Wrocław, 13.06.2017

Laboratorium

Technologii Sieciowych

Temat:

Sprawozdanie 5

Autor: Bartosz Banasik
Informatyka, semestr: 4
Prowadzący: dr Łukasz Krzywiecki

Spis treści:

1. Cel pracy

Celem zajęć laboratoryjnych jest przetestowanie działania prostego serwera `HTML` napisanego w języku `Perl`. Jego odpowiednia modyfikacja oraz przebudowa w celu zrozumienia działania protokołu `HTML`.

2. Realizacja

Głównym przedmiotem badań będzie następujący skrypt napisany w języku `Perl`, demonstrujący działanie serwera `HTML`. (listing 1)

2.1 Opis programu

Działanie skryptu przedstawionego w listingu 1 opiera się na o mechanizm udostępniany przez bibliotekę `libwww-perl`. Na początku tworzona jest nowa instancja obiektu typu `HTTP::Daemon`. Obiekt ten jest serwerem `HTTP/1.1`, który nasłuchuje na adresie `localhost`, gnieździe `4321` nadchodzące żądania. Następnie wyświetlana jest informacja pod jakim adresem stacjonuje nasz serwer. W linii 13 rozpoczyna się pierwsza pętla programu. Konstrukcja `my $c = $d -> accept` zapisuje w zmiennej `$c` wskaźnik do obiektu `HTTP::Daemon::ClientConn`. Wykonanie instrukcji `my $r = $c -> get request` na tym obiekcie (linia 14) powoduje zapisanie w zmiennej `$r` wskaźnika do obiektu `HTTP::Request`. W linii 15 następuje sprawdzenie, czy żądanie zostało wykonane metodą `GET`. Jeśli nie, to do klienta zostaje przesłana informacja o błędzie (linia 21). W przypadku gdy zapytanie było prawidłowe, klient otrzyma treść pliku index.html, który zapisany jest w bieżącym katalogu.

```
 1   use HTTP::Daemon;
 2   use HTTP::Status;
 3   #use IO::File;
 4
 5   my $d = HTTP::Daemon->new(
 6            LocalAddr => 'localhost',
 7            LocalPort => 4321,
 8         )|| die;
 9
10   print "Please contact me at: <URL:", $d->url, ">\n";
11
12
13   while (my $c = $d->accept) {
14       while (my $r = $c->get_request) {
15           if ($r->method eq 'GET') {
16
17               $file_s= "./index.html";    # index.html - jakis istniejacy
plik
18               $c->send_file_response($file_s);
19
20           }
21           else {
22               $c->send_error(RC_FORBIDDEN)
23           }
24
25       }
26       $c->close;
27       undef($c);
28   }
```

listing 1: Program server3.pl

2.2 Prosty test serwera

Spróbujemy przetestować nasz serwer w dwóch metodach.

1. Przy użyciu usługi internetowej `Telnet.`

2. Za pomocą przeglądarki internetowej `Firefox`.

1. Po uruchomieniu skryptu, nawiązano połączenie za pomocą programu `telnet` przez port `4321`. Następnie przesłano znak `'Y'` i potwierdzono kombinacją znaków `CR LF`. Odpowiedź serwera `HTTP` przedstawia `listing 2`. Listing 2 przedstawia nagłówek `HTTP/1.1` wraz ze stroną o treści `Bad Request`. Uzyskano taką odpowiedź, ponieważ skrypt `server3.pl` traktuje wszystkie zapytania typu innego niż `GET` za nieprawidłowe. Przeprowadzono kolejny eksperyment. Nawiązano połączenie, takie samo jak przy poprzedniej próbie, jednak wysłano treść `'GET /'` i potwierdzono ją kombinacją znaków `CR LF`. Odpowiedź serwera przestawia listing 3.

```
Input: $telnet localhost 4321

Output:
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Input: Y

Output:
HTTP/1.1 400 Bad Request
Date: Tue, 13 Jun 2017 17:52:04 GMT
Server: libwww-perl-daemon/6.01
Content-Type: text/html
Content-Length: 57

<title>400 Bad Request</title>
<h1>400 Bad Request</h1>

Connection closed by foreign host.
```

listing 2: Odpowiedź serwera HTTP

```
Input: $telnet localhost 4321

Output:
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Input: GET /

Output:
Hello hello,
HTTP Server,
Some file here.
I Hate programming...
I Hate programming...
I Hate programming...
IT WORKS!!!
I love programming...
Connection closed by foreign host.
```

listing 3: Odpowiedź serwera HTTP na zapytanie 'GET /'

