Лабораторная работа №5

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

Бочаров Андрей

Содержание

# 1 Цель работы

Целью работы является приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера mov и int.

# 2 Выполнение лабораторной работы

Открыл файловый менеджер mc(рис. 1).

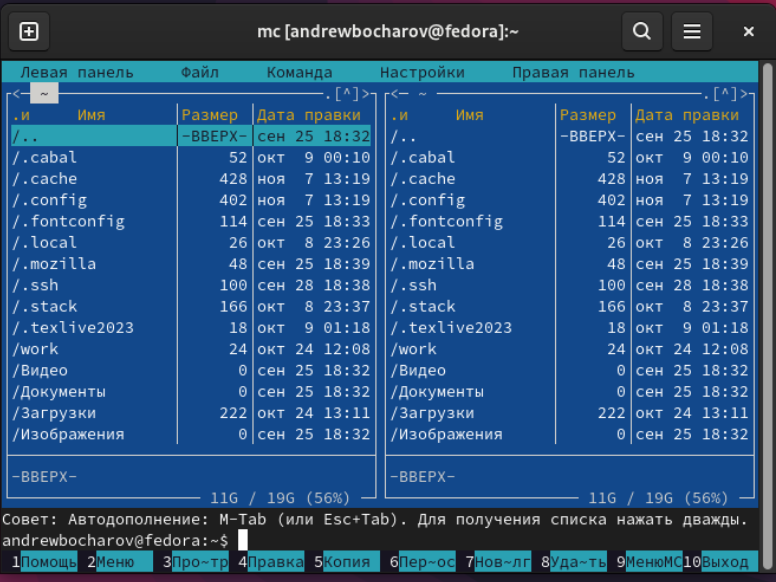


Рис. 1: Интерфейс mc

Перешел в папку ~/work/arch-pc при помощи стрелочек и кнопки ввода (рис. 2).

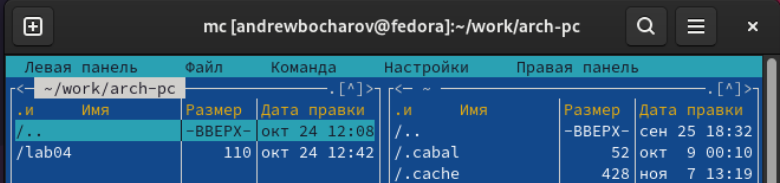


Рис. 2: Папка ~/work/arch-pc

Создал папку lab05 (рис. 3).

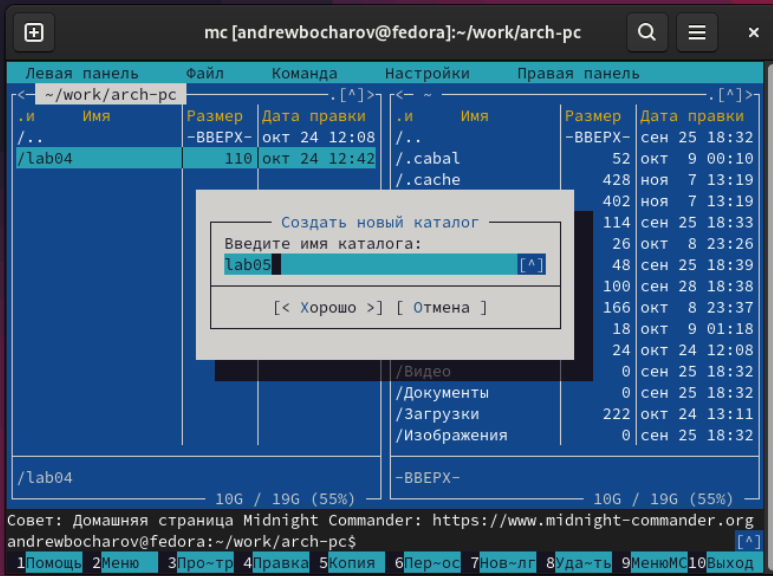


Рис. 3: Создание папки lab05

Перешел в новую папку и ввел команду для создания файла lab5-1.asm (рис. 4).

