Zadanie programistyczne nr 4
z Sieci komputerowych

1 Opis zadania

Napisz program `webserver` będący prostym serwerem WWW, wyświetlającym strony z zadanego katalogu. Zadanie można wykonywać na maszynie wirtualnej Virbian lub na linuksowym komputerze w sali 109.

Rozpocznij od następujących czynności.

1. Ze strony wykładu pobierz plik `webpages.tgz` zawierający strony WWW i rozpakuj go do wybranego katalogu. Dostępny na stronie wykładu obraz maszyny wirtualnej *Virbian* zawiera już te strony WWW w katalogu `/var/local/webpages/`.

2. Umieść w pliku `/etc/hosts` następującą zawartość.

```
127.0.0.1 localhost
127.0.1.1 lab108-18
```

Takie wpisy istnieją w pliku `/etc/hosts` na komputerach w pracowni 109. Odwołania do domen `localhost` i `lab108-18` kierowane będą do lokalnego komputera.

Serwer powinien akceptować dwa parametry podawane w wierszu poleceń: *port* i *katalog*. Pierwszy z nich jest numerem portu, na którym serwer będzie oczekiwać na przychodzące połączenia, zaś drugi katalogiem zawierającym strony WWW (np. te pobrane). Program powinien obsługiwać błędne dane wejściowe, typu nieistniejący katalog, zgłaszając odpowiedni komunikat.

Otwarcie w przeglądarce WWW (działającej na tym samym komputerze lub na tej samej maszynie wirtualnej) adresu `http://`*nazwa_domeny*`:`*port*`/`*strona.html* powinno spowodować wyświetlenie zawartości strony *katalog*`/`*nazwa_domeny*`/`*strona.html*.

1.1 Implementacja

Oczywiście nie trzeba implementować pełnego protokołu HTTP. Twój serwer powinien natomiast obsługiwać żądania `GET`. Wystarczy obsługiwać pierwszy wiersz takiego zapytania (czyli pole zawierające adres strony), pole `Host` i pole `Connection` (patrz niżej). W szczególności nie trzeba obsługiwać warunkowych żądań `GET`, np. pól `If-Modified-Since`.

Twój serwer powinien odsyłać następujące informacje: kod odpowiedzi, pole `Content-Type`, pole `Content-Length` i inne potrzebne dane (przykładowo pole `Location` dla odpowiedzi 301).

1. Zwracane powinny być następujące kody odpowiedzi (w razie potrzeby można zaimplementować też inne zdefiniowane w standardzie HTTP).
   - `200 OK`: w przypadku powodzenia;

- `301 Moved Permanently`: jeśli przeglądarka chce pobrać obiekt, który jest katalogiem należy przekierować ją do strony `index.html`, która znajduje się w tym katalogu;
- `403 Forbidden`: jeśli przeglądarka będzie chciała pobrać adres prowadzący do strony spoza danej domeny;
- `404 Not Found`: jeśli przeglądarka będzie chciała pobrać nieistniejącą stronę;
- `501 Not Implemented`: jeśli przeglądarka wyśle dane niezrozumiałe dla serwera.

Również w przypadku komunikatów różnych od 200 Twój serwer powinien wysyłać zawartość HTML, stanowiącą prosty opis błędu, który wystąpił. Zawartość ta zostanie wyświetlona przez przeglądarkę.

2. Pole `Content-Type` powinno być ustalane na podstawie rozszerzenia pliku. Poprawnie powinny być obsługiwane pliki `txt`, `html`, `css`, `jpg`, `jpeg`, `png` i `pdf`. Pozostałe pliki mogą mieć typ `application/octet-stream`). W przypadku pliku tekstowego można założyć, że jest on zapisany w UTF-8 i nie trzeba tego sprawdzać. Przykładowo w przypadku strony HTML zawartość wiersza z `Content-Type` może być równa `Content-Type: text/html; charset=utf-8`.

Twój serwer nie powinien rozłączać się po obsłużeniu pojedynczego zapytania, lecz utrzymywać połączenie z nieaktywnym klientem przez pewien zdefiniowany czas (np. 250-1000 milisekund).[1] Wyjątkiem jest obsługa zapytania, w którego nagłówku znajduje się pole `Connection: close`. W takim przypadku należy zamknąć połączenie zaraz po obsłużeniu zapytania. W tym samym połączeniu Twój serwer powinien potrafić obsłużyć wiele zapytań (np. o stronę WWW i znajdujące się na niej obrazki). Twój program nie musi obsługiwać połączeń od wielu klientów jednocześnie.

