Лабораторная работа №5

Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux

Анастасия Романовна Сырцева

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целю работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера mov и int.

2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

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. В Midnight Commander используются функциональные клавиши F1 — F10, к которым привязаны часто выполняемые операции.

Функциональные клавиши	Выполняемое действие
F1	Вызов контекстно-зависимой подсказки
F2	Вызов меню, созданного пользователем
F3	Просмотр файла, на который указывает подсветка в активной панели
F4	Вызов встроенного редактора для файла, на который указывает подсветка в активной панели
F5	Копирование файла или группы отмеченных файлов из каталога, отображаемого в активной панели, в каталог, отображаемый на второй панели
F6	Перенос файла или группы отмеченных файлов из каталога, отображаемого

Функциональные клавиши	Выполняемое действие
в активной панели, в каталог, отображаемый на второй панели	
F7	Создание подкаталога в каталоге, отображаемом в активной панели
F8	Удаление файла (подкаталога) или группы отмеченных файлов
F9	Вызов основного меню программы
F10	Выход из программы

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 (учетверённое 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. Для объявления неинициированных данных в секции .bss используются директивы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо зарезервировать заданное количество ячеек памяти.

4 Выполнение лабораторной работы

Открываю Midnight Commander с помощью команды mc (рис. 4.1), (рис. 4.2).

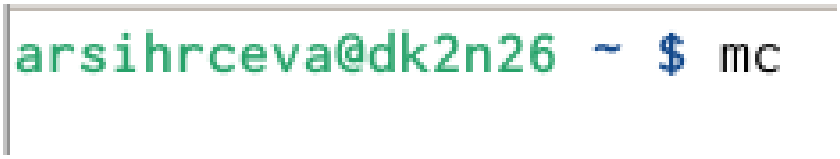


Рис. 4.1: Команда для открытия Midnight Commander

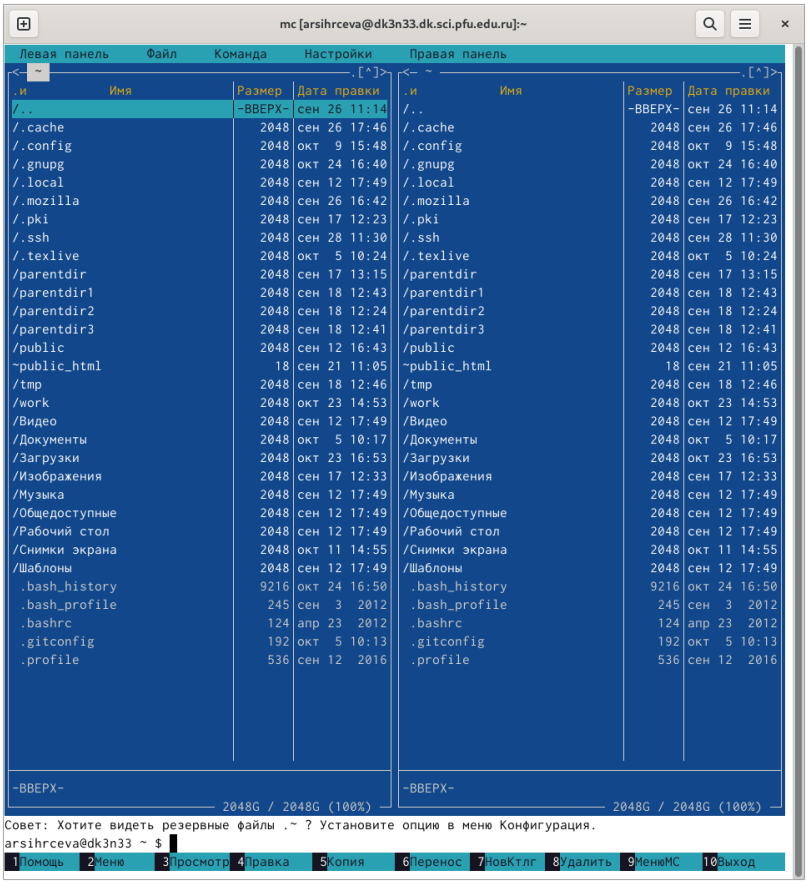


Рис. 4.2: Открытое окно Midnight Commander

Перехожу в каталог ~/work/arch-pc, созданный в 4 лабораторной работе (рис. 4.3).

