Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ

к лабораторной работе №1

на тему

**УПРАВЛЕНИЕ ПРОЦЕССАМИ, ПОТОКАМИ, НИТЯМИ**

|  |
| --- |
| Выполнил: студент гр. 253503  Кудош А.С. |
| Проверил: ассистент кафедры информатики Гриценко Н.Ю. |

Минск 2024

**СОДЕРЖАНИЕ**

[1 Постановка задачи и программа для её решения 3](#_Toc178870353)

[1.1 Постановка задачи 3](#_Toc178870354)

[1.2 Демонстрация работы программы 3](#_Toc178870355)

[Заключение 6](#_Toc178870356)

[Список использованных источников 7](#_Toc178870357)

[Приложение А (справочное) Исходный код  
 (к пункту 1.2) 8](#_Toc178870358)

**1 ПОСТАНОВКА ЗАДАЧИ И ПРОГРАММА ДЛЯ ЕЁ РЕШЕНИЯ**

**1.1 Постановка задачи**

В данной лабораторной работе необходимо разработать процесс-диспетчер, который будет управлять выполнением и состоянием нескольких процессов. Основной задачей является реализация функциональности выбора и запуска исполняемого файла, а также хранение информации о порожденных процессах, включая их идентификаторы и текущие состояния. Диспетчер должен отображать состояния процессов, различая «выполняется» и «завершился». Исходный код программы находится в приложении А [1].

Также необходимо реализовать возможность отправки сообщения WM\_CLOSE [1] для завершения выбранного процесса [2] и обеспечить отображение возникающих ошибок в ходе работы. Для проверки состояния процессов будет использоваться периодический опрос с применением функций WaitForSingleObject [3], при этом важно, чтобы проверка не блокировала выполнение диспетчера на длительное время. Контролируемые процессы могут быть как произвольными программами, так и специально разработанным оконным приложением, которое наглядно демонстрирует свое выполнение. Целью работы является создание удобного интерфейса для управления процессами в операционной системе.

**1.2 Демонстрация работы программы**

Консольное приложение при запуске предлагает выбрать один из пунктов меню (рисунок 1.1).

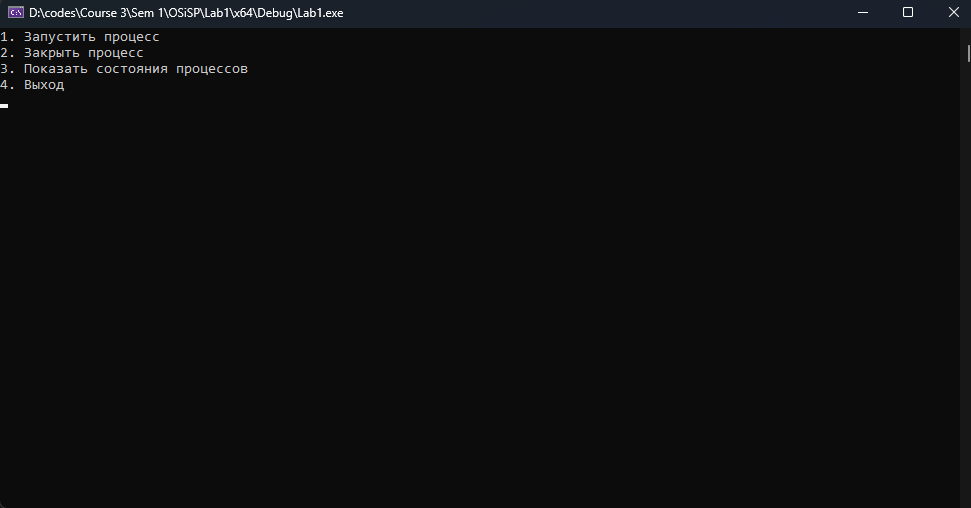


Рисунок 1.1 – Меню консольного приложения

Имеется обработка ошибок, как строкового ввода, так и выхода за пределы пунктов меню (рисунок 1.2).