Jak widzimy w odpowiedzi serwer przesłał nam zawartość pliku umieszczonego pod nazwą `index.html` w tym samym katalogu co podany skrypt. Na listingu 4 znajduje się zawartość pliku. Odpowiedź jest zatem prawidłowa. Przesłana została prawidłowa zawartość. Oznacza to, że skrypt działa poprawnie.

```
  1 Hello hello,
  2 HTTP Server,
  3 Some file here.
  4 I Hate programming...
  5 I Hate programming...
  6 I Hate programming...
  7 IT WORKS!!!
  8 I love programming...
```

listing 4: zawartość index.html

2. Otwieramy przeglądarkę internetową `firefox` i przechodzimy na stronę `localhost:4321`. Przeglądarka automatycznie wygeneruje dla nas zapytanie i wyśle je do serwera. Domyślnie jest to zapytanie ‘`GET`’. W odpowiedzi przeglądarka wyświetla nam następującą treść. (listing 5)

```
Hello hello, HTTP Server, Some file here. I Hate programming... I
Hate programming... I Hate programming... IT WORKS!!! I love
programming...
```

listing 5: Odpowiedź serwera na zapytanie przeglądarki

Widzimy, że przeglądarka prawidłowo wysłała żądanie ‘GET’ oraz odebrała je i wyświetliła dla użytkownika.

2.3 Modyfikacja serwera.

Celem tego zadania jest odpowiednia modyfikacja skryptu `serwer3.pl` tak, aby niezależnie od żądania zwracał on jego nagłówek . Listing 6 przedstawia skrypt po modyfikacji. Kolorem fioletowym zaznaczone są linie skryptu, które uległy zmianie.

```
 1   use HTTP::Daemon;
 2   use HTTP::Status;
 3   #use IO::File;
 4
 5   my $d = HTTP::Daemon->new(
 6             LocalAddr => 'localhost',
 7             LocalPort => 4321,
 8         )|| die;
 9
10   print "Please contact me at: <URL:", $d->url, ">\n";
11
12
13   while (my $c = $d->accept) {
14       while (my $r = $c->get_request) {
15
16
17         $response = HTTP::Response->new(200,'OK');
18         $response->content($r->as_string);
19         $c->send_response($response);
20
21       }
22       $c->close;
23       undef($c);
24   }
```

listing 6: Skrypt zwracający nagłówki.

Skrypt różni się jedynie zawartością drugiej pętli. Zrezygnowano tutaj ze sprawdzania, czy metoda zapytania jest typu ‘GET’. W celu wysłania odpowiedzi, należy ją najpierw przygotować, co następuje w linii 17. Treść zapytania w formie tekstowej (`$r->as_string`) zostaje dołączona do obiektu `$response`. W linii 19 następuje przesłanie przygotowanej odpowiedzi. Działanie skryptu zamieszczonego na listingu 7. Skrypt przetestowano używając przeglądarki internetowej `firefox`. W pasku adresu ponownie wpisano adres serwera `localhost:4321`. Odpowiedź wyświetloną przez program przedstawia listing 7.

```
GET / HTTP/1.1
1:Cache-Control: max-age=0
2:Connection: keep-alive
3:Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
4:Accept-Encoding: gzip, deflate
5:Accept-Language: en-US,en;q=0.5
6:Host: localhost:4321
7:If-Modified-Since: Sun, 11 Jun 2017 21:29:29 GMT
8:User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:53.0) Gecko/20100101
9:Firefox/53.0
10:Upgrade-Insecure-Requests: 1
```

listing 7: Odpowiedź zmodyfikowanego serwera.

