

Шаблон отчёта по лабораторной работе

Простейший вариант

Дмитрий Сергеевич Кулябов

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основы работы с mc	9
4.2	Структура программы на языке ассемблера NASM	11
4.3	Подключение внешнего файла	13
5	Выполнение заданий для самостоятельной работы	17
6	Выводы	20

Список иллюстраций

4.1	Открываю Midnight commander	9
4.2	Перемещение между директориями	10
4.3	Создание каталога	10
4.4	Создание каталога	10
4.5	Команда touch	11
4.6	Клавиша F4 и редактирование файла	11
4.7	Проверка	12
4.8	Ввод команд	12
4.9	Запуск	12
4.10	Файл in_out.asm	13
4.11	Перемещение файла	13
4.12	Копирую файл и меняю названия	14
4.13	Изменение содержимого файла.	14
4.14	Транслирование файла в объектный файл и компоновка.	15
4.15	Изменение файла	15
4.16	Транслирование файла в объектный файл и компоновка.	15
5.1	Копирование файла.	17
5.2	Изменение файла.	18
5.3	Проверка.	18
5.4	Изменение файла.	19
5.5	Проверка.	19

Список таблиц

1 Цель работы

Приобрести практические навыки работы в Midnight Commander. Освоить 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 предоставляет возможность просматривать структуру каталогов и выполнять основные операции управления файловой системой, то есть mc является файловым менеджером. Midnight Commander упрощает работу с файлами, делая её более удобной и наглядной. Программа на языке ассемблера NASM обычно состоит из трёх секций: секция кода программы (SECTION .text), секция инициализированных данных (SECTION .data), известная на этапе компиляции, и секция неинициализированных данных (SECTION .bss), для которых память отводится во время компиляции, но значения присваиваются в ходе выполнения программы. Для объявления инициализированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения будут храниться в этой памяти:

- DB (define byte) — определяет переменную размером в 1 байт;
- DW (define word) — определяет переменную размером 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 (четверное слово);
- DT (define ten bytes) — определяет переменную размером в 10 байт.

Эти директивы применяются для объявления простых переменных и массивов. Для определения строк обычно используется директива DB из-за особенностей хранения данных в оперативной памяти. Инструкция языка ассемблера mov предназначена для копирования данных из источника в приёмник.

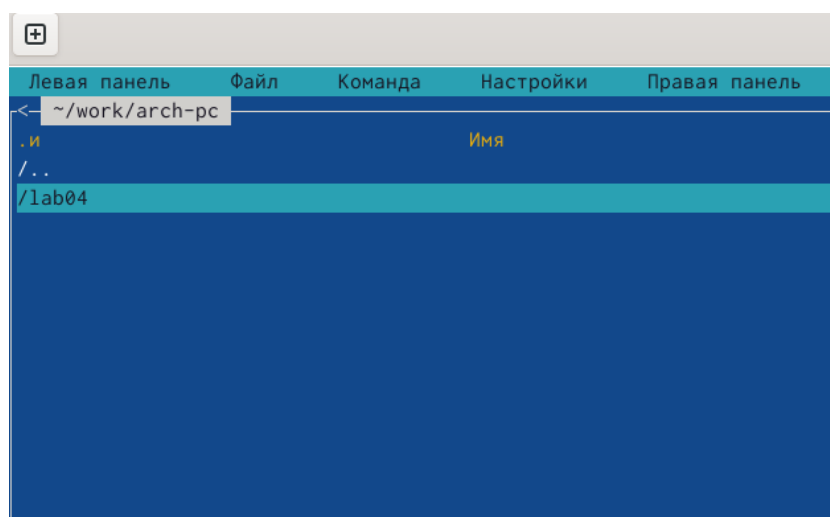


Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 4.3).

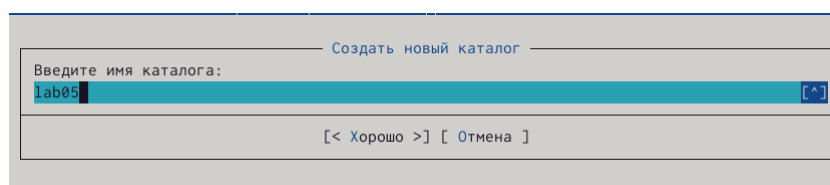


Рис. 4.3: Создание каталога

Вижу, что каталог создан (рис. 4.4).

