

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 nagłówki jego żądania.
4. Zmień skrypt (lub napisz własny serwer w dowolnym języku programowania) tak aby obsługiwał żądania klienta do prostego tekstowego serwisu WWW (kilka statycznych stron 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
1  <html>
2
3  <head>
4  <title>Server 3</title>
5  </head>
6
7  <body>
8  <p>Strona testowa stworzona na potrzeby zadania 1 z Listy 5</p>
9  </body>
10
11 </html>
12
```

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

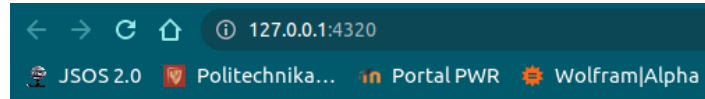
```
server3.pl
1  #!/usr/bin/perl
2  use HTTP::Daemon;
3  use HTTP::Status;
4  #use IO::File;
5
6  # Serwer, który chce powiązać się z określonym adresem na ustalonym porcie HTTP, zostanie skonstruowany w następujący
   sposób:
7  my $d = HTTP::Daemon->new(
8      LocalAddr => 'localhost',
9      LocalPort => 4320,
10     ) || die; # zatrzymujemy skrypt
11
12  print "Please contact me at: <URL:", $d->url, ">\n";
13
14  # d = serwer
15  # c = klient
16  # r = request = żądanie
17
18  while (my $c = $d->accept) { # Prawda kiedy połączenie z klientem jest dostępne.
19     while (my $r = $c->get_request) { # Pobieramy żądania od klienta (naszej strony)
20         # Pobieramy atrybut metody i sprawdzamy czy jest równy stringowi GET (inne możliwe atrybuty to HEAD, PUT,
           PATCH lub POST)
21         if ($r->method eq 'GET') {
22             $file_s = "./strony/index.html"; # index.html - jakiś istniejący plik
23             $c->send_file_response($file_s); # wysyłamy odpowiedź do klienta
24         }
25         else {
26             $c->send_error(RC_FORBIDDEN)
27         }
28     }
29 }
30 $c->close; # zamykamy klienta
31 undef($c); # "oddefiniujemy klienta"
32 }
33
```

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

Modifikacje 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 wpisujemy 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.

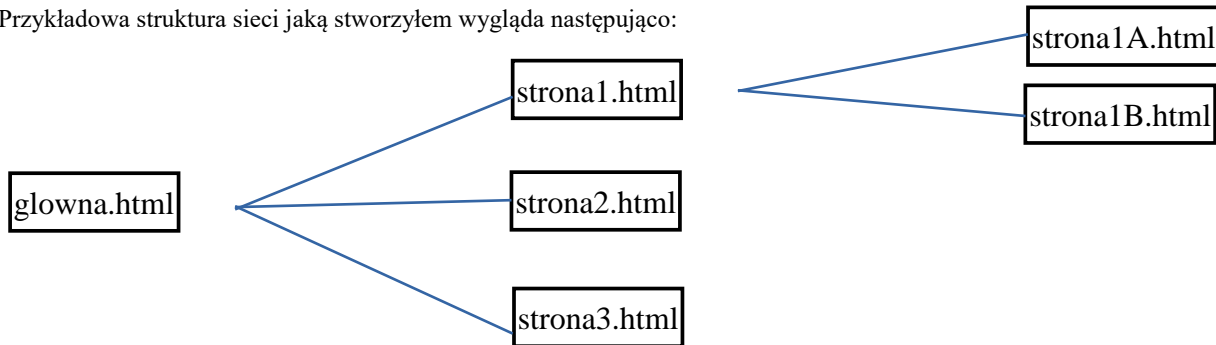
```
Connection -----> keep-alive
Accept -----> text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding -----> gzip, deflate, br
Accept-Language -----> pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Host -----> 127.0.0.1:4321
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:



- ⑩ Strona glowna.html zawiera hiperłącza do stron strona1.html, strona2.html, strona3.html,
- ⑩ Strona strona1.html zawiera hiperłącza do stron strona1A.html, strona1B.html,
- ⑩ Strony strona1A.html i strona1B.html zawierają hiperłącza do strony glowna.html.

