsługę błędów.

3.12 Zmodyfikuj program z zadania **3.11** w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). Zadbaj o prawidłową obsługę błędów.

3.16 Zmodyfikuj program z zadania **3.15** w taki sposób, aby łączył się z serwerem posiadającym adres IPv6. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). 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. Zadanie rozwiąż bez użycia dodatkowych bibliotek (korzystaj jedynie z gniazd). Zadbaj o prawidłową obsługę błędów.