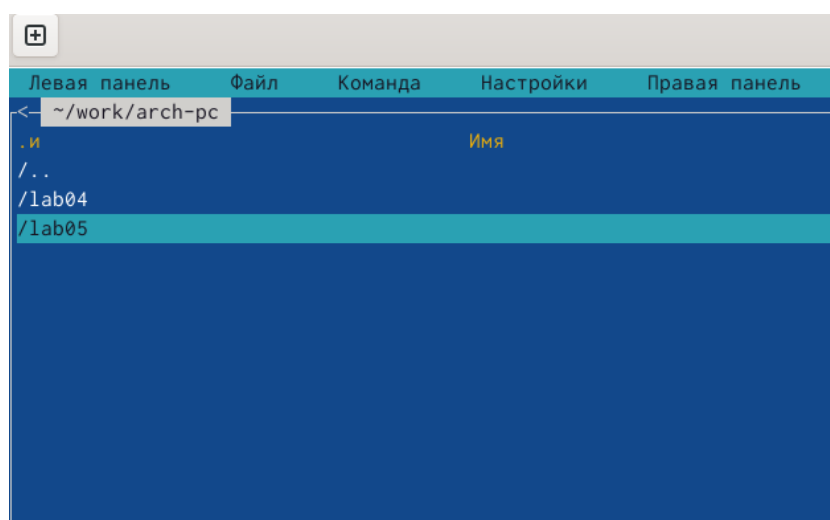


Рис. 4.4: Создание каталога

Захожу в каталог и в строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в котором буду работать (рис. 4.5).

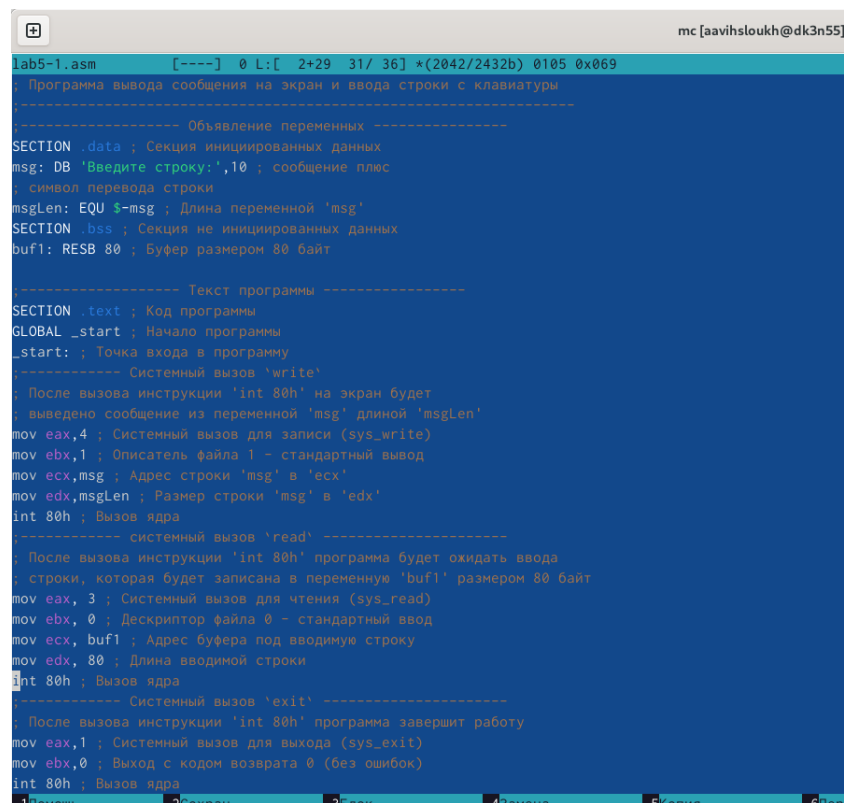
```
Совет: Макросы % работают даже в командной строке.
aavihsloukh@dk3n55 ~/work/arch-pc $ touch lab5-1.asm
```

1Помощь	2Меню	3Просмотр	4Правка	5Копия
---------	-------	-----------	---------	--------

