

## **Отчет по лабораторной работе №5**

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

Маркеш Виейра Нанке Грасимилде

## **Содержание**

- 1. Цель работы.....4**
- 2. Задание.....5**
- 3. Теоретическое введение.....6**
- 4. Выполнение лабораторной работы.....8**
  - 4.1 Основы работы с mc.....8**
  - 4.2 Структура программы на языке ассемблера NASM.....10**
  - 4.3 Подключение внешнего файла.....12**
  - 4.4 Выполнение заданий для самостоятельной работы.....15**
- 5. Выводы.....21**
- 6. Список литературы.....22**

## **Список иллюстраций**

- 4.1 Открытый mc.....8**
- 4.2 Перемещение между директориями.....9**
- 4.3 Создание каталога.....9**
- 4.4 Создание файла.....10**
- 4.5 Открытие файла для редактирования.....10**
- 4.6 Редактирование файла.....11**
- 4.7 Компиляция файла и передача на обработку компоновщику.....11**
- 4.8 Исполнение файла.....12**
- 4.9 Скачанный файл.....12**
- 4.10 Копирование файла.....12**
- 4.11 Копирование файла.....13**
- 4.12 Редактирование файла.....13**
- 4.13 Компиляция файла и передача на обработку компоновщику.....14**
- 4.14 Редактирование файла.....14**
- 4.15 Компиляция файла и передача на обработку компоновщику.....15**
- 4.16 Копирование файла.....15**
- 4.17 Редактирование файла.....16**
- 4.18 Компиляция файла и передача на обработку компоновщику.....17**
- 4.19 Копирование файла.....18**
- 4.20 Редактирую файл.....19**
- 4.21 Компиляция файла и передача на обработку компоновщику.....19**

## **1 Цель работы**

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

## **2 Задание**

1. Основы работы с Midnight Commander
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 позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. те является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной. Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION\_text), секция инициализированных (известных во время компиляции) данных (SECTION.data) и секция ненициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION.bss). Для объявления инициализированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны 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 (четыре-рёвное слово);
- DT (define ten bytes) — определяет переменную размером в 10 байт. Директивы используют mov\_dst,src

Здесь operand dst — приёмник, a src — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера int предназначена для вызова прерывания с указанным номером. int n

Здесь n — номер прерывания, принадлежащий диапазону 0–255. При программировании в Linux с использованием вызовов ядра sys\_calls n=80h (принято задавать в шестнадцатеричной системе счисления).

## 4 Выполнение лабораторной работы

### 4.1 Основы работы с mc

Открываю Midnight Commander, введя в терминал mc (рис. 4.1).

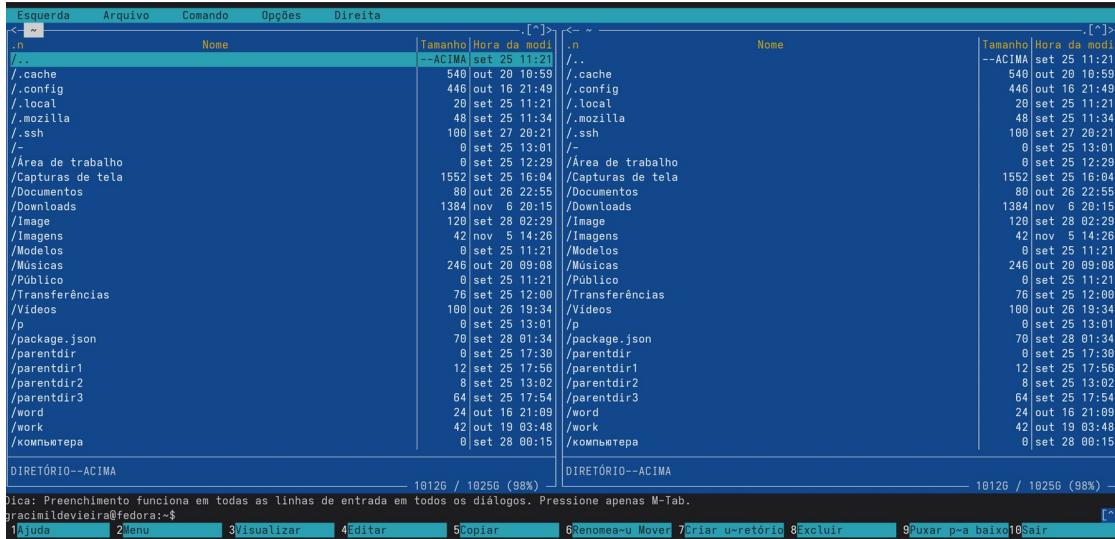


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2023-2024/Архитектура Компьютера/arch-rcs, используя файловый менеджер mc (рис. 4.2)

