Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Закиров Нурислам Дамирович

Содержание

# 1 Цель работы

Цель данной лабораторной работы состоит в том, чтобы приобрести практические навыки работы в Midnight Commander и освоить использование инструкции языка ассемблера mov и int.

# 2 Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной. Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION .bss). Для объявления инициированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны храниться в этой памяти: - DB (define byte) — определяет переменную размером в 1 байт; - DW (define word) — определяет переменную размеров в 2 байта (слово); - DD (define double word) — определяет переменную размером в 4 байта (двойное слово); - DQ (define quad word) — определяет переменную размером в 8 байт (учетве- рённое слово); - DT (define ten bytes) — определяет переменную размером в 10 байт. Директивы используются для объявления простых переменных и для объявления массивов. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике.

mov dst,src

Здесь операнд dst — приёмник, а src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера intпредназначена для вызова прерывания с указанным номером.

int n

Здесь n — номер прерывания, принадлежащий диапазону 0–255. При программировании в Linux с использованием вызовов ядра sys\_calls n=80h (принято задавать в шестнадцатеричной системе счисления).

# 4 Выполнение лабораторной работы

## 4.1 Основы работы с mc

Открываю Midnight Commander, введя в терминал mc (рис. -[1](#fig:001)).

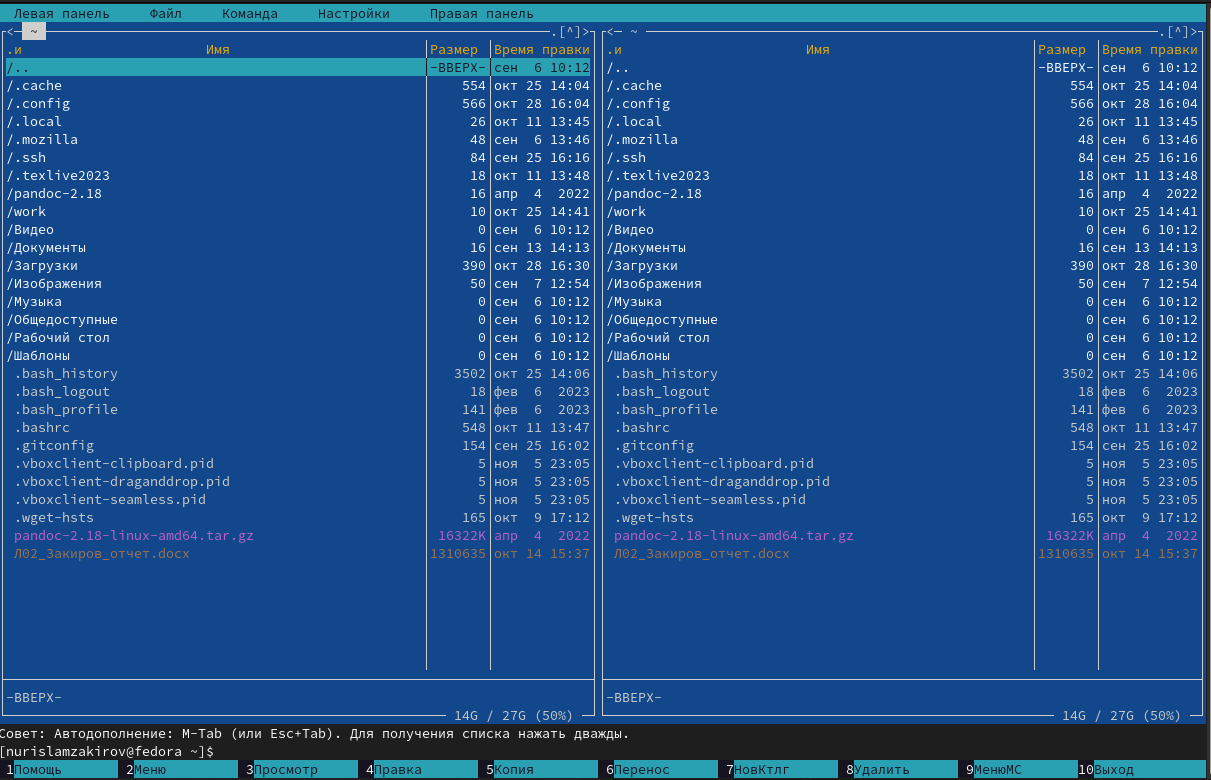


Figure 1: Открытый mc

Используя файловый менеджер mc, перехожу в каталог ~/work/study/2023-2024/Архитектура Компьютера/arch-pc. (рис. -[2](#fig:002))

