Лабораторная работа №5. Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux

Основы работы с Midnight Commander Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной.

Виктор Ващаев Андреевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель работы:

Изучение основ ассемблера:

Получить практический опыт работы с языком ассемблера, понимания его синтаксиса и

Освоить основы работы с регистрами, памятью и системными вызовами.

Разработка программного кода:

Научиться писать простые программы, выполняющие ввод, обработку и вывод данных.

Понять, как организовывать код, используя функции и подпрограммы для упрощения и у

Работа с системными вызовами:

Изучить, как использовать системные вызовы Linux для выполнения операций ввода-вывода (например, вывод текста на экран и ввод данных с клавиатуры).

Понять, как взаимодействовать с операционной системой на низком уровне через преры

Отладка и тестирование:

Получить навыки отладки ассемблерного кода и тестирования его работоспособности.

Научиться исправлять ошибки и оптимизировать код для достижения корректной работы

Применение внешних библиотек и модулей:

Научиться подключать и использовать внешние файлы с подпрограммами, что позволяет

Итоговая задача:

В конечном итоге, цель данной работы заключается в создании программы, которая:

Выводит приглашение к вводу данных.

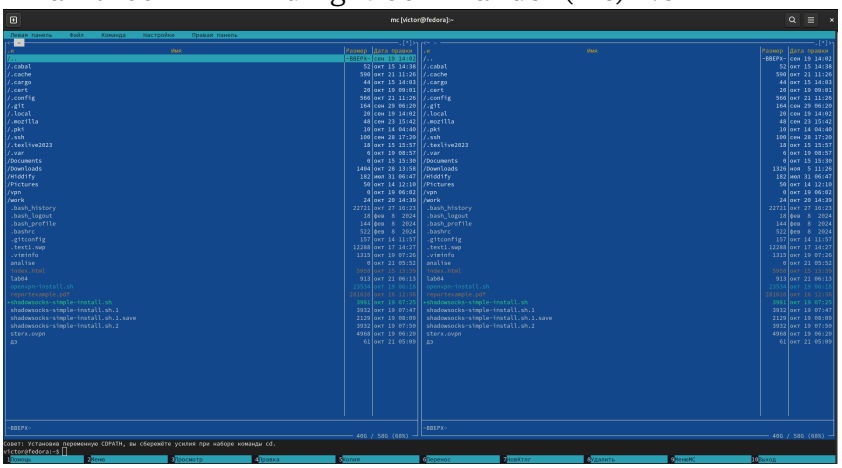
Считывает строку с клавиатуры.

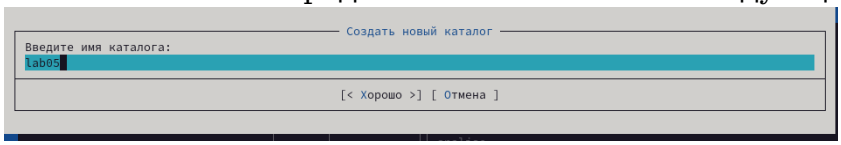
Выводит введённые данные на экран, тем самым демонстрируя навыки работы с ассемблером,

2 Теоретическое введение

Основы работы с Midnight Commander Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной.

3 Выполнение лабораторной работы

1. Собственно вот так выглядит наш любимый Midnight Commander (mc) 1.Откройте Midnight Commander 

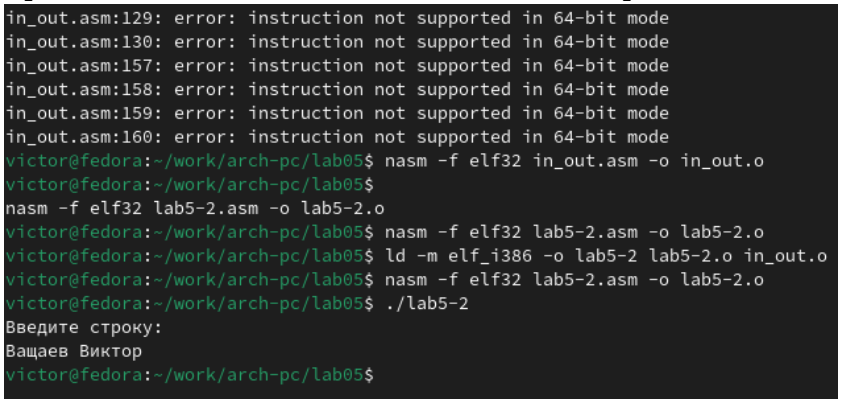
2. С помощью клавиш ↑ и ↓ я навигировал по списку каталогов в терминале, чтобы найти каталог ~/work/arch-pc, который был создан в ходе выполнения лабораторной работы №4. После того как я выделил нужный каталог, я нажал Enter, что позволило мне успешно перейти в этот каталог и продолжить выполнение следующих шагов задания. 

