

Лабораторная работа №6

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

Вершинина Ангелина Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Подключение внешнего файла in_out.asm	12
4.2	Выполнение заданий для самостоятельной работы	14
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Открытие Midnight Commander	8
4.2	Переход в каталог	9
4.3	Создание каталога	9
4.4	Создание файла	10
4.5	Открытие файла	10
4.6	Ввод текста программы	11
4.7	Сохранение текста программы	11
4.8	Запуск исполняемого файла	12
4.9	Открытие нужных файлов в панели	12
4.10	Копирование файла	13
4.11	Открытие файла	13
4.12	Запуск исполняемого файла	14
4.13	Редактирование файла	14
4.14	Запуск исполняемого файла	14
4.15	Редактирование файла	15
4.16	Запуск исполняемого файла	15
4.17	Запуск исполняемого файла	17

Список таблиц

1 Цель работы

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

2 Задание

Написать программы принимающие на ввод строку и вывод ее.

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.

4 Выполнение лабораторной работы

Открою Midnight Commander (рис. 4.1)

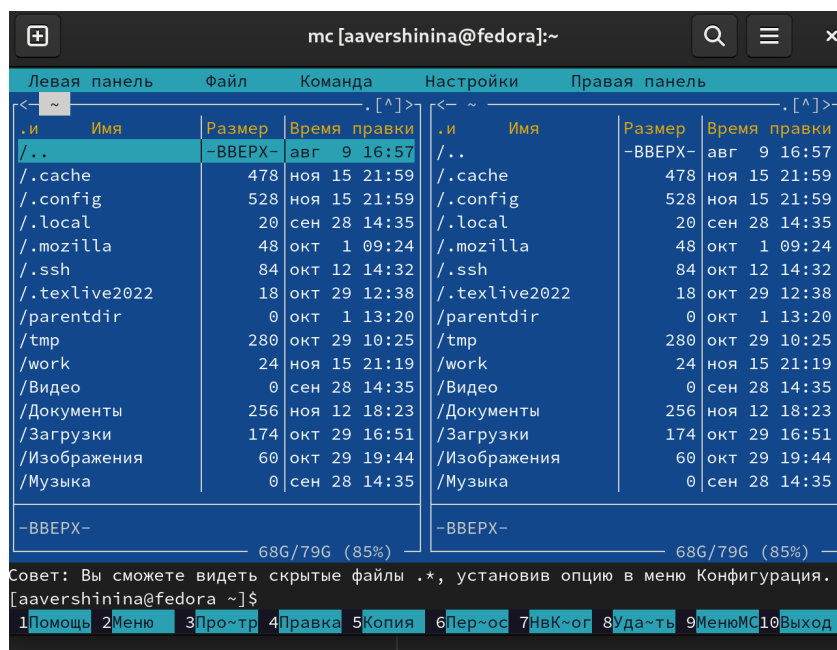


Рис. 4.1: Открытие Midnight Commander

Пользуясь клавишами **⌘**, **⌘** и Enter перейду в каталог ~/work/arch-rc (рис. 4.2)