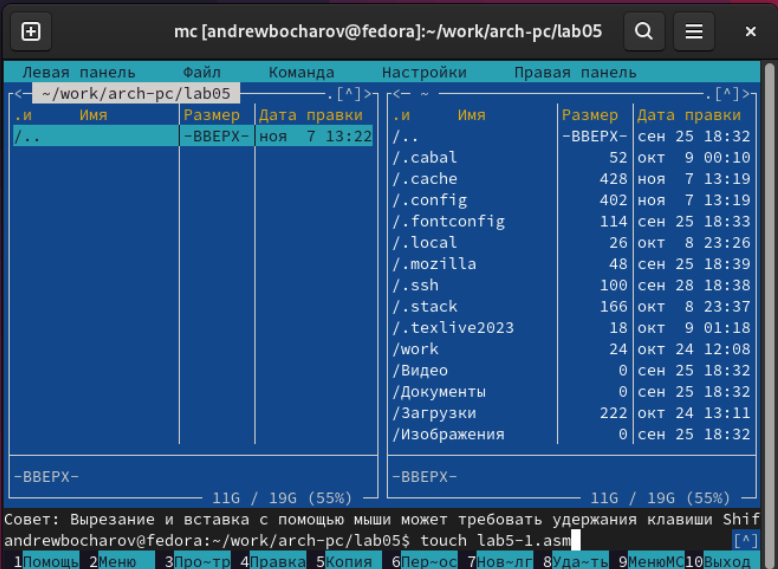


Рис. 4: Пустая папка и введенная команда

Открыл редактор mcedit клавишей f4 (рис. 5).

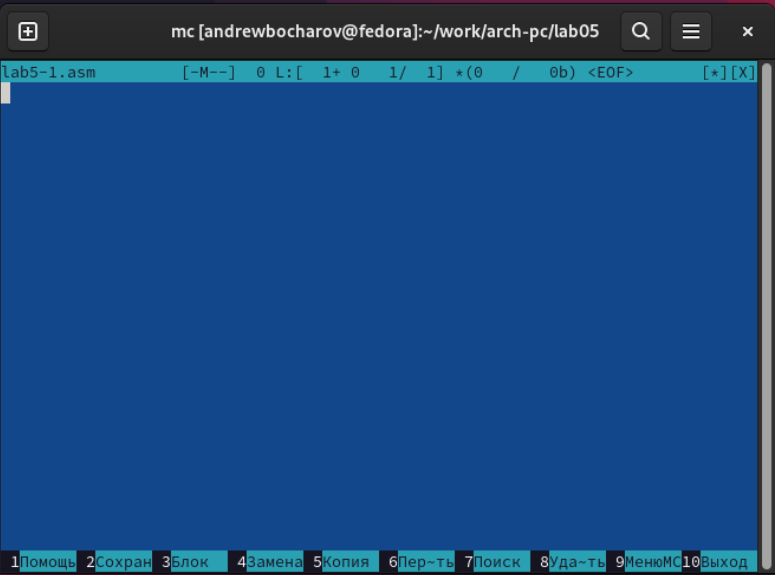


Рис. 5: Редактор mcedit

Переписал код для вывода сообщения и ввода строки с клавиатуры (рис. 6).

Листинг кода:

; Программа вывода сообщения на экран и ввода строки с клавиатуры  
;---------------------------------------------------------------------  
SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку: ',10 ; сообщение  
  
msgLen: EQU $-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  
  
;------------ Cистемный 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  
  
;------------ 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  
 mov ebx, 0  
 mov ecx, buf1 ; запись адреса переменной в `EAX`  
 mov edx, 80  
 int 80h ; запись длины вводимого сообщения в `EBX`  
  
