

Лабораторная работа №5

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

Андрюшин Никита Сергеевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение задания для самостоятельной работы	17
4	Выводы	21

Список иллюстраций

2.1	Запуск Midnight commander	6
2.2	Интерфейс midnight commander	6
2.3	Переход в нужный каталог (~/.work/arch-pc)	7
2.4	Создание папки	8
2.5	Создание файла lab5-1.asm с помощью команды touch прямо в mc	9
2.6	Выбор текстового редактора	10
2.7	Редактирование файла lab5-1.asm	11
2.8	Проверка успешного редактирования	12
2.9	Компиляция файла с помощью nasm	12
2.10	Сборка исполняемого файла с помощью ld	12
2.11	Запуск исполняемого файла	13
2.12	Взаимодействие с программой	13
2.13	Открытие папки с файлом in_out.asm в правой панели	13
2.14	Копирование файла с помощью F6	14
2.15	Копирование файла с помощью F5	14
2.16	Текущий вид рабочей папки	15
2.17	Редактирование файла lab5-2.asm	15
2.18	Создание исполняемого файла	15
2.19	Запуск исполняемого файла	16
2.20	Изменение файла lab5-2.asm	16
2.21	Запуск изменённого файла	16
3.1	Создание копии файла lab5-1.asm	17
3.2	Изменение файла lab5-1-1.asm	18
3.3	Создание исполняемого файла	18
3.4	Проверка работы программы	18
3.5	Создание копии файла lab5-2.asm	19
3.6	Изменение файла lab5-2-1.asm	19
3.7	Создание исполняемого файла	20
3.8	Проверка работы программы	20

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 необходимо открыть Midnight commander с помощью команды `mc` (Рис. 2.1):

```
nsandryushin@nsandryushin:~$ mc
```

Рис. 2.1: Запуск Midnight commander

После ввода команды мы увидим такой интерфейс (Рис. 2.2):

