

# **Лабораторная работа №5**

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

Богданюк Анна Васильевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	21
	Список литературы	22

## **Список таблиц**

## Список иллюстраций

4.1	Открываю Midnight Commander . . . . .	9
4.2	Midnight Commander . . . . .	9
4.3	Перекожу в каталог . . . . .	10
4.4	Создаю lab05 . . . . .	10
4.5	Перехожу в lab05 . . . . .	10
4.6	Создание файла . . . . .	11
4.7	Редактирую файл . . . . .	11
4.8	Ввод текста . . . . .	11
4.9	Открываю файл для просмотра . . . . .	13
4.10	Оттранслирую текст программы в объектный файл . . . . .	13
4.11	Выполнение компоновки . . . . .	13
4.12	Запуск файла . . . . .	14
4.13	Файл in_out.asm . . . . .	14
4.14	Открываю каталоги в двух панелях . . . . .	14
4.15	Копирую файл . . . . .	15
4.16	Копирую файл . . . . .	15
4.17	Исправляю файл . . . . .	15
4.18	Создание и вывод . . . . .	16
4.19	Копирую файл . . . . .	17
4.20	Изменение файла . . . . .	17
4.21	Вывод . . . . .	19
4.22	Копирование файла . . . . .	19
4.23	Изменение файла . . . . .	19
4.24	Вывод . . . . .	20

# 1 Цель работы

Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`.

## **2 Задание**

1. Выполнение лабораторной работы
2. Задания 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).

Простейший диалог с пользователем требует наличия двух функций — вывода текста на экран и ввода текста с клавиатуры. Простейший способ вывести строку на экран — использовать системный вызов `write`. Этот системный вызов имеет номер 4, поэтому перед вызовом инструкции `int` необходимо поместить значение 4 в регистр `eax`. Первым аргументом `write`, помещаемым в регистр `ebx`, задаётся дескриптор файла. Для вывода на экран в качестве дескриптора файла нужно указать 1 (это означает «стандартный вывод», т. е. вывод на экран). Вторым аргументом задаётся адрес выводимой строки (помещаем его в регистр `ecx`, например, инструкцией `mov ecx, msg`). Строка может иметь любую длину. Последним аргументом (т.е. в регистре `edx`) должна задаваться максимальная длина выводимой строки. Для ввода строки с клавиатуры можно использовать аналогичный системный вызов `read`. Его аргументы — такие же, как у вызова `write`, только для «чтения» с клавиатуры используется файловый дескриптор 0

(стандартный ввод). Системный вызов `exit` является обязательным в конце любой программы на языке ассемблер. Для обозначения конца программы перед вызовом инструкции `int 80h` необходимо поместить в регистр `eax` значение 1, а в регистр `ebx` код завершения 0.



## 4 Выполнение лабораторной работы

### 1. Выполнение лабораторной работы

Открываю Midnight Commander с помощью mc (рис. 4.1).



Рис. 4.1: Открываю Midnight Commander

Midnight Commander (рис. 4.2).

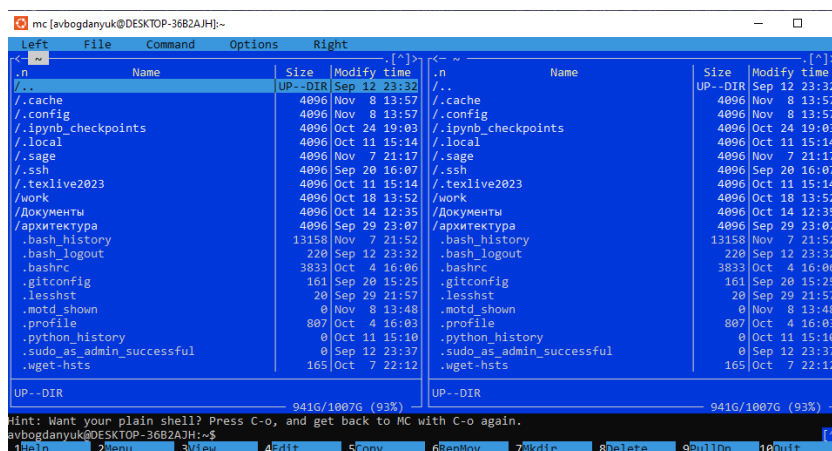


Рис. 4.2: Midnight Commander

Перехожу в каталог, созданный в ходе лабораторной работы №4 (рис. 4.3).