Рис. 4.5: Команда touch

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл в редакторе mcedit и ввожу в файл код программы (рис. 4.6).



```
lab5-1.asm  [----]  0 L:[ 2+29 31/ 36] *(2042/2432b) 0105 0x069
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
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
;-----
;----- 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 ; Описание 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: Клавиша F4 и редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 4.7).

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 4.10).

Левая панель				Файл	Команда	Настройки	Правая панель				
<- ~/Загрузки								<- ~			
.[^]>								.[^]>			
.и	Имя	Размер	Дата правки	.и	Имя	Размер	Дата правки	.и	Имя	Размер	Дата правки
/..		-BBEPX-	окт 24 17:18	/..		-BBEPX-	окт 10 16:58	/..		-BBEPX-	окт 10 16:58
in_out.asm		3942	окт 24 17:27	/.cache		2048	сен 26 13:20	/.cache		2048	сен 26 13:20
report.pdf		293741	окт 11 18:18	/.config		2048	окт 23 19:05	/.config		2048	окт 23 19:05
Лаборато~(1).pdf		3163684	окт 23 15:41	/.gnupg		2048	окт 24 16:31	/.gnupg		2048	окт 24 16:31
Лаборато~иса.pdf		3163684	окт 11 15:54	/.local		2048	сен 26 12:09	/.local		2048	сен 26 12:09
Файлы.zip		263193	окт 12 14:12	/.mozilla		2048	сен 26 13:18	/.mozilla		2048	сен 26 13:18
				/.pki		2048	сен 26 13:17	/.pki		2048	сен 26 13:17
				/.ssh		2048	сен 27 17:18	/.ssh		2048	сен 27 17:18
				/.texlive		2048	окт 10 12:31	/.texlive		2048	окт 10 12:31
				/arch-pc		2048	сен 27 17:18	/arch-pc		2048	сен 27 17:18
				/public		2048	сен 6 18:27	/public		2048	сен 6 18:27
				~public_html		18	сен 21 09:17	~public_html		18	сен 21 09:17
				/tmp		2048	окт 11 18:12	/tmp		2048	окт 11 18:12
				/work		2048	окт 24 16:37	/work		2048	окт 24 16:37
				/Видео		2048	сен 26 12:07	/Видео		2048	сен 26 12:07
in_out.asm				-BBEPX-							

Рис. 4.10: Файл in out.asm

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 4.11).

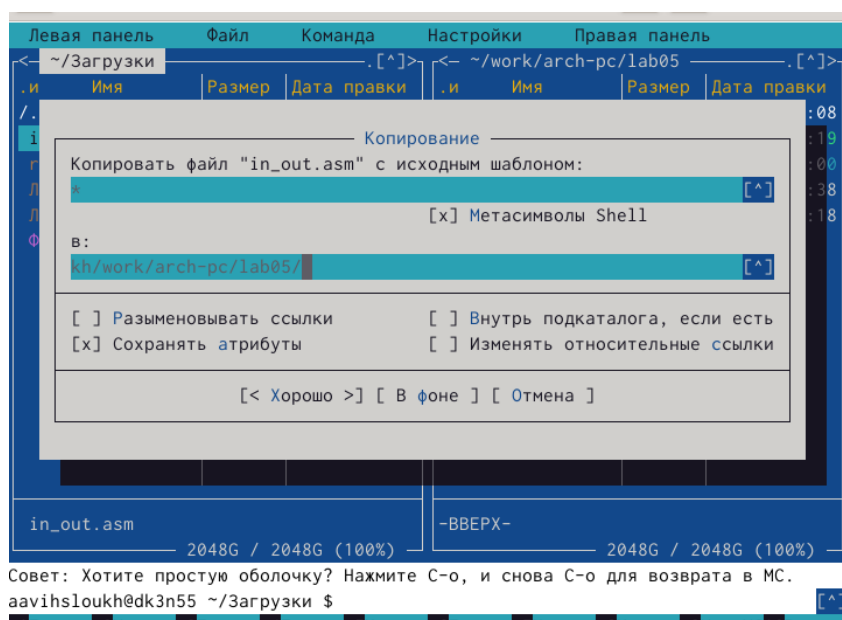


Рис. 4.11: Перемещение файла

С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с именем lab5-2 (рис. 4.12).

Левая панель	Файл	Команда	Настройки	Правая панель
~ /work/arch-pc/lab05 .[^]>				
.и	Имя		Размер	Дата правки
./..			-ВВЕРХ-	окт 24 17:08
in_out.asm			3942	окт 24 17:27
*lab5-1			8744	окт 24 17:19
lab5-1.asm			2432	окт 24 17:00
lab5-1.asm.save			2433	окт 24 17:38
lab5-1.o			752	окт 24 17:18
lab5-2.asm			2432	окт 24 17:00

Рис. 4.12: Копирую файл и меняю названия

Изменяю содержимое файла lab5-2.asm во встроенном редакторе mscedit (рис. 4.13), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

```
lab5-2.asm [----] 41 L: [ 1+ 0 1/ 17] *(41 /1224b) 0045 0x02D
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных 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 sprintf ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в 'ECX'
mov edx, 80 ; запись длины вводимого сообщения в 'EDX'
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 4.13: Изменение содержимого файла.

