Dominik Kaczmarek, nr albumu 261757

Lista 5

~~~~~~

Plik server3.pl zawiera przykładowy program serwera protokołu HTTP.

1. Uruchom ten skrypt, przetestuj, zastanów się jak działa.
2. Nawiąż połączenie za pomocą przeglądarki internetowej.
3. Zmień skrypt (lub napisz własny serwer w dowolnym języku programowania) tak aby wysyłał do klienta nagłówek jego żądania.
4. Zmień skrypt (lub napisz własny serwer w dowolnym języku programowania) tak aby obsugiwał żądania klienta do prostego tekstowego serwisu WWW (kilka statycznych ston z wzajemnymi odwołaniami) zapisanego w pewnym katalogu dysku lokalnego komputera na którym uruchomiony jest skrypt serwera.
5. Przechwyć komunikaty do/od serwera za pomocą analizatora sieciowego - przeanalizuj ich konstrukcję.
6. Napisz zwięzłe sprawozdanie.

~~~~~~

Zad 1.

Utworzyłem plik index.html zawierający prostą stronę internetową. Jego zawartość jest na poniższym zdjęciu.

```
strony > index.html > html
<html>

<head>
<title>Server 3</title>
</head>

<body>
<p>Strona testowa stworzona na potrzeby zadania 1 z Listy 5</p>
</body>

</html>

```

Analizując plik server3.pl opisałem fragmenty kodu które były niejasne.

```
server3.pl
#!usr/bin/perl
use HTTP::Daemon;
use HTTP::Status;
#use IO::File;

# Serwer, który chce powiązać się z określonym adresem na ustalonym porcie HTTP, zostanie skonstruowany w następujący sposób:
my $d = HTTP::Daemon->new(
            LocalAddr => 'localhost',
            LocalPort => 4320,
        ) || die; # zatrzymujemy skrypt

print "Please contact me at: <URL:", $d->url, ">\n";

# d = serwer
# c = klient
# r = request = żądanie

while (my $c = $d->accept) { # Prawda kiedy połacznie z klientem jest dostępne.
    while (my $r = $c->get_request) { # Pobieramy żądania od klienta (naszej strony)
        # Pobieramy atrybut metody i sprawdzamy czy jest równy stringowi GET (inne możliwe atrybuty to HEAD, PUT, PATCH lub POST)
        if ($r->method eq 'GET') {

            $file_s = "./strony/index.html";    # index.html - jakis istniejacy plik
            $c->send_file_response($file_s);    # wysyłamy odpowiedź do klienta
        }
        else {
            $c->send_error(RC_FORBIDDEN)
        }
    }
    $c->close; # zamykamy klienta
    undef($c); # "oddefinujemy klienta"
}

```

~~~~~~
~~~~~~

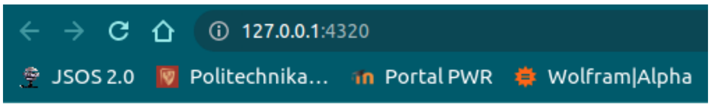
\~~~~~~

Zad 2

Odpalając program server3.pl wszedłem w link, który wyświetlił się w oknie terminala (Adres mojego serwera).

```
dominik@dominik-Nitro-AN515-55:~/Studia/TS/Lab5$ perl server3.pl
Please contact me at: <URL:http://127.0.0.1:4320/>
```

Otworzyła mi się strona internetowa z treścią identyczną jak we wcześniej stworzonym pliku index.html.



Strona testowa stworzona na potrzeby zadania 1 z Listy 5

\~~~~~~

Zad 3

Modyfikacje wprowadziłem wyłącznie między linijkami 18 – 32 pierwotnego kodu, dlatego umieszczam zrzut ekranu z tego fragmentu pliku po modyfikacji. Dodałem przydatne komentarze pozwalające szybciej zrozumieć zmiany.

```
18 while (my $c = $d->accept) {
19     while (my $r = $c->get_request) {
20         if ($r->method eq 'GET') {
21 
22             $file_s = "./strony/zad3.html";
23             # otwieramy plik do którego wpiszemy nagłówek żądania klienta
24             open(FH, '>', $file_s) or die $!;
25             # iterujemy po każdym polu nagłówka
26             foreach($r->header_field_names) {
27                 # wpisujemy do pliku każdy nagłówek wraz z jej wartością
28                 print FH ($_ . "   ----->   " . $r->header($_) . "<br>" );
29             }
30             close(FH);
31             $c->send_file_response($file_s);
32 
33         }
34         else {
35             $c->send_error(RC_FORBIDDEN)
36         }
37     }
38     $c->close;
39     undef($c);
40 }
```

Nagłówek żądania klienta lekko zmodyfikowałem, żeby łatwiej można było przeanalizować jego składowe.
Oto co wysłaliśmy z powrotem klientowi.