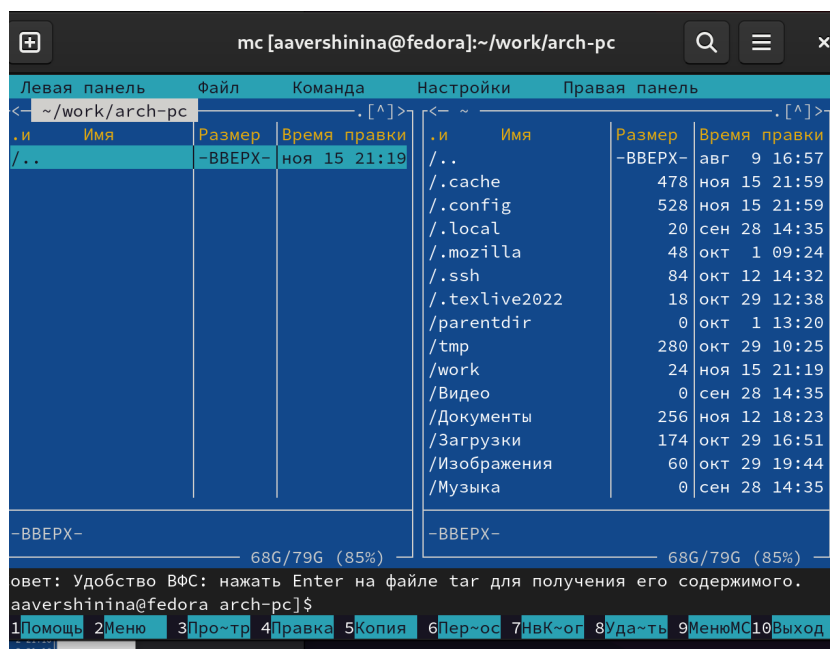


Рис. 4.2: Переход в каталог

С помощью функциональной клавиши F7 создам папку lab06 (рис. 4.3) и перейду в созданный каталог.

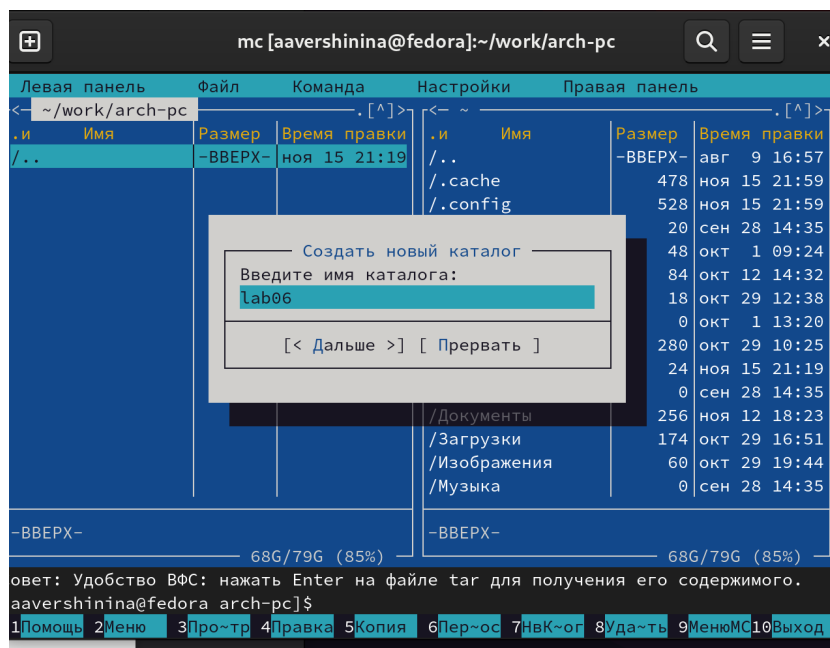


Рис. 4.3: Создание каталога

Пользуясь строкой ввода и командой touch создам файл lab6-1.asm (рис. 4.4)

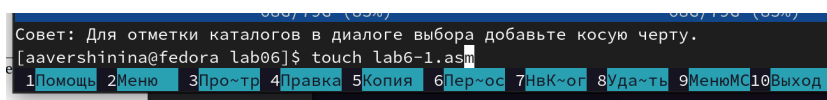


Рис. 4.4: Создание файла

С помощью функциональной клавиши F4 открою файл lab6-1.asm для редактирования во встроенном редакторе (рис. 4.5)

