

Лабораторная работа №5

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

Казначеев Сергей Ильич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	14

Список иллюстраций

2.1	scr1	6
2.2	scr2	7
2.3	scr3	7
2.4	scr4	8
2.5	scr5	8
2.6	scr6	9
2.7	scr7	9
2.8	scr8	9
2.9	scr9	10
2.10	scr10	10
2.11	scr11	11
2.12	scr12	11
2.13	scr13	12
2.14	scr13	12
2.15	scr14	12
2.16	scr15	13
2.17	scr16	13

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1 Я открыл Midnight commander с помощью команды mc

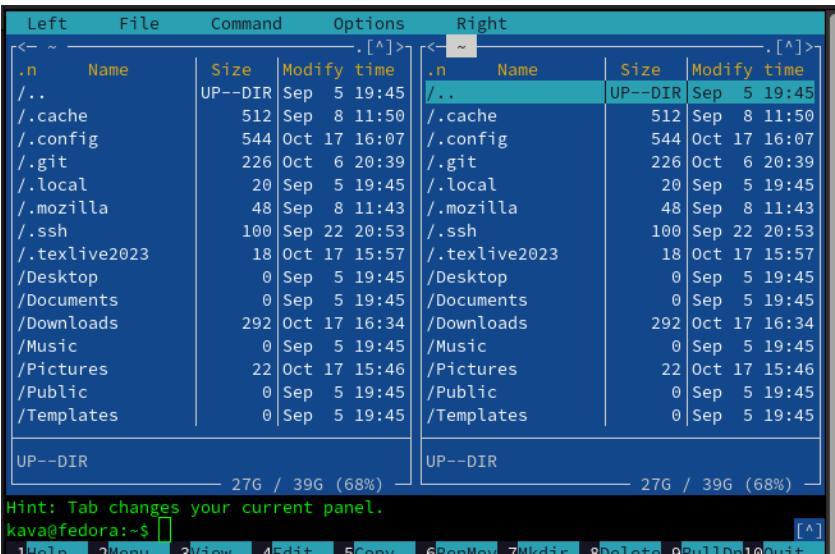


Рис. 2.1: scr1

С помощью стрелок и клавиши Enter я перехожу в каталог ~/work/arch-pc и создаю папку lab05

