**Цель работы:** получить понимание принципов работы DOS-прерываний и методов их использования в программировании на языке C++

**Задание:** Консольный файловый менеджер

Реализовано на стороне assembler:

Вывод информации о текущем пути, каталоге, информации и файлах (размере и дате последнего изменения)

Открытие файлов в сторонних программах или открытие других программ по ярлыку или по самому файлу.

Создание файла в выбранной директории

На стороне C++

Обработка нажатий на клавиши

Меню

Открытие папок

**ТУТ ЧАСТЬ КОДА НА C++, ЧАСТЬ НА NASM, ЧАСТЬ НА MASM!!!!!!!**

**СОБИРАЙТЕ СТАТ БИБИЛИОТЕКЕ И ЛИНКУЙТЕ К С++ И ТОГДА БУДЕТ РАБОТАТЬ**

**Ход работы:**

#include <iostream>  
#include <windows.h>  
#include <filesystem>  
  
namespace fs = std::filesystem;  
  
extern "C" void print\_text(const char \*str);  
extern "C" void create(const char \*filepath);  
extern "C" void open\_file(const char \*filepath);  
  
std::string get\_file\_change\_date(std::string &dirPath) {  
 FILETIME report;  
 WIN32\_FIND\_DATAA findData;  
 HANDLE hFind;  
 SYSTEMTIME sutc, st;  
 hFind = FindFirstFileA(dirPath.c\_str(), &findData);  
 report = findData.ftLastWriteTime;  
 FindClose(hFind);  
 FileTimeToSystemTime(&(report), &sutc);  
 SystemTimeToTzSpecificLocalTime(NULL, &sutc, &st);  
 std::string ans = std::to\_string(st.wDay) + '-' + std::to\_string(st.wMonth) + '-' + std::to\_string(st.wYear) + ' ' +  
 std::to\_string(st.wHour) + ':' + std::to\_string(st.wMinute) + ':' + std::to\_string(st.wSecond);  
 return ans;  
}  
  
int response\_from\_click() {  
 if (GetKeyState(VK\_UP) & 0x8000) {  
 return 18432;  
 }  
 if (GetKeyState(VK\_DOWN) & 0x8000) {  
 return 20480;  
 }  
 if (GetKeyState(VK\_F1) & 0x8000) {  
 return 12654;  
 }  
 if (GetKeyState(VK\_ESCAPE) & 0x8000) {  
 return 283;  
 }  
 if (GetKeyState(VK\_RETURN) & 0x8000) {  
 return 7181;  
 }  
 return 0;  
  
}  
  
  
int wait() {  
 int a;  
 for (int i = 0; i < 8.8e7; i++) {  
 a = i \* i;  
 }  
 return a;  
}  
  
  
int out\_files(const int &cur, std::string &path) {  
 system("cls");  
 int tmp = 0;  
 std::string tmp\_path = path;  
 std::string result\_text;  
 if (path[path.length() - 1] != '\\') tmp\_path += '\\';  
 for (const auto &entry: fs::directory\_iterator(path)) {  
 std::string a = entry.path().string();  
 a = a.substr(tmp\_path.length(), a.length() - path.length());  
 if (tmp == cur) result\_text += char(126);  
 a += " ";  
 tmp++;  
 result\_text += (entry.is\_directory() ? char(212) : char(196));  
 result\_text += " " + a;  
 if (!entry.is\_directory()) {  
 for (int i = 0; i < 40 - a.size() && i >= 0; i++) result\_text += " ";  
 std::string ww = entry.path().string();  
 result\_text += get\_file\_change\_date(ww);  
 result\_text += "\t";  
 auto filesize = entry.file\_size();  
 int type\_of\_size = 0;  
 while (filesize > 1024 && type\_of\_size < 5) {  
 filesize /= 1024;  
 type\_of\_size++;  
 }  
 result\_text += std::to\_string(filesize) + " ";  
 switch (type\_of\_size) {  
 case 0:  
 result\_text += "B";  
 break;  
 case 1:  
 result\_text += "KB";  
 break;  
 case 2:  
 result\_text += "MB";  
 break;  
 case 3:  
 result\_text += "GB";  
 break;  
 default:  
 result\_text += "TB";  
 break;  
 }  
  
 }  
  
 result\_text += "\n";  
  
 }  
 if (tmp == 0) result\_text += "No files | Tap ESC to go back";  
 result\_text +=  
 "\n\nUp\\Down to change selection | Return to Select | ESC to go UP on catalog | F1 - create file\n\nFiles & Folders: " +  
 std::to\_string(tmp) +  
 "\nPath: " + path + "\n";  
 char \*t = const\_cast<char \*>(result\_text.c\_str());  
 print\_text(t);  
 return tmp;  
}  
  
int cnt\_files(std::string &path) {  
 int tmp = 0;  
 std::string tmp\_path = path;  
 if (path[path.length() - 1] != '\\') tmp\_path += '\\';  
 for (const auto &entry: fs::directory\_iterator(path)) {  
 std::string a = entry.path().string();  
 tmp++;  
 }  
 return tmp;  
}  
  