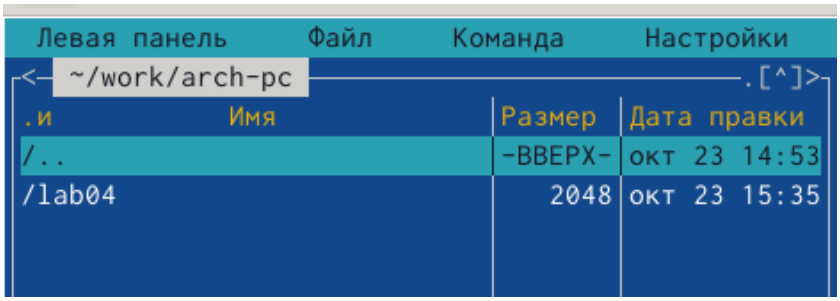


Рис. 4.3: Каталог ~/work/arch-pc

Создаю каталог lab05 и перехожу в него (рис. 4.4).

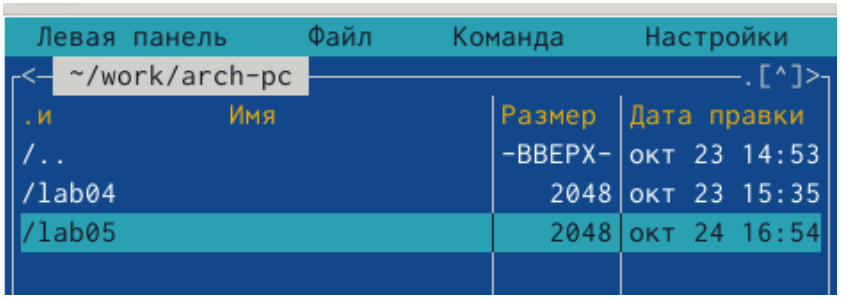


Рис. 4.4: Каталог lab05

Создаю файл lab5-1.asm (рис. 4.5).

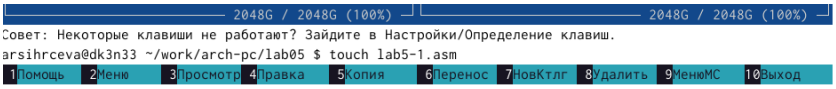
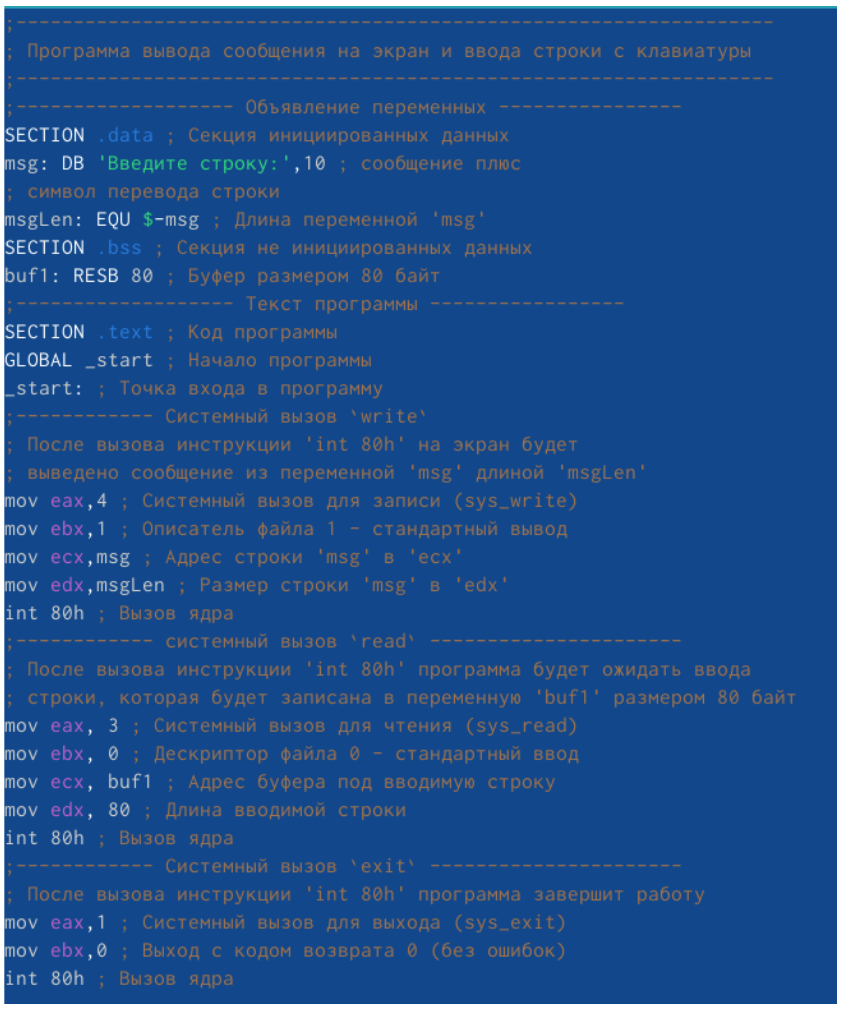