Serwer zwrócił nagłówek żądania. W pierwszej linii widoczne jest, że program `Firefox` wysłał żądanie typu `GET`. Ponieważ w przesyłanym adresie nie podano nazwy żadnego konkretnego pliku, to program wybiera domyślny katalog, czyli `/`. Druga linia zawiera informacje o rodzaju nawiązanego połączenia. W tym wypadku jest to `keep-alive`. Linia 3 zawiera akceptowane przez przeglądarkę treści, linia 4 - informacje na temat akceptowanego kodowania znaków, linia 5 - informacje na temat kodowania danych, a linia 6 akceptowane języki. W linii 7 znajduje się pełny adres serwera `HTTP`. Linia 8, oznaczona tagiem `User-Agent`, zawiera informacje o programie przy pomocy którego dokonano zapytania. Ostatnia linia zawiera wartość `Keep-Alive`, czyli czas w jakim kolejne zapytania do serwera będą wykonywane przy pomocy tego samego połączenia. Zmodyfikowany skrypt działa poprawnie i spełnia założenia zadania, t.j. zwraca nagłówki żądań wysyłane przez progamy klienckie.

2.4 Rozszerzenie działalności serwera

Kolejnym zadaniem było rozszerzenie działalności serwera w taki sposób 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. Po raz kolejny wykorzystano skrypt `server3.pl`. Dokonano niewielkich zmian, które umożliwiają przesyłanie żądanego pliku znajdującego się w wyznaczonym katalogu. Listing 11 przedstawia wyżej opisany skrypt. Odpowiednio zmienna `$SOURCE_PATH` przetrzymuje lokalizacje naszych trzech plików. Są to: `index.html`, `filmy.html` oraz `ksiazki.html` (listing 8, 9, 10). Kolorem niebieskim zaznaczyłem odnośniki to kolejnych stron.

```
 1 <!DOCTYPE html>
 2 <html>
 3     <body>
 4         <h3>To jest glowna strona!!</h3>
 5         <p>Podziele sie w wami tutuaj ksiazkami oraz filmami, ktore
                    lubie</p>
 6         <ol>
 7             <li><a href="filmy.html">Zbior filmow</a></li>
 8             <li><a href="ksiazki.html">Zbior ksiazek</a></li>
 9         </ol>
10     </body>
11 </html>
```

listing 8: Tresc pliku index.html

```
 1 <!DOCTYPE html>
 2 <html>
 3     <head>
 4         <title>Zbior moich ulubionych filmow</title>
 5     </head>
 6     <body>
 7
 8         <h3>Spis moich ulubionych filmow</h3>
 9         <ol>
10             <li>Titanic</li>
11             <li>Avengers</li>
12             <li>Avatar</li>
13             <li>Zielona Mila</li>
14             <li>Taxi 3</li>
15             <li>Wilk z Wallstreet</li>
16         </ol>
17
18         <p> A tutaj maly dodatek matematyczny</p>
19         <p>
20             Twierdzenie Pitagorasa:
21             <math>
22                 <mrow>
23                     <msup><mi>a</mi><mn>2</mn></msup>
24                     <mo>+</mo>
25                     <msup><mi>b</mi><mn>2</mn></msup>
26                     <mo>=</mo>
27                     <msup><mi>c</mi><mn>2</mn></msup>
28                 </mrow>
29             </math>
30         </p>
31         <p>Zielony okrag:</p>
32         <p>
33             <svg>
34                 <circle r="50" cx="50" cy="50" fill="green" />
35             </svg>
36         </p>
37
38         <a href="index.html">Powrot do strony glownej</a>
39
40     </body>
41 </html>
```

listing 9: Treść pliku filmy.html

```
 1 <!DOCTYPE html>
 2 <html>
 3     <head>
 4         <title>Zbior moich ulubionych ksiazek</title>
 5     </head>
 6     <body>
 7 
 8         <h3>Spis moich ulubionych ksiazek</h3>
 9         <ol>
10             <li>Saga Wiedzmin</li>
11             <li>Ani slowa prawdy</li>
12             <li>Pani jeziora</li>
13             <li>Jakub Wedrowycz</li>
14             <li>Zloty kompas</li>
15             <li>Morfina</li>
16         </ol>
17 
18         <p> A tutaj maly dodatek fizyczny</p>
19         <p>
20             Rownowaznosc masy i energii
21             <math>
22                 <mrow>
23 
24                     <mo>e</mo>
25 
26                     <mo>=</mo>
27 
28                     <mo>m</mo>
29 
30                     <mo>*</mo>
31 
32                     <msup><mi>c</mi><mn>2</mn></msup>
33 
34                 </mrow>
35             </math>
36         </p>
37 
38         <p>Niebieski wielokat:</p>
39         <p>
40             <svg height="210" width="500">
41                 <polygon points="200,10 250,190 160,210"
42                     style="fill:blue;stroke:purple;stroke-width:1" />
43             </svg>
44         </p>
45         <a href="index.html">Powrot do strony glownej</a>
46     </body>
47 </html>
```