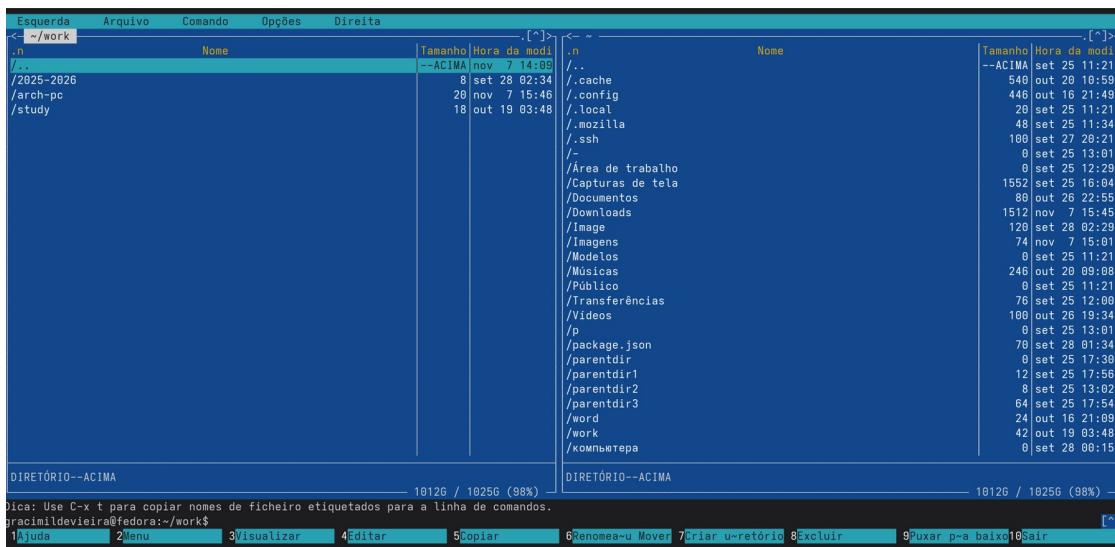


Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 и перехожу в него (рис. 4.3).

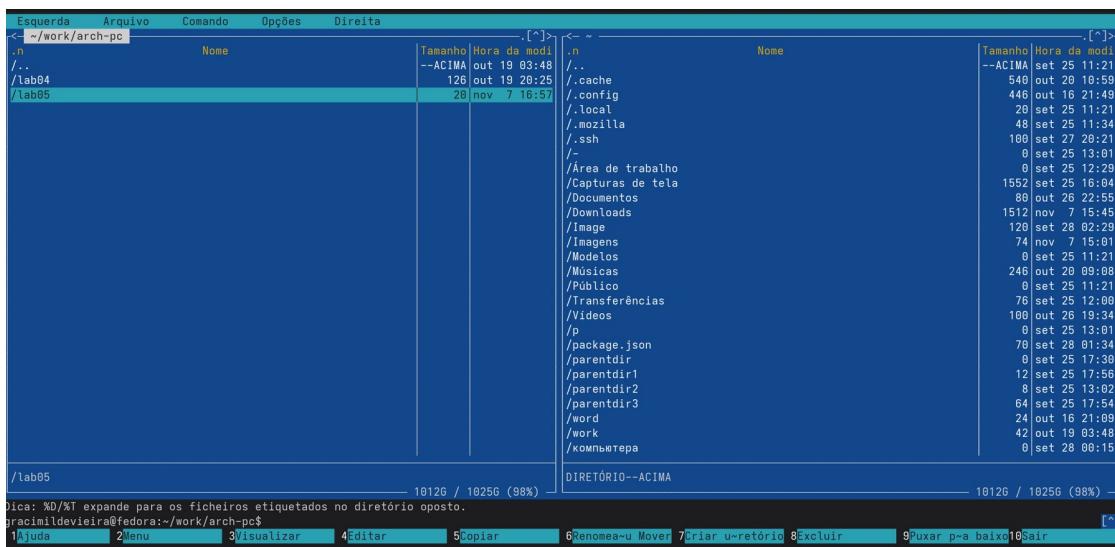


Рис. 4.3: Создание каталога

В строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в котором буду работать (рис. 4.4).