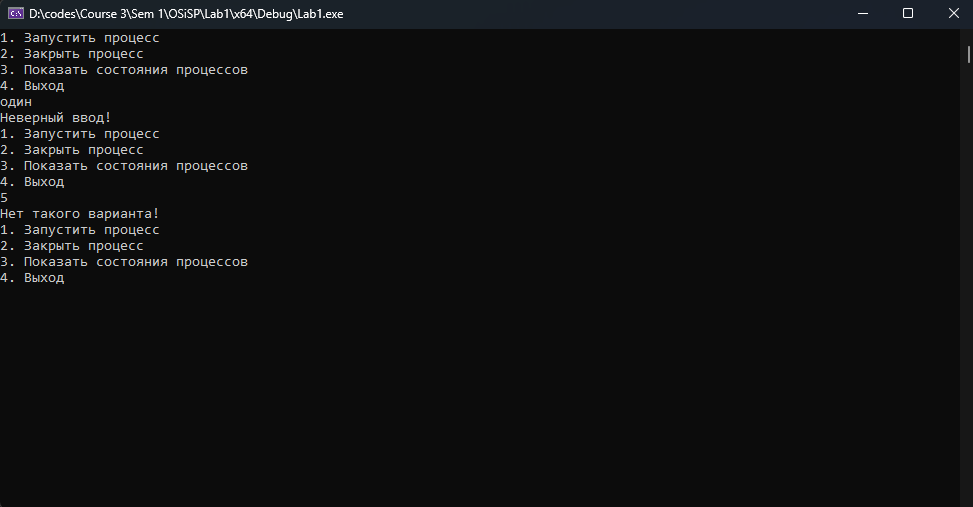


Рисунок 1.2 – Обработка неверного ввода

При выборе пункта запуска процесса, программа предложит ввести путь к исполняемому файлу. Если файл был успешно запущен, приложение выведет сообщение об этом и вернет меню (рисунок 1.3).

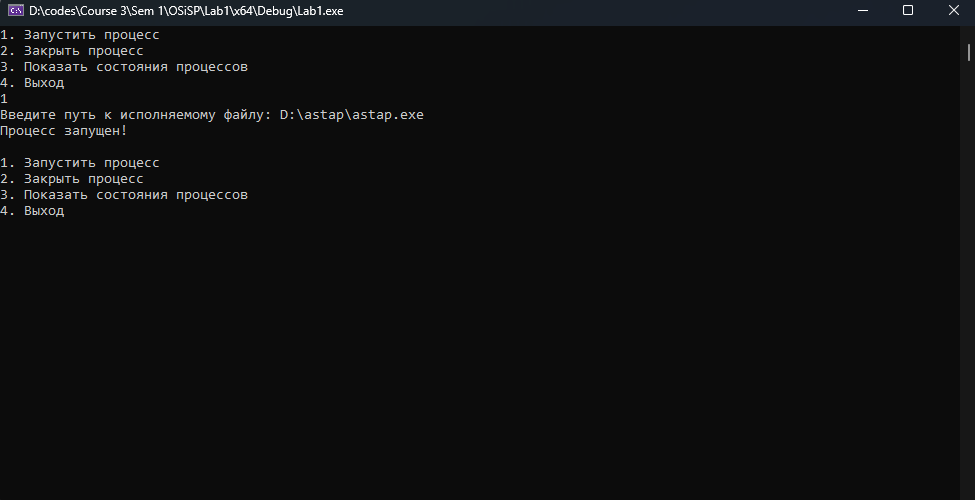


Рисунок 1.3 – Запуск процесса

При выборе пункта просмотра состояний процессов будут выведены все запущенные или завершенные процессы (рисунок 1.4).