listing 10: Treść pliku ksiazki.html

```
 1 use HTTP::Daemon;
 2 use HTTP::Status;
 3
 4 #domyslny folder z plikami
 5 $SOURCE_PATH = '/home/aedd/www';
 6
 7 my $d = HTTP::Daemon->new(
 8         ReuseAddr => 1,
 9         LocalAddr => 'localhost',
10         LocalPort => 4321,
11     ) || die;
12
13 print "Please contact me at : <URL:", $d->url,">\n";
14
15 while (my $c = $d->accept) {
16     while (my $r = $c->get_request){
17         if ($r->method eq 'GET'){
18             $file_s = $r->uri;
19
20             if ($file_s eq "/") {
21                 $file_s = "/index.html";
22             }
23
24             $c->send_file_response($SOURCE_PATH.$file_s);
25         }
26         else {
27             $c->send_error(RC_FORBIDDEN)
28         }
29     }
30     $c->close;
31     undef($c);
32 }
```

listing 11: Zmodyfikowany skrypt serwera HTTP

Skrypt po modyfikacji zawiera w linii 5 zmienną `$SOURCE_PATH`, która przechowuje ścieżkę do katalogu na dysku, w którym znajdują się pliki `HTML`. W pętli rozpoczynającej się od linii 16 sprawdzany jest warunek, czy zapytanie wykonane zostało metodą `GET`. Jeśli tak, to w zmiennej `$file_s` zapisany zostaje adres pliku podany w zapytaniu. Jeśli nie podano żadnego konkretnego pliku, to `$file_s` zawiera jedynie znak '/'. W takim wypadku należy wyświetlić domyślną stronę, czyli `index.html` (linie 20-22). W lini 24 następuje wysłanie zawartości pliku o nazwie podanej w żądaniu klienta i znajdującym się w katalogu `/home/aedd/www`.

Zmodyfikowany skrypt serwera działa dobrze i wykonuje prawidłowo swoje zadania.

2.5 Podgląd komunikatów przesyłanych pomiędzy klientem a serwerem HTTP

Ostatnim punktem tego ćwiczenia jest sprawdzenie komunikatów przesyłanych pomiędzy serwerem a klientem. W tym celu użyto programu `Wireshark`. Nasłuchiwanie będzie się odbywać na interfejsie `Loopback`. Dodatkowo ustawimy filtr na `HTTP`. Dzięki opcji '`follow http stream`' możemy odczytać całą komunikację. Przykładowy `output` zawiera listing 12.

```
GET / HTTP/1.1
Host: localhost:4321
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:53.0) Gecko/20100101
Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Wed, 14 Jun 2017 00:06:48 GMT
Server: libwww-perl-daemon/6.01
Content-Type: text/html
Content-Length: 272
Last-Modified: Mon, 12 Jun 2017 21:51:15 GMT

<!DOCTYPE html>
<html>
.<body>
..<h3>To jest glowna strona!!</h3>
..<p>Podziele sie w wami tutuaj ksiazkami oraz filmami, ktore lubie</p>
..<ol>
...<li><a href="filmy.html">Zbior filmow</a></li>
...<li><a href="ksiazki.html">Zbior ksiazek</a></li>
..</ol>
.</body>
</html>
```

listing 12: Przykładowa analiza połaczeń (Wireshark)

3. Wnioski

Podstawa działania serwera protokołu HTTP jest bardzo prosta. Usługa tego rodzaju daje olbrzymie możliwości, które stały się podstawą dzisiejszego Internetu. Protokół HTTP ma prostą, tekstową budowę, dzięki czemu jego implementacja jest prosta i jednoznaczna.