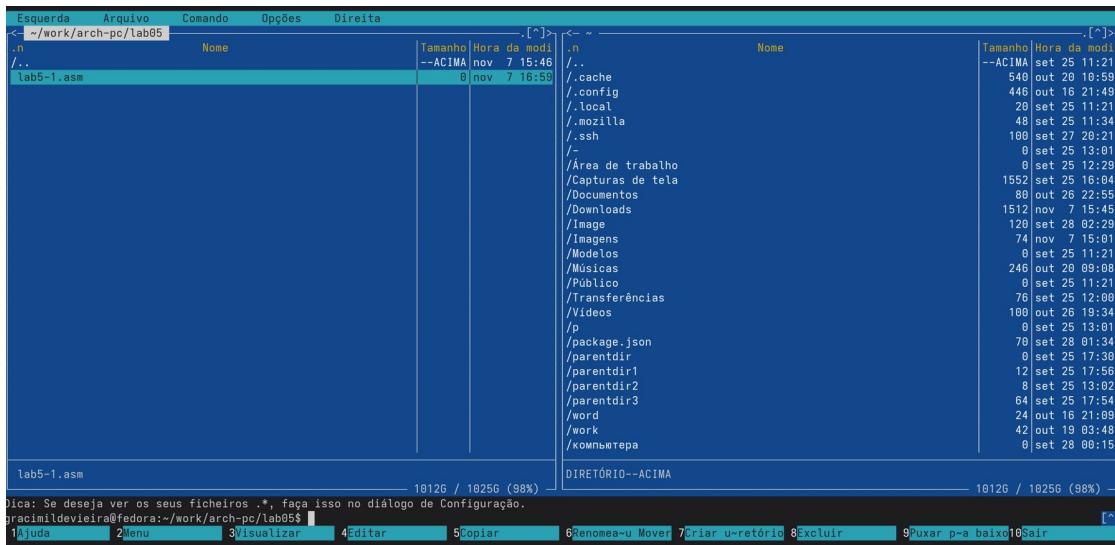


Рис. 4.4: Создание файла

## 4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе (рис. 4.5).

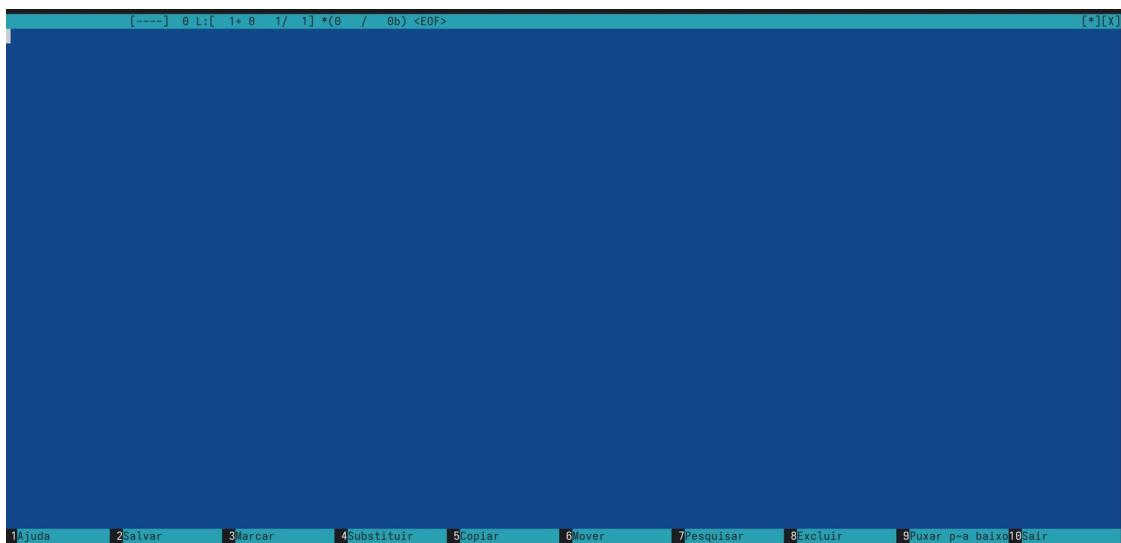


Рис. 4.5: Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя (рис. 4.6). Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter). С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы

A screenshot of a terminal window showing the assembly code for 'lab5-1.asm'. The code includes a message prompt and buffer definitions. At the top, there are status indicators: '/home/gracimildevieira/work/arch-pc/lab85/lab5-1.asm' (file path), '350/350' (line count), and '100%' (percentage). Below the code area, there is a menu bar with options: Ajuda, Desenrolar, Sair, Hexadecimal, Ir para, Pesquisar, Analisar, Formatar, and Sair. The menu bar has a blue background with white text.

