Sprawozdanie 5

Józef Piechaczek

2019-06-09

1 Omówienie skryptu

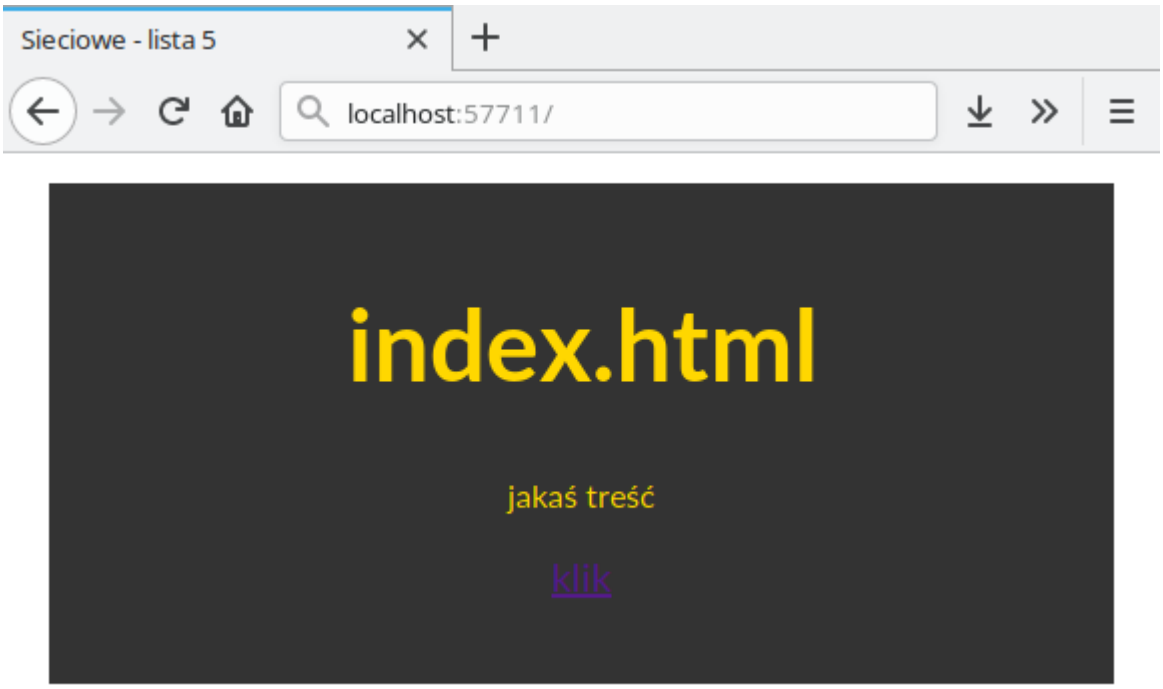
Podany skrypt napisany jest w języku Perl. Dzięki obiektowi *HTTP::Daemon* umożliwia on utworzenie prostego serwera HTTP. Skrypt akceptuje przychodzące połączenie za pomocą metody *$d->accept*. Zapytanie wysłane przez użytkownika trafia do zmiennej *$r* i jeśli jest równe *GET* serwer odpowiada na zapytanie, wysyłając plik *index.html*. Jeśli zapytanie jest różne od *GET* serwer odpowiada wysyłając błąd *403*.

Kod programu:

```
use HTTP::Daemon;
use HTTP::Status;

my $d = HTTP::Daemon->new(
          LocalAddr => 'lukim',
          LocalPort => 4321,
      ) || die;

print "Please contact me at: <URL:", $d->url, ">\n";
while (my $c = $d->accept) {
    while (my $r = $c->get_request) {
        if ($r->method eq 'GET') {

            $file_s= "./index.html";
            $c->send_file_response($file_s);
        } else {
            $c->send_error(RC_FORBIDDEN)
        }
    }
    $c->close;
    undef($c);
}
```

2 Nawiązanie połączenie za pomocą przeglądarki internetowej

W celu nawiązania połączenia posłużono się przeglądarką *Mozilla Firefox*. W folderze w którym znajdował się skrypt utworzono plik *index.html* zawierający źródło pewnej strony internetowej. Skrypt edytowano tak, aby tworzył serwer pod adresem *127.0.0.1* i pod losowym portem. Po wpisaniu w adres przeglądarki adresu *localhost:<port >* uzyskano następujący wynik:

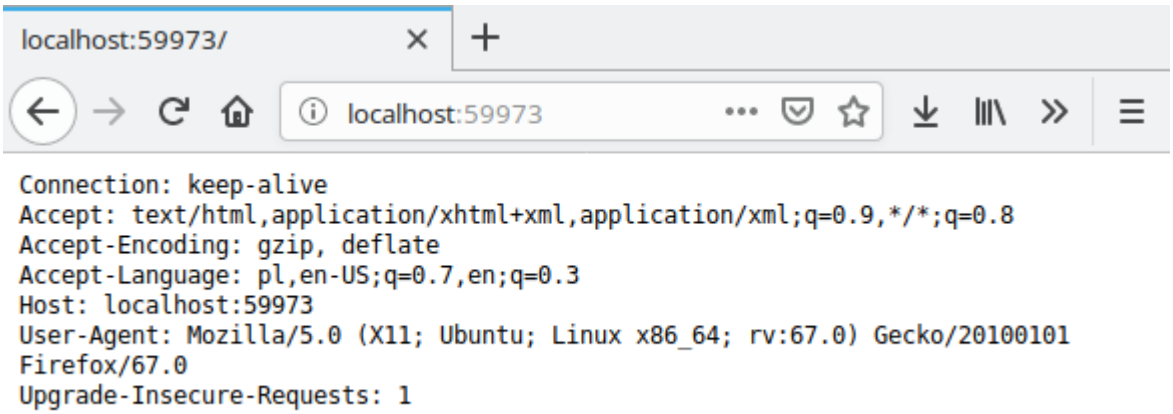


3 Wysyłanie nagłówka żądania

W celu realizacji zadania zmodyfikowano główną pętlę programu w następujący sposób:

```
$response = HTTP::Response->new;
$response->code(200);
$response->content($r->headers_as_string);
$c->send_response($response);
```

Na początku tworzony jest obiekt *HTTP:Response*, który umożliwia utworzenie odpowiedzi. Jako kod odpowiedzi wybrano *200*, czyli *OK*. Jako treść odpowiedzi ustawiono nagłówek żądania klienta, a następnie wysłano odpowiedź. Po nawiązania połączenia przy użyciu przeglądarki uzyskano następujący wynik:

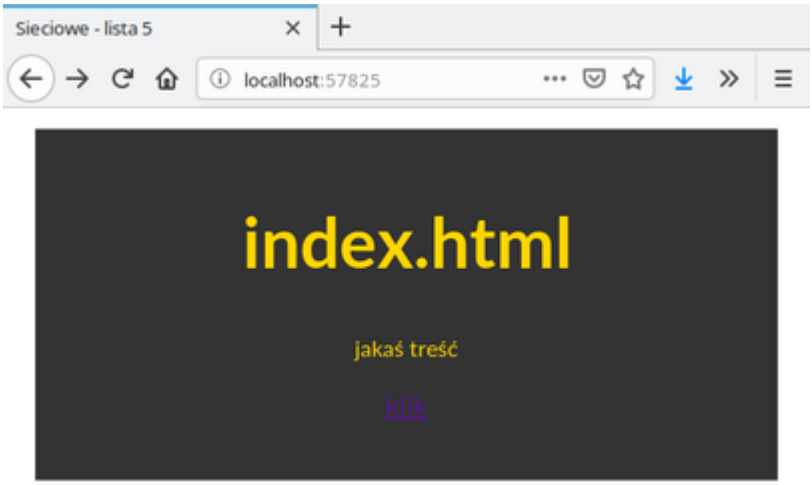


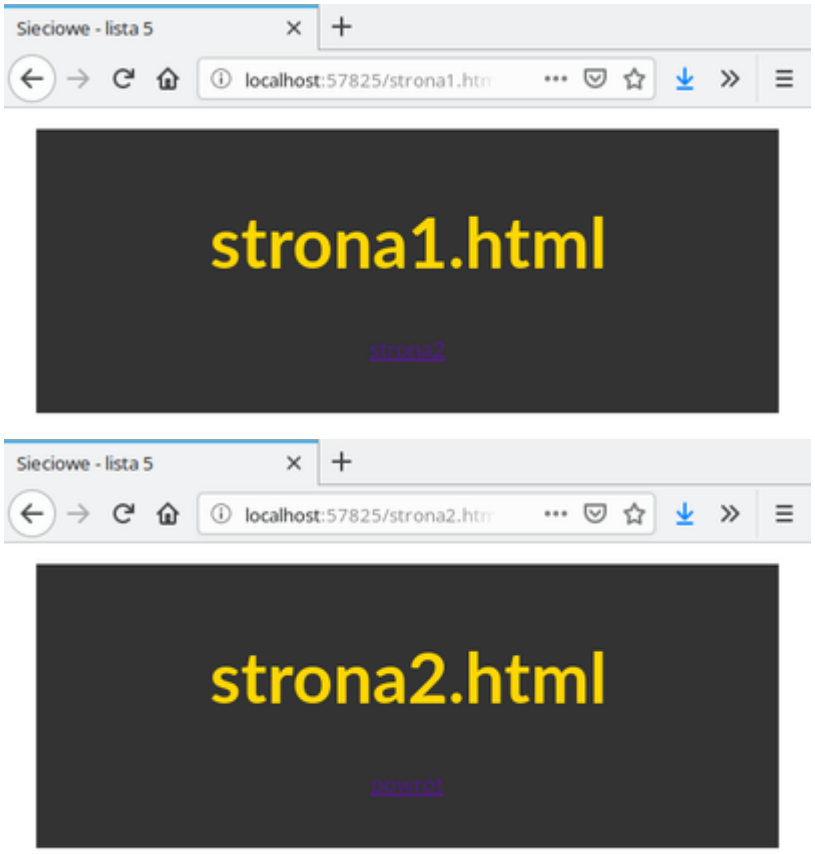
4 Obsługiwanie odwołań

W celu realizacji zadania zmodyfikowano główną pętlę programu w następujący sposób:

```
if ($r->method eq 'GET') {
    if ($r->uri eq '/') {
        $path = "./www/index.html";
    } else {
        $path = "./www" . $r->uri;
    }
    $c->send_file_response($path);
} else {
    $c->send_error(RC_FORBIDDEN)
}
```

Jeśli metoda HTTP to *GET* sprawdzana jest żądana ścieżka. Jeśli nie została ona podana, wysyłany jest plik *index.html*, w przeciwnym wypadku żądany plik, znajdujący się w katalogu *./www*. Jeśli żądana metoda jest inna niż *GET* wysyłany jest błąd 403. Uzyskano następujący wynik:





Jak można zauważyć, za pomocą łączy można przenieść się do poszczególnych podstron.

5 Przechwytywanie zapytań przy pomocy programu WireShark

Przy pomocy programu WireShark przechwytane zostały komunikaty pomiędzy klientem i serwerem. Na poniższej grafice widać listę przykładowych pakietów:

```
1867 86.672792496  127.0.0.1          127.0.0.1          HTTP      476 GET / HTTP/1.1
1882 86.674095182  127.0.0.1          127.0.0.1          HTTP      998 HTTP/1.1 200 OK  (text/html)
1888 87.710660684  127.0.0.1          127.0.0.1          HTTP      446 GET /strona1.html HTTP/1.1
1899 87.713204594  127.0.0.1          127.0.0.1          HTTP      963 HTTP/1.1 200 OK  (text/html)
1901 89.131731151  127.0.0.1          127.0.0.1          HTTP      458 GET /strona2.html HTTP/1.1
1916 89.133612301  127.0.0.1          127.0.0.1          HTTP      961 HTTP/1.1 200 OK  (text/html)
```

Przykładowy pakiet z żądaniem *GET*:

```
▾ Hypertext Transfer Protocol
  ▸ GET / HTTP/1.1\r\n
    Host: localhost:47785\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:67.0) Gecko/20100101 Firefox/67.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: pl,en-US;q=0.7,en;q=0.3\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    If-Modified-Since: Sat, 08 Jun 2019 22:26:46 GMT\r\n
    Cache-Control: max-age=0\r\n
    \r\n
    [Full request URI: http://localhost:47785/]
    [HTTP request 7/9]
    [Prev request in frame: 441]
    [Response in frame: 1882]
    [Next request in frame: 1888]
```

W pakiecie można dostrzec np. nagłówek żądania, badany w jednym z poprzednich zadań oraz żądany adres strony(w tym wypadku /)

Przykładowa odpowiedź:

```
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Date: Sat, 08 Jun 2019 23:28:40 GMT\r\n
    Server: libwww-perl-daemon/6.01\r\n
    Content-Type: text/html\r\n
  ▸ Content-Length: 930\r\n
    Last-Modified: Sat, 08 Jun 2019 22:26:46 GMT\r\n
    \r\n
    [HTTP response 7/9]
    [Time since request: 0.001302686 seconds]
    [Prev request in frame: 441]
    [Prev response in frame: 456]
    [Request in frame: 1867]
    [Next request in frame: 1888]
    [Next response in frame: 1899]
    [Request URI: http://localhost:47785/]
    File Data: 930 bytes
▾ Line-based text data: text/html (39 lines)
    <!DOCTYPE html>\n
    <html lang="en">\n
    <meta name="viewport" content="width=device-width, initial-scale=1.0"> \n
    <head>\n
        <meta charset="UTF-8">\n
        <title>Sieciowe - lista 5</title>\n
        <style>\n
            div {\n
                background-color: #333333;\n
                margin: 15px;\n
                padding: 20px;\n
            }\n
            #header1 {\n
```

W odpowiedzi można dostrzec m. in. kod odpowiedzi (200 - OK) oraz źródło strony.