;------------ 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 ядр

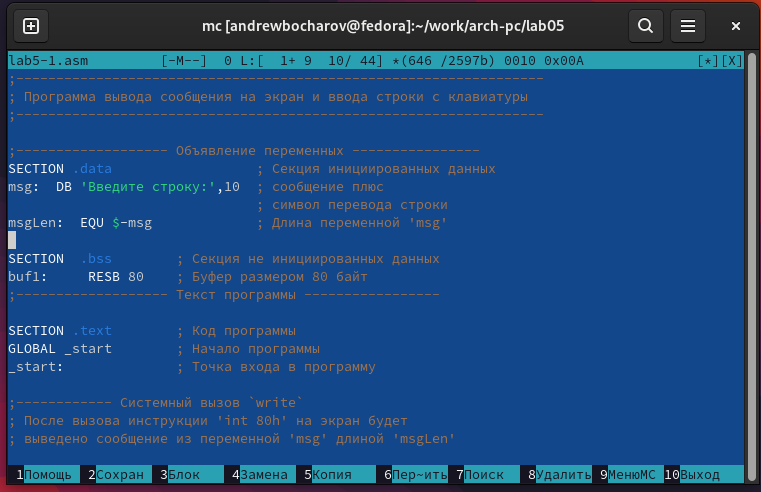


Рис. 6: Листинг кода

Сохранил и проверил содержимое фалйа клавишей f3 (рис. 7).

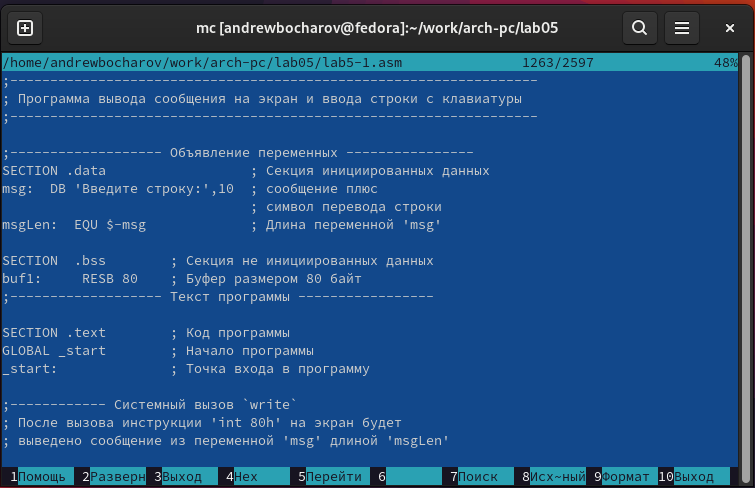


Рис. 7: Содержимое файла

Создал объектный файл lab5-1.o при помощи ассемблера nasm. При помощи объктново компоновщика ld сделал исполняемый файл lab5-1 (рис. 8 и рис. 9).

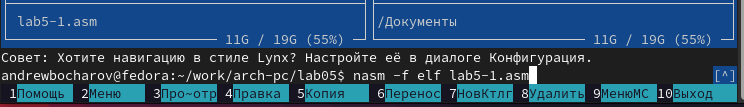


Рис. 8: Трансляция файла

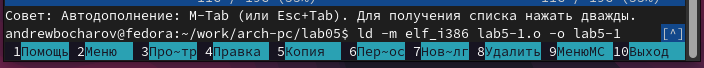


Рис. 9: Компоновка файла

Проверил наличие файла (рис. 10).

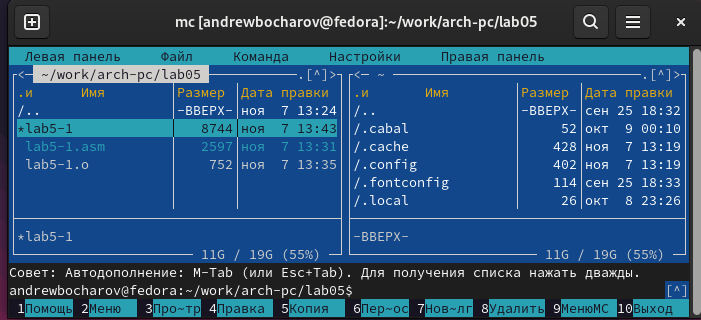


Рис. 10: Компоновка файла

Запустил исполняемый файл, он работает (рис. 11).