Рис. 4.6: Редактирование файла

Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-1.asm. Создался объектный файл lab5-1.o. Выполняю компоновку объектного файла с помощью команды ld -m elf\_1386 -o lab5-1 lab5-1.o (рис. 4.7). Создался исполняемый файл lab5-1.

```
nasm -f elf lab5-1.asm  
ld -m elf_i386 -o lab5-1 lab5-1.o
```

*Рис. 4.7: Компиляция файла и передача на обработку компоновщику*

Запускаю исполняемый файл. Программа выводит строку “Ведите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 4.8).

```
gracimildevieira@fedora:~/work/arch-pc/lab05$ ./lab5-1  
Введите строку:  
Маркеш Виейра Нанке Грасимилде  
gracimildevieira@fedora:~/work/arch-pc/lab05$
```

*Рис. 4.8: Исполнение файла*

#### 4.3 Подключение внешнего файла

Скачиваю файл in\_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 4.9).



*Рис. 4.9: Скачанный файл*

С помощью функциональной клавиши F5 копирую файл in\_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 4.10).

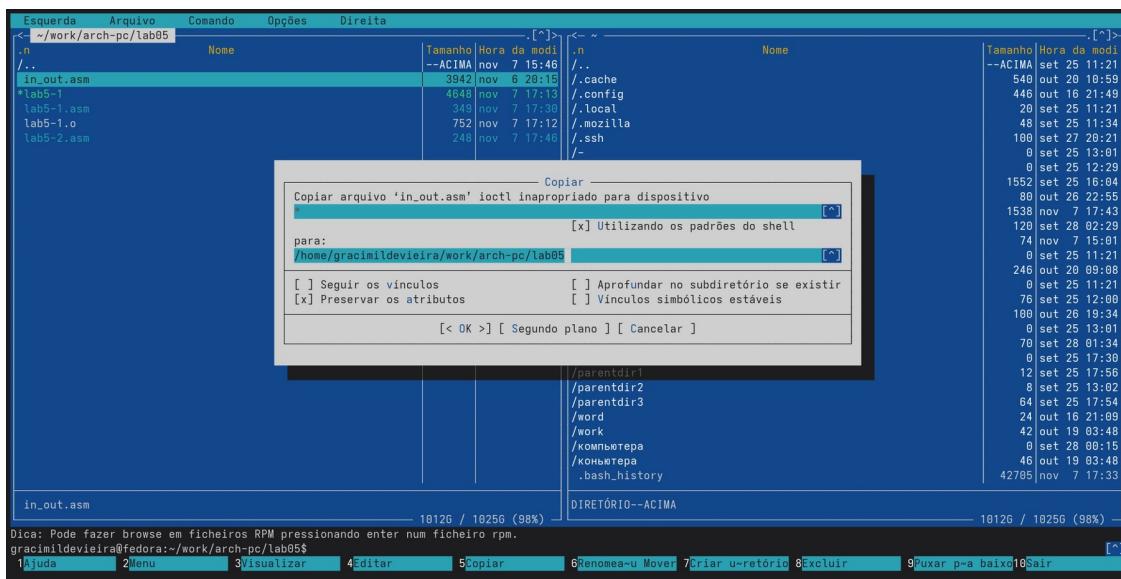


Рис. 4.10: Копирование файла

С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне тс прописываю имя для копии файла (рис. 4.11).