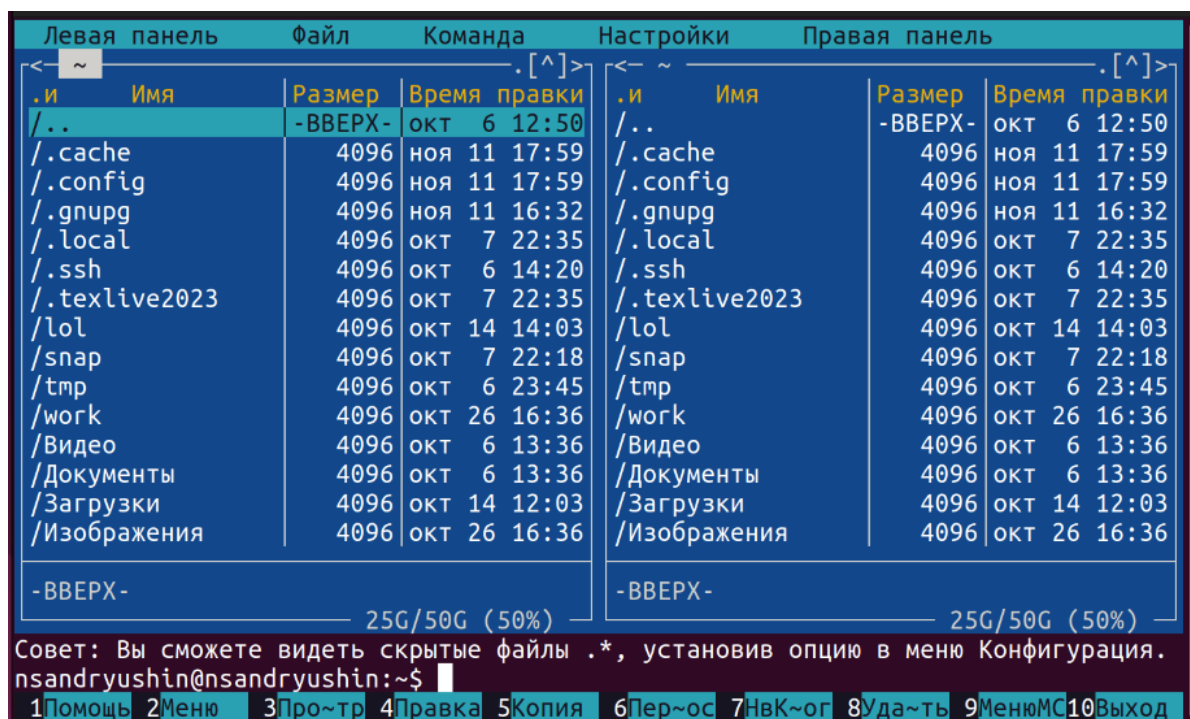


Рис. 2.2: Интерфейс midnight commander

2.3):

Совет: Хотите простую оболочку? Нажмите C-o, и снова C-o для возврата в MS.

Рис. 2.3: Переход в нужный каталог (~/.work/arch-rc)

Создадим папку lab05 с помощью клавиши F7 (Рис. 2.4):

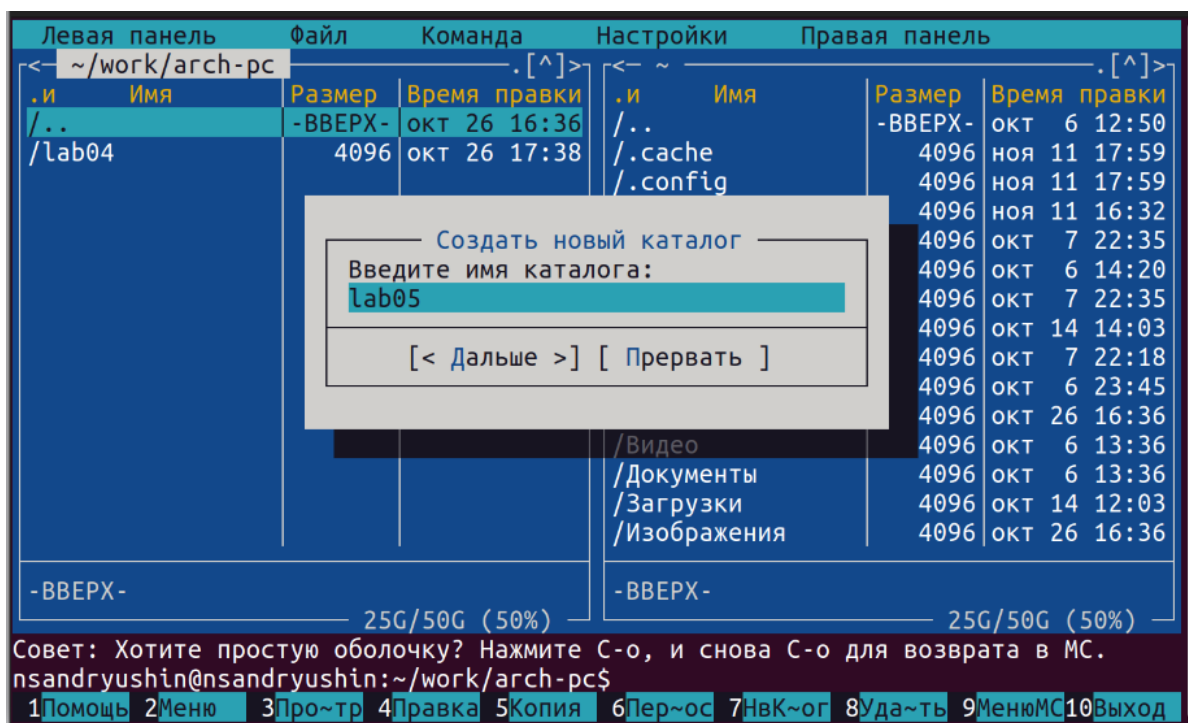


Рис. 2.4: Создание папки

Теперь с помощью команды `touch` создадим файл `lab5-1.asm` (Рис. 2.5):