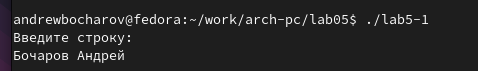


Рис. 11: Файл работает

Скачал файл in\_out.asm (рис. 12).

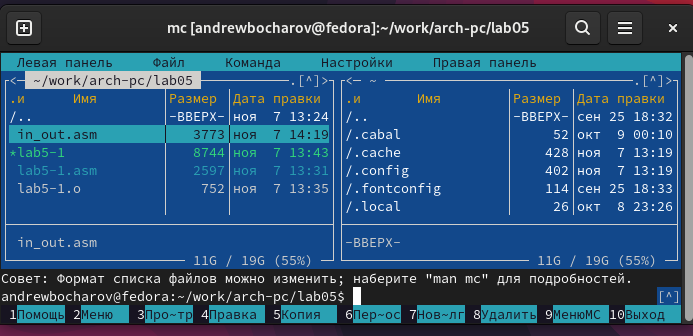


Рис. 12: Наличие файла

Сделал копию файла lab5-1.asm при помощи клавиши f5 (рис. 13).

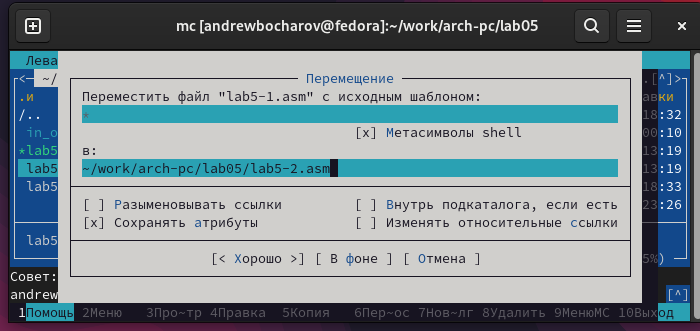


Рис. 13: Копирование файла

Переписал код в файле lab5-2.asm (рис. 14).

Листинг кода:

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

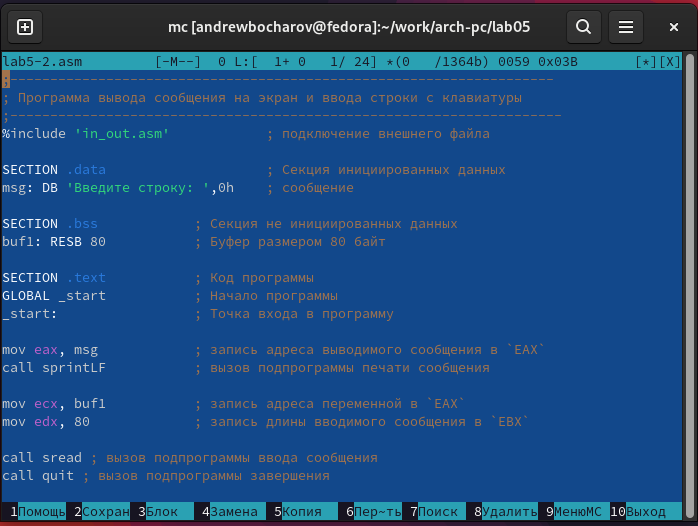


Рис. 14: Копирование файла

Сохранил изменения в файле lab5-2.asm (рис. 15).

