

Лабораторная работа №5

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

Коровкин Никита Михайлович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение самостоятельной работы	11
4	Выводы	14

Список иллюстраций

2.1	Запуск Midnight commander при помощи mc	6
2.2	Midnight commander	6
2.3	файл lab5-1.asm	7
2.4	Вставляем нужный код в файл	7
2.5	Компиляция кода	8
2.6	Сборка файла	8
2.7	Запуск исполняемого файла	8
2.8	Файл in_out.asm	8
2.9	Копируем файл в нужную директорию	9
2.10	Копируем файл lab5-1 и переименовываем	9
2.11	Редактируем файл	10
2.12	Компилируем и запускаем файл	10
2.13	Запускаем код после редакции	10
3.1	Копируем изначальный файл	11
3.2	Редактируем файл, чтобы на выводе мы получали то же, что и вводили	12
3.3	Запускаем файл	12
3.4	Копируем второй файл и пишем код для вывода того же текста, что и при вводе	13
3.5	Проверяем работоспособность кода	13

Список таблиц

1 Цель работы

Ознакомиться с программой Midnight commander и освоить написание программ на языке ассемблера с помощью инструкций `mov` и `int`

2 Выполнение лабораторной работы

Для начала выполнения лабораторной работы нам нужно открыть Midnight commander с помощью команды `mc` (Рис.1):

```
[nikitak@fedora ~]$ mc
```

Рис. 2.1: Запуск Midnight commander при помощи `mc`

Перед нами появится такой интерфейс (Рис.2)

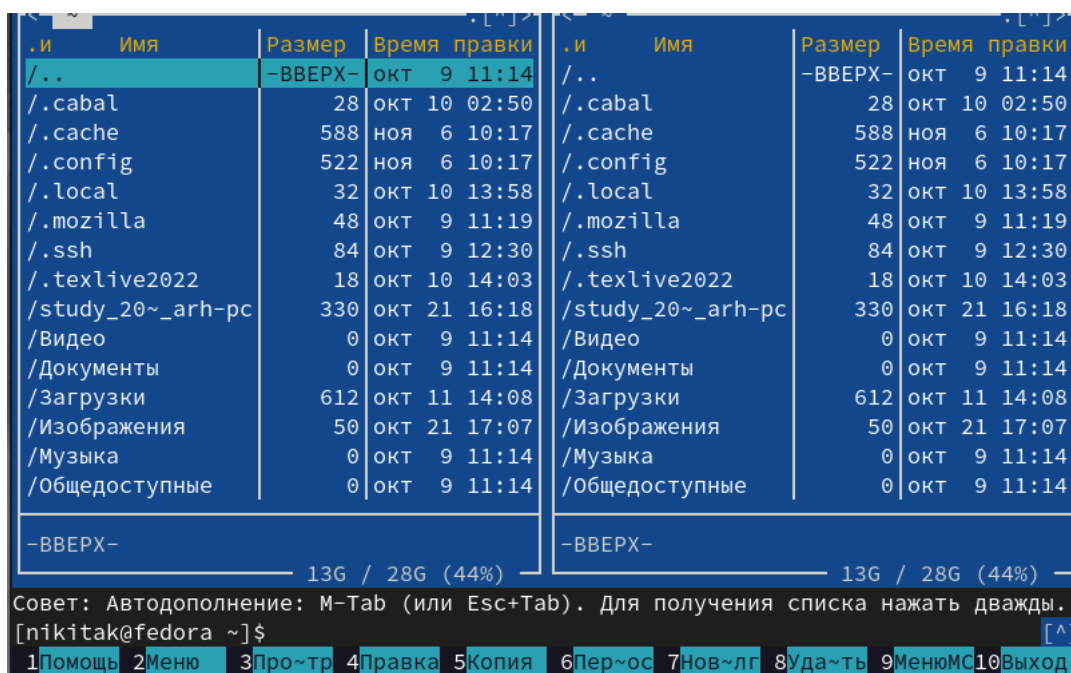


Рис. 2.2: Midnight commander

Создадим файл `lab5-1.asm` в нужной директории. (Рис.3)