Рис. 4.5: Код в строке ввода для создания файла

Открываю файл lab5-1.asm для редактирования и ввожу текст программы, указанный в условии лабораторной работы (рис. 4.6).



```
;-----------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----------------------------------------------------------------
;------------------- Объявление переменных ----------------
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;------------------- Текст программы -----------------
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
;------------ Системный вызов `write`
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
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
;------------ системный вызов `read` ----------------------
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
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
;------------ Системный вызов `exit` ----------------------
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
```

Рис. 4.6: Текст программы во встроенном редакторе

Закрываю файл и открываю его для просмотра, чтобы убедиться в наличии текста программы в нём (рис. 4.7).

```
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
52 Демидова А. В.
Архитектура ЭВМ
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h
mov eax,1
mov ebx,0
int 80h
```

Рис. 4.7: Файл открытый для просмотра

Транслирую текст программы lab5-1.asm в объектный файл,убеждаюсь в правильности выполнения команды и выполняю компановку. 4.8).

```
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ ls
lab5-1.asm  lab5-1.o
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 4.8: Трансляция программы и компановка

Запускаю получившийся исполняемый файл. На запрос команды ввожу своё ФИО. (рис. 4.9).

```
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Сырцева Анастасия Романовна
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $
```

Рис. 4.9: Запуск файла

Скачиваю файл in_out.asm из ТУИС(рис. **??**).