Connection -----> keep-alive
Accept -----> text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding -----> gzip, deflate, br
Accept-Language -----> pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Host -----> 127.0.0.1:4321
If-Modified-Since -----> Sun, 12 Jun 2022 16:03:11 GMT
User-Agent -----> Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36
Sec-Ch-Ua -----> " Not A;Brand";v="99", "Chromium";v="100", "Google Chrome";v="100"
Sec-Ch-Ua-Mobile -----> ?0
Sec-Ch-Ua-Platform -----> "Linux"
Sec-Fetch-Dest -----> document
Sec-Fetch-Mode -----> navigate
Sec-Fetch-Site -----> none
Sec-Fetch-User -----> ?1
Upgrade-Insecure-Requests -----> 1

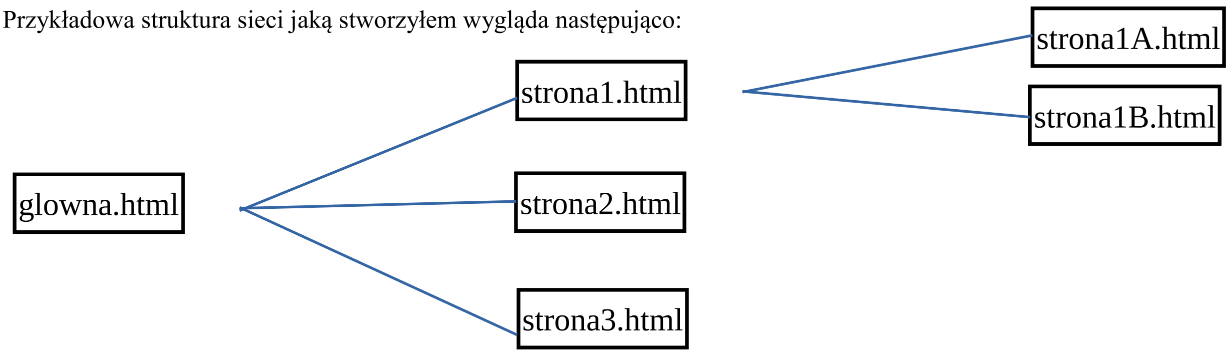
Listę nagłówków HTTP można znaleźć tu:
https://pl.wikipedia.org/wiki/Lista_nag%C5%82%C3%B3wk%C3%B3w_HTTP

\~~~~~~

\~~~~~~

Zad 4

Przykładowa struktura sieci jaką stworzyłem wygląda następująco:



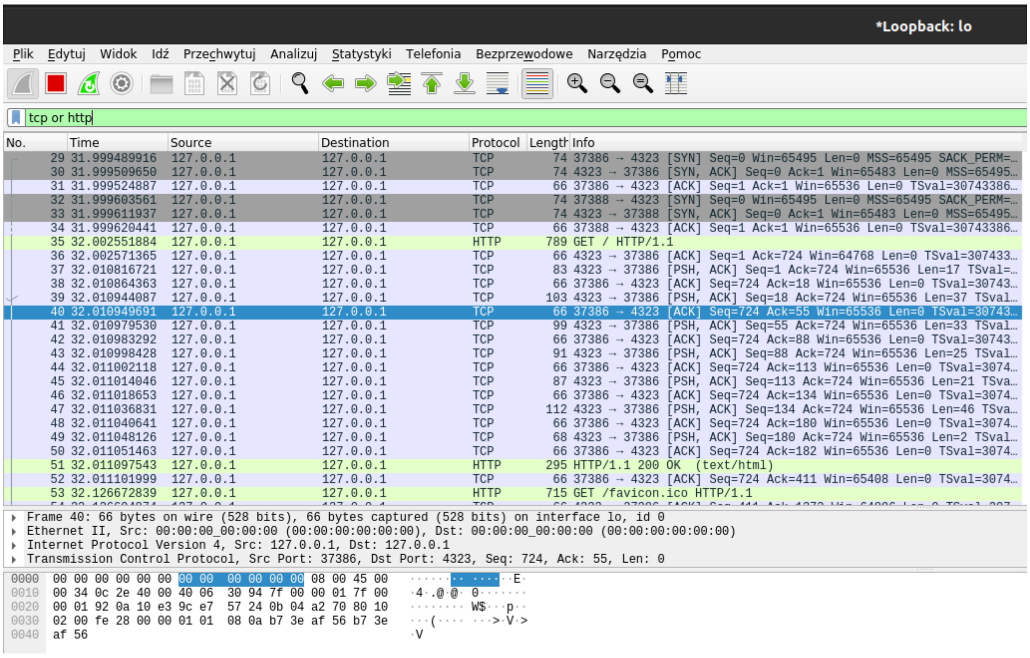
- Strona glowna.html zawiera hiperłącza do stron strona1.html, strona2.html, strona3.html,
- Strona strona1.html zawiera hiperłącza do stron strona1A.html, strona1B.html,
- Strony strona1A.html i strona1B.html zawierają hiperłącza do strony glowna.html.

Kod po dodaniu modyfikacji prezentuje się następująco.

```perl
my $dir = "./strony"; # katalog ze wszystkimi stronami

while (my $c = $d->accept) {
    while (my $r = $c->get_request) {
        if ($r->method eq 'GET') {

            # patrzymy czy znajdujemy się na stronie głównej = adres ma postać http://127.0.0.1:4323/
            if($r->uri eq "/") {

                my $file_s = "./strony/glowna.html";
                $c->send_file_response($file_s);

            } else { # znajdujemy się na którejś z podstron = adres ma postać http://127.0.0.1:4323/strona(X).html

                $file_s = $dir.$r->uri;
                $c->send_file_response($file_s);
            }
        }
        else {
            $c->send_error(RC_FORBIDDEN)
        }
    }
    $c->close;
    undef($c);
}
```

~~~~~~

Zad 5


~~~~~~