Левая панель				Правая панель			
Файл		Команда		Настройки		Правая панель	
.и	Имя	Размер	Время правки	.и	Имя	Размер	Время правки
/..	-ВВЕРХ-	оكت 9 13:08		/..	-ВВЕРХ-	оكت 9 11:14	
/presentation	100	оكت 9 13:08		/.cabal	28	оكت 10 02:50	
/report	62	ноя 6 10:15		/.cache	588	ноя 6 10:17	
lab5-1.asm	0	ноя 6 10:35		/.config	522	ноя 6 10:17	
				/.local	32	оكت 10 13:58	
				/.mozilla	48	оكت 9 11:19	
				/.ssh	84	оكت 9 12:30	
				/.texlive2022	18	оكت 10 14:03	
				/study_20~_arh-pc	330	оكت 21 16:18	
				/Видео	0	оكت 9 11:14	
				/Документы	0	оكت 9 11:14	
				/Загрузки	612	оكت 11 14:08	
				/Изображения	50	оكت 21 17:07	
				/Музыка	0	оكت 9 11:14	
				/Общедоступные	0	оكت 9 11:14	
-ВВЕРХ-				-ВВЕРХ-			
13G / 28G (44%)				13G / 28G (44%)			

Совет: Автодополнение: M-Tab (или Esc+Tab). Для получения списка нажать дважды.

Рис. 2.3: файл lab5-1.asm

Откроем созданный файл и отредактируем его, вставив туда нужный код.
(Рис.4)

```

/home/nikitak/study_2024-2025_arh-pc/labs/lab05/lab5-1.asm  Изменён
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

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

```

Рис. 2.4: Вставляем нужный код в файл

Сохранив его, скомпилируем файл. (Рис.5)

```
liveuser@localhost-live:~$ nasm -f elf lab5-1.asm
```

Рис. 2.5: Компиляция кода

Теперь соберем файл. (Рис.6)

```
liveuser@localhost-live:~$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 2.6: Сборка файла

После этого запустим получившийся исполняемый файл. (Рис. 7):

