­

**Programowanie aplikacji klient-serwer**

Sprawdzarka cen (Temat 7)

**Projekt**

(wersja 1.1)

**AUTORZY PROJEKTU:**

Roman Uliashyn

Michał Pawłowski

Spis treści

1 Wstęp 3

Treść zadania 3

Dodatkowe założenia 3

Serwer 3

Klient 3

2 Użyte protokoły 3

Protokół komunikacji klienta z serwerem 3

Wstęp 3

Technologia 3

Funkcjonalność: 3

Dokładny opis funkcjonalny poszczególnych metod/komend 4

Przypadki użycia 5

Klient wpisuje kod towaru 5

Server odpowiada klientowi 5

Klient wpisuje admin 5

Serwer czeka na hasło 5

Admin ma do wyboru funkcje użytkowania 5

Admin wyloguje się z servera 5

Aplikacja klienta 5

Wstęp 5

Moduł Algorytm 5

Aplikacja serwera 5

Wstęp 5

Moduł Algorytm 6

Przechowywane dane 6

Wstęp 6

Interfejs danych 6

Spis komunikatów błędów 6

Planowany podział prac 6

# Wstęp

## Treść zadania

Serwer przechowuje bazę danych o towarach (kod, nazwa, cena).

Klient wprowadza kod towaru (zgodny z kodem paskowym) i w odpowiedzi dostaje parę nazwa i cena.

## Dodatkowe założenia

Administrator:

Loguje się na serwer,

Wprowadza i modyfikuje dane towarów

### Serwer

Serwer wykonany zostanie w języku „C”. Towary będą przechowywane w pliku. Serwer po otrzymaniu zapytania od klienta będzie pobierał wartość z bazy(plik .txt) i wysyłał wszystkie dane o towarze(nazwa, cena, id ...).

### Klient

Klient będzie aplikacją napisaną w języku „C”. Będzie wysyłał zapytanie do serwera z kodem produktu i będzie otrzymywał odpowiedź z dokładnymi danymi towaru(nazwa, cena, id...).

# Użyte protokoły

## Protokół komunikacji klienta z serwerem TCP

### Wstęp

TCP / IP określa sposób wymiany danych za pośrednictwem internetu poprzez zapewnienie bezpośredniego związku, który określa, jak powinna być podzielona na pakiety, adresowana przekazana, transportowana i uzyskana w miejscu przeznaczenia

#### Technologia

Podstawowym założeniem modelu TCP/IP jest podział całego zagadnienia komunikacji sieciowej na szereg współpracujących ze sobą warstw.

Każda z nich może być tworzona przez programistów zupełnie niezależnie, jeżeli narzucimy pewne protokoły według których wymieniają się one informacjami. Założenia modelu TCP/IP są pod względem organizacji warstw zbliżone do modelu OSI. Jednak liczba warstw jest mniejsza i bardziej odzwierciedla prawdziwą strukturę Internetu. Model TCP/IP składa się z czterech warstw.

Warstwa:

1. Warstwa aplikacji
2. Warstwa transportowa
3. Warstwa Internetu
4. Warstwa dostępu do sieci

#### Funkcjonalność:

Protokół ten realizuje następującą funkcjonalność:

Aby protokół TCP wystartował musimy z serwerem jawnie nawiązać kontakt, wysyłamy żądanie i czekamy na odpowiedź, że mamy połączenie. TCP nie jest najszybsze, jednak przy pojedynczych żądaniach nie ma potrzeby, aby działało jak najszybciej.

### Dokładny opis funkcjonalny poszczególnych metod/komend

Klient:

1. Metoda socket() służy do utworzenia socketu, zwraca socket
2. Metoda connect() tworzy polaczenie z serwerem, zwraca informacje czy się udało
3. Metoda writeAndSend(fe, buffer, size) będzie pobierała od użytkownika nr towaru, sprawdzała poprawność według kryteriów oraz wysyłała do serwera
4. Metoda ReadFromServer() wyświetli wszystkie informacje
5. Metoda loginAsAdmin() loguje na admina i pozwala na nowe opcje
6. Metoda Add() dodaje
7. Cd nastąpi

Serwer:

1. Metoda socket() służy do utworzenia socketu, zwraca socket
2. Metoda listen() czeka na zadanie od klienta
3. Metoda func() będzie pobierała informacje o towarze i wysyłała do klienta

Jak zaczniemy pisać od razu wyślemy poprawiona i ulepszona wersje oraz napiszemy o co urozmaicimy program.

# Przypadki użycia

### Klient wpisuje kod towaru

bzero(buff, MAX);

printf("Enter the product number : ");

n = 0;

while ((buff[n++] = getchar()) != '\n') //wprowadzanie tekstu

;

if ((strncmp(buff, "exit", 4)) == 0) { //jesli podane "exit" rozlacz sie z serwerem

printf("Client Exit...\n");

break;

}

if(strncmp(buff, "admin", 5)==0) //jesli wpisane admin

{

(...)

}

else

{

write(sockfd, buff, MAX);

(..)