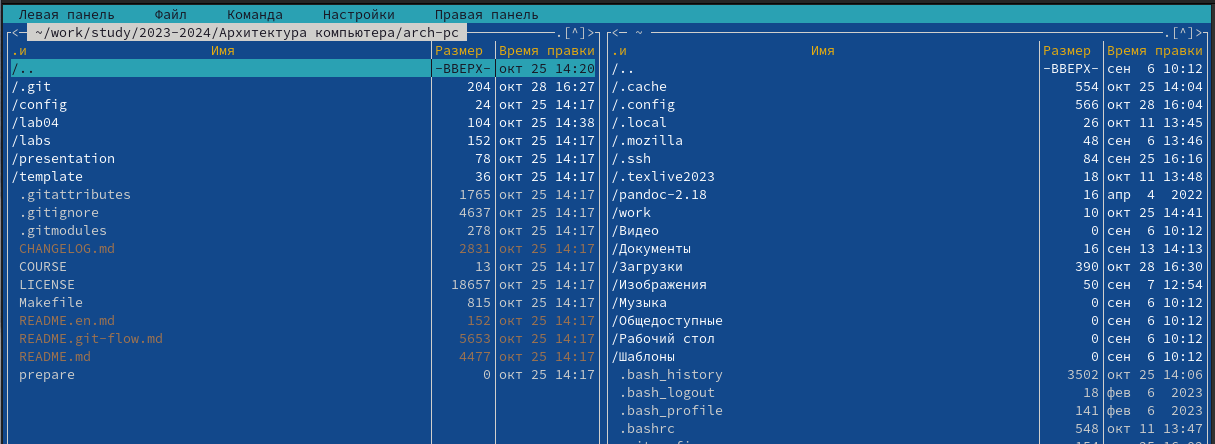


Figure 2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 (рис. -[3](#fig:003)).

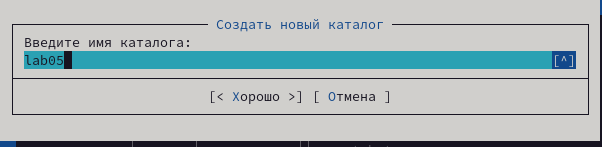


Figure 3: Создание каталога

Перехожу в созданный каталог (рис. -[4](#fig:005)).

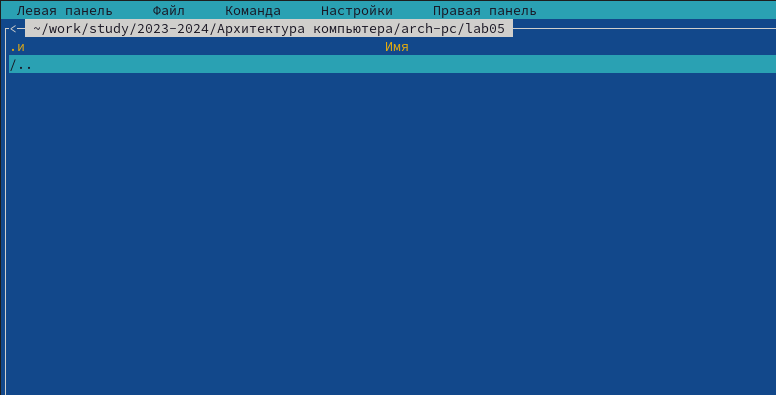


Figure 4: Перемещение между директориями

Я прописываю команду touch lab5-1.asm в строке ввода, чтобы создать файл, в котором буду работать. (рис. -[5](#fig:006)).

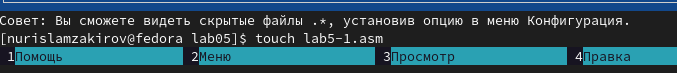


Figure 5: Создание файла

## 4.2 Структура программы на языке ассемблера NASM

Открываю созданный файл для редактирования в редакторе с помощью функциональной клавиши F4. (рис. -[6](#fig:007)).



Figure 6: Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя (рис. -[7](#fig:008)). Далее выхожу из файла, используя комбинацию Ctrl+X, сохраняя изменения, используя комбинацию Y, Enter.