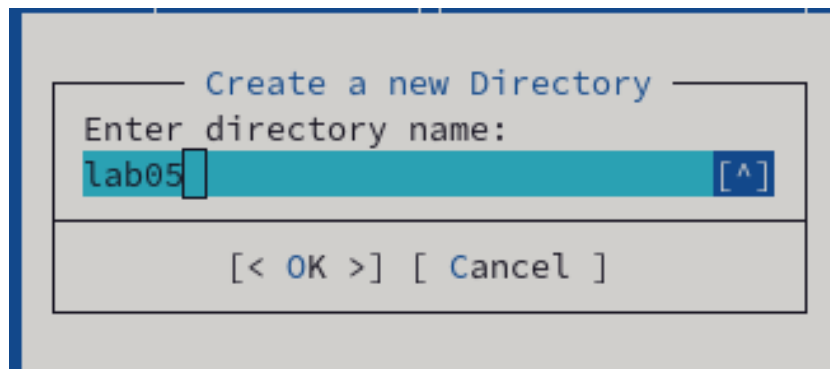


Рис. 2.2: scr2

Переходим в нее и создаем файл с помощью команды `touch lab5-1.asm`

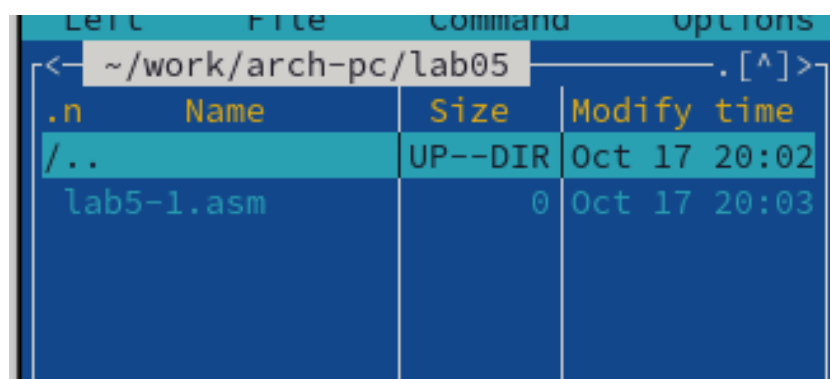


Рис. 2.3: scr3

С помощью клавиши F4 откроем только что созданный файл

```
lab5-1.asm      [-M--] 20 L:[ 1+ 2 3/ 35] *(208 /2431b) 0045 0x02D [*][X]
;-----
; 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 ; 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
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 2.4: scr4

Далее редактируем файл

```
/home/kava/work/arch-pc/lab05/lab5-1.asm      1448/2432      59%
;-----
; 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 ; 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
1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format10Quit
```

Рис. 2.5: scr5

Теперь скомпилируем его


```

Hint: Shell commands will not work when you are on a non-local file system.
kava@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDn10Quit

```

Рис. 2.6: scr6

и соберем

```

kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDn10Quit

```

проверяем что файлы создались и то что можно ввести ФИО

..	Name	Size	Modify Time
./..		UP--DIR	Oct 17 20:02
*lab5-1		8744	Oct 17 20:11
	lab5-1.asm	2432	Oct 17 20:07
	lab5-1.o	752	Oct 17 20:10

Рис. 2.7: scr7

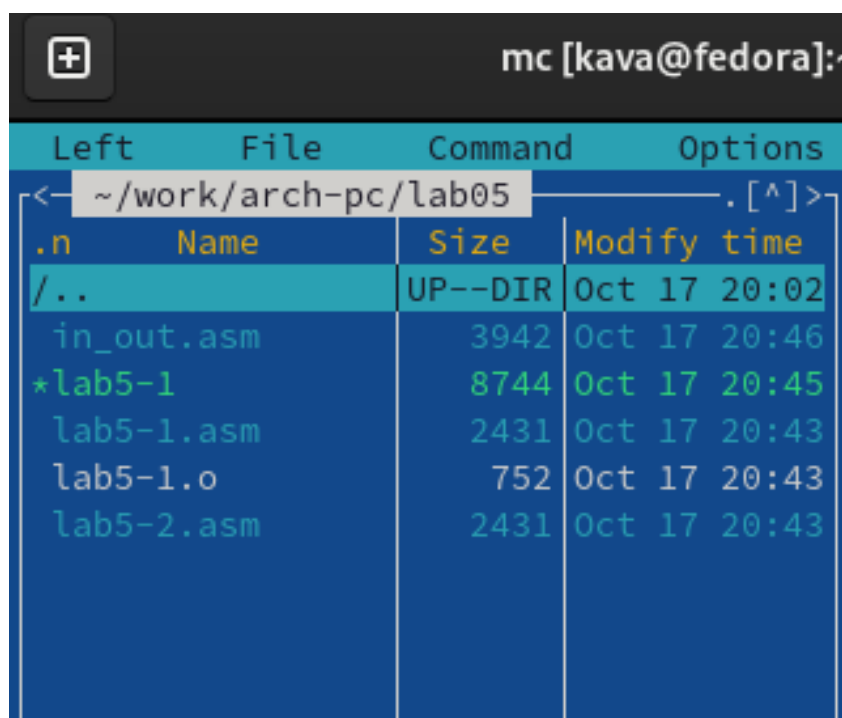
```

