Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №7

на тему:

**«Интеграция ассемблерных прерываний в проекты на C++.»**

БГУИР 6-05-0612-02 39

|  |
| --- |
| Выполнил студент группы 353503  ДЕРГУН Николай Олегович |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 9.** Текстовый редактор с шифрованием.

Реализуйте базовый текстовый редактор.

На стороне assembler: добавьте функцию шифрования текста с помощью ассемблерного прерывания. Реализуйте функции сохранения и открытия зашифрованных файлов.

Встроенное сообщение о нарушениях: если кто-то пытается открыть зашифрованный файл без соответствующего ключа более определенного числа раз, программа автоматически уничтожает или блокирует файл. Предложите несколько различных методов шифрования, таких как Цезарь, подстановка, транспозиция и другие (минимум 3 метода). Пользователь должен иметь возможность выбирать метод шифрования перед его применением. Методы должны быть реализованы как assembler-функции.

На стороне C++: меню и интерфейс пользователя. Управление функциями ассемблера: выбор метода шифрования, ввод ключа, чтение и запись файла и т.д. Логика по учету неудачных попыток доступа к файлу и последующего его уничтожения.

# 2 Выполнение работы

Программа реализована с использованием ассемблера nasm, библиотеки Qt, языка С++ для ОС Linux. Логика шифрования и дешифрования реализована с помощью ассемблерных функций, вызываемых из С++ кода с помощью экспортированных функций. Интерфейс и логика удаления файлов при трёхкратном вводе неверного пароля реализованы на стороне С++.

Листинг кода 1 – Внешние функции в С++

extern "C" void caesar\_cipher(int shift);

extern "C" int rand\_d(int limit);

extern "C" void vigenere\_cipher(int size, const char\* key);

extern "C" void vigenere\_decipher(int size, const char\* key);

extern "C" void transposition\_cipher(int div1, int div2);

Шифрование и дешифрование зачастую может быть реализовано с использованием одного и того же алгоритма, но с использованием разных ключей, что и было повсеместно использовано в коде.

Листинг кода 2 – Вызов assembler-функции из С++ кода

void CipherManager::EncryptVigenere(const QString& key)

{

vigenere\_cipher(key.size(), key.toStdString().data());

}

Листинг кода 3 – Пример реализованной функции для шифрования

section .data

src\_path db "/home/mikd/asm/cypher\_temp", 0

dst\_path db "/home/mikd/asm/cypher\_temp\_enc", 0

buffer times 4096 db 0

key db "12345678901234567890", 0

key\_size dq 0

src\_desc dq 0

dst\_desc dq 0

time dq 0

section .text

global vigenere\_cipher

vigenere\_cipher:

mov [rel key\_size], rdi

mov rcx, rdi

lea rdi, [rel key]

en\_start:

mov bl, [rsi]

mov [rdi], bl

inc rsi

inc rdi

loop en\_start

mov rax, 2

lea rdi, [rel src\_path]

mov rsi, 0 ; O\_RDONLY

mov rdx, 0

syscall

test rax, rax

js file\_error

mov [rel src\_desc], rax

mov rax, 2

lea rdi, [rel dst\_path]

mov rsi, 0x241 ; O\_WRONLY | O\_CREAT | O\_TRUNC (0x1 | 0x40 | 0x200)

mov rdx, 0o644 ;(rw-r--r--)

syscall

test rax, rax

js file\_error

mov [rel dst\_desc], rax

copy\_loop:

mov rax, 0 ; sys\_read

mov rdi, [rel src\_desc]

lea rsi, [rel buffer]

mov rdx, 4096

syscall

test rax, rax

jz close\_files

js file\_error

mov rcx, rax

mov [rel time], rcx

lea rsi, [rel buffer]

xor rdi, rdi

cipher\_loop:

xor rbx, rbx

xor rax, rax

lea rdx, [rel key]

add rdx, rdi

mov bl, byte [rdx]

mov al, [rsi]

add al, bl

and al, 0x7F

mov [rsi], al

inc rsi

inc rdi

cmp rdi, [rel key\_size]

jne .continue

xor rdi, rdi

.continue:

loop cipher\_loop

xor rdi, rdi

en\_end\_loop:

mov rcx, [rel time]

mov rax, 1 ; sys\_write

mov rdi, [rel dst\_desc]

lea rsi, [rel buffer]

mov rdx, rcx

syscall

js file\_error

jmp copy\_loop

close\_files:

mov rax, 3 ; sys\_close

mov rdi, [rel dst\_desc]

syscall

mov rax, 3 ; sys\_close

mov rdi, [rel src\_desc]

syscall

ret

file\_error:

mov rdi, 1

mov rax, 60

syscall

Объектные файлы assembler-функций подключаются в файле CMake.

Листинг кода 4 – Подключение объектных файлов

qt\_add\_executable(7

MANUAL\_FINALIZATION

${PROJECT\_SOURCES}

cf.o

tf.o

rand.o

vf.o

)

# Вывод

В ходе выполнения лабораторной работы были получены навыки работы с системными вызовами ОС Linux с использованием языка assembler; интеграции кода на языках С++ и языке assembler.