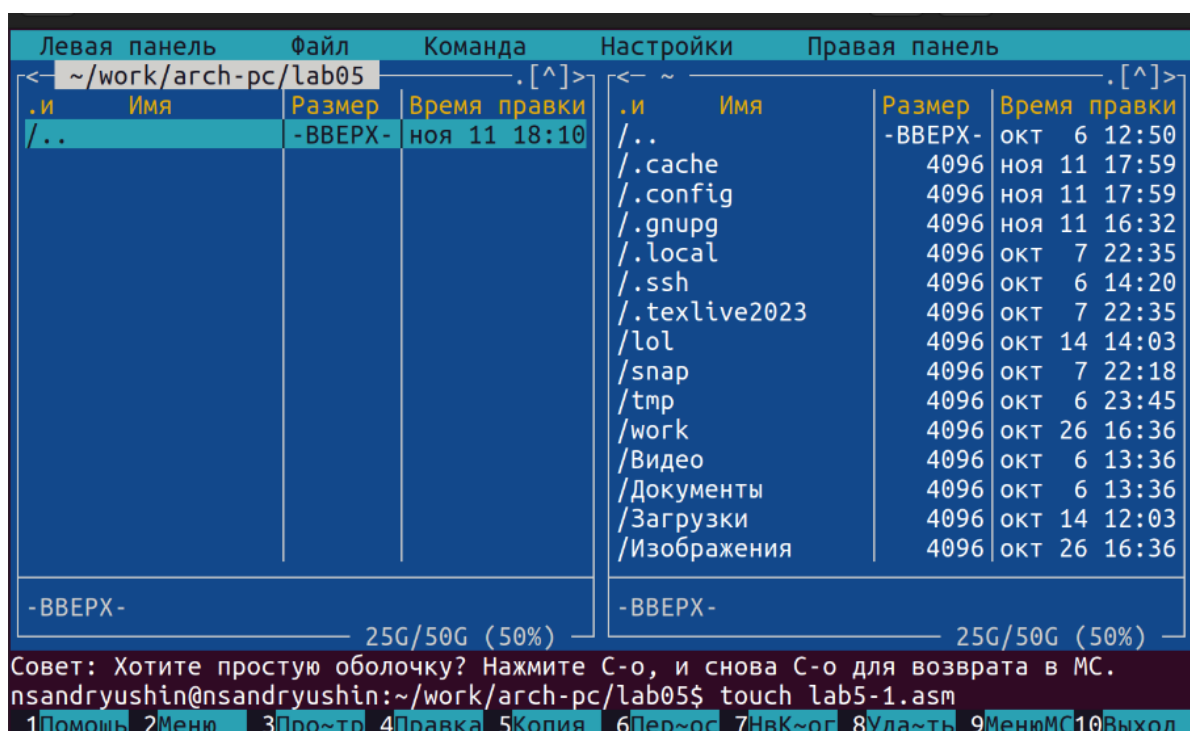


Рис. 2.5: Создание файла lab5-1.asm с помощью команды touch прямо в mc

Теперь с помощью клавиши F4 откроем только что созданный файл. Нам предложат выбор между несколькими редакторами. Мы выберем редактор по умолчанию, nano (Рис. 2.6):

```
nsandryushin@nsandryushin:~$ mc
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ touch lab5-1.asm
Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/mcedit
 3. /usr/bin/vim.tiny
 4. /bin/ed
Choose 1-4 [1]: █
```

Рис. 2.6: Выбор текстового редактора

Теперь отредактируем файл и поместим в него следующий код (Рис. 2.7):

```
GNU nano 6.2 /home/nsandryushin/work/arch-pc/lab05/lab5-1.asm *
;-----
; 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'

^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить    ^C Позиция
^X Выход        ^R ЧитФайл     ^\ Замена       ^U Вставить     ^J Выровнять    ^/_ К строке
```