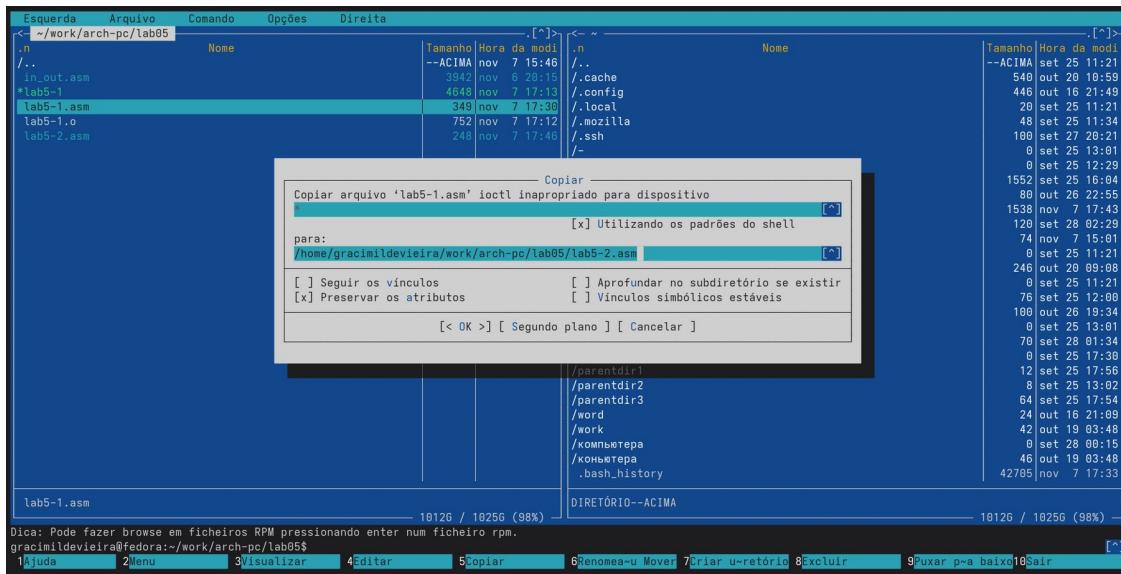


Рис. 4.11: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенным редакторе (рис. 4.12), чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm.



```
/home/gracimildevieira/work/arch-pc/lab05/lab5-2.asm
%include 'in_out.asm'

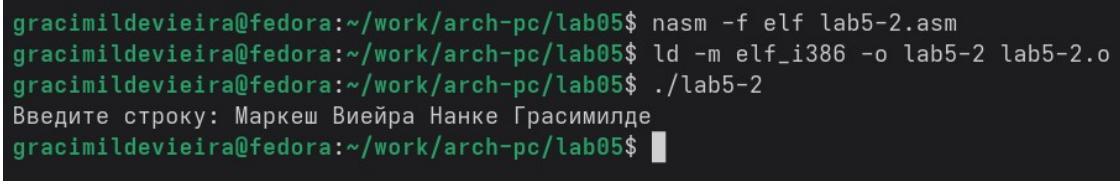
SECTION .data
    msg: DB 'Введите строку: ',0h

SECTION .bss
    buf1: RESB 80

SECTION .text
GLOBAL _start
_start:
    mov eax, msg
    call sprint
    mov ecx, buf1
    mov edx, 80
    call sread
    call quit
```

Рис. 4.12: Редактирование файла

Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-2.asm. Создался объектный файл lab5-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf\_i386 -o lab5-2 lab5-2.o Создался исполняемый файл lab5-2. Запускаю исполняемый файл (рис. 4.13).



```
gracimildevieira@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
gracimildevieira@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Маркеш Виейра Нанке Грасимилде
gracimildevieira@fedora:~/work/arch-pc/lab05$ █
```

Рис. 4.13: Компиляция файла и передача на обработку компоновщику

Открываю файл lab5-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 4.14).

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,msg
    call sprint
    mov ecx,buf1
    mov edx,80
    call sread
    call quit
```

Рис. 4.14: Редактирование файла

Снова транслирую файл, выполняя компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.15).

```
gracimildevieira@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
gracimildevieira@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Маркеш Виейра Нанке Грасимилде
gracimildevieira@fedora:~/work/arch-pc/lab05$ █
```

Рис. 4.15: Компиляция файла и передача на обработку компоновщику

Разница между первым исполняемым файлом и вторым в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

#### 4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 4.16).

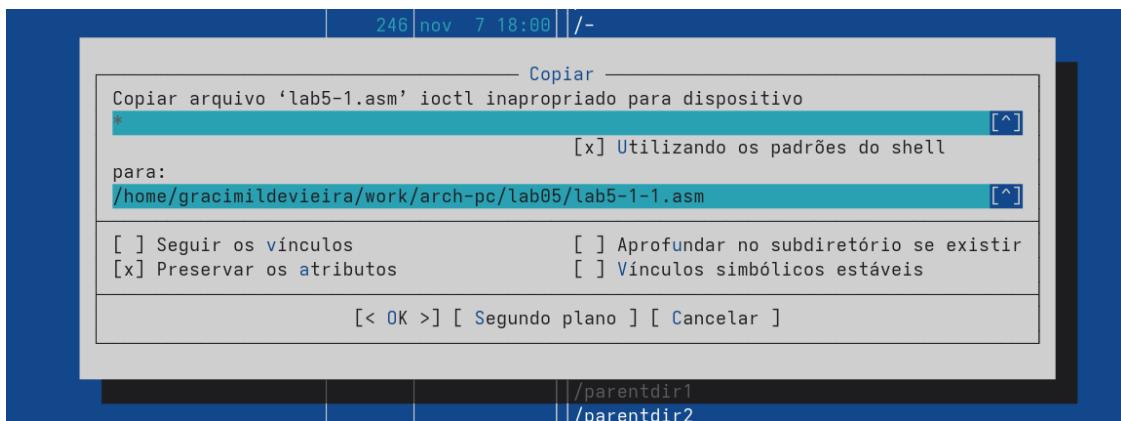


Рис. 4.16: Копирование файла

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. 4.17).