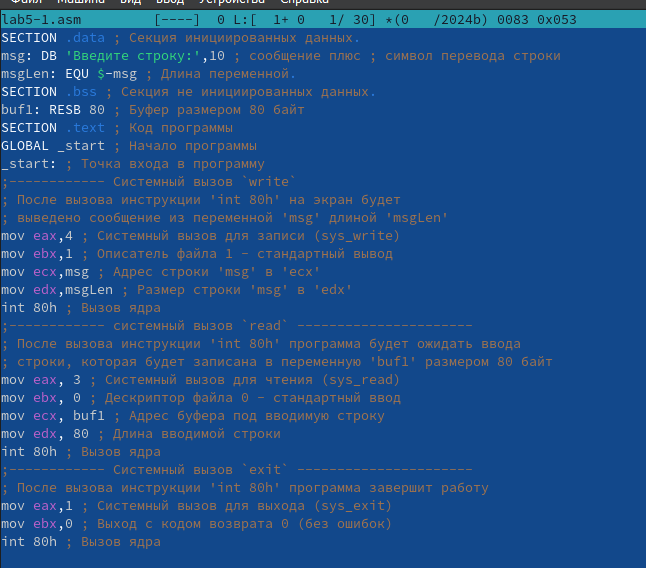


Figure 7: Редактирование файла

Открываю файл с помощью функциональной клавиши F3, чтобы увидеть, содержит ли он текст программы. (рис. -[8](#fig:009)).

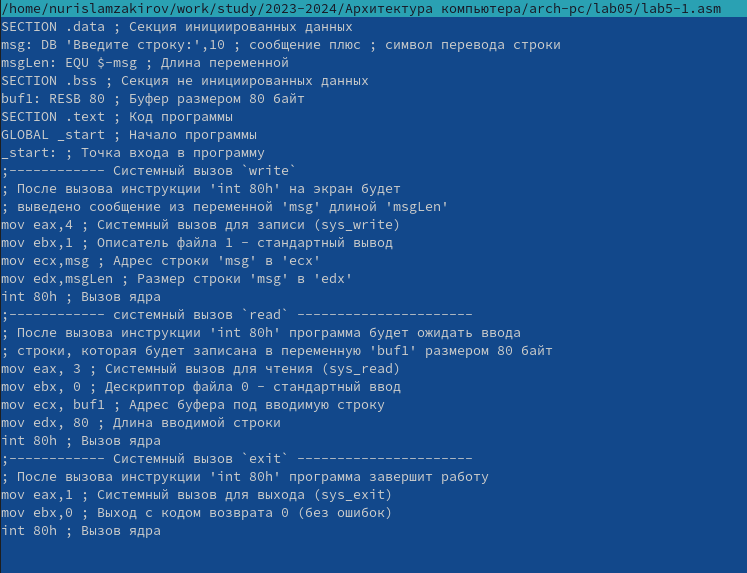


Figure 8: Открытие файла для просмотра

Командой nasm -f elf lab5-1.asm транслирую текст программы файла в объектный файл.Я использую команду ld -m elf\_i386 -o lab5-1 lab5-1.o, чтобы выполнить компоновку объектного файла. (рис. -[9](#fig:010)). Создался исполняемый файл lab5-1.

Figure 9: Компиляция файла и передача на обработку компоновщику

Figure 9: Компиляция файла и передача на обработку компоновщику

Программа выдает строку «Введите строку» и ожидает ввода с клавиатуры; я ввожу свои ФИО, и программа завершает свою работу. (рис. -[10](#fig:011)).

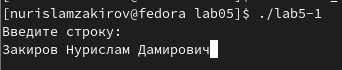


Figure 10: Исполнение файла

## 4.3 Подключение внешнего файла

Скачиваю файл in\_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. -[11](#fig:012)).

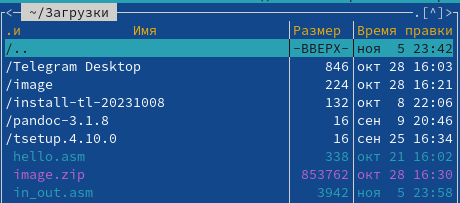


Figure 11: Скачанный файл

Копирую файл in\_out.asm из каталога Загрузки в созданный каталог lab05 с помощью функциональной клавиши F5. (рис. -[12](#fig:013)).