Рис. 2.7: Редактирование файла lab5-1.asm

Теперь сохраним его (сочетанием клавиш `ctrl+x` и согласившись с сохранением) и с помощью `F3` откроем для просмотра, чтобы убедиться, что он сохранился корректно (Рис. 2.8):

```
/home/nsandryushin/work~ch-pc/lab05/lab5-1.asm 1448/2432 59%
;-----
; 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
1Помощь 2Раз~рн 3Выход 4Нех 5Пер~ти 6 7Поиск 8Исх~ый 9Формат10Выход
```

Рис. 2.8: Проверка успешного редактирования

Теперь скомпилируем его (Рис. 2.9):

```
Совет: Хотите простую оболочку? Нажмите C-o, и снова C-o для возврата в MC.
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm [^]
1Помощь 2Меню 3Про~тр 4Правка 5Копия 6Пер~ос 7НвК~ог 8Уда~ть 9МенюМС10Выход
```

Рис. 2.9: Компиляция файла с помощью nasm

И соберём (Рис. 2.10):

```
Совет: Хотите простую оболочку? Нажмите C-o, и снова C-o для возврата в MC.
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o [^]
1Помощь 2Меню 3Просмотр 4Правка 5Копия 6Перенос 7НвК~ог 8Удалить 9МенюМС 10Выход
```

Рис. 2.10: Сборка исполняемого файла с помощью ld

После этого запустим получившийся исполняемый файл (Рис. 2.11):

```
Совет: Хотите простую оболочку? Нажмите C-o, и снова C-o для возврата в MS.
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ./lab5-1
1Помощь 2Меню 3Просмотр 4Правка 5Копия 6Перенос 7НвКтлог 8Удалить 9МенюМС 10Выход
```

Рис. 2.11: Запуск исполняемого файла

Теперь введём ФИО (Рис. 2.12):

```
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Андрюшин Никита Сергеевич
```

Рис. 2.12: Взаимодействие с программой

После нажатия Enter программа завершится и ничего не произойдёт. Теперь скачаем файл `in_out.asm` и откроем папку с ним в правой панели (Рис. 2.13):

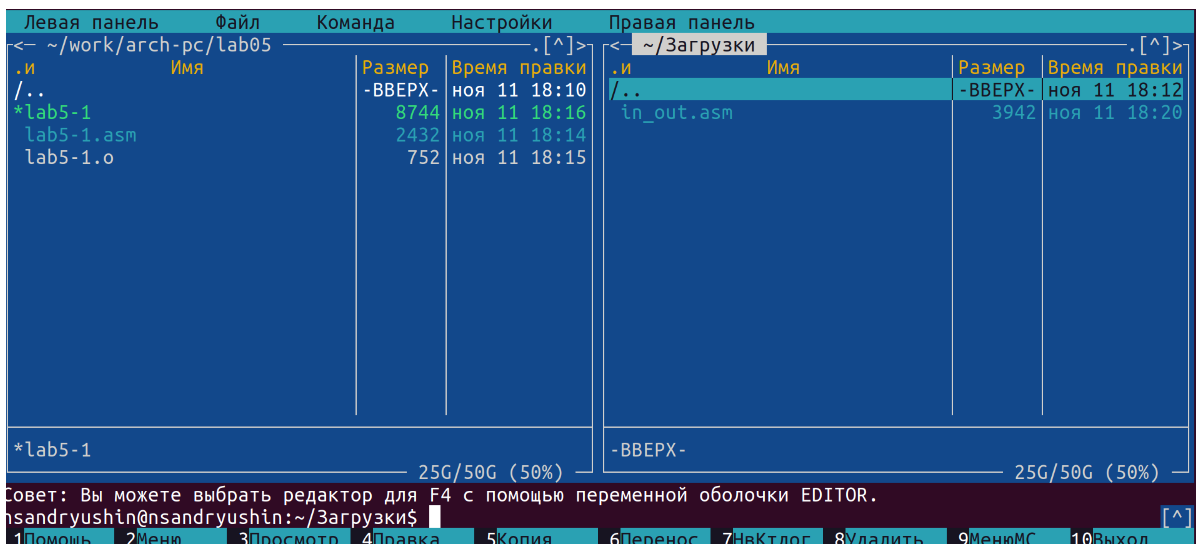


Рис. 2.13: Открытие папки с файлом `in_out.asm` в правой панели

Скопируем его в нашу рабочую папку с помощью F6 (Рис. 2.14):

