МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Лабораторная работа №2

по дисциплине

«Информатика и программирование»

Классы и шифры

Группа: **АБс-822**

Студент: **Соленый Н.Е.**

Преподаватель:

НОВОСИБИРСК

2019

**Задача:**

Реализовать интерфейс для теста алгоритмов шифрования на выбор пользователя. Реализовать и сами алгоритмы на C++, применив ООП. Для секретной информации использовать атрибуты private. Задачу выполнить для следующих алгоритмов: RSA, шифр Шамира, шифр Эль-Гамаля, система Диффи-Хеллмана.

UPD: В ходе работы помимо основной задачи была решена задача обмена данными через канал двойной связи (прямой канал и обратный), реализованный элементарно через два бинарных файла, задача не оказалась тривиальной: блок-схема и описание подводных камней прилагаются.

**Структура проекта**

**Интерфейс взаимодействия с пользователем**

// (main.cpp)

#include <iostream>

#include <typeinfo>

#include "shifres.hpp"

using namespace std;

enum class Type {SHAMIR=1, DIFFI\_HELLMAN, RSA, EL\_GHAMAL};

template <typename Enumeration>

auto as\_integer(Enumeration const value) -> typename std::underlying\_type<Enumeration>::type {

return static\_cast<typename std::underlying\_type<Enumeration>::type>(value);

}

template<>

constexpr auto as\_integer<Type>(Type const value) -> typename std::underlying\_type<Type>::type {

int n = static\_cast<typename std::underlying\_type<Type>::type>(value);

if (n < 1 || n > 4)

throw std::runtime\_error("Incorrect input");

return n;

}

Type chooseEncode() {

int t;

inputerror:

cout << "1 - Shamir`s\n"

"2 - Diffi-Hellman`s\n"

"3 - RSA\n"

"4 - El-Ghamal`s\n"

"Enter type of simulated encryption algorythm: ";

cin >> t;

cout << endl;

try {

Type encode = static\_cast<Type>(t);

cout << as\_integer(encode) << endl;

return encode;

}

catch (std::exception &e) {

cerr << e.what() << endl;

goto inputerror;

}

}

void emulateRecipient(Type type) {

init(".recipient", ".dispatcher"); /// this is a crutch. it shouldn't be here, but we have it

switch (type) {

case Type::DIFFI\_HELLMAN: {

Diffi\_Hellman session;

cout << "\nMy shared key is " << session.sharedKey <<

". Just reporting, nothing more. I won`t tell you my secret key." << endl;

cout << "processing..." << endl;

session.recipient\_protocol();

cout << "\nI`m abonent B. My secret number is " << session.getEvaluatedNumber() <<

"; abonent A has equal secret number" << endl;

break;

}

case Type::EL\_GHAMAL: {

El\_Ghamal session;

cout << "processing..." << endl;

session.recipient\_protocol();

cout << "Got message! (output.txt)" << endl;

break;

}

case Type::SHAMIR: {

Shamir session;

cout << "processing..." << endl;

session.recipient\_protocol();

cout << "Got message! (output.txt)" << endl;

break;

}

case Type::RSA: {

RSA session;

cout << "processing..." << endl;

session.recipient\_protocol();

cout << "Got message! (output.txt)" << endl;

break;

}

}

}

void emulateDispatcher(Type type) {

init(".dispatcher", ".recipient"); /// this is a crutch. it shouldn't be here, but we have it

switch (type) {

case Type::DIFFI\_HELLMAN: {

Diffi\_Hellman session;

cout << "\nMy shared key is " << session.sharedKey <<

". Just reporting, nothing more. I won`t tell you my secret key." << endl;

cout << "processing..." << endl;

session.dispatcher\_protocol();

cout << "\nI`m abonent A. My secret number is " << session.getEvaluatedNumber() <<

"; abonent B has equal secret number" << endl;

break;

}

case Type::EL\_GHAMAL: {

El\_Ghamal session;

cout << "processing..." << endl;

session.dispatcher\_protocol();

cout << "Message input.txt was sent" << endl;

break;

}

case Type::SHAMIR: {

Shamir session;

cout << "processing..." << endl;

session.dispatcher\_protocol();

cout << "Message input.txt was sent" << endl;

break;

}

case Type::RSA: {

RSA session;

cout << "processing..." << endl;

session.dispatcher\_protocol();

cout << "Message input.txt was sent" << endl;

break;