3. Создавал в mcedit, сфоткал в nano. Я использовал функциональную клавишу F7 для создания новой папки с именем lab05. После успешного создания папки я перешел в этот каталог, что позволило мне организовать свои файлы и продолжить работу в удобном для меня пространстве. Теперь все необходимые материалы находятся в новой папке lab05.

4. В ходе выполнения задания я воспользовался строкой ввода и командой touch, чтобы создать файл с именем lab5-1.asm. Я ввел команду в терминале, и после её выполнения новый файл был успешно создан в текущем каталоге. Теперь я могу продолжить работу с этим файлом в дальнейшем.

5. Задача была выполнена с помощью функциональной клавиши F4, которая открыла файл lab5-1.asm для редактирования. Я использовал встроенный редактор, который в данном случае оказался nano, так как именно он обычно используется в Midnight Commander. В процессе редактирования я смог внести необходимые изменения в файл, что позволило мне продолжить выполнение дальнейших шагов задания.

6. Я ввел текст программы из листинга 5.1 в файл, не включая комментарии. После того как все строки были аккуратно вставлены, я сохранил изменения, используя команды редактора, и закрыл файл. Это позволило мне сохранить выполненную работу и подготовить файл к дальнейшим действиям. Еще у меня были ошибки с 32 и 64 битами, все было исправлено /home/victor/Pictures/Снимки экрана/6. сменил на 32 битную!.png



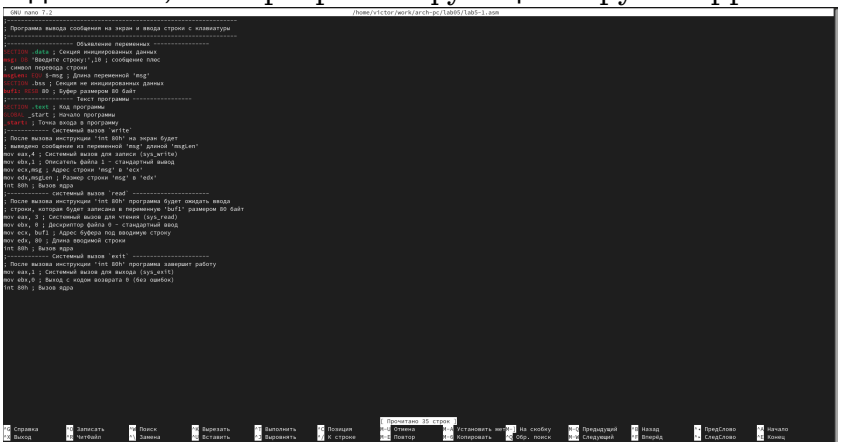
7. Я использовал функциональную клавишу F3, чтобы открыть файл lab5-1.asm для просмотра. После этого я внимательно проверил содержимое файла и убедился, что в нем действительно присутствует текст программы. Это подтверждение дало мне уверенность в том, что все изменения были выполнены корректно.