}

### Serwer opdowiada klientowi

read(sockfd, buff, MAX);

id = atoi(buff);

//bzero(buff, MAX);

int k;

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

if (strncmp("admin", buff, 5) == 0) { //sprawdzamy czy admin

printf("Ooo admin...\n");

bzero(buff, MAX);

buff[0] = 'H';

buff[1] = 'a';

buff[2] = 's';

buff[3] = 'l';

buff[4] = 'o';

buff[5] = ':';

write(sockfd, buff, MAX); //wysylamy komunikat

bzero(buff, MAX);

read(sockfd, buff, MAX); //odczytujemy haslo

if (strncmp("admin", buff, 5) == 0) //haslo to admin sprawdzamy czy podane prawdidlowe

{

bzero(buff, MAX);

buff[0] = 'Z';

buff[1] = 'a';

buff[2] = 'l';

buff[3] = 'o';

buff[4] = 'g';

buff[5] = 'o';

buff[6] = 'w';

buff[7] = 'a';

buff[8] = 'n';

buff[9] = 'o';

write(sockfd, buff, MAX); //wysylamy komunikat zalogowano

bzero(buff, MAX);

}

else //jesli nie to odsylamy wiadomosc bledne haslo

{

bzero(buff, MAX);

buff[0] = 'B';

buff[1] = 'l';

buff[2] = 'e';

buff[3] = 'd';

buff[4] = 'n';

buff[5] = 'e';

buff[6] = ' ';

buff[7] = 'H';

buff[8] = 'a';

buff[9] = 's';

buff[10] = 'l';

buff[11] = 'o';

write(sockfd, buff, MAX); //wysylamy komunikat bledne haslo

bzero(buff, MAX);

}

}

else

{

for(k=0; k<100; ++k)

{

if(prod[k].id==id) //jezeli jest taki produkt to wykonuje wysylanie do klienta danych

{

// bzero(buff, MAX);

buff[0] = prod[k].id;

write(sockfd, buff, MAX);

bzero(buff, MAX);

unsigned char\* bytedFloat = (unsigned char\*) &prod[k].cena;

buff[0] = bytedFloat[3];

buff[1] = bytedFloat[2];

buff[2] = bytedFloat[1];

buff[3] = bytedFloat[0];

write(sockfd, buff, MAX);

bzero(buff, MAX);

int n = 0;

strcat (prod[k].nazwa, " ");

while (prod[k].nazwa[n] != '\0')

{

buff[n] = prod[k].nazwa[n];

n++;

}

write(sockfd, buff, sizeof(buff));

### 

### Admin ma do wyboru funkcje użytkowania

Adminowi nadane uprawnienia punkt z listy

1 – Dodaj towar

2 – Wyloguj

printf("Wybierz opcje:\n1-Dodaj towar\n2-wyloguj \n ");

choose = getchar();

printf("\n");

switch (choose) {

case '1':

add(sockfd);

break;

case '2':

printf("\nWylogowano\n");

break;

# Aplikacja klienta

## Wstęp

Klient.c obsługuje zapytania klienta , a dokładnie wprowadzenie kodu towaru + przesyłanie kodu do serwera. Celem jest uzyskać informacje odnośnie towaru.  
Klient może tak samo wejść w trybie administratora a dodać towar. Proces logowania pojawia się po wpisaniu komendy „admin”. Wpisywane hasło jest schowane. Poprawne hasło jest zaszyfrowane w pliku. Jest zrealizowana integralność.

## Algorytm

1)Klient tworzy socket -> potem robi connect do serwera -> potem wywoła funkcje „*func*”

Ta funkcja przyjmuję socket dla dalszego wykorzystania.

2)func prosi wpisać string do buffera. W tej chwili można wpisać trzy opcje oczekujące (admin,numer towaru,exit)

3)Status

-Admin pyta o hasło oraz daje możliwości admina.

-Wpisanie numeru towaru – serwer odpowiada wiadomoscią z informacjej dotyczącej towaru.

-Exit zamyka tryby oraz klienta

# Aplikacja serwera

## Wstęp

Serwer.c ustawia potrzebne zmienne -> wywoła otwiera plik a synchronicznie zapisuje wszystko do structury -> stara się usłychaćklienta a przyjąć pakiet.

## Algorytm

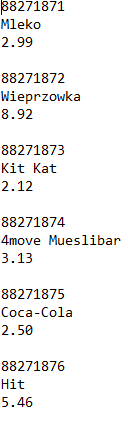
1)Serwer po otrzymaniu pakieta, przypisuję to do buffera a od razu sprawdza na identyczność z funkcjami.  
2)Jeśli serwer spawdził warunek „admin” lub „exit” – wykona kod wewnętrzny

Jeśli nie to działa w trybie złykłym , czyli wybiera dane powiązane z otrzymanym ID oraz przesyła do klienta

# Przechowywane dane

Dane są przychowywane w pliku file.txt. Są w nim linijki wierszy. Odczytujemy za pomocą fopen file. Potem nadajemu tryb pliku „r” czy „a” . zapisujemy wszystkie linijki rozdzielnie od innych towarów do structury products.

# Interfejs danych



## Funkcja 1

Dodaj func

## Spis komunikatów błędów

**Socked creation failed**

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

if (sockfd == -1)

{

printf("socket creation failed...\n");

exit(0);

}

**Failed binding**

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0)

{

printf("socket bind failed...\n");

exit(0);

}

**Listening**

if ((listen(sockfd, 5)) != 0)

{

printf("Listen failed...\n");

exit(0);

}

## Odebranie pakietu od klienta

## connfd = accept(sockfd, (SA\*)&cli, &len);

## if (connfd < 0){

## printf("server acccept failed...\n");

## exit(0);

## }

**Plik jest pusty**

if(pFile==NULL)

{

perror("Error opening file.");

}

**Wprowadzony numer istnieje**

**Numer musi być 8 cyfrowy**

# Planowany podział prac

Jest podzieliona 50 na 50

Wszystkie podejścia do robienia zadania były organizowane żeby każda osoba wypełniła takie zadania jak *programowanie , testowanie,myślenia logiki, opisanie sprawozdania*.