Klient TCP daytime

TCP vs UDP

TCP

- Strumieniowy
- Gwarancja dostarczenia
- Porządek FIFO
- Kontrola przepływu

UDP

- Pakietowy
- Brak gwarancji dostarczenia
- Brak porządku FIFO

Funkcje wywoływane przez klienta TCP

socket() - connect() - read()/write() - close()

socket()

Creates an endpoint for communication and returns a **file descriptor** that refers to that endpoint.

int socket(int domain, int type, int protocol);
domain e.g. AF_INET, AF_INET6, AF_UNIX
type e.g. SOCK_STREAM, SOCK_DGRAM
protocol e.g. 0 //wybierany automatycznie

Zwraca: deskryptor utworzonego gniazda lub -1 (błąd, ustawia errno).

connect()

The connect() system call connects the socket [...] to the address [...].

int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

sockfd - deskryptor pliku gniazda, z którym będzie związane połączenie (lokalnie)
addr - adres, z którym będzie nawiązana łączność
addrlen - wielkość struktury przekazanej jako addr

Zwraca: 0 (ok) lub -1 (błąd, errno ustawione na kod błędu).

read()

attempts to read **up to** count bytes from file descriptor fd into the buffer starting at buf.

ssize_t read(int fd, void *buf, size_t count);
fd - deskryptor gniazda, które będzie oczekiwało na dane
buf - bufor, do którego zostaną przekazane odebrane dane
count - maksymalna ilość danych, która będzie przekazana do bufora

Zwraca: liczbę odebranych bajtów lub -1 (błąd, ustawia errno) lub 0 (gniazdo zamknięte).

write()

writes up to count bytes from the buffer pointed buf to the file referred to by the file descriptor fd

ssize_t write(int fd, const void *buf, size_t count);
fd - deskryptor gniazda, które będzie wysyłało na dane
buf - bufor, z którego zostaną odczytane dane do wysłania
count - maksymalna ilość danych, która będzie odczytana z bufora

Zwraca: liczbę wysłanych bajtów lub -1 (błąd, ustawia errno).

close()

closes a file descriptor, so that it no longer refers to any file and may be reused.

int close(int fd);
fd - deskryptor gniazda, które zostanie zamknięte

Zwraca: 0 (ok) lub -1 (błąd, ustawia errno).

sockaddr_in

Do funkcji connect należy podać jako argument addr instancję struktury sockaddr_in (in = internet) rzutując na typ (struct sockaddr *). Więcej w “man 7 ip”.

sockaddr_in {
sin_family //AF_INET, AF_INET6
sin_port //nr portu aplikacji w porządku sieciowym
sin_addr //adres ip w porządku sieciowym
sin_zero //kompatybilność z ogólną strukturą sockaddr, powinno zawierać zera
}

memset(&sa, 0, sizeof(sa)); //zapewnia, że sin_zero zawiera zera

Zamiana na reprezentację sieciową

htons (host to network short) / ntohs (network to host short)
htonl (host to network long) / ntohl (network to host long)

e.g. sa.sin_port = htons(1337);

inet_pton (presentation to network)
inet_ntop (network to presentation)

e.g. inet_pton(AF_INET, “127.0.0.1”, &(sa.sin_addr));

Zadania cz. 1

Zadanie 1

Prowadzący uruchomi serwer daytime oraz udostępni jego adres IP. Usługa daytime standardowo działa na porcie 13.

Napisz program, który połączy się z serwerem, odczyta ciąg znaków zakończony nową linią i wyświetli go na ekranie. Następnie zamknie połączenie (gniazdo) oraz zakończy działanie.

Zadania cz. 2

Zadanie 2
Przekształć poprzedni program tak, by czytał z adresu IP i portu podanego w argumentach programu.

Zadanie 3
Dodaj do programu obsługę błędów zwracanych przez funkcje connect i read.

Zadanie 4
Zmień IP na losowe (tak, by nie odpowiadało na próbę połączenia). Programem netstat -tnp / ss -tnp wyświetl utworzone połączenie.