}

}

}

void chooseMode(Type type) {

int m;

err:

cout << "1 - Emulate recipient\n"

"2 - Emulate dispatcher to encrypt file (shared key of any recipient required)\n"

"Choose mode: ";

cin >> m;

cout << endl;

switch(m) {

case DISPATCHER:

emulateDispatcher(type);

break;

case RECIPIENT:

emulateRecipient(type);

break;

default:

goto err;

}

};

int main(int argc, char\*argv[]) {

if(argc==1) {

Type type = chooseEncode();

chooseMode(type);

system("pause");

} else {

Environment &Environment = Environment::Instance();

Type type = static\_cast<Type>(0);

void (\*f)(Type) = nullptr;

for(int i = 1; i < argc; i++) {

if(argv[i][0]!='-') throw std::runtime\_error("Incorrect input\n");

else {

switch(argv[i][1]) {

case 'i':

Environment.input = string(argv[++i]);

break;

case 'o':

Environment.output = string(argv[++i]);

break;

case 'b':

Environment.enableBinary = true;

break;

case '-': {

string str(argv[i]+2);

if(str==string("diffi-hellman")) type = Type::DIFFI\_HELLMAN;

else if (str==string("shamir")) type = Type::SHAMIR;

else if (str==string("el-ghamal")) type = Type::EL\_GHAMAL;

else if (str==string("rsa")) type = Type::RSA;

else throw std::runtime\_error("Incorrect input\n");

break;

}

case 'r':

f = emulateRecipient;

break;

case 'd':

f = emulateDispatcher;

break;

}

}

}

if(f == nullptr || type == static\_cast<Type>(0)) throw std::runtime\_error("Incorrect input\n");

f(type);

}

return 0;

}

Выбор, какого клиента эмулировать, отправителя или получателя. Выбор алгоритма для теста. Защита ввода. Пользователю нужно запустить два экземпляра программы и дождаться выполнения (передачи файла или удачного соединения). Также присутствует небольшой скрипт для тестов:

// (bin/Release/script.sh)

echo '1 - Shamir`s

2 - Diffi-Hellman`s

3 - RSA

4 - El-Ghamal`s

Enter type of simulated encryption algorythm: '

read T

if [ $T == 2 ]

then

echo "$T 1 " | ./001.exe | grep ' secret number is ' &

echo "$T 2 " | ./001.exe | grep ' secret number is '

else

echo "$T 1 " | ./001.exe > /dev/null &

echo "$T 2 " | ./001.exe > /dev/null

if [ -e ./.dispatcher ]

then

rm ./.dispatcher

fi

if [ -e ./.recipient ]

then

rm ./.recipient

fi

fi

**Описание классов алгоритмов через наследование**

// (shifres.hpp)

#include "locker.h"

#include <fstream>

#include <iostream>

#include <boost/multiprecision/cpp\_int.hpp> //1\_71\_0

#include <boost/random.hpp>

#include "sh.h"

#undef EXIT\_SUCCESS

enum Mode {RECIPIENT=1, DISPATCHER};

using namespace std;

using namespace boost::multiprecision;

class Environment {

public:

int encrMaxNumber, encrSuppNumber;

string input, output;

bool enableBinary;

static Environment& Instance() { static Environment s; return s; }

private:

Environment() { encrMaxNumber = 1000667, encrSuppNumber = 500333; input = "input.txt", output = "output.txt"; }

~Environment() { }

Environment(Environment const&) = delete;

Environment& operator= (Environment const&) = delete;

};

class Encoded\_Structure {

public:

int sharedKey;

virtual void recipient\_protocol();

virtual void dispatcher\_protocol();

enum class FileStat{ EXIT\_SUCCESS, FAILED, UNNAMED\_ERROR };

protected:

virtual void decode(void\* data) = 0;

virtual void encode(void\* data) = 0;

virtual void takeSharedKey();

virtual void giveSharedKey();

void waitTilReady(WaitMode m);

uint8\_t takeByte();

uint8\_t sendByte(uint8\_t data);

void sendEOF();

void write(int byte);

int read();

int takenSharedKey;

};

class Diffi\_Hellman : public Encoded\_Structure {

public:

Diffi\_Hellman();

void recipient\_protocol() override final;

void dispatcher\_protocol() override final;

int getEvaluatedNumber() { return evaluatedNumber; };

private:

inline void decode(void\* data) override;

inline void encode(void\* data) override;

int hiddenKey, evaluatedNumber;