Рис. 4.10: Скачивание файла

Открываю в двух панелях каталог с файлом lab5-1.asm и католог со скачанным файлом in_out.asm (рис. 4.11).

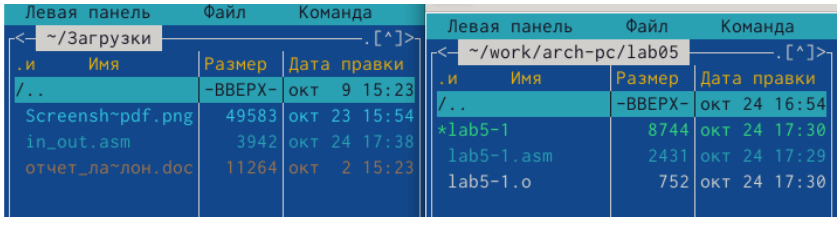


Рис. 4.11: Панели с каталогами

Копирую файл in_out.asm с помощью функциональной клавиши F5 (рис. 4.12).

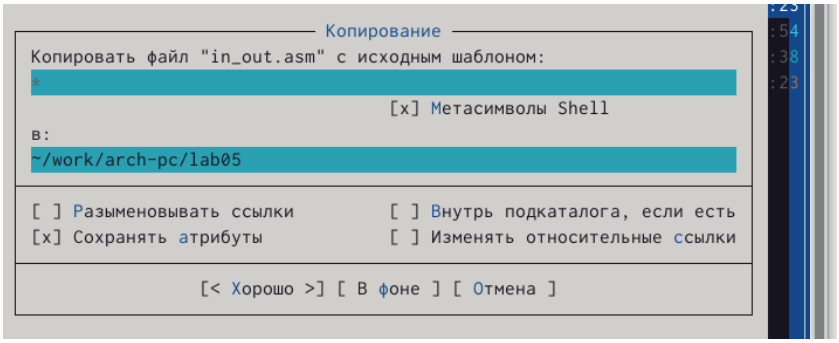


Рис. 4.12: Копирование файла

Убеждаюсь в правильности выполненных действий(рис. 4.13).

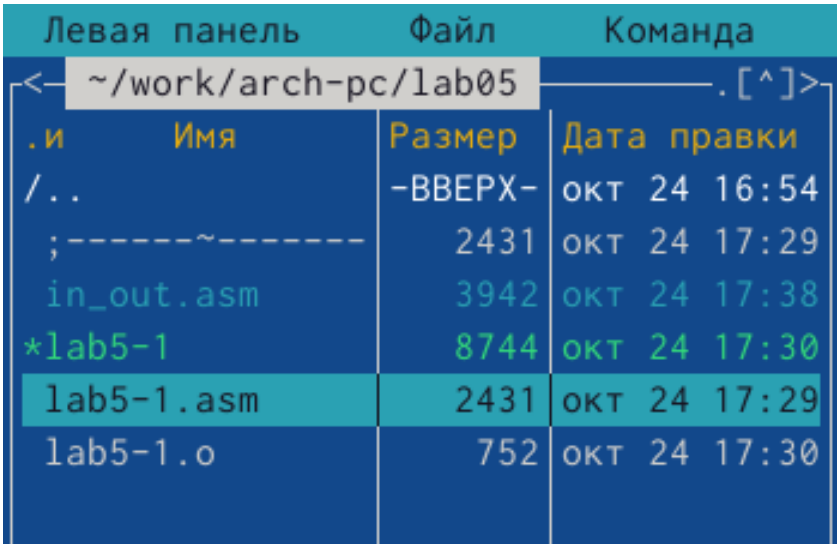


Рис. 4.13: Каталог с файлами lab5-1.asm и in_out.asm

Копирую файл lab5-1.asm изменив имя на lab5-2.asm (рис. 4.14).

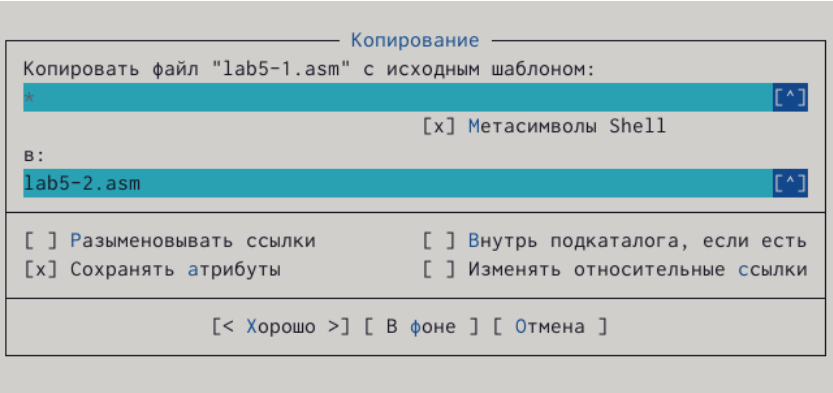


Рис. 4.14: Копирование файла с изменением имени

Меняю текст программы в файле lab5-1.asm, используя подпрограммы из внешнего файла in_out.asm(рис. 4.15).

```
;------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;------------------------------------------------------------
;------------------- Объявление переменных ----------------
%include 'in_out.asm' ;подключение внешнего файла

SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',0h ; сообщение.

SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;------------------- Текст программы -----------------
SECTION .text ; Код программы
    GLOBAL _start ; Начало программы
    _start: ; Точка входа в программу
;------------ Системный вызов `write`
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
    mov eax,msg
    call sprintLF
....
    mov ecx, buf1
    mov edx, 80
    call spread
....
    call quit
```

Рис. 4.15: Изменённый текст программы

Транслирую объектный файл, компилирую его и запускаю исполняемый файл. На запрос команды ввожу своё ФИО (рис. 4.16).

```
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Syrtseva Anastasia Romanovna
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $
```

Рис. 4.16: Исполнение программы

В данном файле заменяю в тексте программы sprintLF на sprint. Выполняю нужные действия для запуска программы (рис. 4.17).