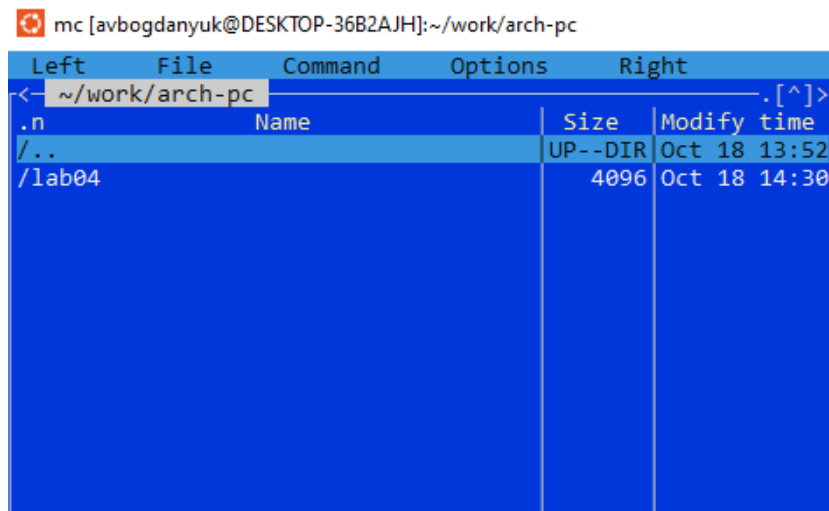


Рис. 4.3: Перекожу в каталог

Создаю новый каталог lab05 (рис. 4.4).

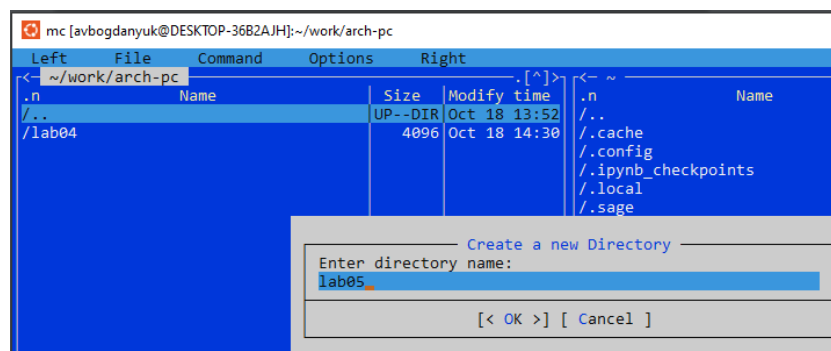


Рис. 4.4: Создаю lab05

Перехожу в новый каталог(рис. 4.5).

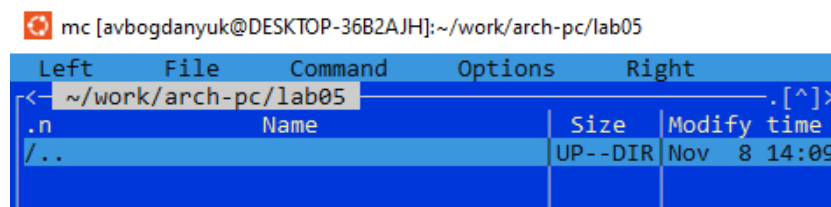


Рис. 4.5: Перехожу в lab05

С помощью touch создаю новый файл lab5-1.asm (рис. 4.6).

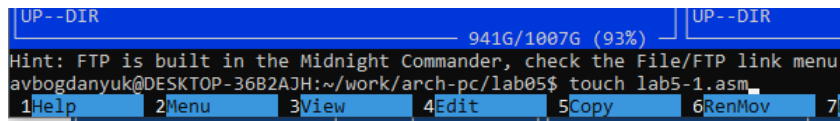


Рис. 4.6: Создание файла

Редактирую созданный файл (рис. 4.7).