```
liveuser@localhost-live:~$ ./lab5-1
Введите строку:
Korovkin Nikita Michilovich
```

Рис. 2.7: Запуск исполняемого файла

Запишем ФИО в строку и нажмем Enter. Ничего не произойдет. Тогда скачаем файл in_out.asm и откроем Midnight Commander. (Рис.8)

*lab5-1	8744	Nov 7 09:59	/Music	4096	Nov 7 09:35
lab5-1.asm	1496	Nov 6 02:48	/Pictures	4096	Nov 7 09:35
lab5-1.o	752	Nov 7 09:57	/Public	4096	Nov 7 09:35
			/Templates	4096	Nov 7 09:35
			/Videos	4096	Nov 7 09:35
			/lab05	4096	Nov 7 10:20
			.bash_history	59	Nov 7 09:42
			.bash_logout	18	Feb 8 2024
			.bash_profile	144	Feb 8 2024
			.bashrc	522	Feb 8 2024
			in_out.asm	3942	Nov 7 10:08
*lab5-1	8744	Nov 7 09:59			
lab5-1.asm	1496	Nov 6 02:48			
lab5-1.o	752	Nov 7 09:57			

Рис. 2.8: Файл in_out.asm

Скопируем файл в нужную папку, с которой мы работаем. (Рис.9)

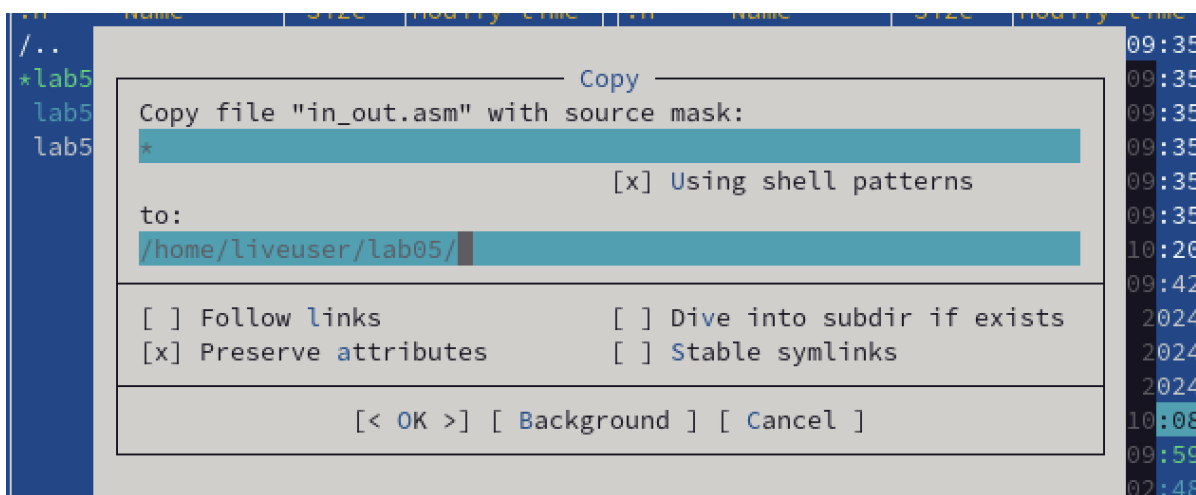


Рис. 2.9: Копируем файл в нужную директорию

Теперь скопируем lab5-1 в ту же папку, назвав его lab5-2. (Рис.10)

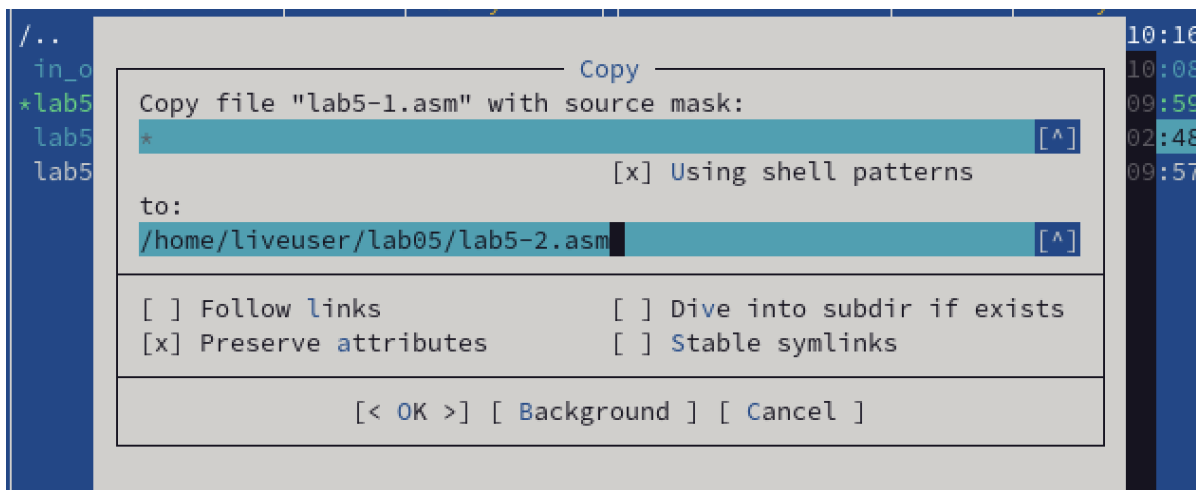
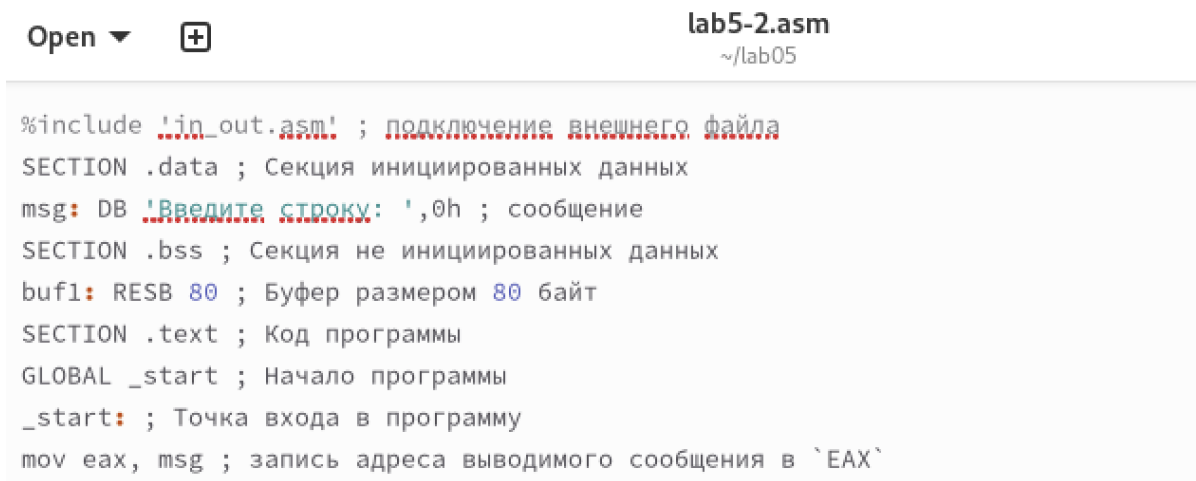


Рис. 2.10: Копируем файл lab5-1 и переименовываем

Откроем новый файл и отредактируем его, вставив туда следующий код. (Рис. 11)

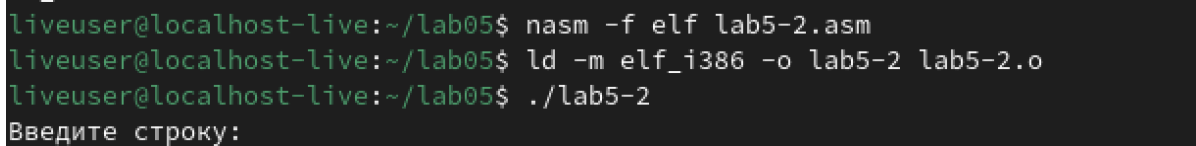