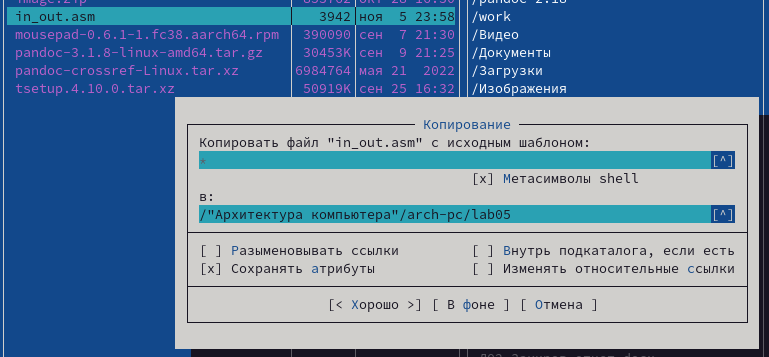


Figure 12: Копирование файла

Копирую файл lab5-1 в тот же каталог под другим именем с помощью функции F5. Это проделываю в окне mc. (рис. -[13](#fig:014)).

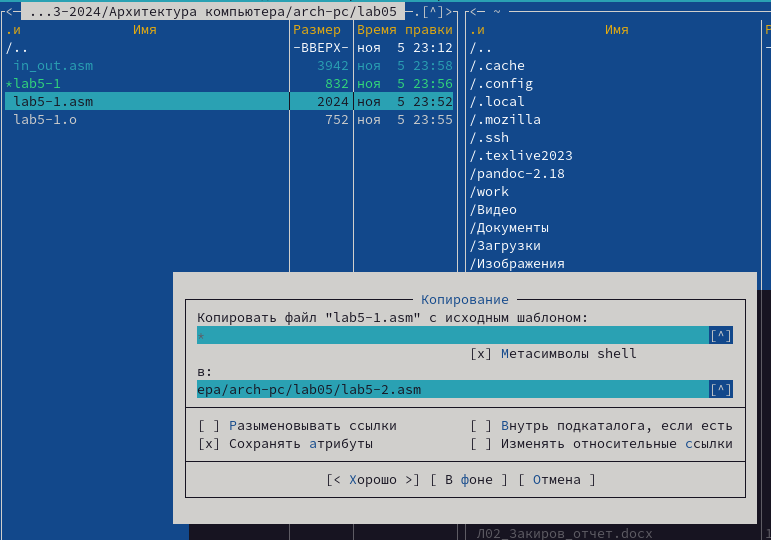


Figure 13: Копирование файла

Меняю содержимое файла lab5-2.asm во встроенном редакторе nano (рис. -[14](#fig:015)), чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm.

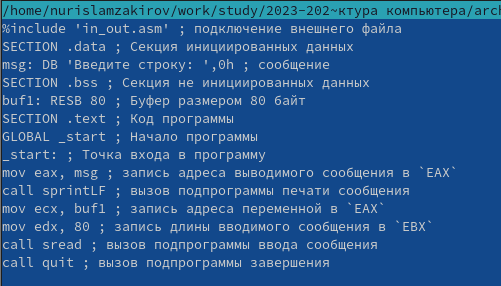


Figure 14: Редактирование файла

Трaнслирую текст программы файла в объектный файл командой nasm -f elf lab5-2.asm. Я использую команду ld -m elf\_i386 -o lab5-2 lab5-2, чтобы выполнить компоновку объектного файла.Файл lab5-2 для выполнения. Запускаю исполняемый файл. (рис. -[15](#fig:016)).

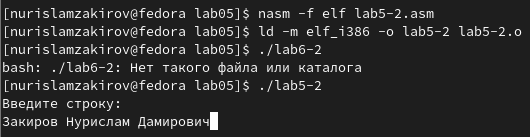


Figure 15: Исполнение файла

Открываю файл lab5-2.asm, чтобы изменить его на nano функциональной клавише F4. В нем изменил подпрограмму sprint LF на sprint. Я сохраняю изменения и открываю файл для проверки сохранения. (рис. -[16](#fig:017)).

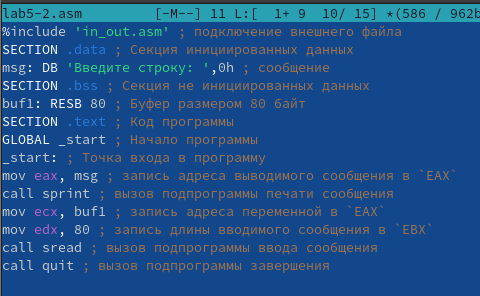


Figure 16: Отредактированный файл

Я сначала транслирую файл, затем выполняю компоновку объектного файла, а затем запускаю новый исполняемый файл. (рис. -[17](#fig:018)).