};

class Shamir : public Encoded\_Structure {

public:

Shamir();

void recipient\_protocol() override final;

void dispatcher\_protocol() override final;

private:

inline void decode(void\* data) override;

inline void encode(void\* data) override;

int hiddenKey1, hiddenKey2;

};

class El\_Ghamal : public Encoded\_Structure {

public:

El\_Ghamal();

private:

int hiddenKey;

inline void decode(void\* data) override;

inline void encode(void\* data) override;

};

class RSA : public Encoded\_Structure {

public:

RSA();

uint1024\_t sharedModulus, sharedExponent;

private:

uint1024\_t hiddenExponent, takenModulus, takenExponent;

void takeSharedKey() override;

void giveSharedKey() override;

inline void decode(void\* data) override;

inline void encode(void\* data) override;

uint1024\_t pows(const uint1024\_t &a, const uint1024\_t &x, const uint1024\_t &p);

uint1024\_t mod\_inverse(const uint1024\_t &a, const uint1024\_t &p);

bool ferma(const uint1024\_t& num);

boost::random::mt11213b gen;

};

Классы алгоритмов наследуются от основного класса, реализующего некоторые общие методы (чтения, записи данных, реализацию каналов их передачи). При этом, наследуясь от базового класса, потомки должны реализовать некоторые виртуальные методы родителя.

**Реализация алгоритмов**

0. Общий класс

Реализация общих, одинаковых для некоторых подклассов методов обмена сообщениями и ключами. Реализация служебных методов: чтение, запись, ожидание готовности потока.

// (encoded\_structure.cpp)

#include "shifres.hpp"

//#define DEB

#ifdef DEB

#define LOG(N) cout << N;

#else

#define LOG(N)

#endif

void Encoded\_Structure::waitTilReady(WaitMode m) {

switch(m) {

case REV\_STRT\_UNLOCKED: {

do {

LOG(1);

trylock\_reverse\_channel();

} while(errno != 0);

do {

LOG(2);

trylock\_straight\_channel();

} while(errno != 0);

break;

}

case OREV\_LOCKED: {

do {

LOG(3);

trylock\_reverse\_channel();

} while(errno != EACCES);

do {

LOG(4);

trylock\_straight\_channel();

} while(errno != 0);

break;

}

case REV\_LOCKED: {

do {

LOG(5);

trylock\_reverse\_channel();

} while(errno != EACCES);

break;

}

case REV\_UNLOCKED: {

do {

LOG(6);

trylock\_reverse\_channel();

} while(errno != 0);

break;

}

}

}

int Encoded\_Structure::read() {

Environment &Environment = Environment::Instance();

static ifstream file(Environment.input, ios::binary|ios::in);

uint8\_t data;

file.get((char&)data);

if(file.eof()) {

file.close();

throw FileStat::EXIT\_SUCCESS;

}

return data;

}

void Encoded\_Structure::write(int data) {

Environment &Environment = Environment::Instance();

static ofstream file(Environment.output, ios::binary|ios::out);

if (data == EOF) {

file.close();

return;

}

file << static\_cast<uint8\_t>(data);

}

uint8\_t Encoded\_Structure::takeByte() { /// deprecated, dye to the impossibility of mixed use with sendByte

waitTilReady(OREV\_LOCKED);

lock\_straight\_channel();

waitTilReady(REV\_UNLOCKED);

int data;

data = readByte();

unlock\_reverse\_channel();

unlock\_straight\_channel();

if (data == EOF)

throw FileStat::EXIT\_SUCCESS;

return data;

}

uint8\_t Encoded\_Structure::sendByte(uint8\_t data) { /// deprecated, dye to the impossibility of mixed use with takeByte

// not turned out to write nice abstraction of transmission protocol

waitTilReady(REV\_STRT\_UNLOCKED);

putByte(data);

waitTilReady(REV\_LOCKED);

unlock\_straight\_channel();

waitTilReady(REV\_STRT\_UNLOCKED);

uint8\_t answer = readByte();

unlock\_reverse\_channel();

return answer;

}

void Encoded\_Structure::sendEOF() {

waitTilReady(REV\_STRT\_UNLOCKED);

putByte(EOF);

waitTilReady(REV\_LOCKED);

unlock\_straight\_channel();

}

void Encoded\_Structure::recipient\_protocol() {

int sz = 1;