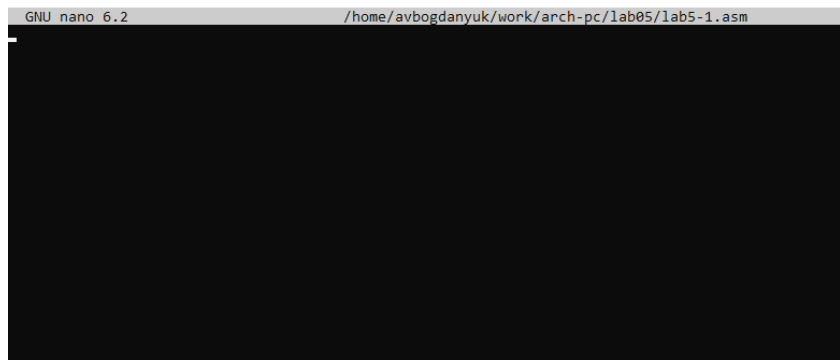


Рис. 4.7: Редактирую файл

Ввожу текст из листинга, сохраняю файл и закрываю его (рис. 4.8).

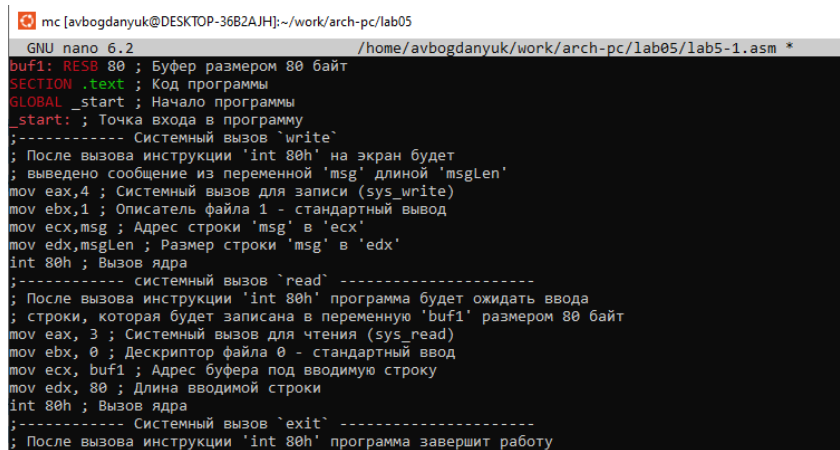


Рис. 4.8: Ввод текста

Листинг программы:

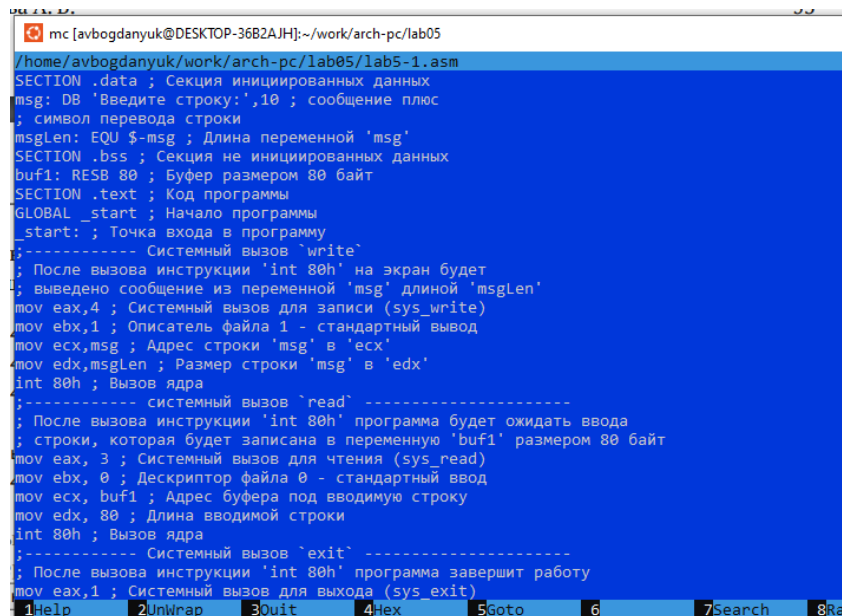
```

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
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"`

С помощью функциональной клавиши F3 открываю файл для просмотра. Убеждаюсь, что файл содержит текст программы (рис. 4.9).



```
mc [avbogdanyuk@DESKTOP-36B2AJH]:~/work/arch-pc/lab05
/home/avbogdanyuk/work/arch-pc/lab05/lab5-1.asm
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msglen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
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 ;Descriptor 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)
```

Рис. 4.9: Открываю файл для просмотра

Оттранслирую текст программы lab5-1.asm в объектный файл (рис. 4.10).