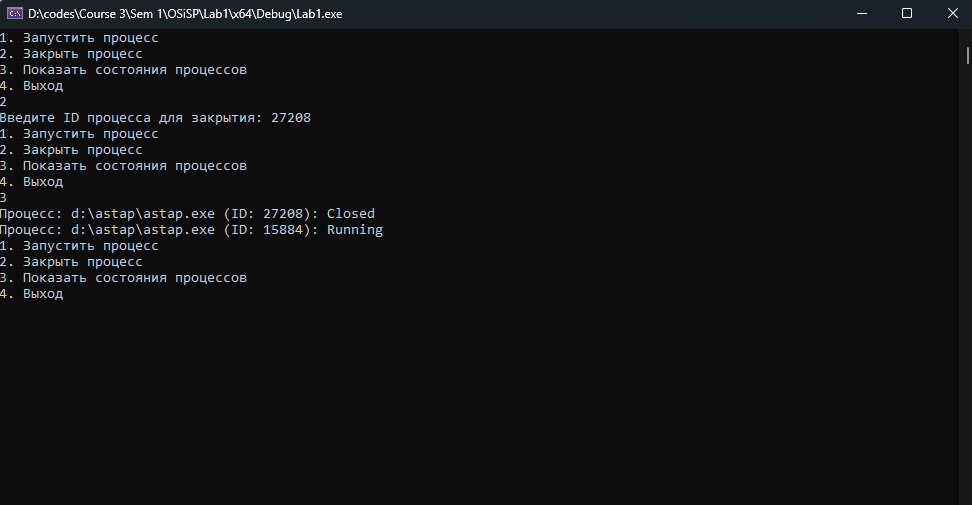


Рисунок 1.4 – Вывод состояния процессов

При выборе пункта выхода в меню, программа завершит своё выполнение (рисунок 1.5).

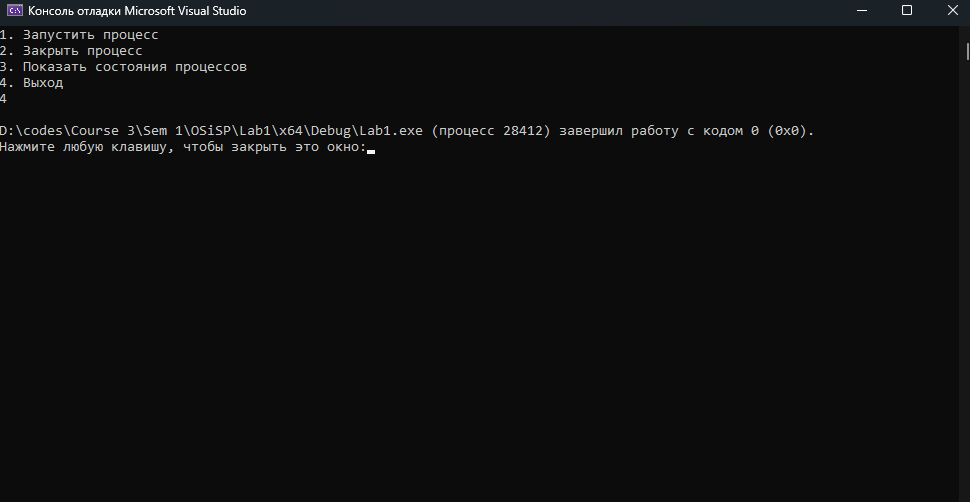


Рисунок 1.5 – Выход из программы

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной лабораторной работы был разработан процесс-диспетчер, способный эффективно управлять выполнением и состоянием нескольких процессов. Основные задачи, поставленные в начале работы, были успешно решены.

Процесс-диспетчер реализует функциональность выбора и запуска исполняемого файла, а также ведет учет информации о порожденных процессах, включая их идентификаторы и текущие состояния. Благодаря различению состояний «выполняется» и «завершился», пользователи могут легко отслеживать активность процессов.

Реализованная возможность отправки сообщения WM\_CLOSE для завершения выбранного процесса обеспечивает гибкость в управлении, а механизм опроса с использованием функций WaitForSingleObject позволяет контролировать состояние процессов без значительных задержек в работе диспетчера. Это особенно важно в условиях, когда требуется быстрое реагирование на изменения в состоянии процессов.