Encoded\_Structure\* r = dynamic\_cast<RSA\*>(this);

if(r)sz = 34;

r = dynamic\_cast<El\_Ghamal\*>(this);

if(r)sz = 2;

takeSharedKey();

while(true) {

waitTilReady(OREV\_LOCKED);

lock\_straight\_channel();

waitTilReady(REV\_UNLOCKED);

int byte[sz] = {0};

if(readBytes(byte, sz\*4) == EOF) { write(EOF); unlock\_reverse\_channel(); unlock\_straight\_channel(); break; }

unlock\_reverse\_channel();

unlock\_straight\_channel();

decode(byte);

write(byte[0]);

}

}

void Encoded\_Structure::dispatcher\_protocol() {

int sz = 1;

Encoded\_Structure\* r = dynamic\_cast<RSA\*>(this);

if(r)sz = 34;

r = dynamic\_cast<El\_Ghamal\*>(this);

if(r)sz = 2;

giveSharedKey();

try {

while(true) {

int byte[sz] = {0};

byte[0] = read();

encode(byte);

waitTilReady(REV\_STRT\_UNLOCKED);

putBytes(byte, sz\*4);

waitTilReady(REV\_LOCKED);

unlock\_straight\_channel();

}

}

catch (FileStat) {

sendEOF();

}

}

void Encoded\_Structure::takeSharedKey() {

waitTilReady(OREV\_LOCKED);

putBytes(&sharedKey, 4);

waitTilReady(REV\_UNLOCKED);

readBytes(&takenSharedKey, 4);

unlock\_reverse\_channel();

unlock\_straight\_channel();

}

void Encoded\_Structure::giveSharedKey() {

waitTilReady(REV\_STRT\_UNLOCKED);

putBytes(&sharedKey, 4);

waitTilReady(REV\_LOCKED);

unlock\_straight\_channel();

waitTilReady(REV\_STRT\_UNLOCKED);

readBytes(&takenSharedKey, 4);

unlock\_reverse\_channel();

}

1. Система Диффи-Хеллмана

Здесь абоненты просто установят связь и, вычислив секретное число, покажут пользователю.

// (diffi\_hellman.cpp)

#include "shifres.hpp"

Diffi\_Hellman::Diffi\_Hellman() {

Environment &Environment = Environment::Instance();

hiddenKey = gen\_k(Environment.encrMaxNumber);

sharedKey = pows(Environment.encrSuppNumber, hiddenKey, Environment.encrMaxNumber);

};

inline void Diffi\_Hellman::decode(void\* data) { }

inline void Diffi\_Hellman::encode(void\* data) { }

void Diffi\_Hellman::recipient\_protocol() {

Environment &Environment = Environment::Instance();

takeSharedKey();

evaluatedNumber = pows(takenSharedKey, hiddenKey, Environment.encrMaxNumber);

}

void Diffi\_Hellman::dispatcher\_protocol() {

Environment &Environment = Environment::Instance();

giveSharedKey();

evaluatedNumber = pows(takenSharedKey, hiddenKey, Environment.encrMaxNumber);

}

2. Шифр Шамира

Протокол передачи сообщения отличается от установленного по умолчанию – требуется три пересылки сообщения. Поэтому виртуальные методы перегружены.

// (shamir.cpp)

#include "shifres.hpp"

Shamir::Shamir() { }

inline void Shamir::decode(void\* data) {

\*((int\*)data) = pows(\*((int\*)data), hiddenKey2, Environment::Instance().encrMaxNumber);

}

inline void Shamir::encode(void\* data) {

\*((int\*)data) = pows(\*((int\*)data), hiddenKey1, Environment::Instance().encrMaxNumber);

}

void Shamir::recipient\_protocol() {

Environment &Environment = Environment::Instance();

takeSharedKey();

while(true){

choose\_c\_d(&hiddenKey1, &hiddenKey2, Environment.encrMaxNumber);

waitTilReady(OREV\_LOCKED);

lock\_straight\_channel();

waitTilReady(REV\_UNLOCKED);

int byte;

if(readBytes(&byte, 4) == EOF) { write(EOF); break; }

encode(&byte);

unlock\_straight\_channel();

putBytes(&byte, 4);

unlock\_reverse\_channel();

unlock\_straight\_channel();

waitTilReady(REV\_LOCKED);

lock\_straight\_channel();

waitTilReady(REV\_UNLOCKED);

readBytes(&byte, 4);

unlock\_reverse\_channel();

unlock\_straight\_channel();

decode(&byte);

write(byte);