```
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
```

Рис. 4.10: Оттранслирую текст программы в объектный файл

Выполняю компоновку объектного файла (рис. 4.11).

```
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 4.11: Выполнение компоновки

Запускаю получившийся файл и ввожу своё ФИО (рис. 4.12).

```
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Богданюк Анна Васильевна
```

Рис. 4.12: Запуск файла

Скачиваю файл in\_out.asm (рис. 4.13).

in\_out.asm 11/8/2023 3:45 PM Assembler Source

Рис. 4.13: Файл in\_out.asm

Так как файл in\_out.asm должен находиться в одном каталоге с файлом с программой, в которой он используется. Открываю в одной панели mc каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in\_out.asm. (рис. 4.14).

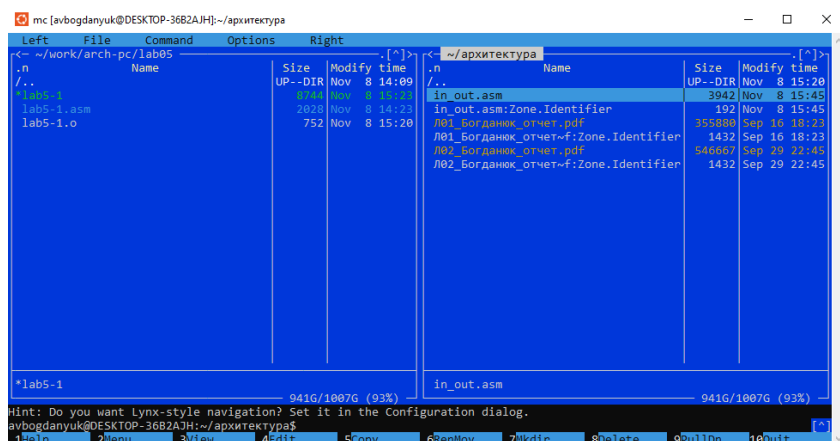


Рис. 4.14: Открываю каталоги в двух панелях

Копирую файл in\_out.asm в каталог с файлом lab5-1.asm (рис. 4.15).

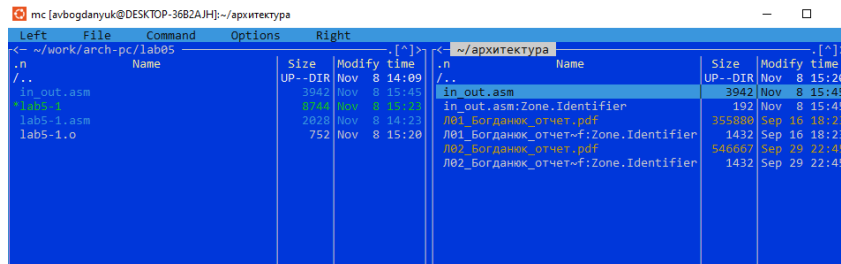


Рис. 4.15: Копирую файл

Копирую файл lab5-1.asm с именем lab5-2.asm (рис. 4.16).

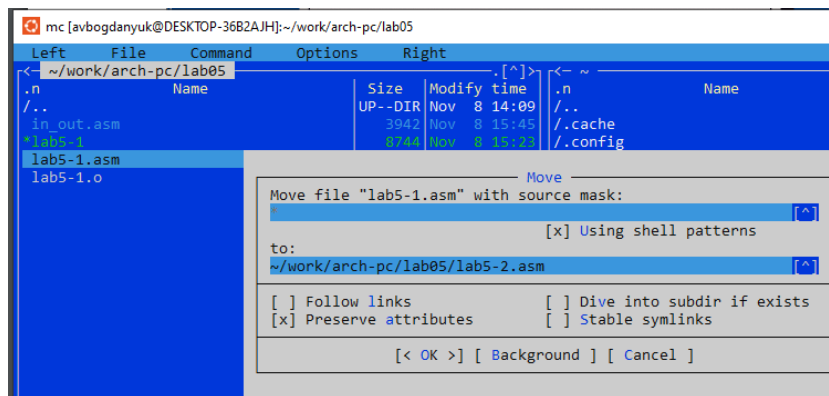


Рис. 4.16: Копирую файл

Исправляю текст программы lab5-2.asm, где меняю sprintf на printf (рис. 4.17).