Я транслирую текст программы из файла в объектный файл, используя команду `nasm -f elf lab5-2.asm`. В результате был создан объектный файл lab5-2.o. Затем выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o`

lab5-2 lab5-2.o, в результате чего создаётся исполняемый файл lab5-2. Запускаю исполняемый файл (рис. 4.14).

```
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Выслух Алиса Александровна
```

Рис. 4.14: Транслирование файла в объектный файл и компоновка.

Открываю файл lab6-2.asm для редактирования с помощью текстового редактора nano, нажав функциональную клавишу F4. Вношу изменения в подпрограмму sprintLF, переименовывая её в sprint. После этого сохраняю изменения и открываю файл для просмотра, чтобы убедиться, что все изменения были сохранены корректно (рис. 4.15).

```
lab5-2.asm [----] 41 L: [ 1+16 17/ 17] *(1222/1222b) <EOF>
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных 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
```

Рис. 4.15: Изменение файла

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.16).

```
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-2 lab5-2.o
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ ./lab5-2-2
Введите строку: Выслух Алиса Алесандровна
```

Рис. 4.16: Транслирование файла в объектный файл и компоновка.

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 заключается в том, что при запуске первого файла программа запрашивает ввод с новой строки, в то время как программа во втором файле запрашивает ввод без переноса на новую строку. Это различие обусловлено изменениями в подпрограммах `sprintLF` и `sprint`: первая добавляет перенос строки после запроса ввода, а вторая — нет.

5 Выполнение заданий для самостоятельной работы

Создаю копию файла lab6-1.asm с именем lab6-1-1.asm с помощью функциональной клавиши F5 (рис. 5.1)

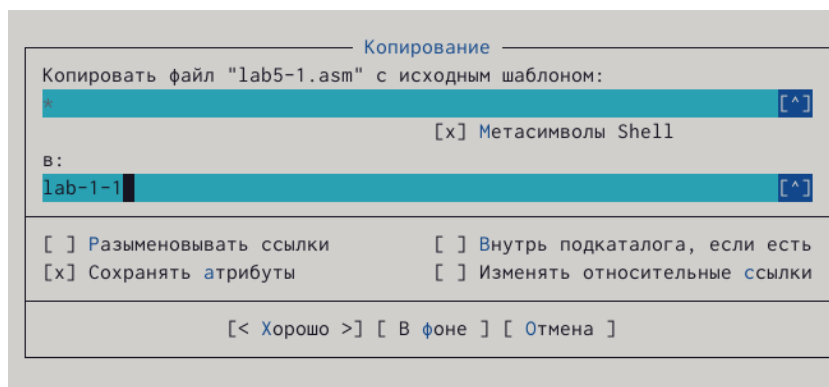


Рис. 5.1: Копирование файла.

С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 5.2).

```

lab5-1-1.asm      [----] 20 L:[ 1+17 18/ 26] *(1057/1521b) 0010
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных 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

```

Рис. 5.2: Изменение файла.

Создаю объектный файл lab5-1-1.o и передаю его на обработку компоновщику. После компоновки получаю исполняемый файл lab5-1-1. Запускаю полученный исполняемый файл. Программа запрашивает ввод, куда я ввожу свои ФИО, после чего программа выводит введенные мною данные (рис. 5.3).

```

aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1-1.asm
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ ./lab5-1-1
Введите строку:
Выслоух
Выслоух

```

Рис. 5.3: Проверка.

Создаю копию файла lab5-2.asm с именем lab6-2-1.asm с помощью функциональной клавиши F5, открываю файл с помощью клавиши F4 и изменяю ее так,

чтобы кроме вывода приглашения и запроса ввода, программа выводила пользовательскую строку (рис. 5.4).

```
lab5-2-1.asm [----] 41 L:[ 1+17 18/ 18] *(1145/1145b) <EOI
#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
```

Рис. 5.4: Изменение файла.

Проверяю аналогично предыдущему (рис. 5.5).

```
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2-1.asm
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
aavihsloukh@dk3n55 ~/work/arch-pc/lab05 $ ./lab5-2-1
Введите строку: Выслоух
Выслоух
```

Рис. 5.5: Проверка.

6 Выводы

При выполнении данной лабораторной работы я приобрела практические навыки работы в Midnight Commander, а также освоила инструкции языка ассемблера `mov` и `int`.