**Rok akademicki: 2019/2020**

**Telekomunikacja i prezentacja sygnałów**

**Ćwiczenie:**

Xmodem – implementacja protokołu transferu plików.

**Autorki:** Karolina Łyskowicz 224362

Monika Karpińska 224321

**Cel ćwiczenia:**

Celem zadania było napisanie programu, który pełni funkcję odbiornika wykorzystującego implementację protokołu Xmodem do transferu plików. Należało wykorzystać konfigurację portu szeregowego RS-232.

**Opis programu:**

Zadanie postanowiłyśmy rozbić na dwie części i wykonać dwa oddzielne programy. Jeden z nich pełni funkcję nadajnika, natomiast drugi odpowiada za odbiornik.

Na początku działania każdego z programów należy wykonać odpowiednią konfigurację portów szeregowych. Wykorzystujemy do tego bibliotekę <windows.h>

bool Odbieranie::ustawPort() {

uchwyt = CreateFile("COM2", GENERIC\_WRITE | GENERIC\_READ, 0, NULL, OPEN\_EXISTING, 0, NULL);

if (uchwyt != INVALID\_HANDLE\_VALUE) {

deviceControlBlock.DCBlength = sizeof(deviceControlBlock);

deviceControlBlock.BaudRate = CBR\_9600;

deviceControlBlock.fParity = TRUE;

deviceControlBlock.Parity = NOPARITY

deviceControlBlock.StopBits = ONESTOPBIT;

deviceControlBlock.ByteSize = 8;

deviceControlBlock.fDtrControl = DTR\_CONTROL\_DISABLE;

deviceControlBlock.fRtsControl = RTS\_CONTROL\_DISABLE;

deviceControlBlock.fOutxCtsFlow = FALSE;

deviceControlBlock.fOutxDsrFlow = FALSE;

deviceControlBlock.fDsrSensitivity = FALSE;

deviceControlBlock.fAbortOnError = FALSE;

deviceControlBlock.fOutX = FALSE;

deviceControlBlock.fInX = FALSE;

deviceControlBlock.fErrorChar = FALSE;

deviceControlBlock.fNull = FALSE;

SetCommState(uchwyt, &deviceControlBlock);

return true;

}

else return false;

}

Powyżej zamieszczony został fragment kodu odpowiedzialny z konfigurację portów w środowisku Windows. Prędkość transmisji została ustawiona na 9600 bitów na sekundę, liczba przesyłanych bitów danych to 8, zrezygnowano z bitu kontroli parzystości oraz ustawiono jeden bit stopu.

Podczas przesyłania plików za pomocą protokołu Xmodem dzielimy plik na bloki 128 bajtów poprzedzając je 3 bajtami nagłówka. Nagłówek składa się ze znaku SOH (Start of Header), numeru wysyłanego bloku (z przedziału 0-255) oraz trzeciego bajtu będącego dopełnieniem numeru bloku do 255. Do bloku dołączana jest również suma kontrolna.

Dla podstawowej wersji protokołu sumę kontrolną stanowi suma wszystkich elementów bloku danych modulo 256. Dla wersji protokołu z CRC suma kontrolna składa się z 2 bajtów zamiast z jednego.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

char Wysylanie::sumaKontrolna() {

char suma = 0;

for(int i=0; i<128; i++) {

suma += this->blok[i];

suma = suma % 256;

}

return suma;

}

//////////////////////////////////////////////////////////////////////////////////////

unsigned int Wysylanie::CRC(char \*dane, int ileZnakow) {// blok danych , ilosc = 128

uint8\_t i;

uint16\_t wCrc = 0xffff;

while (ileZnakow--) {

wCrc ^= \*(unsigned char \*)dane++ << 8;

for (i=0; i < 8; i++)

wCrc = wCrc & 0x8000 ? (wCrc << 1) ^ 0x1021 : wCrc << 1;

}

return wCrc & 0xffff;

}

char Wysylanie::SumaCRC(int liczba, int ktoryBajt)

{

int binarna[16];

int reszta = 0;

for(int i = 0; i < 16; i++) {

binarna[i] = 0;