mc [kava@fedora]:~/work/arch-pc/lab05
kava@fedora:~/work/arch-pc/lab05$ mc
kava@fedora:~/work/arch-pc/lab05$ touch lab5-1.asm
kava@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1
kava@fedora:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Казаначеев Сергей Ильич

```

Рис. 2.8: scr8

После скачиваем файл и копируем в рабочую папку

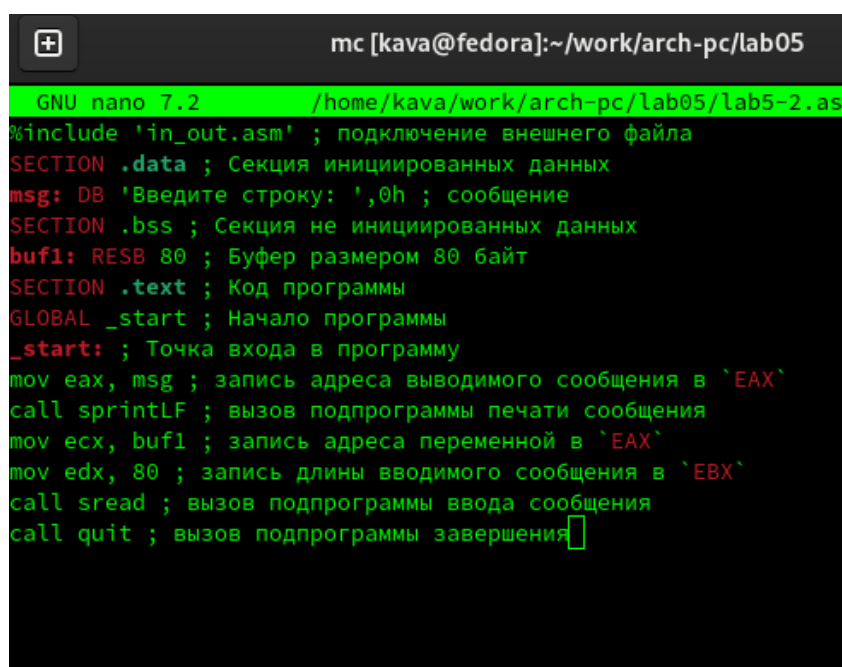


mc [kava@fedora]:~

Left	File	Command	Options
<-	~/work/arch-pc/lab05		. [^]>
.n	Name	Size	Modify time
	/..	UP--DIR	Oct 17 20:02
	in_out.asm	3942	Oct 17 20:46
*	lab5-1	8744	Oct 17 20:45
	lab5-1.asm	2431	Oct 17 20:43
	lab5-1.o	752	Oct 17 20:43
	lab5-2.asm	2431	Oct 17 20:43

Рис. 2.9: scr9

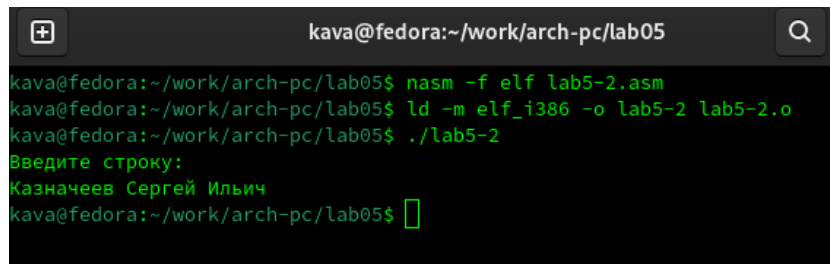
После я скопировал файлы и написал следующий код



```
GNU nano 7.2 /home/kava/work/arch-pc/lab05/lab5-2.asm
#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. 2.10: scr10

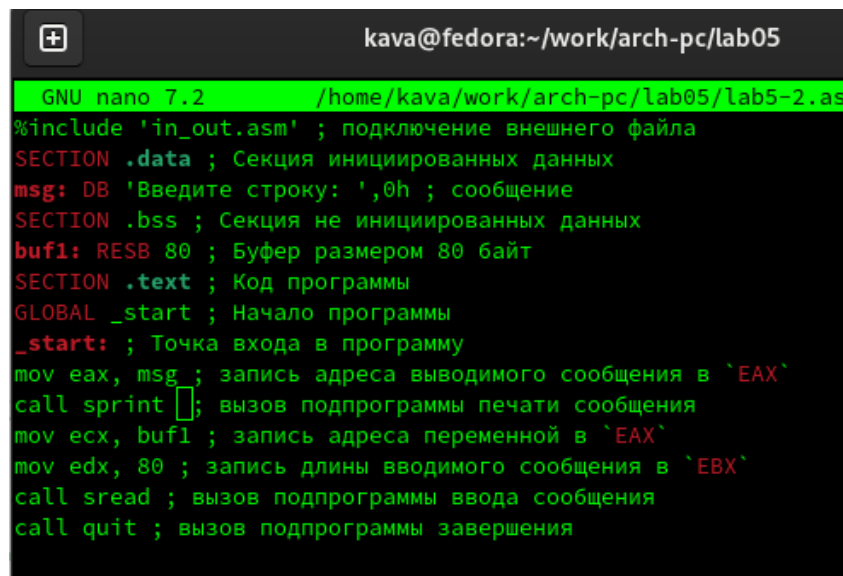
Затем я создал исполняемый файл с помощью `nasm` и `ld`



```
kava@fedora:~/work/arch-pc/lab05
kava@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
kava@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Казначеев Сергей Ильич
kava@fedora:~/work/arch-pc/lab05$
```