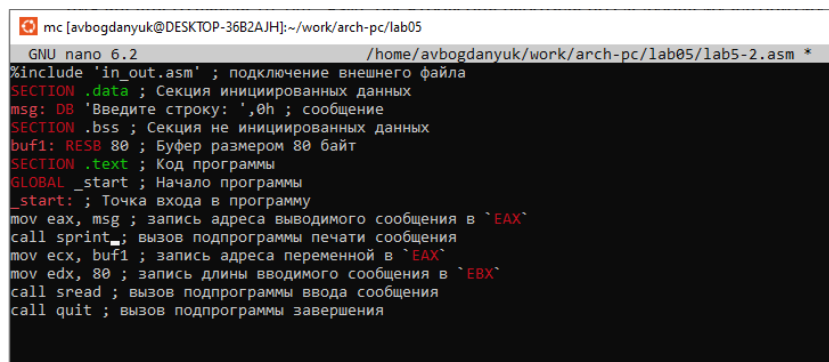


Рис. 4.17: Исправляю файл

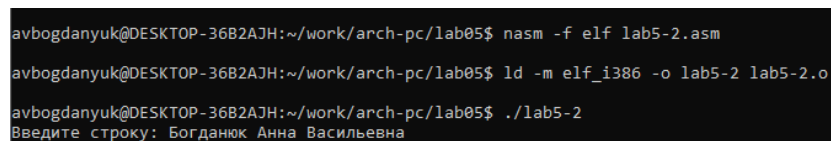
Листинг программы:

```

"%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
call quit ; вызов подпрограммы завершения"

```

Создаю исполняемый файл для lab5-2.asm и проверю его работу. Разница: sprintLF – работает аналогично sprint, но при выводе на экран добавляет к сообщению символ перевода строки (рис. 4.18).



```

avbogdanyuk@DESKTOP-36B2A3H:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
avbogdanyuk@DESKTOP-36B2A3H:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
avbogdanyuk@DESKTOP-36B2A3H:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Богданюк Анна Васильевна

```

Рис. 4.18: Создание и вывод

## 2. Задания для самостоятельной работы

Создаю копию файла lab5-1.asm с названием lab5-3.asm (рис. 4.19).



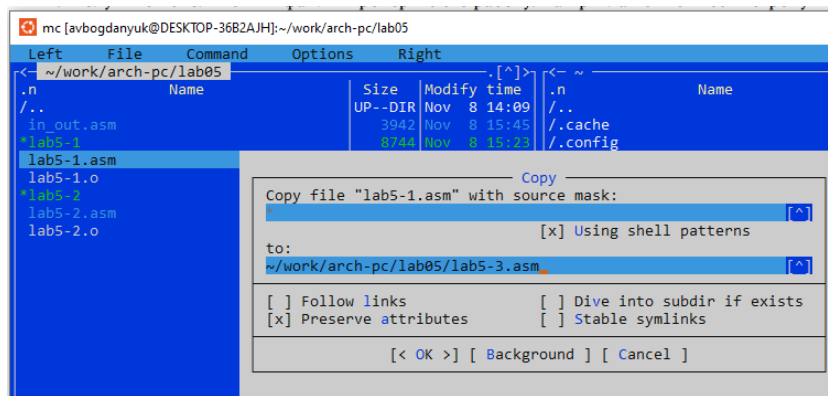


Рис. 4.19: Копирую файл

Изменяю файл, чтобы был вывод строки, ввод пользователем и вывод строки (рис. 4.20).

