Celem zadania jest napisanie programu pełniącego funkcję uproszczonego serwera protokołu HTTP. Program może być uruchomiony na dwa sposoby (uruchomienie bądź wyłączenie serwera):

1. w celu **wystartowania** serwera jako procesu-demona:

   `./prog -s -p` *`port`* `-d` *`dir`*

   gdzie:
   - *port* to numer portu, na którym będzie nasłuchiwał serwer;
   - *dir* to nazwa katalogu, który będzie pełnił rolę *document root* serwera.

   Np., uruchomienie programu poleceniem:

   ```
   ./prog -s -p 8080 -d ~/dokumenty
   ```

   uruchomi serwer http nasłuchujący na porcie `8080` i serwujący dokumenty z katalogu `~/dokumenty`.

2. w celu zatrzymania pracującego serwera:

   ```
   ./prog -q
   ```

Serwer powinien realizować następujące funkcjonalności:

- poprawnie reagować na błędne argumenty (niepoprawny numer portu, numer portu już obsadzony usługą, nieistniejący katalog bądź nieodczytywalny katalog, daemon już uruchomiony dla opcji `-s` oraz daemon nieuruchomiony dla opcji `-q`, itp);
- obsługiwać polecenie `GET` protokołu http w wersji 1.1;
- reagować odesłaniem błędu `HTTP 501` (*Not Implemented*) w odpowiedzi na odebranie jakiegokolwiek innego polecenia;
- na polecenie postaci `GET /` program powinien reagować odesłaniem dokumentu `index.html` ze swojego katalogu *document root*
- poprawnie zgłaszać podstawowe błędy, w szczególności `HTTP 404`;
- prowadzić plik kroniki (log), w którym zapisywane są w czytelnej postaci wszystkie odebrane żądania poprzedzone datą, czasem, adresem IP klienta i wynikiem wykonania żądania; plik kroniki powinien być zapisywany w tym samym katalogu, w którym znajduje się plik wykonywalny i mieć taką samą nazwę jak plik wykonywalny oraz rozszerzenie .log (a więc w przypadku uruchomienia serwera z pliku `a.out` kronika będzie zapisywana do pliku `a.out.log`)
- program powinien uruchomić osobny wątek dla każdego nadchodzącego połączenia; wątek obsługuje przydzielone mu żądanie, zamyka połączenie i kończy pracę;

- do pliku kroniki należy również zapisywać TID wątku obsługującego żądanie;
- należy zadbać o prawidłową synchronizację zapisów do kroniki pochodzących z różnych wątków;
- należy obsłużyć przynajmniej typy MIME odpowiadające dokumentom HTML(L) i podstawowym formatom graficznym (JP(E)G, PNG, GIF).
- nie wymaga się implementowania mechanizmu *keep-alive*, tzn. serwer odpowiadając na żądanie powinien odesłać klientowi nagłówek *Connection: close*, a po realizacji żądania zamknąć połączenie;
- do testowania zachowania serwera można użyć zestawu plików spakowanych w archiwum *httptest.zip*, dostępnym w plikach zespołu naszego przedmiotu oraz dowolnej przeglądarki internetowej.

Na antyplagiat wysyłamy kody jednego programu, zmieniając przed wysłaniem jego nazwę na: ***numer_indeksu.ps.lab09.c*** *(czyli np. 66666.ps.lab09.c),*

Kod źródłowy (1 plik) po oddaniu prowadzącemu zajęcia laboratoryjne musi zostać jako załącznik przesłany na adres **pss1@zut.edu.pl** (wysyłamy jeden mail z jednym załącznikiem):

- plik z kodem źródłowym musi mieć nazwę zgodną ze wzorcem podanym w treści zadania,
- mail musi zostać wysłany z poczty uczelnianej (domena **zut.edu.pl**),
- temat maila musi mieć postać: **PS IS1 999X LAB09**, gdzie 999X to numer grupy laboratoryjnej (np. PS IS1 321 LAB09),
- w pierwszych trzech liniach kodu źródłowego w komentarzach (każda linia komentowana osobno) musi znaleźć się:
  - informacja identyczna z zamieszczoną w temacie maila,
  - imię i nazwisko osoby wysyłającej maila,
  - adres e-mail, z którego wysłano wiadomość,
    np.:

```
// PS IS1 321 LAB09
// Jan Nowak
// nj66666@zut.edu.pl
```

- e-mail nie może zawierać żadnej treści (tylko załącznik).

Dostarczone kody programów będą analizowane pod kątem wykrywania plagiatów. Niewysłanie wiadomości, wysłanie jej w formie niezgodnej z powyższymi wymaganiami lub wysłanie pliku, który nie będzie się kompilował i uruchamiał, będzie traktowane jako brak programu i skutkowało otrzymaniem za niego oceny niedostatecznej.