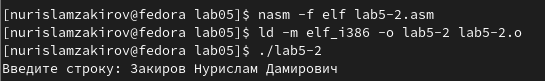


Figure 17: Исполнение файла

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

## 4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. -[18](#fig:019)).

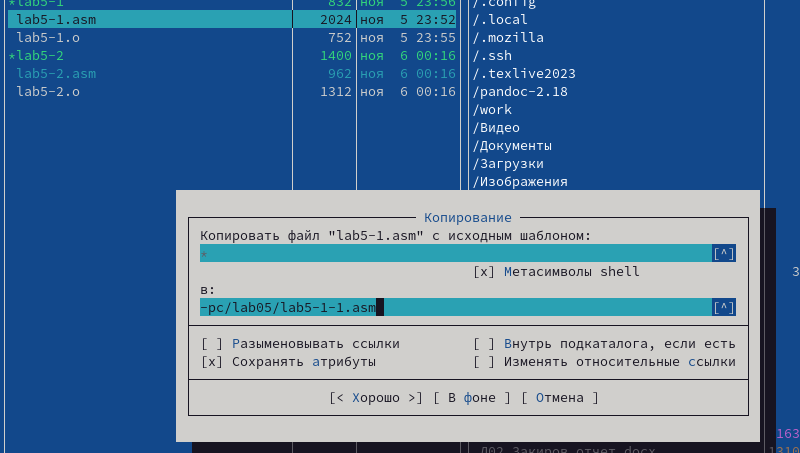


Figure 18: Копирование файла

Открываю созданный файл для редактирования с помощью функционной клавиши F4. Я изменил программу, чтобы она выдавала строку, вводимую пользователем, помимо приглашения и запроса ввода. (рис. -[19](#fig:020)).

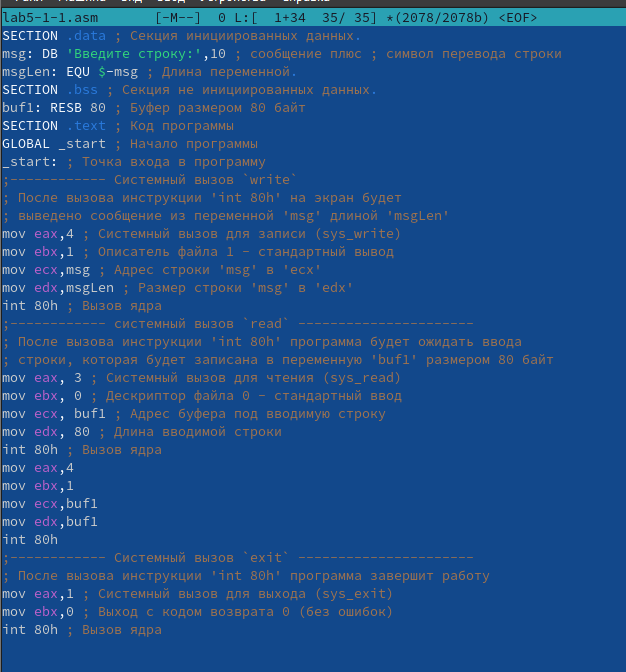


Figure 19: Редактирование файла

1. Я создаю проектный файл lab5-1-1.o, передаю его компоновщику для обработки, получаю исполняемый файл lab5-1-1 и запускаю полученный исполняемый файл. Программа запрашивает ввод данных, а затем выводит мои данные. (рис. -[20](#fig:021)).

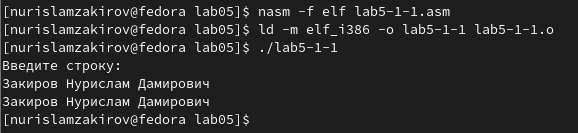


Figure 20: Исполнение файла

Код программы из пункта 1:

SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку:',10  
msgLen: EQU $-msg ; Длина переменной 'msg'  
SECTION .bss ; Секция не инициированных данных  
buf1: RESB 80 ; Буфер размером 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
mov eax,4 ; Системный вызов для записи (sys\_write)  
mov ebx,1 ; Описатель файла 1 - стандартный вывод  
mov ecx,msg ; Адрес строки 'msg' в 'ecx'  
mov edx,msgLen ; Размер строки 'msg' в 'edx'  
int 80h ; Вызов ядра  
mov eax, 3 ; Системный вызов для чтения (sys\_read)  
mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод  
mov ecx, buf1 ; Адрес буфера под вводимую строку  
mov edx, 80 ; Длина вводимой строки  
int 80h ; Вызов ядра  
mov eax,4 ;  
mov ebx,1 ;  
mov ecx,buf1 ;  
mov edx,buf1 ;  
int 80h ;  
mov eax,1 ; Системный вызов для выхода (sys\_exit)  
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)  
int 80h ; Вызов ядра

1. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. -[21](#fig:022)).