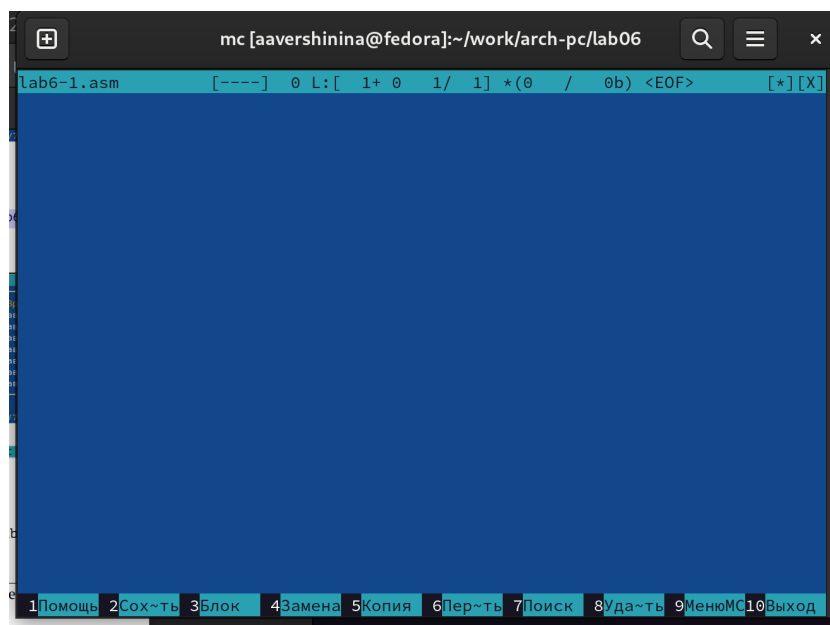
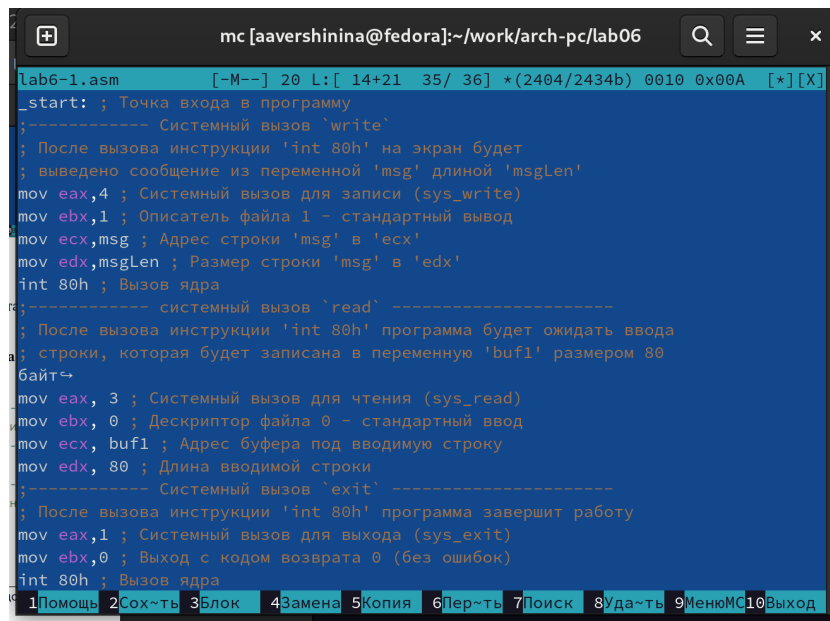


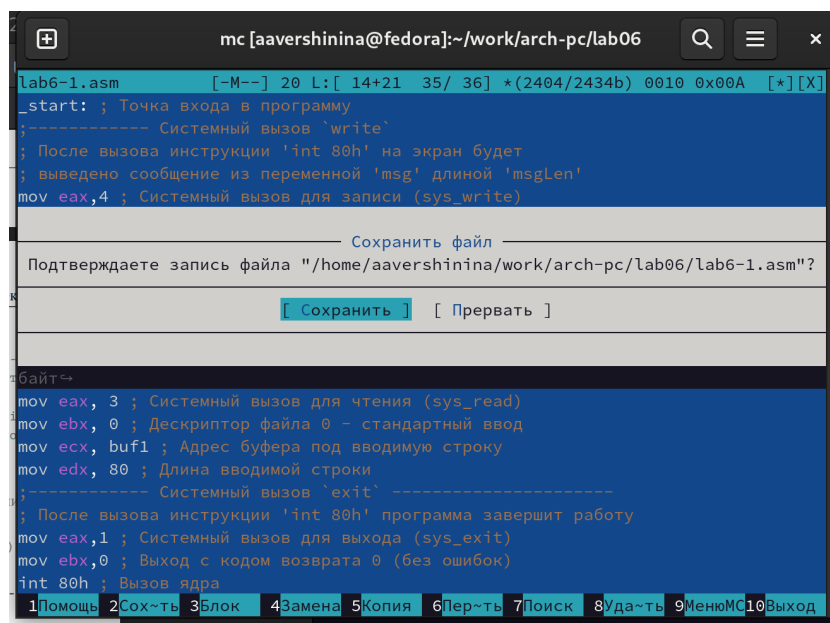
Рис. 4.5: Открытие файла

Введу текст программы из листинга 6.1 (рис. 4.6), со-храню изменения (рис. 4.7) и закрою файл.



```
lab6-1.asm [-M--] 20 L:[ 14+21 35/ 36] *(2404/2434b) 0010 0x00A [*][X]
_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
;----- 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
1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.6: Ввод текста программы