Kod po dodaniu modyfikacji prezentuje się następująco.

```

14 my $dir = "./strony"; # katalog ze wszystkimi stronami
15
16 while (my $c = $d->accept) {
17     while (my $r = $c->get_request) {
18         if ($r->method eq 'GET') {
19
20             # patrzmy czy znajdujemy się na stronie głównej = adres ma postać http://127.0.0.1:4323/
21             if ($r->uri eq "/" ) {
22
23                 my $file_s = "./strony/glowna.html";
24                 $c->send_file_response($file_s);
25
26             } else { # znajdujemy się na którejś z podstron = adres ma postać http://127.0.0.1:4323/strona(X).html
27
28                 $file_s = $dir.$r->uri;
29                 $c->send_file_response($file_s);
30             }
31         }
32     } else {
33         $c->send_error(RC_FORBIDDEN)
34     }
35 }
36 $c->close;
37 undef($c);
38 }
  
```

Zad 5

*Loopback: lo

Plik Edytuj Widok Idź Przechwytyj Analizuj Statystyki Telefonnia Bezprzewodowe Narzędzia Pomoc

tcp or http

No.	Time	Source	Destination	Protocol	Length	Info
29	31.999489916	127.0.0.1	127.0.0.1	TCP	74	37386 → 4323 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=...
30	31.999509650	127.0.0.1	127.0.0.1	TCP	74	4323 → 37386 [ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495...
31	31.999524887	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=30743386...
32	31.999603561	127.0.0.1	127.0.0.1	TCP	74	37388 → 4323 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=...
33	31.999611937	127.0.0.1	127.0.0.1	TCP	74	4323 → 37388 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495...
34	31.999620441	127.0.0.1	127.0.0.1	TCP	66	37388 → 4323 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=30743386...
35	32.002551884	127.0.0.1	127.0.0.1	HTTP	789	GET / HTTP/1.1
36	32.002571365	127.0.0.1	127.0.0.1	TCP	66	4323 → 37386 [ACK] Seq=1 Ack=724 Win=64768 Len=0 TSval=307433...
37	32.010816721	127.0.0.1	127.0.0.1	TCP	83	4323 → 37386 [PSH, ACK] Seq=1 Ack=724 Win=65536 Len=17 TSval=...
38	32.010846363	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=724 Ack=18 Win=65536 Len=0 TSval=30743...
39	32.010944087	127.0.0.1	127.0.0.1	TCP	193	4323 → 37386 [PSH, ACK] Seq=18 Ack=724 Win=65536 Len=37 TSval=...
40	32.010949051	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=724 Ack=55 Win=65536 Len=0 TSval=30743...
41	32.010979530	127.0.0.1	127.0.0.1	TCP	99	4323 → 37386 [PSH, ACK] Seq=55 Ack=724 Win=65536 Len=33 TSval=...
42	32.010983292	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=724 Ack=88 Win=65536 Len=0 TSval=30743...
43	32.010998428	127.0.0.1	127.0.0.1	TCP	91	4323 → 37386 [PSH, ACK] Seq=88 Ack=724 Win=65536 Len=25 TSval=...
44	32.011002118	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=724 Ack=113 Win=65536 Len=0 TSval=3074...
45	32.011014046	127.0.0.1	127.0.0.1	TCP	87	4323 → 37386 [PSH, ACK] Seq=113 Ack=724 Win=65536 Len=21 TSva...
46	32.011018653	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=724 Ack=134 Win=65536 Len=0 TSval=3074...
47	32.011036831	127.0.0.1	127.0.0.1	TCP	112	4323 → 37386 [PSH, ACK] Seq=134 Ack=724 Win=65536 Len=46 TSva...
48	32.011040641	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=724 Ack=180 Win=65536 Len=0 TSval=3074...
49	32.011048126	127.0.0.1	127.0.0.1	TCP	68	4323 → 37386 [PSH, ACK] Seq=180 Ack=724 Win=65536 Len=2 TSval=...
50	32.011051463	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=724 Ack=182 Win=65536 Len=0 TSval=3074...
51	32.011097543	127.0.0.1	127.0.0.1	HTTP	295	HTTP/1.1 200 OK (text/html)
52	32.011101999	127.0.0.1	127.0.0.1	TCP	66	37386 → 4323 [ACK] Seq=724 Ack=411 Win=65408 Len=0 TSval=3074...
53	32.126572839	127.0.0.1	127.0.0.1	HTTP	715	GET /favicon.ico HTTP/1.1

Frame 40: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface lo, id 0

Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 37386, Dst Port: 4323, Seq: 724, Ack: 55, Len: 0

```

0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....E
0010  00 34 0c 2e 40 00 00 06 30 94 7f 00 00 01 7f 00  -4..@.0.....
0020  00 01 92 0a 10 e3 9c e7 57 24 0b 04 a2 70 08 10  .....WS...p...
0030  02 00 fe 28 00 00 01 01 08 0a b7 3e af 56 b7 3e  ...(..>V>
0040  af 56
  
```