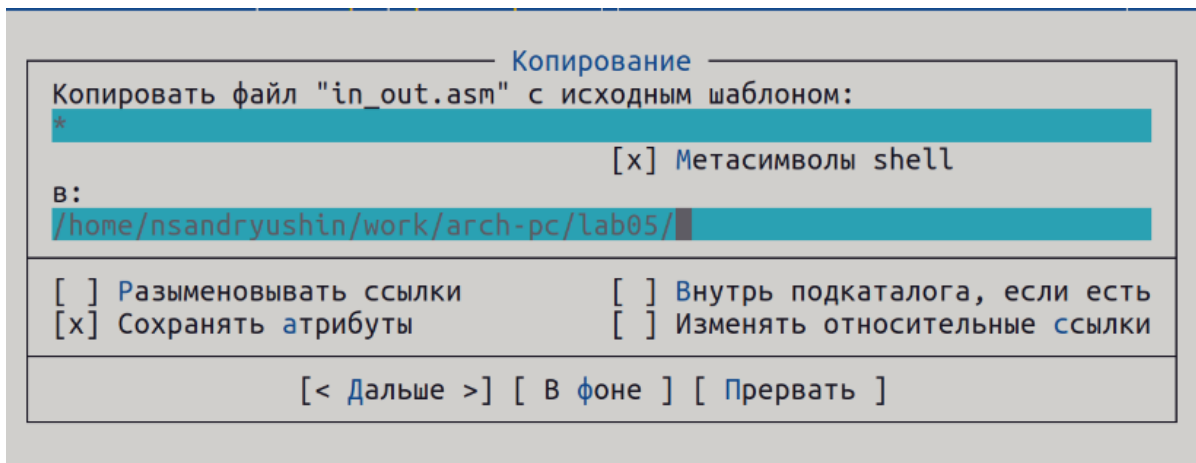


Рис. 2.14: Копирование файла с помощью F6

Теперь сделаем копию файла lab5-1.asm с помощью команды F5. Назовём копию lab5-2.asm (Рис. 2.15):

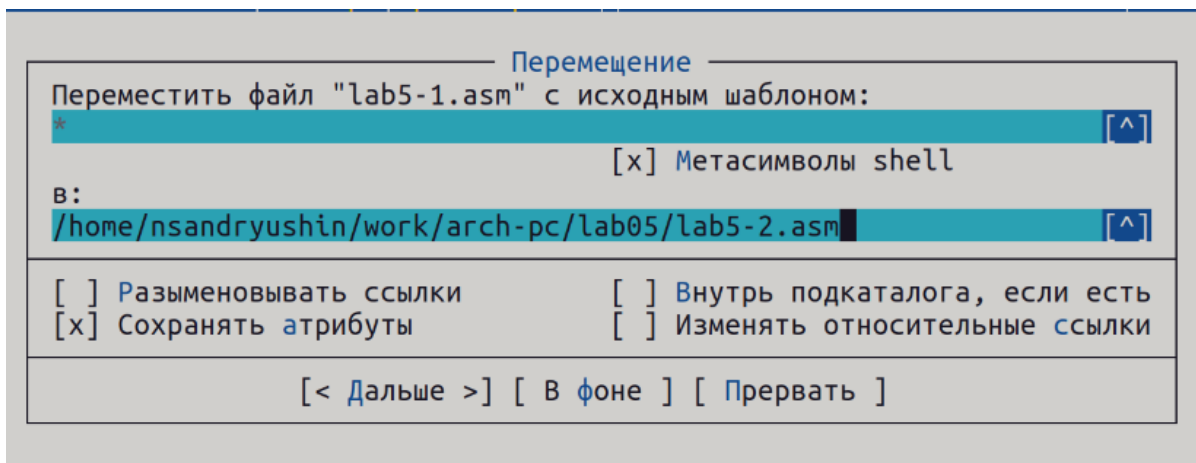


Рис. 2.15: Копирование файла с помощью F5

Теперь наша папка выглядит следующим образом (Рис. 2.16):