```
lab6-1.asm [-M--] 20 L:[ 14+21 35/ 36] *(2404/2434b) 0010 0x00A [*][X]
_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)
;----- 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
;----- 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
1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.7: Сохранение текста программы

С помощью функциональной клавиши F3 открою файл lab6-1.asm для просмотра. Файл действительно содержит текст программы.

Оттранслирую текст программы lab6-1.asm в объектный файл. Выполню компоновку объектного файла и запущу получившийся исполняемый файл. Программа

выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос введу фамилию и имя (рис. 4.8)

```
[aavershinina@fedora lab06]$ nasm -f elf lab6-1.asm
[aavershinina@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[aavershinina@fedora lab06]$ ./lab6-1
Введите строку:
Vershinina Angelina
[aavershinina@fedora lab06]$
```

Рис. 4.8: Запуск исполняемого файла

4.1 Подключение внешнего файла in_out.asm

Скачаю файл in_out.asm со страницы курса в ТУИС. В одной из панелей mc открою каталог с файлом lab6-1.asm. В другой панели каталог со скаченным файлом in_out.asm. (рис. 4.9)

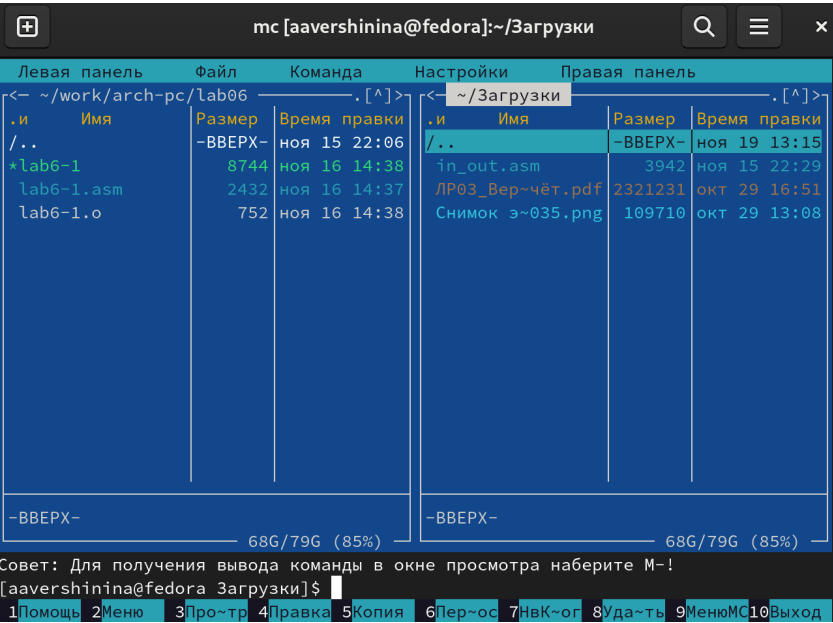


Рис. 4.9: Открытие нужных файлов в панели

Скопирую файл in_out.asm в каталог с файлом lab6-1.asm с помощью функциональной клавиши F5 (рис. 4.10)