```
Open ▾ ⊕ lab5-2.asm
~/lab05

%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`
```

Рис. 2.11: Редактируем файл

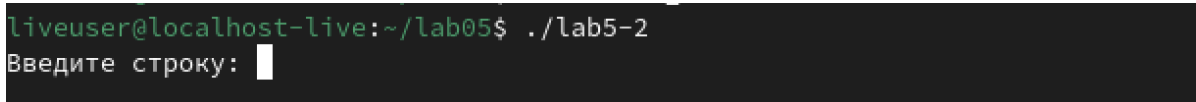
Теперь сделаем то же самое, что и с первым файлом, и запустим его. (Рис. 12)



```
liveuser@localhost-live:~/lab05$ nasm -f elf lab5-2.asm
liveuser@localhost-live:~/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
liveuser@localhost-live:~/lab05$ ./lab5-2
Введите строку:
```

Рис. 2.12: Компилируем и запускаем файл

Он работает также, как и файл lab5-1, но использует для работы сторонний файл. Теперь отредактируем код еще раз и заменим `sprintfL` на `sprintf`. Запускаем код. (Рис.13)



```
liveuser@localhost-live:~/lab05$ ./lab5-2
Введите строку: █
```

Рис. 2.13: Запускаем код после редакции

Нетрудно заметить, что теперь нет переноса на следующую строку. В этом и отличие команды `sprintfLF` от `sprintf`. Первая добавляет перенос после текста, а вторая нет.

3 Выполнение самостоятельной работы

Теперь еще раз скопируем изначальный файл. (Рис.14)

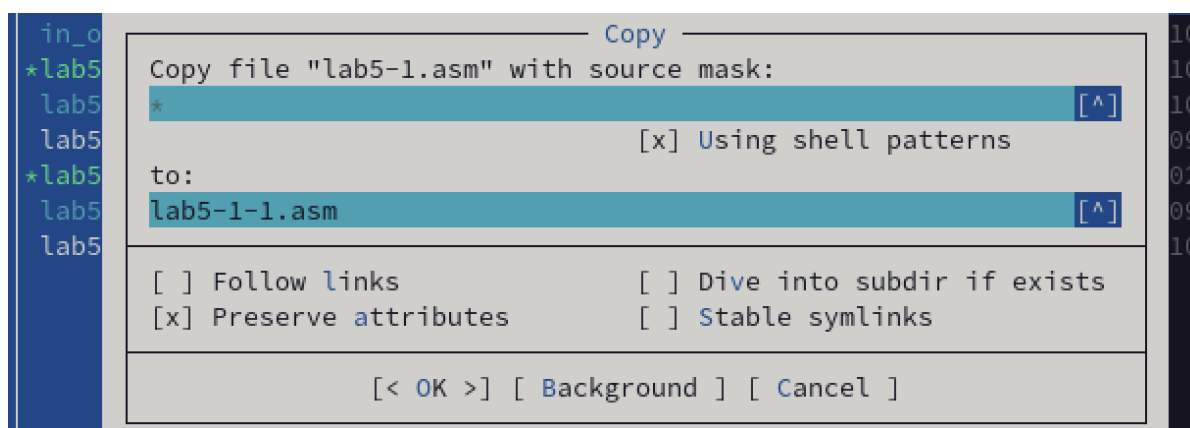


Рис. 3.1: Копируем изначальный файл

Нам нужно сделать так, чтобы код выводил полученную на вводе строку. Для этого перед системным вызовом `exit` нам нужно вставить часть кода с вызовом `write`. Нам необходимо переместить адрес строки `buf1` в `ecx` и размер строки `buf1` (80) в `edx`. (Рис.15)

```

;----- 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

```

Рис. 3.2: Редактируем файл, чтобы на выводе мы получали то же, что и вводили

Запускаем файл и проверяем работоспособность. (Рис.16)

```

liveuser@localhost-live:~/lab05$ nasm -f elf lab5-1-1.asm
liveuser@localhost-live:~/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
liveuser@localhost-live:~/lab05$ ./lab5-1-1
Введите строку:
Nikita
Nikita

```

Рис. 3.3: Запускаем файл

Теперь создадим копию файла lab5-2.asm и сделаем так, чтобы код тоже выводил тот же текст, что получает на ввод. Для этого перед последней строкой добавим строчку, которая записывает в eax адрес buf1 и строку, вызывающую подпрограмму sprintLF. (Рис.17)

```

;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#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 sprintf ; 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, buf1 ; запись адреса выводимого сообщения в `EAX`
call sprintf ; вызов подпрограммы печати сообщения
call quit ; вызов подпрограммы завершения

```

Рис. 3.4: Копируем второй файл и пишем код для вывода того же текста, что и при вводе

Скомпилируем файл и запустим код.(Рис.18)

```

liveuser@localhost-live:~/lab05$ nasm -f elf lab5-2-1.asm
liveuser@localhost-live:~/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
liveuser@localhost-live:~/lab05$ ./lab5-2-1
Введите строку:
Nikita K
Nikita K

```

Рис. 3.5: Проверяем работоспособность кода

Программа выполняется верно, следовательно, самостоятельная работа выполнена правильно.

4 Выводы

В ходе выполнения данной лабораторной работы были получены навыки работы с Midnight commander и навыки создания программ ввода и вывода на ассемблере.