}

for(int i = 0; i < 16; i++) {

reszta = liczba % 2;

if (reszta == 1) liczba = (liczba - 1) / 2;

if (reszta == 0) liczba = liczba / 2;

binarna[15-i] = reszta;

}

int koniec;

int x = 0;

if(ktoryBajt == 1) koniec = 7;

if(ktoryBajt == 2) koniec = 15;

for (int i = 0; i < 8; i++)

x = x + Potega2(i) \* binarna[koniec - i];

std::cout << "crc bajt: " << ktoryBajt<< ". " << x <<'\n';

return (char)x;

}

//////////////////////////////////////////////////////////////////////////////////////

Powyżej zamieszczone zostały odpowiednie fragmenty kodu odpowiadające za obliczenie sum kontrolnych w obu wariantach.

Transmisja plików rozpoczyna się wysłaniem przez odbiorcę znaku NAK (wersja podstawa) lub C (wersja z CRC). Nadawca odpowiada wysyłając znak ACK.

Następnie rozpoczynana jest transmisja bloku danych (128 bajtów) wraz z nagłówkiem (trzy bajty) i sumą kontrolną(jeden bądź dwa bajty). Odbiorca sprawdza poprawność sumy kontrolnej i odpowiada znakiem ACK, jeżeli otrzymane dane są poprawne. W innym przypadku odpowiada znakiem NAK, który wymaga ponownego przesłania bloku przez nadawcę. Transmisję kończy wysłanie przez nadawcę znaku EOT(End of Transmision) oraz odpowiedzi ACK ze strony odbiorcy.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

bool Wysylanie::wyslaniePliku(){

int flaga;

std::cout << "Oczekiwanie na znak....\n";

ReadFile(uchwyt, &znak, licznikZnakow, &rozmiarZnaku, NULL);

std::cout << "Otrzymano znak NAK/C : " << znak <<'\n';

if(znak == NAK) {

flaga = 1;

}

else if( znak == C){

flaga = 2;

}

std::ifstream plik;

plik.open(this->nazwa,std::fstream::binary);

int nrBloku = 0;

if(plik.good()){

std::vector<char> buforPliku;

int rozmiarPliku = 0;

while(plik){

char b = plik.get();

if(plik){

buforPliku.push\_back(b);

rozmiarPliku ++;

}

}

plik.close();

int iter = 0;

while(iter <= rozmiarPliku){

bool czyWyslano = false;

this->naglowek[0] = SOH;

this->naglowek[1] = (char)nrBloku;

this->naglowek[2] = (char)(255 - nrBloku);

for(int i=0; i<128; i++){

this->blok[i] =buforPliku[i+iter];

if(i+iter > rozmiarPliku){

this->blok[i] = '0';

}

}

iter +=128;

do{

WriteFile(uchwyt, &naglowek[0],licznikZnakow,&rozmiarZnaku, NULL);

WriteFile(uchwyt, &naglowek[1],licznikZnakow,&rozmiarZnaku, NULL);

WriteFile(uchwyt, &naglowek[2],licznikZnakow,&rozmiarZnaku, NULL);

for(int i =0; i < 128; i++){

WriteFile(uchwyt, &blok[i], 1, &rozmiarZnaku, NULL);

}

if(flaga == 1){

char suma;

suma = sumaKontrolna();

WriteFile(uchwyt, &suma, 1, &rozmiarZnaku, NULL);

}

else if(flaga == 2){

int crc = CRC(this->blok,128);

char suma[2];

suma[0] = SumaCRC(crc, 1);

suma[1] = SumaCRC(crc, 2);

std::cout << crc << '\n';

WriteFile(uchwyt, &suma, 2, &rozmiarZnaku, NULL);

}

ReadFile(uchwyt, &znak, 1, &rozmiarZnaku, NULL);

if(znak == ACK){

czyWyslano = true;

std::cout << "Otrzymano ACK, blok danych nr: " << nrBloku <<" zostal wyslany\n";

}

if(znak == NAK){

std::cout << "Pakiet nie zostal wyslany poprawnie. Ponowienie proby........\n";