}

}

void Shamir::dispatcher\_protocol() {

Environment &Environment = Environment::Instance();

giveSharedKey();

try {

while(true) {

choose\_c\_d(&hiddenKey1, &hiddenKey2, Environment.encrMaxNumber);

int byte = read();

encode(&byte);

waitTilReady(REV\_STRT\_UNLOCKED);

putBytes(&byte, 4);

waitTilReady(REV\_LOCKED);

unlock\_straight\_channel();

waitTilReady(REV\_STRT\_UNLOCKED);

int answer;

readBytes(&answer, 4);

decode(&answer);

unlock\_reverse\_channel();

waitTilReady(REV\_STRT\_UNLOCKED); // as a precaution

putBytes(&answer, 4);

waitTilReady(REV\_LOCKED);

unlock\_straight\_channel();

}

} catch(FileStat) {

sendEOF();

}

}

3. Шифр Эль-Гамаля

Перегружены только необходимые методы кодирования.

// (el\_ghamal.cpp)

#include "shifres.hpp"

El\_Ghamal::El\_Ghamal() {

Environment &Environment = Environment::Instance();

hiddenKey = gen\_c(Environment.encrMaxNumber);

sharedKey = eval\_d(hiddenKey, Environment.encrSuppNumber, Environment.encrMaxNumber);

}

inline void El\_Ghamal::decode(void\* data) {

int r = \*((int\*)data), e = \*((int\*)data+1);

\*((int\*)data) = eval\_m(e, r, hiddenKey, Environment::Instance().encrMaxNumber);

}

inline void El\_Ghamal::encode(void\* data) {

Environment &Environment = Environment::Instance();

int k = gen\_k(Environment.encrMaxNumber);

int b = \*((int\*)data);

\*((int\*)data) = eval\_r(Environment.encrSuppNumber, k, Environment.encrMaxNumber);

\*((int\*)data+1) = eval\_e(b, takenSharedKey, k, Environment.encrMaxNumber);

}

4. Алгоритм RSA

Алгоритм реализован в полном объеме, с 1024-битными ключами. Для этого пришлось подключить boost. Некоторые методы переписаны на длинную арифметику. Переопределены методы кодирования и обмена ключами.

// (rsa.cpp)

#include "shifres.hpp"

#include <boost/integer/common\_factor\_rt.hpp> // gcd

#include <time.h>

using namespace boost::multiprecision;

inline bool RSA::ferma(const uint1024\_t &num) { // checks if prime; O(K\*logN)

if(num == 2)

return true;

boost::random::uniform\_int\_distribution<uint1024\_t> rand(1, std::numeric\_limits<uint1024\_t>::max());

for(int i=0;i<200;i++){

uint1024\_t a = (rand(gen) % (num - 2)) + 2;

if (boost::integer::gcd\_evaluator<uint1024\_t>()(a, num) != 1)

return false;

if( pows(a, num-1, num) != 1)

return false;

}

return true;

}

inline uint1024\_t RSA::pows(const uint1024\_t &a, const uint1024\_t &x, const uint1024\_t &p) {

number<cpp\_int\_backend<2048, 2048, unsigned\_magnitude, unchecked, void> > y = 1;

uint16\_t offset = 1023;

for(;;offset--) {

y\*=y;

y%=p;

if((x>>offset)&0x1) {

y\*=a;

y%=p;

}

if(offset==0)

break;

}

return static\_cast<uint1024\_t>(y);

}

inline uint1024\_t RSA::mod\_inverse(const uint1024\_t &a, const uint1024\_t &p) {

number<cpp\_int\_backend<2048, 2048, signed\_magnitude, unchecked, void> > u1=p,u2=0,v1=a,v2=1,t1,t2,q;

while (v1!=0) {

q=u1/v1; t1=u1%v1; t2=u2-q\*v2;

u1=v1;u2=v2;v1=t1;v2=t2;

}

while(u2<0)u2+=p;

return static\_cast<uint1024\_t>(u2);