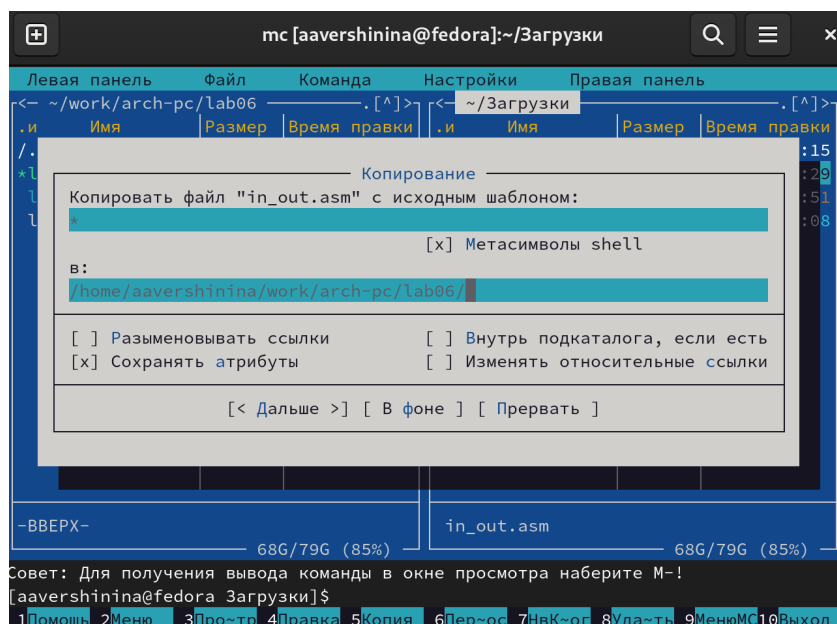


Рис. 4.10: Копирование файла

С помощью функциональной клавиши F6 создам копию файла lab6- 1.asm с именем lab6-2.asm. Выделю файл lab6-1.asm, нажму клавишу F6 , введу имя файла lab6-2.asm и нажму клавишу Enter (рис. 4.11)

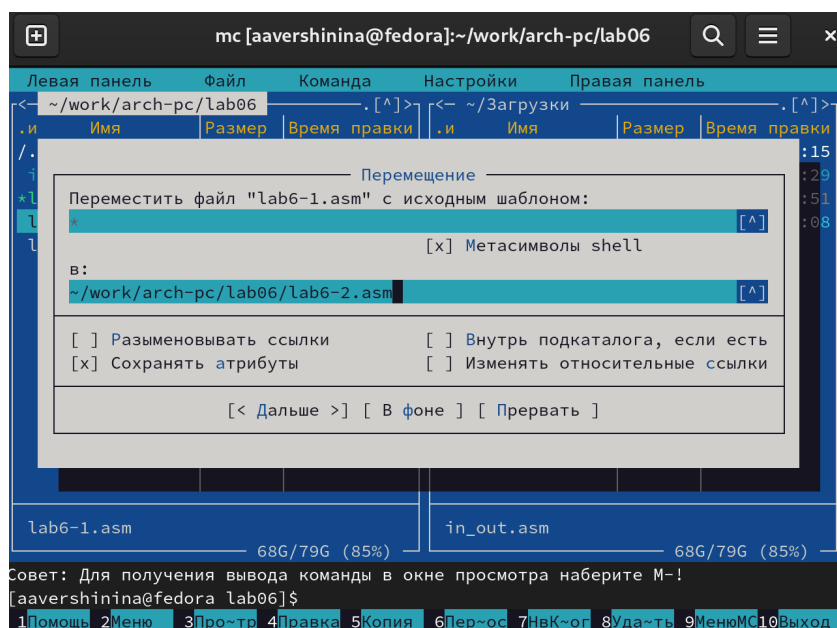


Рис. 4.11: Открытие файла

Исправлю текст программы в файле lab6-2.asm с использованием под-программ

из внешнего файла `in_out.asm` в соответствии с листингом 6.2. Создам исполняемый файл и проверьте его работу (рис. 4.12)

```
[aavershinina@fedora lab06]$ nasm -f elf lab6-2.asm
[aavershinina@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[aavershinina@fedora lab06]$ ./lab6-2
Введите строку:
Vershinina Angelina
[aavershinina@fedora lab06]$
```

Рис. 4.12: Запуск исполняемого файла

В файле `lab6-2.asm` заменю подпрограмму `sprintLF` на `sprint`. (рис. 4.13) Создам исполняемый файл и проверю его работу. (рис. 4.14) Отличие `sprintLF` и `sprint` заключается в том, что в первом случае программа запрашивает ввод данных с новой строки, а во втором не происходит переноса строки для ввода данных.

```
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'
```