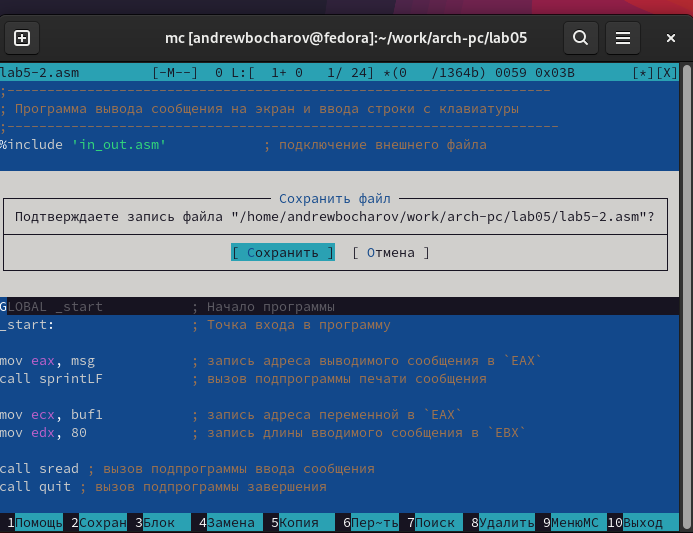


Рис. 15: Сохранение изменений

Создал объектный файл lab5-1.o при помощи ассемблера nasm. При помощи объктново компоновщика ld сделал исполняемый файл lab5-2. (рис. 16).

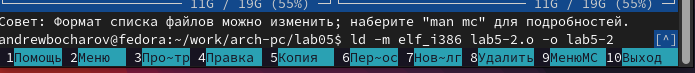


Рис. 16: Компоновка файла

Проверил работу исполняемого файла lab5-2 (рис. 17).

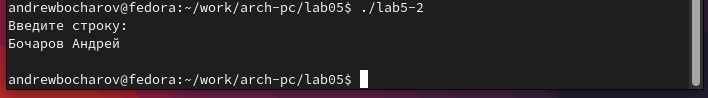


Рис. 17: Работает файла

Заменил call stringLF на call string (рис. 18).

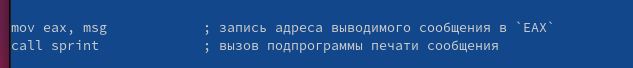


Рис. 18: Листинг кода

Посмотерл на результат работы, команда call stringLF после вывода строки выводит символ переноса строки, поэтому после изменения команды, Фамилия и имя остались на том же уровне, что и “Введите данные:” (рис. 19).

Рис. 19: Листинг кода

Рис. 19: Листинг кода

# 3 Выполнение самостоятельной работы

Создал копию файла lab5-1.asm и внес в него изменения, что бы после ввода данных, они выводилилсь на экран. mov eax,4

Листинг кода:

; Программа вывода сообщения на экран и ввода строки с клавиатуры  
;---------------------------------------------------------------------  
SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку: ',10 ; сообщение  
  
msgLen: EQU $-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  
  
;------------ Cистемный 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  
  
;------------ 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  
 mov ebx, 0  
 mov ecx, buf1 ; запись адреса переменной в `EAX`  
 mov edx, 80  
 int 80h ; запись длины вводимого сообщения в `EBX`  
  
 mov eax, 4 ; Системный вызов для записи (sys\_write)  
 mov ebx, 1 ; Описатель файла 1 - стандартный вывод  
 mov ecx, buf1 ; Адрес буфера  
 mov edx, 80 ; Размер буфера  
 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 ядр

Сделал трансляцию, компоновку и запустил и проверил код. Код работает (рис. 20).

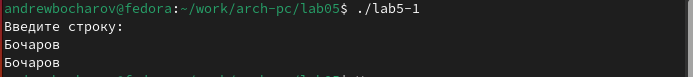


Рис. 20: Результат выполнения

Создал копию файла lab5-2.asm и внес в него изменения, что бы после ввода данных, они выводилилсь на экран с использованием команд из in\_out.asm. Листинг кода:

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

Сделал трансляцию, компоновку и запустил и проверил код. Код работает (рис. 21).

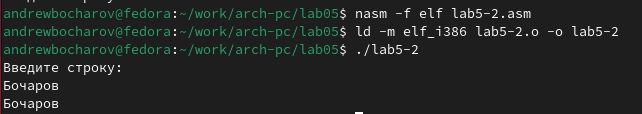


Рис. 21: Результат выполнения

# 4 Выводы

Выполнив данную лабораторную работу я обрел теоретические и практические знания в использовании Midnight Commander. Освоенил инструкции языка ассемблера mov и int.