}

}while(!czyWyslano);

nrBloku++;

if(nrBloku == 256)

nrBloku = 0;

}

WriteFile(uchwyt, &EOT, 1, &rozmiarZnaku, NULL);

std::cout << "Wyslano EOT\n";

ReadFile(uchwyt, &znak, 1, &rozmiarZnaku, NULL);

if(znak == ACK){

std::cout << "Przesylanie zakonczone" << std::endl;

}

}

else{

std::cout << "Nie odnaleziono pliku\n";

CloseHandle(uchwyt);

return false;

}

CloseHandle(uchwyt);

return true;

}

///////////////////////////////////////////////////////////////////////////////////////////

bool Odbieranie::odbieraniePliku(int flaga) {

std::ofstream plik;

plik.open(nazwa, std::ios::binary);

int nrBloku = 0;

char sumaSprawdzenie;

if(flaga == 1)

znak = NAK;

else if(flaga == 2)

znak = C;

else {

std::cout <<"Niepoprawny Wybor\n";

return 0;

}

std::cout <<"Wysylam NAK/C\n";

for(int i = 0; i < 20; i++){

WriteFile(uchwyt, &znak, ileZnakow, &rozmiarZnaku, NULL);

ReadFile(uchwyt, &znak, 1, &rozmiarZnaku, NULL);

if( znak == SOH) {

break;

}

sleep(3000);

}

while(znak == SOH){

bool czyOdebrane = false;

bool poprawnaSuma = false;

while(!czyOdebrane){

ReadFile(uchwyt, &nrBloku, 1, &rozmiarZnaku, NULL);

std::cout <<"Odbieram blok danych nr: " << (int)nrBloku <<'\n';

ReadFile(uchwyt, &znak, 1, &rozmiarZnaku, NULL);

if(znak != (char)(255 - nrBloku)){

std::cout << "Numer pakietu nie jest poprawny!!!\n";

}

for(int i =0; i < 128; i++){

ReadFile(uchwyt, &blok[i], 1, &rozmiarZnaku, NULL);

}

if(flaga == 1 ){

char suma;

ReadFile(uchwyt, &suma, 1, &rozmiarZnaku, NULL);

sumaSprawdzenie = sumaKontrolna();

if(suma == sumaSprawdzenie){

poprawnaSuma = true;

}

}

else{

char suma[2];

char sumaCrcSprawdzenie[2];

ReadFile(uchwyt, &suma, 2, &rozmiarZnaku, NULL);

int crc = CRC(blok,128);

sumaCrcSprawdzenie[0] = SumaCRC(crc,1);

sumaCrcSprawdzenie[1] = SumaCRC(crc,2);

if(suma[0] == sumaCrcSprawdzenie[0] && suma[1] == sumaCrcSprawdzenie[1]){

poprawnaSuma = true;

}

}

if(poprawnaSuma) {

std::cout << "Udalo sie przeslac pakiet! \n";

int ile = 128;

if(blok[127] == '0' and blok[126] == '0' and blok[125] == '0') {

for (int i = 127; i >= 0 and this->blok[i] == '0'; i--) {

ile--;

}

}

std::vector<char> buforPliku;

for(int i = 0 ; i < ile; i++){

buforPliku.push\_back(blok[i]);

}

plik.write(buforPliku.data(), buforPliku.size());

WriteFile(uchwyt, &ACK, 1, &rozmiarZnaku, NULL);

czyOdebrane = true;

}

else{

std::cout << "Nie udalo sie przeslac pakietu :/\n"

"Przesylam NAK\n";

WriteFile(uchwyt, &NAK, 1, &rozmiarZnaku, NULL);

}

}

ReadFile(uchwyt, &znak, 1, &rozmiarZnaku, NULL);

}

if(znak == EOT){

std::cout << "Odebrano EOT, transmisja zakonczona sukcesem! \n";

WriteFile(uchwyt, &ACK, 1, &rozmiarZnaku, NULL);

}

plik.close();

}

//////////////////////////////////////////////////////////////////////////////////////

Powyższe fragmenty kodu zawierają funkcje odpowiadające za wysłanie oraz odbieranie pliku.