```

SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,4
    mov ebx,1
    mov ecx,msg
    mov edx,msgLen
    int 80h
    mov eax, 3
    mov ebx, 0
    mov ecx, buf1
    mov edx, 80
    int 80h
    mov eax,4
    mov ebx,1
    mov ecx,buf1
    mov edx,buf1
    int 80h
    mov eax,1
    mov ebx,0
    int 80h

```

*Рис. 4.17: Редактирование файла*

2. Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 4.18).

```

gracimildevieira@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
gracimildevieira@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
gracimildevieira@fedora:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Маркеш Виейра Нанке Грасимилде
Маркеш Виейра Нанке Грасимилде
gracimildevieira@fedora:~/work/arch-pc/lab05$ █

```

*Рис. 4.18: Компиляция файла и передача на обработку компоновщику*

Код программы из пункта 1:

```
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'

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,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 ввод uniqueую 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 buff1 в ecx
        mov edx,buf1 ; Размер строки buff1
        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
```

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 4.19).

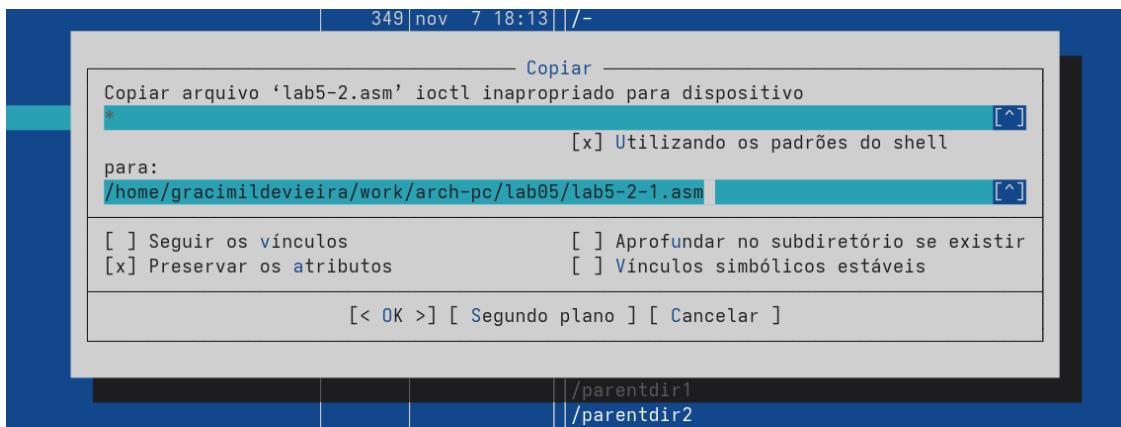


Рис. 4.19: Копирование файла

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. 4.20).

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,msg
    call sprint
    mov ecx,buf1
    mov edx,80
    call sread
    mov eax,4
    mov ebx,1
    mov ecx,buf1
    int 80h
    call quit
```

Рис. 4.20: Редактирую файл

4. Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без

переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 4.21).

```
gracimildevieira@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
gracimildevieira@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
gracimildevieira@fedora:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку: Маркеш Виейра Нанке Грасимилде
Маркеш Виейра Нанке Грасимилде
gracimildevieira@fedora:~/work/arch-pc/lab05$
```

Рис. 4.21: Компиляция файла и передача на обработку компоновщику

Код программы из пункта 3:

%include 'in\_out.asm'

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 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 еса

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.

## **6 Список литературы**

1. Лабораторная работа №5