}

RSA::RSA() {

int t = time(NULL);

boost::random::uniform\_int\_distribution<uint1024\_t> rand( RSA::pows(2, 510, std::numeric\_limits<uint512\_t>::max())-t, std::numeric\_limits<uint512\_t>::max()-t );

uint1024\_t P = rand(gen)+t, Q = rand(gen)+t;

while(!ferma(++P)) if(P >= std::numeric\_limits<uint512\_t>::max()-2) { P = rand(gen)+t; }

while(!ferma(++Q)) if(Q >= std::numeric\_limits<uint512\_t>::max()-2) { Q = rand(gen)+t; }

sharedModulus = P\*Q;

uint1024\_t eul = (P-1)\*(Q-1);

rand = boost::random::uniform\_int\_distribution<uint1024\_t>( RSA::pows(2, 510, std::numeric\_limits<uint512\_t>::max())-t, sharedModulus-t );

auto gcd = boost::integer::gcd\_evaluator<uint1024\_t>();

do { sharedExponent = rand(gen)+t; } while(gcd(sharedExponent, eul) != 1);

hiddenExponent = RSA::mod\_inverse(sharedExponent, eul);

}

inline void RSA::decode(void\* data) {

uint1024\_t n = \*((uint1024\_t\*)data);

\*((uint1024\_t\*)data) = RSA::pows(n, hiddenExponent, sharedModulus);

}

inline void RSA::encode(void\* data) {

uint1024\_t n = static\_cast<uint1024\_t>( \*((int\*)data) );

\*((uint1024\_t\*)data) = RSA::pows(n, takenExponent, takenModulus);

}

void RSA::takeSharedKey() {

waitTilReady(OREV\_LOCKED);

uint1024\_t arr[2]; arr[0] = sharedModulus, arr[1] = sharedExponent;

putBytes(arr, 272);

waitTilReady(REV\_UNLOCKED);

readBytes(arr, 272);

unlock\_reverse\_channel();

unlock\_straight\_channel();

takenModulus = arr[0], takenExponent = arr[1];

}

void RSA::giveSharedKey() {

waitTilReady(REV\_STRT\_UNLOCKED);

uint1024\_t arr[2]; arr[0] = sharedModulus, arr[1] = sharedExponent;

putBytes(arr, 272);

waitTilReady(REV\_LOCKED);

unlock\_straight\_channel();

waitTilReady(REV\_STRT\_UNLOCKED);

readBytes(arr, 272);

unlock\_reverse\_channel();

takenModulus = arr[0], takenExponent = arr[1];

}

**Реализация канала двойной связи**

Если провести мысленный эксперимент, нетрудно догадаться, что одного файла не хватит, чтобы провести передачу данных синхронно. Потребуется как минимум два файла-канала, один будет выступать в роли прямого канала, другой – обратного.

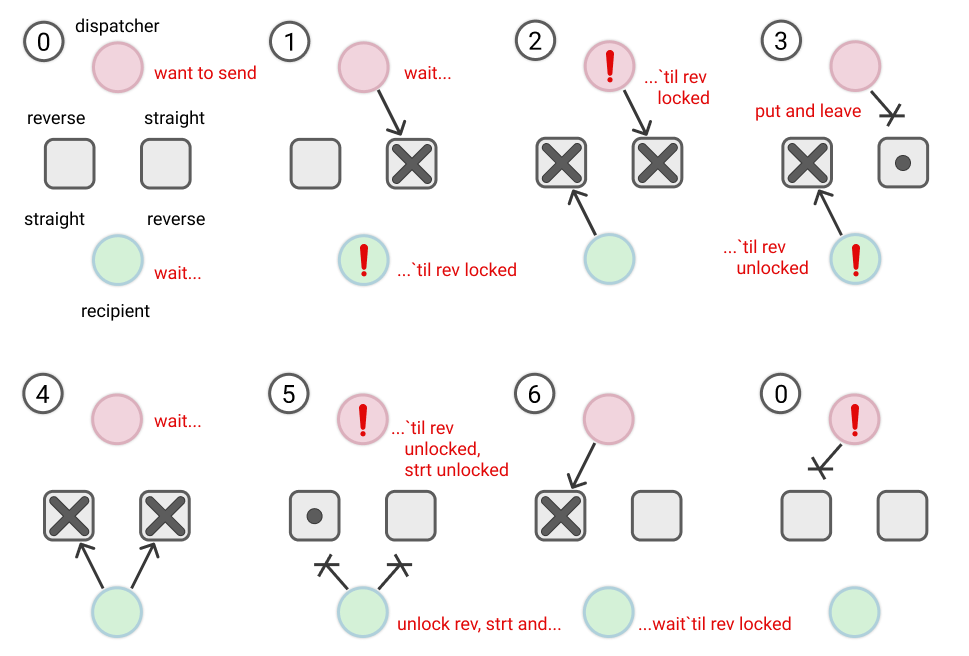
****

Рис.1 Единственный способ синхронной передачи данных через двойной канал

Имея возможности (реализованные в виде функций из библиотеки ниже) блокирования/разблокирования файла, чтения/записи(с блокировкой соответственно) и проверки, заблокирован ли файл, передачу организовать можно только изображенным на схеме образом.