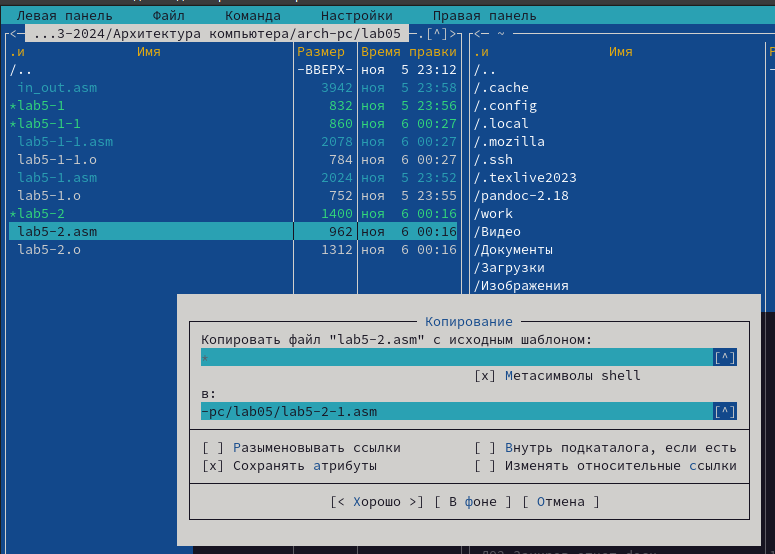


Figure 21: Копирование файла

Открываю созданный файл для редактирования с помощью функционной клавиши F4. Я изменил программу, чтобы она выдавала строку, вводимую пользователем, помимо приглашения и запроса ввода. (рис. -[22](#fig:023)).

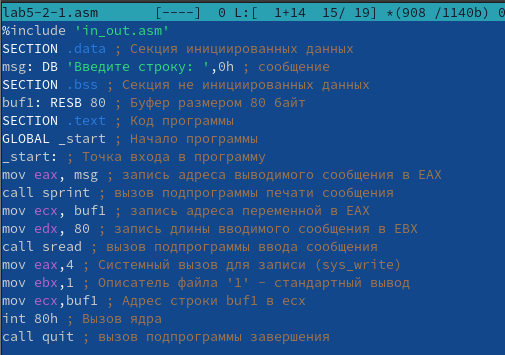


Figure 22: Редактирование файла

1. Я создаю проектный файл lab5-2-1, отдаю его компоновщику, получаю исполняемый файл lab5-2-1, а затем запускаю полученный файл. Программа запрашивает ввод без переноса данных на новую строку. Затем она вводит мои ФИО и выводит введенные мною данные. (рис. -[23](#fig:024)).

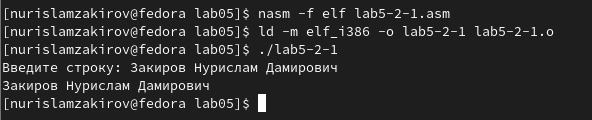


Figure 23: Исполнение файла

Код программы из пункта 3:

%include 'in\_out.asm'  
SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку: ',0h ; сообщение  
SECTION .bss ; Секция не инициированных данных  
buf1: RESB 80 ; Буфер размером 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
mov eax, msg ; запись адреса выводимого сообщения в `EAX`  
call sprint ; вызов подпрограммы печати сообщения  
mov ecx, buf1 ; запись адреса переменной в `EAX`  
mov edx, 80 ; запись длины вводимого сообщения в `EBX`  
call sread ; вызов подпрограммы ввода сообщения  
mov eax,4 ; Системный вызов для записи (sys\_write)  
mov ebx,1 ; Описатель файла '1' - стандартный вывод  
mov ecx,buf1 ; Адрес строки buf1 в ecx  
int 80h ; Вызов ядра  
call quit ; вызов подпрограммы завершения

# 5 Выводы

При выполнении данной лабораторной работы я приобрел практические навыки работы в Midnight Commander и освоил инструкции языка ассемблера mov и int.

# 6 Список литературы

1. [Лабораторная работа №5](https://esystem.rudn.ru/pluginfile.php/2089085/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%965.%20%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B%20%D1%81%20Midnight%20Commander%20%28%29.%20%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D1%8B%20%D0%B2%20%D0%9E%D0%A1%20GNU%20Linux.pdf)