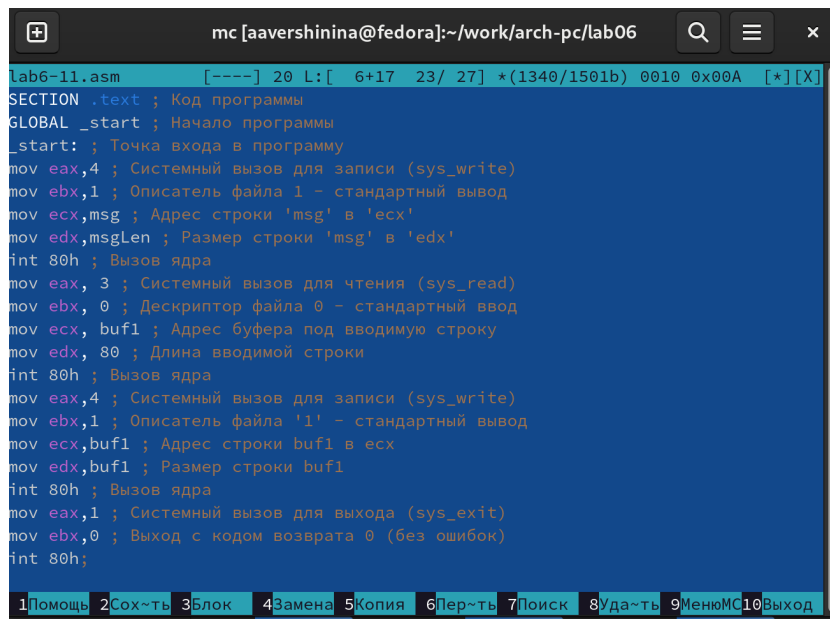
Рис. 4.13: Редактирование файла

```
[aavershinina@fedora lab06]$ nasm -f elf lab6-2.asm
[aavershinina@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[aavershinina@fedora lab06]$ ./lab6-2
Введите строку: Vershinina Angelina
[aavershinina@fedora lab06]$
```

Рис. 4.14: Запуск исполняемого файла

4.2 Выполнение заданий для самостоятельной работы

1. Создам копию файла `lab6-1.asm` с именем `lab6-11.asm` с помощью функциональной клавиши F5. С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, производился вывод вводимой пользователем строки (рис. 4.15)

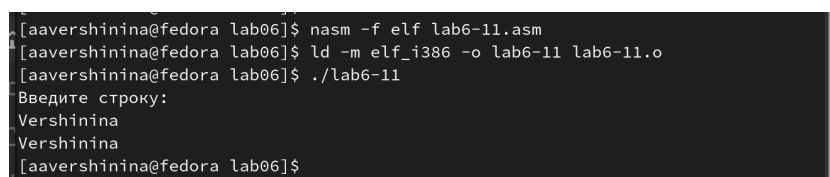


```
lab6-11.asm [----] 20 L: [ 6+17 23/ 27] *(1340/1501b) 0010 0x00A [*][X]
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки '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;

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню10Выход
```

Рис. 4.15: Редактирование файла

Создам исполняемый файл и проверю его работу. Программа запрашивает ввод строки, ввожу свою фамилию, программа выводит введенную строку(рис. 4.16)



```
[aavershinina@fedora lab06]$ nasm -f elf lab6-11.asm
[aavershinina@fedora lab06]$ ld -m elf_i386 -o lab6-11 lab6-11.o
[aavershinina@fedora lab06]$ ./lab6-11
Введите строку:
Vershinina
Vershinina
[aavershinina@fedora lab06]$
```

Рис. 4.16: Запуск исполняемого файла

Листинг программы 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 ; 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

```

2. Создам копию файла lab6-2.asm с именем lab6-2-1.asm с помощью функциональной клавиши F5. С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, производился вывод вводимой пользователем строки.

Создам исполняемый файл и проверю его работу. Программа запрашивает ввод строки, ввожу свою фамилию, программа выводит введенную строку(рис. 4.17)


```

[aaavershinina@fedora lab06]$ nasm -f elf lab6-21.asm
[aaavershinina@fedora lab06]$ ld -m elf_i386 -o lab6-21 lab6-21.o
[aaavershinina@fedora lab06]$ ./lab6-21
Введите строку: Vershinina
Vershinina
[aaavershinina@fedora lab06]$

```

Рис. 4.17: Запуск исполняемого файла

Листинг программы 2

```

#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 приобрела практические навыки работы в Midnight Commander, а также освоила инструкции языка ассемблера `mov` и `int`.

Список литературы