В итоге, созданный интерфейс управления процессами в операционной системе является удобным и функциональным инструментом для пользователей, что соответствует поставленной цели работы. Он предоставляет интуитивно понятный доступ к ключевым функциям управления процессами, что делает его полезным как для опытных пользователей, так и для тех, кто только начинает знакомиться с процессами и их управлением.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] WinAPI: WM\_CLOSE [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/winmsg/wm-close – Дата доступа: 01.10.2024

[2] WinAPI: Потоки в Win32 [Электронный ресурс]. – Режим доступа: https://club.shelek.ru/viewart.php?id=71 – Дата доступа: 01.10.2024

[3] WinAPI: WaitForSingleObject [Электронный ресурс]. – Режим доступа: https://stackoverflow.com/questions/2554447/how-to-use-waitforsingleobject – Дата доступа: 01.10.2024

ПРИЛОЖЕНИЕ А  
(справочное)  
Исходный код

#include <windows.h>

#include <tlhelp32.h>

#include <vector>

#include <string>

#include <iostream>

struct ProcessInfo {

DWORD processID;

std::wstring processName;

PROCESS\_INFORMATION processInfo;

ProcessInfo(DWORD id, const std::wstring& name, const PROCESS\_INFORMATION& info)

: processID(id), processName(name), processInfo(info) {}

};

DWORD targetProcessID;

std::vector<ProcessInfo> processes;

void LaunchProcess(const std::wstring& filePath) {

STARTUPINFO si = { sizeof(si) };

PROCESS\_INFORMATION pi;

if (CreateProcess(filePath.c\_str(), NULL, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi)) {

Sleep(1000);

processes.push\_back({ pi.dwProcessId, filePath, pi });

std::cout << "Процесс запущен!\n\n";

}

else {

MessageBox(NULL, L"Не удалось запустить процесс", L"Ошибка", MB\_OK | MB\_ICONERROR);

}

}

BOOL CALLBACK CloseProcess(HWND hwnd, LPARAM lParam) {

DWORD processID;

GetWindowThreadProcessId(hwnd, &processID);

if (processID == targetProcessID) {

LRESULT res = ::SendMessage(hwnd, WM\_CLOSE, NULL, NULL);

return FALSE;

}

return TRUE;

}

void EnumProcesses() {

for (const auto& proc : processes) {

DWORD waitResult = WaitForSingleObject(proc.processInfo.hProcess, 0);

if (waitResult == WAIT\_TIMEOUT) {

std::wcout << L"Процесс: " << proc.processName

<< L" (ID: " << proc.processID << "): Running" << std::endl;

}

else {

std::wcout << L"Процесс: " << proc.processName

<< L" (ID: " << proc.processID << "): Closed" << std::endl;

}

}

}

int main() {

setlocale(LC\_ALL, "Russian");

while (true) {

std::cout << "1. Запустить процесс\n";

std::cout << "2. Закрыть процесс\n";

std::cout << "3. Показать состояния процессов\n";

std::cout << "4. Выход\n";

int choice;

std::cin >> choice;

if (std::cin.fail()) {

std::cout << "Неверный ввод!\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

else if (choice == 1) {

std::cout << "Введите путь к исполняемому файлу: ";

std::wstring filePath;

std::wcin >> filePath;

LaunchProcess(filePath);

}

else if (choice == 2) {

std::cout << "Введите ID процесса для закрытия: ";

std::cin >> targetProcessID;

EnumWindows(CloseProcess, 0);

}

else if (choice == 3) {

EnumProcesses();

}

else if (choice == 4) {

break;

}

else {

std::cout << "Нет такого варианта!\n";

}

}

for (auto& proc : processes) {

CloseHandle(proc.processInfo.hProcess);

CloseHandle(proc.processInfo.hThread);

}

return 0;

}