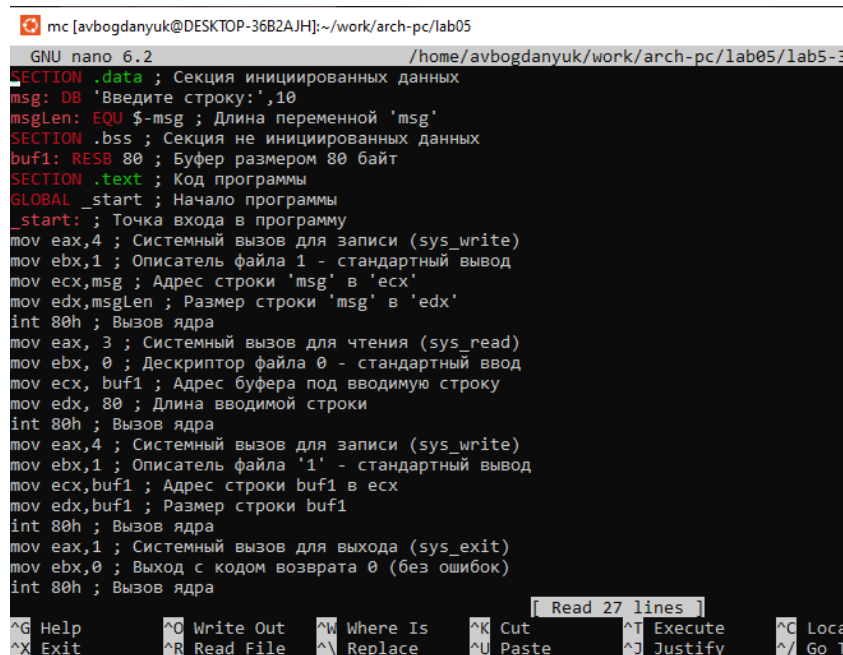


Рис. 4.20: Изменение файла

Листинг программы:

```
"SECTION .data ; Секция иницированных 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 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
mov edx,buf1 ; Размер строки buf1
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра"

```

Запускаю исполняемый файл (рис. 4.21).

```

avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ nasm -f elf lab5-3.asm
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-3 lab5-3.o
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ ./lab5-3
Введите строку:
Богданюк Анна Васильевна
Богданюк Анна Васильевна

```

Рис. 4.21: Вывод

Копирую файл lab5-2.asm с именем lab5-4.asm (рис. 4.22).

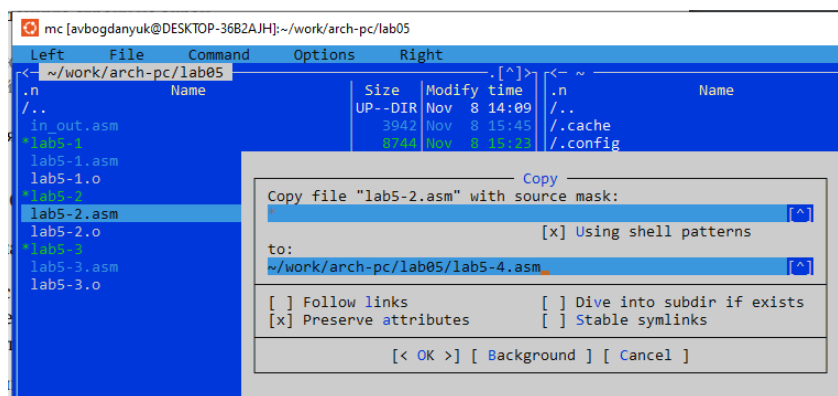


Рис. 4.22: Копирование файла

Изменяю файл, чтобы был вывод строки, ввод пользователем и вывод строки (рис. 4.23).

```

mc [avbogdanyuk@DESKTOP-36B2AJH]:~/work/arch-pc/lab05
GNU nano 6.2 /home/avbogdanyuk/work/arch-pc/lab05/lab5-4.asm *
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициализированных 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

```

Рис. 4.23: Изменение файла

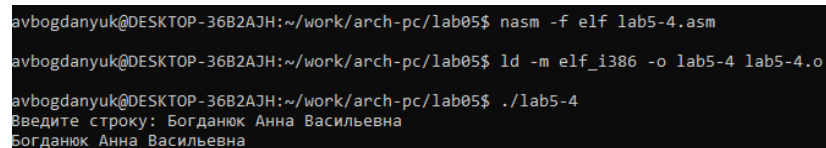
Листинг программы:

```

"%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 ; 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"

```

Запускаю исполняемый файл (рис. 4.24).



```

avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ nasm -f elf lab5-4.asm
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-4 lab5-4.o
avbogdanyuk@DESKTOP-36B2AJH:~/work/arch-pc/lab05$ ./lab5-4
Введите строку: Богданюк Анна Васильевна
Богданюк Анна Васильевна

```

Рис. 4.24: Вывод

## 5 Выводы

Во время выполнения лабораторной работы приобрела практические навыки работы в Midnight Commander. Основание инструкций языка ассемблера `mov` и `int`.

## **Список литературы**