**Wojskowa Akademia Techniczna**

**Wydział Elektroniki**

**Instytut Telekomunikacji**

**Studia II°**

**Projektowanie aplikacji sieciowych**

Materiały pomocnicze do zajęć

Cz.6.

**dr inż. Jarosław Krygier**

Warszawa 2017

Spis treści

[1 Serwer wielowątkowy 3](#_Toc496616124)

# Serwer wielowątkowy

/\*

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

Name : PAS\_7\_serwer\_wielowatkowy.c

Author : J. Krygier

Version :

Copyright : na podstawie: http://www.binarytides.com/socket-programming-c-linux-tutorial/

Description : Serwer wielowatkowy

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

\*/

#include <stdio.h>

#include <string.h> //strlen

#include <stdlib.h> //strlen

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

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

#include <unistd.h> //write

#include<pthread.h> //watki, pamietac o lpthread

//pamietac o dodaniu pthread jako opcji linkera: gcc program.c -l pthread

// wyswietlenie w konsoli watkow: ps -eL | grep PAS\

// wiele informacji mozna znalezc tutaj: http://www.binarytides.com/socket-programming-c-linux-tutorial/

// deklaracja fukcji

void \*watek\_obslugi\_polaczenia(void \*); // funckajs - watek obslugi polaczenia

int main(void) {

int deskryptor\_gniazda , nowe\_gniazdo , c;

int \*nowe\_gniazdo\_ptr;

struct sockaddr\_in strukt\_gniazdo\_serwera; // struktura adresacji gniazda serwera

struct sockaddr\_in strukt\_gniazdo\_klienta; // struktura adresacji gniazda klienta

char \*wiadomosc;

int optval;

//Create socket

deskryptor\_gniazda = socket(AF\_INET , SOCK\_STREAM , 0);

if (deskryptor\_gniazda == -1) {

printf("SERWER: blad funkcji soclet() \n");

}

//zapewnienie nieblokowalnosci gniazda po zwolnieniu

optval = 1;

setsockopt(deskryptor\_gniazda, SOL\_SOCKET, SO\_REUSEADDR, (const void \*)&optval , sizeof(int)); // w socket.h

//Przygotowanie adresacji gniazda

strukt\_gniazdo\_serwera.sin\_family = AF\_INET; //IPv4

strukt\_gniazdo\_serwera.sin\_addr.s\_addr = INADDR\_ANY; //dowolny klinet moze sie polaczyc

strukt\_gniazdo\_serwera.sin\_port = htons( 8888 ); //na jakim porcie nasluchuje serwer

//Powiaz deskryptor gniazda z adresem gniazda

if( bind(deskryptor\_gniazda,(struct sockaddr \*)&strukt\_gniazdo\_serwera , sizeof(strukt\_gniazdo\_serwera)) < 0) {

printf ("SERWER: blad funkcji bind() \n");

return 1;

} else {

printf ("SERWER: Polaczono sie z gniazdem. \n");

}

//Nasluchiwanie na porcie

listen(deskryptor\_gniazda , 3);

//Accept and incoming connection

printf ("SERWER: Czekam na klianta ... \n");

c = sizeof(struct sockaddr\_in); //rozmiar struktury dla adresacji gniazda klienta

while( (nowe\_gniazdo = accept(deskryptor\_gniazda, (struct sockaddr \*) &strukt\_gniazdo\_klienta, (socklen\_t\*) &c)) ) {

printf ("SERWER: Polaczenie zaakceptowano. \n");

//Odpowiedz do klienta

wiadomosc = "Polaczyles sie z serwerem TCP. Przydzielono watek obslugi polaczenia ...\n";

// wyslij do klienta wiadomosc

write(nowe\_gniazdo , wiadomosc , strlen(wiadomosc));

pthread\_t id\_watku; // deklaracja id\_watku

// jako argument do fukncji watku przekazemy wskaznik na strukture\_adresu\_gniazda

// do komunikacji z klientem utworzonego przez 'accept()'

nowe\_gniazdo\_ptr = malloc(sizeof(int));

\*nowe\_gniazdo\_ptr = nowe\_gniazdo;

if (pthread\_create( &id\_watku , NULL , watek\_obslugi\_polaczenia , (void\*) nowe\_gniazdo\_ptr) < 0) {

printf ("SERWER: Nie moge utworzyc watku. \n");

return 1;

}

}

//Czekamy na zakoczenie watku

//pthread\_join( id\_watku, NULL);

if (nowe\_gniazdo < 0)

{

printf ("SERWER: Nie zaakceptowano polaczenia. \n");

return 1;

}

return 0;

}

/\*

\* Obsluga polaczenia przez watek

\* \*/

void \*watek\_obslugi\_polaczenia(void \*gniazdo\_do\_komunikacji\_z\_klientem\_ptr)

{

//Get the socket descriptor

int gniazdo;

int i, rozmiar\_wiad;

char \*wiadomosc;

char wiadomosc1 [100];

gniazdo = \*(int\*)gniazdo\_do\_komunikacji\_z\_klientem\_ptr;

bzero(wiadomosc1, 100);

//Wyslemy cos do klienta

sprintf (wiadomosc1, "SERWER: Obsluguje cie watek z id gniazda: %d \n", gniazdo);

//wiadomosc = "SERWER: Obsluguje cie watek\n";

write(gniazdo , wiadomosc1 , 100);

//write(gniazdo , wiadomosc , strlen(wiadomosc));

for (i=0;i<5; i++) {

wiadomosc = ".";

write(gniazdo , wiadomosc , strlen(wiadomosc));

sleep(1);

wiadomosc = "\_";

write(gniazdo , wiadomosc , strlen(wiadomosc));

sleep(1);

}

#if 1

//jak chcesz dluzej podadac z klientem to tutaj

//pobierz wiadomosc od klienta

bzero(wiadomosc1, 100);

while( 1 ) {

wiadomosc = "\nMozesz cos napisac...\n";

write(gniazdo , wiadomosc , strlen(wiadomosc));

rozmiar\_wiad = recv(gniazdo , wiadomosc1 , 2000 , 0);

//wyslij echo

wiadomosc = "\nOdebralem: ";

write(gniazdo , wiadomosc , strlen(wiadomosc));

write(gniazdo , wiadomosc1 , strlen(wiadomosc1));

bzero(wiadomosc1, 100);

}

#endif

//rozlacz

close (gniazdo);

//zwolnij wskaznik

free(gniazdo\_do\_komunikacji\_z\_klientem\_ptr);

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

return 0;

}