  
void select\_return(int &cur, std::string &path) {  
 int tmp = 0;  
 std::string tmp\_path = path;  
 if (path[path.length() - 1] != '\\') tmp\_path += '\\';  
 for (const auto &entry: fs::directory\_iterator(path)) {  
 std::string a = entry.path().string();  
 if (tmp == cur) {  
 if (entry.is\_directory()) {  
 path = entry.path().string();  
 cur = 0;  
 out\_files(cur, path);  
 return;  
 } else {  
 system("cls");  
 char \*close\_text = "Close program to continue...";  
 print\_text(close\_text);  
 open\_file(entry.path().string().c\_str());  
 return;  
 }  
 }  
 tmp++;  
 }  
}  
  
void select\_esc(int &cur, std::string &path) {  
 int tmp = path.find\_last\_of('\\');  
 path = path.substr(0, tmp);  
 if (path.length() < 3) path += "\\";  
 cur = 0;  
 out\_files(cur, path);  
}  
  
void crete\_file(std::string &path, int cnt) {  
  
 std::string t = path;  
 if (t[t.size() - 1] != '\\') t += '\\';  
 system("cls");  
 std::cout << "Type \"x\" to Exit\n";  
 std::cout << "Create file: " << t;  
 std::string tmp;  
 std::cin >> tmp;  
 if (t == "x") return;  
 t += tmp;  
 std::cout << "\n\n" << t << "\n\n";  
 char \*pat = const\_cast<char \*>(t.c\_str());  
 create(pat);  
 int new\_cnt = cnt\_files(path);  
 if (new\_cnt - cnt == 1) {  
 std::cout << "\nFile created.\n";  
 } else {  
 std::cout << "\nSomething went wrong.\n";  
 }  
 system("pause");  
}  
  
  
int main() {  
 int current\_key = 0;  
 std::string path = "C:\\";  
 int cur = 0;  
 wait();  
 int number\_of\_files = out\_files(cur, path);  
 while (1) {  
 current\_key = response\_from\_click();  
 switch (current\_key) {  
 case 18432: *// up* cur = (number\_of\_files + cur - 1) % number\_of\_files;  
 number\_of\_files = out\_files(cur, path);  
 wait();  
 break;  
 case 20480: *// down* cur = (cur + 1) % number\_of\_files;  
 number\_of\_files = out\_files(cur, path);  
 wait();  
 break;  
 case 12654: *// n* case 12622: *// N*  wait();  
 wait();  
 crete\_file(path, number\_of\_files);  
 number\_of\_files = out\_files(cur, path);  
 wait();  
 break;  
 case 7181: *// return* select\_return(cur, path);  
 number\_of\_files = out\_files(cur, path);  
 wait();  
 break;  
 case 283: *// esc* select\_esc(cur, path);  
 number\_of\_files = out\_files(cur, path);  
 wait();  
 break;  
 default:  
 continue;  
 }  
 }  
  
}

includelib kernel32.lib  
  
extrn CreateFileA: PROC  
  
.code  
create proc  
 sub rsp, 40  
 mov rcx, [cls\_str]  
 mov eax, 0  
 add rsp, 40  
 ret  
create endp  
end

section .data  
 output\_format db "%s", 0  
  
section .text  
 extern printf  
  
 global print\_text, \_start  
  
print\_text:  
 sub rsp, 40  
 mov rdi, output\_format  
 mov rsi, [rdi + 8]  
 mov eax, 0  
 call printf  
 add rsp, 40  
 ret

section .text  
 extern system  
 global open\_file  
  
open\_file:  
 sub rsp, 40  
 mov eax, 0  
 call system  
 add rsp, 40  
 ret

int response\_from\_click() {  
 int a = 0;  
 asm {  
 mov ah, 0h  
 int 16h  
 mov a, ax  
 }  
 return a;  
  
}

**Контрольные вопросы**

1. Чем отличаются far pointer и near pointer в контексте сегментированной модели памяти DOS?

Ответ: far pointer указывает на сегмент:смещение, а near pointer хранит только значение смещения.

1. Как осуществляется передача данных между функциями, написанными на C++ и ассемблере, и какие основные точки следует учитывать для обеспечения корректного взаимодействия

Ответ: На примере Intel x86-64 вызывающий код передает первые четыре параметра в регистры. Для передачи в функцию первых четырех параметров (целочисленных) используются регистры RCX, RDX, R8 и R9 соответственно. Если параметры представляют числа с плавающей точкой, то они передаются через регистры XMM0, XMM1, XMM2 и XMM3.

Параметры начиная с 5-го передаются через стек. Так, 5-й параметр должен занимать в стеке место по адресу RSP+32, 6-й параметр - по адресу RSP+40 и так далее.

Если процедура имеет смешанные параметры - одновременно и целочисленные, и с плавающей запятой, то каждый параметр помещается в соответствующий его номеру регистр RCX/RDX/R8/R9 или XMM0/XMM1/XMM2/XMM3. Например, если у нас есть следующая функция C/C++

Параметры всегда представляют собой 8-байтовые значения.

Указатель стека RSP должен быть выровнен по 16 байтам непосредственно перед тем, как инструкция call поместит адрес возврата в стек.

Если функция возвращает целое число, то оно помещается регистр RAX, если число с плавающей точкой - то в регистр XMM0

**Выводы**

В ходе выполнения лабораторной работы я научился интегрировать код assembler в с++, генерировать статические библиотеке из кода на assembler и подключать к с++, вызывать функции assembler из с++ и функции c/c++ из assembler.