UPD: Зачем передавать через файлы, ведь это можно сделать через localhost или разделяемую память? Это была первая идея. К тому же функции отправки можно заменить на использующие localhost, не трогая остальной код. Почти.

**Реализация блокировки файловых потоков**

// (locker.h)

#ifdef BUILD\_DLL

#define DLL\_EXPORT \_\_declspec(dllexport)

#else

#define DLL\_EXPORT \_\_declspec(dllimport)

#endif

#include <stdint-gcc.h>

#define EOF (-1)

enum WaitMode { OREV\_LOCKED=1, REV\_LOCKED=2, REV\_UNLOCKED=3, REV\_STRT\_UNLOCKED=5 };

#ifdef \_\_cplusplus

extern "C"

{

#endif

void DLL\_EXPORT init(const char\* straightChannel, const char\* inverseChannel);

int DLL\_EXPORT readByte(); /// only from reverse channel

int DLL\_EXPORT readBytes(void \*bytes, int cnt); /// only from reverse channel

void DLL\_EXPORT putByte(int data); /// only to straight channel

void DLL\_EXPORT putBytes(void \*bytes, int cnt); /// only to straight channel

void DLL\_EXPORT unlock\_straight\_channel();

void DLL\_EXPORT unlock\_reverse\_channel();

void DLL\_EXPORT lock\_reverse\_channel();

void DLL\_EXPORT lock\_straight\_channel();

void DLL\_EXPORT trylock\_reverse\_channel(); /// this function don`t tell, if file locked 100% probability

void DLL\_EXPORT trylock\_straight\_channel(); /// this function don`t tell, if file locked 100% probability

#ifdef \_\_cplusplus

}

#endif

// (locker.cpp)

#include "locker.h"

#include <stdio.h>

#include <share.h>

#include <string.h>

#include <errno.h>

static FILE \*stFile, \*invFile;

static char straightFilename[40] = {0}, inverseFilename[40] = {0};

void DLL\_EXPORT init(const char\* straight, const char\* inverse) {

int i = 0;

while(straight[i++]);

memcpy(straightFilename, straight, i);

i = 0;

while(inverse[i++]);

memcpy(inverseFilename, inverse, i);

fclose(fopen(straightFilename, "wb")), fclose(fopen(inverseFilename, "wb"));

}

void DLL\_EXPORT unlock\_straight\_channel() { fclose(stFile); }

void DLL\_EXPORT unlock\_reverse\_channel() { fclose(invFile); }

void DLL\_EXPORT lock\_reverse\_channel() {

do {

errno=0;

invFile = \_fsopen(inverseFilename, "rb", \_SH\_DENYRW);

} while(errno!=0);

}

void DLL\_EXPORT lock\_straight\_channel() {

do {

errno=0;

stFile = \_fsopen(straightFilename, "wb", \_SH\_DENYRW);

} while(errno!=0);

}

void DLL\_EXPORT trylock\_reverse\_channel() {

FILE \*f = \_fsopen(inverseFilename, "rb", \_SH\_DENYNO);

if(f) fclose(f), errno=0; /// anyway, fclose locks file

else errno=EACCES;

}

void DLL\_EXPORT trylock\_straight\_channel() {

FILE \*f = \_fsopen(straightFilename, "rb", \_SH\_DENYNO);

if(f) fclose(f), errno=0;

else errno=EACCES;

}

int DLL\_EXPORT readByte() {

do{ errno=0;

invFile = \_fsopen(inverseFilename, "rb", \_SH\_DENYRW);

}while(errno!=0);

int n = fgetc(invFile);

fgetc(invFile);

return feof(invFile) ? n : EOF;

}

int DLL\_EXPORT readBytes(void \*bytes, int cnt) {

do{ errno=0;

invFile = \_fsopen(inverseFilename, "rb", \_SH\_DENYRW);

}while(errno!=0);

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

int n = fgetc(invFile);

if(feof(invFile)) return EOF;

\*((uint8\_t\*)bytes+i) = n;

}

return 0;

}

void DLL\_EXPORT putByte(int data) {

do{ errno=0;

stFile = \_fsopen(straightFilename, "wb", \_SH\_DENYRW);