Рис. 2.11: scr11

Далее поменяем команду `sprintf` использовать просто команду `sprint`



```
GNU nano 7.2 /home/kava/work/arch-pc/lab05/lab5-2.as
%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. 2.12: scr12

Точно также соберем файл и запустим его

```
kava@fedora:~/work/arch-pc/lab05 — ./lab5-2
kava@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab
kava@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Казначеев Сергей Ильич
kava@fedora:~/work/arch-pc/lab05$ mc
kava@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Казначеев Сергей Ильич
kava@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab
kava@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Казанчеев Сергей Ильич
```

Рис. 2.13: scr13

Задания для самостоятельной работы

Создадим с помощью F6 копию файла lab5-1.asm



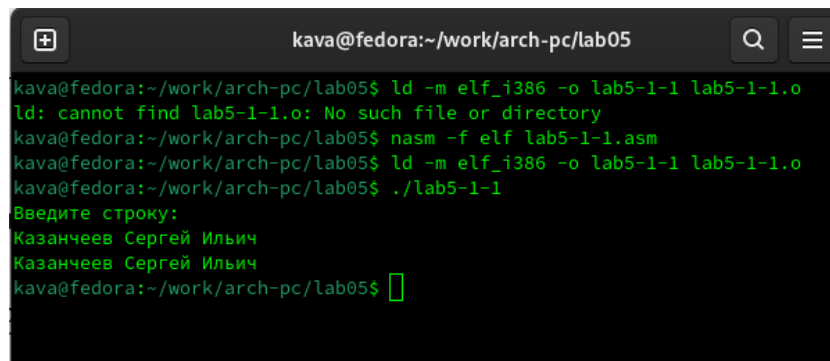
Рис. 2.14: scr13

Меняем файл так чтобы он выводил тот текст который получил на ввод

```
;----- 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 ; Файловый дескриптор 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 ebx, 4 ; Системный вызов для записи (sys_write)
mov ebx, 1 ; Файловый дескриптор 1 - стандартный вывод
mov ecx, buf1 ; Адрес строки 'buf1' в 'ecx'
mov edx, 80 ; Размер строки 'buf1' в 'edx'
int 80h
;----- 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. 2.15: scr14

Сохраняем изменения и запускаем



```
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
ld: cannot find lab5-1-1.o: No such file or directory
kava@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
kava@fedora:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Казанчев Сергей Ильич
Казанчев Сергей Ильич
kava@fedora:~/work/arch-pc/lab05$
```


Рис. 2.16: scr15

Создаем файл с помощью F5 копию файла



Рис. 2.17: scr16

Опять создадим файл и проверим его на корректность работы



```
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
ld: cannot find lab5-1-1.o: No such file or directory
kava@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
kava@fedora:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Казанчев Сергей Ильич
Казанчев Сергей Ильич
kava@fedora:~/work/arch-pc/lab05$ mc
kava@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
kava@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1-1 lab5-2-1-1.o
kava@fedora:~/work/arch-pc/lab05$ ./lab5-2-1-1
Введите строку:
Казанчев Сергей Ильич
Казанчев Сергей Ильич
```

3 Выводы

После выполнения лабораторной работы приобрел практические навыки работы Midnight Commander и освоил инструкции языка ассемблера mov и int #
Список литературы{.unnumbered}