in_out.asm	3942	ноя 11 18:10
*lab5-1	8744	ноя 11 18:20
lab5-1.asm	2432	ноя 11 18:16
lab5-1.o	752	ноя 11 18:14
lab5-2.asm	2432	ноя 11 18:15

Рис. 2.16: Текущий вид рабочей папки

Откроем в текстовом редакторе файл lab5-2.asm и напишем туда следующий код (Рис. 2.17):

```

;-----
; 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 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
call quit ; вызов подпрограммы завершения

```

Рис. 2.17: Редактирование файла lab5-2.asm

После чего создадим исполняемый файл с помощью nasm и ld (Рис. 2.18):

```

nsandryushin@nsandryushin:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ./lab5-2

```

Рис. 2.18: Создание исполняемого файла

Запустим созданный файл (Рис. 2.19):

```
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Андрюшин Никита Сергеевич
```

Рис. 2.19: Запуск исполняемого файла

Он работает также, как и файл lab5-1, но использует для работы сторонний файл. Попробуем теперь вместо команды `sprintLF` использовать просто команду `sprint` (Рис. 2.20):

```
;-----
; 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
call quit ; вызов подпрограммы завершения
```

Рис. 2.20: Изменение файла lab5-2.asm

Точно также соберём исполняемый файл и запустим его (Рис. 2.21):

```
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Андрюшин Никита Сергеевич
```

Рис. 2.21: Запуск изменённого файла

Как мы видим, теперь нет переноса на следующую строку. Этим и отличаются команды `sprintLF` от `sprint`. Первая добавляет перенос после текста, а вторая нет

3 Выполнение задания для самостоятельной работы

Теперь создадим с помощью F6 копию файла lab5-1.asm (Рис. 3.1):