Pamiętaj, że serwerowi WWW nie wolno wysłać zawartości pliku leżącego poza katalogiem ze stronami WWW danej domeny. Twój serwer powinien być odporny na złośliwego użytkownika zdalnego, który wyśle śmieci zamiast poprawnego żądania HTTP. Twój serwer może wyświetlać komunikaty diagnostyczne (np. informacje o żądaniach HTTP i odpowiedziach) na standardowe wyjście, ale powinny być one zwięzłe i przejrzyste.

2 Uwagi techniczne

Pliki Sposób utworzenia napisu oznaczanego dalej jako *imie_nazwisko*: Swoje (pierwsze) imię oraz nazwisko proszę zapisać wyłącznie małymi literami zastępując litery ze znakami diakrytycznymi przez ich łacińskie odpowiedniki. Pomiędzy imię i nazwisko należy wstawić znak podkreślenia.

Prowadzącemu ćwiczeniopracownię należy dostarczyć plik *imie_nazwisko*`.tar.xz` z archiwum (w formacie `tar`, spakowane programem `xz`) zawierającym pojedynczy katalog o nazwie *imie_nazwisko* z następującymi plikami.

- Kod źródłowy w C lub C++, czyli pliki `*.c` i `*.h` lub pliki `*.cpp` i `*.h`. Każdy plik `*.c` i `*.cpp` na początku powinien zawierać w komentarzu imię, nazwisko i numer indeksu autora.

[1] Uwaga: niektóre przeglądarki otwierają więcej niż jedno połaczenie do serwera. Jeśli Twój program będzie obsługiwać tylko jedno z tych połączeń naraz, zbyt duże oczekiwanie na zamknięcie połączenia spowolni cały proces. W poważniejszych zastosowaniach należałoby obsługiwać więcej niż jedno połączenie naraz; tutaj nie jest to wymagane.

- Plik `Makefile` pozwalający na kompilację programu po uruchomieniu `make`.
- Ewentualnie plik `README.txt` lub `README.md`

W katalogu tym **nie** powinno być żadnych innych plików, w szczególności skompilowanego programu, obiektów `*.o`, czy plików źródłowych nie należących do projektu.

Kompilacja Kompilacja i uruchamianie przeprowadzone zostaną w 64-bitowym środowisku Linux. Kompilacja w przypadku C ma wykorzystywać standard C99 lub C17 z ewentualnymi rozszerzeniami GNU (opcja kompilatora `-std=c99`, `-std=gnu99`, `-std=c17` lub `-std=gnu17`), zaś w przypadku C++ — standard C++11, C++14 lub C++17 z ewentualnymi rozszerzeniami GNU (opcje kompilatora `-std=c++11`, `-std=gnu++11`, `-std=c++14`, `-std=gnu++14`, `-std=c++17` lub `-std=gnu++17`). Kompilacja powinna korzystać z opcji `-Wall` i `-Wextra`. Podczas kompilacji nie powinny pojawiać się ostrzeżenia.

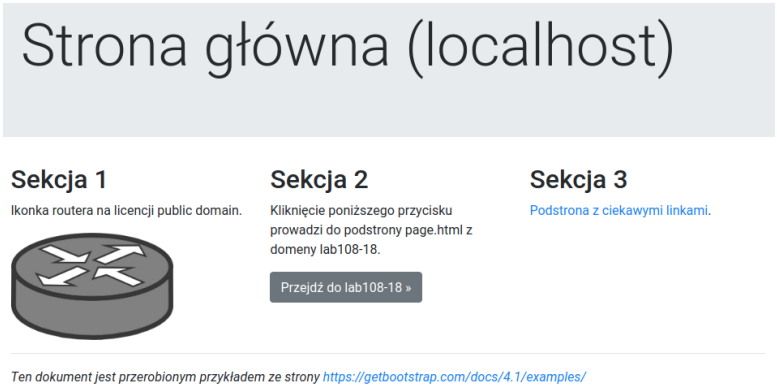
3 Sposób oceniania programów

Poniższe uwagi służą ujednoliceniu oceniania w poszczególnych grupach. Napisane są jako polecenia dla prowadzących, ale studenci powinni **koniecznie się** z nimi zapoznać, gdyż będziemy się ściśle trzymać poniższych wytycznych. Programy będą testowane na zajęciach w obecności autora programu. Na początku program uruchamiany jest w różnych warunkach i otrzymuje za te uruchomienia od 0 do 10 punktów. Następnie obliczane są ewentualne punkty karne. Oceniamy z dokładnością do 0,5 punktu. Jeśli ostateczna liczba punktów wyjdzie ujemna wstawiamy zero. (Ostatnia uwaga nie dotyczy przypadków plagiatów lub niesamodzielnych programów).