```
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:Syrtseva Anastasia Romanovna
arsihrceva@dk3n33 ~/work/arch-pc/lab05 $
```

Рис. 4.17: Исполнение изменённой программы

В первом случае ввод ответа происходит на следующей после запроса строки, во втором случае - на той же строке, что и запрос.

#Самостоятельная работа

Копирую файл lab5-2.asm (рис. 4.18).

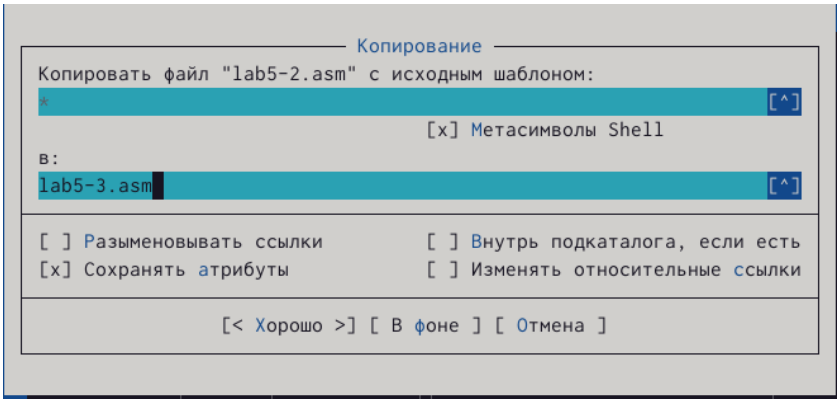


Рис. 4.18: Копирование файла

Изменяю текст программы в скопированном файле так, чтобы послле ввода ответа на запрос выводился введённый ответ(рис. 4.19).

```
;-----------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----------------------------------------------------------------
;------------------ Объявление переменных ----------------
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;------------------ Текст программы -----------------
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
;------------ Системный вызов `write`
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
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
;------------ системный вызов `read` ----------------------
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
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
;------------ Системный вызов `exit` ----------------------
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
```

Рис. 4.19: Изменённый код

Транслирую программу в объектный файл, компилирую и запускаю. Ввожу свою фамилию на запрос команды. (рис. 4.20).

```
arsihrceva@dk2n26 ~/work/arch-pc/lab05 $ nasm -f elf lab5-3.asm
arsihrceva@dk2n26 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-3 lab5-3.o
arsihrceva@dk2n26 ~/work/arch-pc/lab05 $ ./lab5-3
Введите строку:
Сырцева
Сырцева
```

Рис. 4.20: Исполнение команды

Программа работает верно и после моего ответа ввыдится введённая фамилия. Копирую файл lab5-1.asm (рис. 4.21).

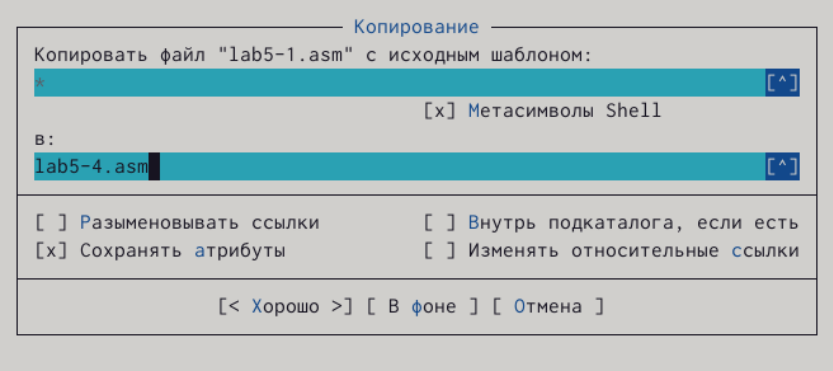


Рис. 4.21: Копирование файла

Открываю скопированный файл и изменяю его код для получения того же результата, что и на прошлом шаге (рис. 4.22).

```
;-------------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-------------------------------------------------------------------
%include 'in_out.asm' ; подключение внешнего файла
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
mov ebx, 1
mov ecx, buf1
int 80h
call quit ; вызов подпрограммы завершения
```

Рис. 4.22: Изменённый код

Ввожу необходимые для запуска программы команды. Ввожу свою фамилию на запрос. (рис. 4.23).

```
arsihrceva@dk2n26 ~/work/arch-pc/lab05 $ nasm -f elf lab5-4.asm
arsihrceva@dk2n26 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-4 lab5-4.o
arsihrceva@dk2n26 ~/work/arch-pc/lab05 $ ./lab5-4
Введите строку: Сырцева
Сырцева
```

Рис. 4.23: Исполнение команды

Код работает верно. После ввода ответа выводится введённая строка.

5 Выводы

Получены навыки работы в Midnight Commander и освоены инструкции языка ассемблера mov и int.