}while(errno!=0);

if(0 <= data && data < 256)

fputc(data, stFile);

else if (data == EOF) {

fputc(0xff, stFile);

fputc(0x01, stFile);

}

}

void DLL\_EXPORT putBytes(void \*bytes, int cnt) {

do{ errno=0;

stFile = \_fsopen(straightFilename, "wb", \_SH\_DENYRW);

}while(errno!=0);

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

fputc( \*((uint8\_t\*)bytes+i), stFile);

}

}

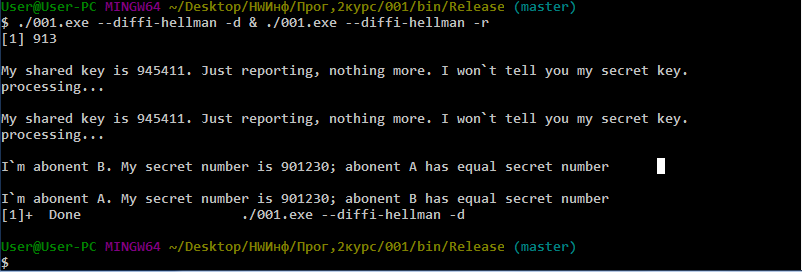
int \_\_attribute\_\_((\_\_stdcall\_\_)) DllMainCRTStartup(int hinstDLL, int fdwReason, void\* lpvReserved){

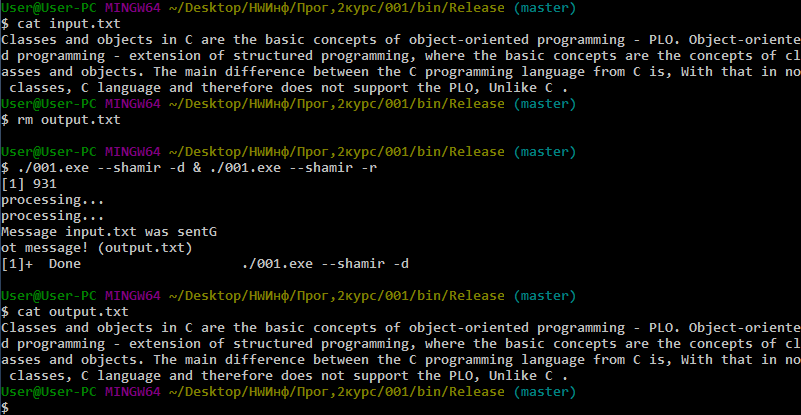
if(fdwReason == 0) remove(inverseFilename), remove(straightFilename);

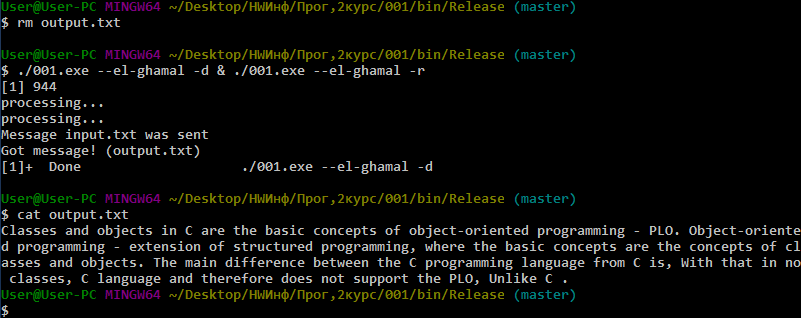
return 1;

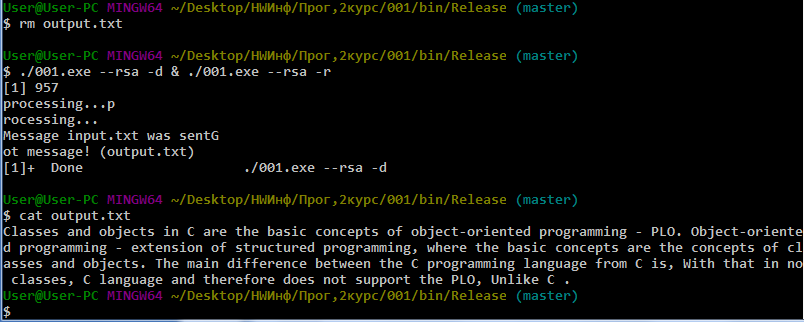
}

**Результат работы программы:**

****



****

****