Testowanie: punkty dodatnie Rozpocząć od kompilacji programu. W przypadku programu niekompilującego się, stawiamy 0 punktów, nawet jeśli program będzie ładnie wyglądał. Następnie należy przygotować środowisko, tj. sprawdzić, czy w pliku `/etc/hosts` są odpowiednie wpisy, a następnie pobrać i rozpakować plik `webpages.tgz`.

Uruchomić serwer nasłuchujący na porcie 8888. Podczas testowania należy każdorazowo sprawdzać (np. rozszerzeniem *HTTP Header Live* przeglądarki), czy wysyłane przez serwer komunikaty są zgodne ze specyfikacją, np. czy pole `Content-Type` jest poprawnie ustawiane.

3 pkt. Wejść na stronę `http://localhost:8888/`. Sprawdzić, czy serwer przekieruje nas za pomocą komunikatu `301` do strony `http://localhost/index.html` i czy wyświetlana później strona wygląda następująco:



Jeśli pobranie zajmuje dłużej niż kilka sekund, należy przydzielić maksymalnie 1 punkt. Jeśli strona nie przypomina powyższego obrazka, nie sprawdzamy dalej.

1 pkt. Kliknąć odnośnik *Podstrona z ciekawymi linkami* i sprawdzić, czy strona `page.html` wyświetla się prawidłowo.

2 pkt. Sprawdzić, czy kliknięcie odnośników *Plik tekstowy ...*, *Kupony deklaracyjne* i *Plik binarny* działa poprawnie. W przypadku dwóch pierwszych przeglądarka powinna otworzyć plik tekstowy i PDF-a (lub poprosić o uruchomienie przeglądarki PDF-ów). Plik binarny powinien mieć typ `application/octet-stream`; przeglądarka powinna zaproponować nam jego zapisanie. Na końcu należy powrócić na stronę główną.

1 pkt. Kliknąć przycisk *Przejdź do lab108-18*. Powinna wyświetlić się zawartość strony `page.html` z domeny `lab108-18`.

1 pkt. Wejść na stronę `http://lab108-18:8888/`. Serwer powinien przekierować nas komunikatem `301` do strony `http://lab108-18/index.html`, a następnie zwrócić komunikat `404` ze względu na brak pliku `index.html`.

1 pkt. Sprawdzić, czy zapytanie o stronę `http://lab108-18:8888/../localhost/index.html` jest poprawnie obsługiwane, tj. nie powoduje wyświetlania żądanej strony. Uwaga: to zapytanie należy wykonać w trybie wsadowym, np. za pomocą polecenia `curl -v --path-as-is http://lab108-18:8888/../localhost/index.html`, bo większość przeglądarek usunie ciąg `../` z pola adresu.

1 pkt. Połączyć się z serwerem za pomocą programu `telnet` i wysłać mu śmieci zakończone pustym wierszem. Serwer powinien odpowiedzieć komunikatem `501`. Sprawdzić, czy serwer potem jest nadal w stanie odpowiadać na żądania.

Punkty karne Punkty karne przewidziane są za następujące usterki.

-2 pkt. Zamykanie połączenia po każdym żądaniu, bez czekania na upłynięcie zadanego czasu (np. 1 sekundy).

-1 pkt. Brak poprawności sprawdzania argumentów wywołania programu.

do -3 pkt. Zła / nieczytelna struktura programu: brak modularności i podziału na funkcjonalne części, niekonsekwentne wcięcia, powtórzenia kodu.

-2 pkt. Aktywne czekanie zamiast zasypiania do momentu otrzymania pakietu.

-1 pkt. Brak sprawdzania poprawności wywołania funkcji systemowych, takich jak `recvfrom()`, `write()` czy `bind()`.

-1 pkt. Nietrzymanie się specyfikacji wejścia i wyjścia. Przykładowo: wyświetlanie nadmiarowych informacji diagnostycznych, inna niż w specyfikacji obsługa parametrów.

-1 pkt. Zły plik `Makefile` lub jego brak: program powinien się kompilować poleceniem `make`: poszczególne pliki `*.c` i `*.cpp` powinny kompilować się do obiektów tymczasowych `*.o` a następnie powinny być konsolidowane do wykonywalnego programu. Polecenie `make clean` powinno czyścić katalog z tymczasowych obiektów (plików `*.o`), zaś polecenie `make distclean` powinno usuwać te obiekty i wykonywalny program pozostawiając tylko pliki źródłowe.

-3 pkt. Kara za wysłanie programu po terminie; opóźnienie nie może być większe niż 1 tydzień.

Marcin Bieńkowski