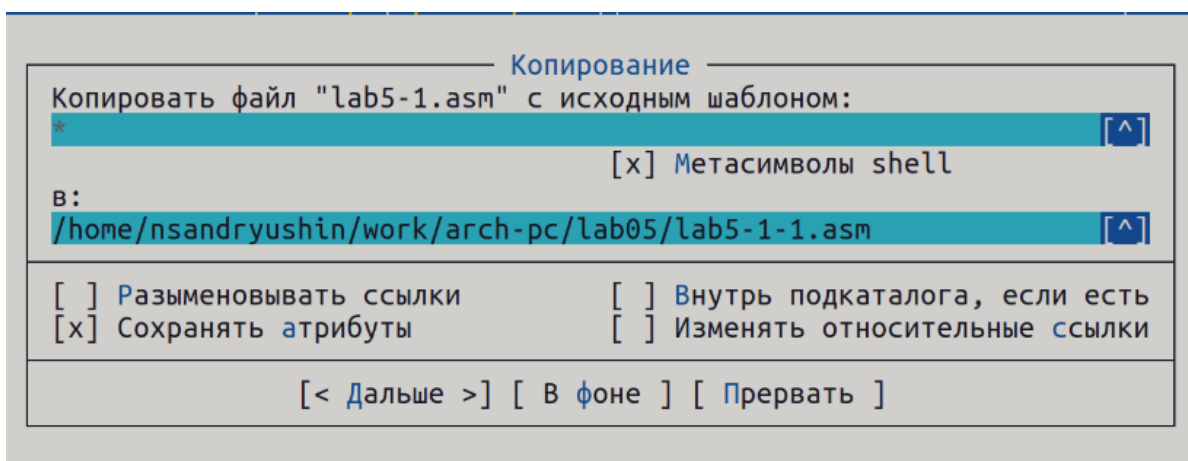


Рис. 3.1: Создание копии файла lab5-1.asm

Изменим копию так, чтобы она выводила тот текст, который получила на ввод. Для этого перед системным вызовом `exit` вставим текст с системным вызовом `write`. Он очень похож на системный вызов `write`, который уже был в коде, но есть несколько отличий. Так, мы перемещаем адрес строки `buf1` в `ecx` и размер строки `buf1` (80) в `edx` (Рис. 3.2):

```

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 `write` -----
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'buf1' длиной 80
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, 80 ; Размер строки 'buf1' в 'edx'
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. 3.2: Изменение файла lab5-1-1.asm

Сохраним изменения и создадим исполняемый файл (Рис. 3.3):

```

nsandryushin@nsandryushin:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o

```

Рис. 3.3: Создание исполняемого файла

Запустим его и проверим, что всё работает (Рис. 3.4):

```

nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Андрюшин Никита Сергеевич
Андрюшин Никита Сергеевич

```

Рис. 3.4: Проверка работы программы

Теперь создадим с помощью F5 копию файла lab5-2.asm (Рис. 3.5):

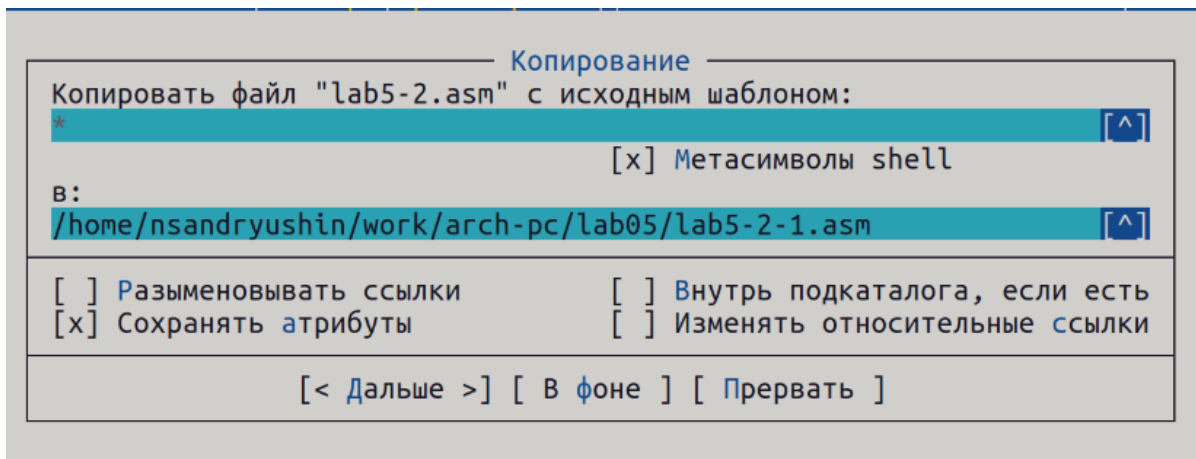


Рис. 3.5: Создание копии файла lab5-2.asm

теперь сделаем так, чтобы этот код также выводил тот текст, что получит на ввод. Для этого перед последней строкой добавим строчку, которая записывает в еах адрес buf1, а также строчку, которая вызывает подпрограмму sprintLF (Рис. 3.6):

```

;-----
; 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 sprintLF ; 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 sprintLF ; вызов подпрограммы печати сообщения
call quit ; вызов подпрограммы завершения

```

Рис. 3.6: Изменение файла lab5-2-1.asm

Теперь создадим исполняемый файл (Рис. 3.7):

```
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
```

Рис. 3.7: Создание исполняемого файла

Теперь запустим программу и убедимся, что она работает (Рис. 3.8):

```
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
nsandryushin@nsandryushin:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку:
Андрюшин Никита Сергеевич
Андрюшин Никита Сергеевич
```

Рис. 3.8: Проверка работы программы

4 Выводы

В результате выполнения работы были получены навыки работы с Midnight commander, а также навыки написания простых программ ввода-вывода на языке ассемблера