**Wojskowa Akademia Techniczna**

**Wydział Elektroniki**

**Instytut Telekomunikacji**

**Studia II°**

**Projektowanie aplikacji sieciowych**

Materiały pomocnicze do zajęć

Cz.7.

**dr inż. Jarosław Krygier**

Warszawa 2017

Spis treści

[1 Serwer TCP: gniazdo strumieniowe, wykorzystanie funkcji select() 3](#_Toc496705696)

# Serwer TCP: gniazdo strumieniowe, wykorzystanie funkcji select()

/\*

============================================================================

Name : PAS\_8\_select.c

Author : J. Krygier

Version :

Copyright : Na podstawie: http://www.binarytides.com/multiple-socket-connections-fdset-select-linux/

Description : Wykorzystanie select do polaczen z gniazdami. Serwer TCP

============================================================================

\*/

// UWAGA: W celu spradzenia dzialania, lacz sie z serwerem za pomoca Telnet

#include <stdio.h>

#include <string.h> //strlen

#include <stdlib.h>

#include <errno.h>

#include <unistd.h> //close

#include <arpa/inet.h> //inet

#include <sys/types.h> //typy zmiennych

#include <sys/socket.h> //gniazda

#include <netinet/in.h> //adresy gniazd

#include <sys/time.h> //Makra FD\_SET, FD\_ISSET, FD\_ZERO

#define TRUE 1

#define FALSE 0

#define PORT 8888

int main(void) {

int opt = TRUE;

int master\_socket , new\_socket; //deskryptor gniazda

int addrlen;

int client\_socket[30]; //tablica na gniazda klientow

int max\_clients = 30 , activity, i , valread , sd;

int max\_sd;

struct sockaddr\_in address; //struktura na adres gniazda

char buffer[1025]; //bufor

fd\_set readfds; //ustawienia deskryptora pliku dla fukcji select (select.h)

//wiadomosc

char \*message = "SERWER TCP: Serwer odpowiada echem. Napisz cos.\r\n";

//inicjalizacja: w client\_socket[], same 0

for (i = 0; i < max\_clients; i++) {

client\_socket[i] = 0;

}

//utworzenie gniazda strumieniowego (TCP) - do nasluchu

if( (master\_socket = socket(AF\_INET , SOCK\_STREAM , 0)) == 0) {

perror("blad w socket ()");

exit(EXIT\_FAILURE);

} else {

printf ("SERWER: Gniazdo nasluchujace: %d \n", master\_socket);

}

// odblokujemy gnizdo zaraz po zamknieciu do ponownego uzycia

//zezwolenie na wiele polaczen z gniazdem

if( setsockopt(master\_socket, SOL\_SOCKET, SO\_REUSEADDR, (char \*)&opt, sizeof(opt)) < 0 ) {

perror("blad w setsockopt()");

exit(EXIT\_FAILURE);

}

//Ustawienie adresu gniazda

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons( PORT );

//skojarzenie utwozronego gniazda z adresem gniazda

if (bind(master\_socket, (struct sockaddr \*)&address, sizeof(address))<0) {

perror("blad bind ()");

exit(EXIT\_FAILURE);

}

printf("SERWER: Nasluchuje na porcie %d \n", PORT);

// Maksimum 3 aktywne polaczenia na tym gniezdzie

if (listen(master\_socket, 3) < 0) {

perror("blad listen()");

exit(EXIT\_FAILURE);

}

// Akceptacja przychodzacego polaczenia

addrlen = sizeof(address);

puts("SERWER: Czekam na polaczenie ...");

while(TRUE) {

// wyczysczenie ustawien deskryptora pliku

FD\_ZERO(&readfds); // FD\_ZERO - makro w select.h

// dodanie gniazda nasuchujacego w celu wysterowania jego ustawien przez f. select

FD\_SET(master\_socket, &readfds);

max\_sd = master\_socket;

// Dodanie gniazd klientow (jak sie pojawia to beda zapisane w client\_socket[])

// w celu wysterowania ich ustawien przez f. select

for ( i = 0 ; i < max\_clients ; i++) {

//odczytanie deskrypora gniaza z client\_socket[i]

sd = client\_socket[i];

// Jesli kolejny i-ty klient sie polaczy to w client\_socket[i] deskryptor gniazda bedzie > 0

// i nastapi dodanie deskrytora do ustawien select

if(sd > 0)

FD\_SET( sd , &readfds);

//znalezenie najwiekszego przydzielonego dla klienta deskryptora gniaza

if(sd > max\_sd)

max\_sd = sd;

}

// czakaj/sprawdzaj aktwynosc na gniazdach klientow

// timeout is NULL , so wait indefinitely

activity = select( max\_sd + 1 , &readfds , NULL , NULL , NULL);

if ((activity < 0) && (errno!=EINTR)) {

printf("blad select ()");

}

//Jesli cos wykryto w gniezdzie nasuczujacym, oznacza to nowe polaczenie

if (FD\_ISSET(master\_socket, &readfds)) {

if ((new\_socket = accept(master\_socket, (struct sockaddr \*)&address, (socklen\_t\*)&addrlen))<0) {

perror("blad accept ()");

exit(EXIT\_FAILURE);

}

//Wypiszemy info o kliencie

printf("SERWER: Nowe polaczenie. \n");

printf("SERWER: Deskryptor gniazda do komunikacji z klientem: %d , Adres IP klienta: %s , Nr portu klienta: %d \n" ,

new\_socket , inet\_ntoa(address.sin\_addr) , ntohs(address.sin\_port));

//wysli powitanie do klinta

if( send(new\_socket, message, strlen(message), 0) != strlen(message) ) {

perror("blad w send ()");

}

puts("SERWER: Wyslano powitanie do klienta");

//Dodamy teraz gnizado do komunikacji z klientem do tablicy gniazd: client\_socket[]

for (i = 0; i < max\_clients; i++) {

//na wolnej pozycji

if( client\_socket[i] == 0 ) {

client\_socket[i] = new\_socket;

printf("SERWER: Dodano gniazdo do komunikacji z klinetem na pozycje %d\n" , i);

break;

}

}

}

// sparwdz czy cos sie zmienilo na innych aktywnych gniazdach

for (i = 0; i < max\_clients; i++) {

sd = client\_socket[i];

if (FD\_ISSET( sd , &readfds)) {

//Jesli czekaja tam dane odczytaj je, ajsli klient zamknal polaczenie zareaguj

if ((valread = read( sd , buffer, 1024)) == 0) {

//read zwraca 0 - to znaczy polczenie zostalo zamkniete przez klienta

// odcztaj dane klineta: getpeername ()

getpeername(sd , (struct sockaddr\*)&address , (socklen\_t\*)&addrlen);

printf("SERWER: Klient o adresie %s (port: %d) sie rozlaczyl \n" , inet\_ntoa(address.sin\_addr) , ntohs(address.sin\_port));

//zamknij gniazdo do komunikacji z klientem i wyzeruj wpis w client\_socket[]

close( sd );

client\_socket[i] = 0;

} else {

// Wyslij echo

// na koncu dodaj '/n'

buffer[valread] = '\0';

send(sd , buffer , strlen(buffer) , 0 );

}

}

}

}

return 0;

}