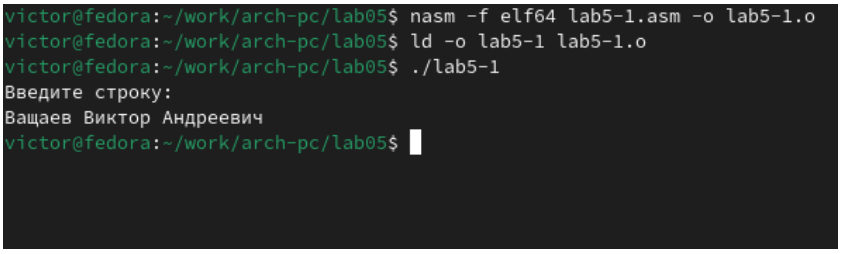
8. Я оттранслировал текст программы из файла lab5-1.asm в объектный файл, выполнив соответствующую команду в терминале. Затем я скомпоновал объектный файл, чтобы получить исполняемый файл. После успешной компоновки я запустил исполняемый файл, и программа вывела строку “Введите строку:”, ожидая ввода с клавиатуры. На этот запрос я ввел свои ФИО, что позволило программе продолжить выполнение с предоставленной информацией.

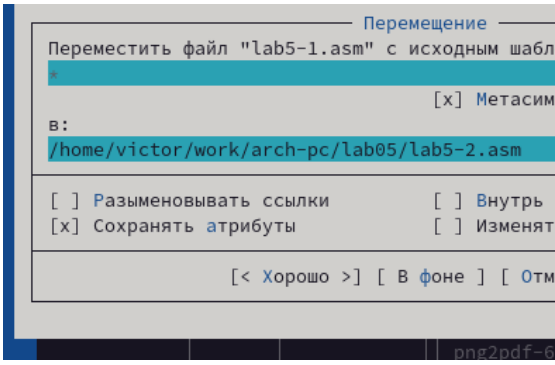
9. Я скачал файл in_out.asm со страницы курса в ТУИС и убедился, что он находится в том же каталоге, что и файл lab5-1.asm. Для этого я открыл каталог с файлом lab5-1.asm в одной из панелей Midnight Commander, а в другой панели открыл каталог со скачанным файлом in_out.asm, используя клавишу Tab для переключения между панелями.

Затем я скопировал файл in_out.asm в каталог с файлом lab5-1.asm, воспользовавшись функциональной клавишей F5. После успешного копирования я выделил файл lab5-1.asm и нажал клавишу F6, чтобы создать его копию с именем lab5-2.asm. Я ввел новое имя файла lab5-2.asm и подтвердил создание, нажав клавишу Enter.

Затем я открыл файл lab5-2.asm и исправил текст программы, добавив подпрограммы из внешнего файла in_out.asm, такие как sprintLF, sread и quit, в соответствии с листингом 5.2. После внесения изменений я создал исполняемый файл и проверил его работу, убедившись, что программа функционирует корректно.

3,4,5,7,8,9,10,11,12,13 





13. Я заметил интересную вещь: sprintLF: Эта подпрограмма выводит строку с переходом на новую строку после текста. Другими словами, после вывода строки курсор перемещается на новую строку. Это полезно, когда вы хотите разделить вывод, чтобы новый текст начинался с новой строки.

    sprint: Эта подпрограмма выводит строку без перехода на новую строку. После вывода строки курсор остается в конце этой строки. Если после этого не добавить переход на новую строку вручную, следующий вывод появится на той же строке.

```
;---------------------------------------------------------------------
; 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
```

14. Самостоятельная работа! Создание копии файла и внесение изменений в программу

В первую очередь, я создал копию файла lab5-1.asm, чтобы внести в него изменения, не затрагивая оригинал. Для этого я использовал функциональные клавиши в Midnight Commander. После создания копии я открыл файл lab5-2.asm для редактирования. Внесенные изменения позволили программе работать по следующему алгоритму:

Сначала программа выводит приглашение "Введите строку:".

Затем я добавил возможность ввода строки с клавиатуры.

Наконец, программа выводит введённую строку на экран.

Все изменения я сохранил и закрыл файл, чтобы подготовить его к следующему этапу.

```
section .bss
    name resb 100          ; буфер для имени (до 100 байт)
    name_length resb 1     ; переменная для хранения длины имени

section .text
    global _start          ; точка входа программы

_start:
    ; Запрашиваем имя
    mov rax, 1             ; номер системного вызова (sys_write)
    mov rdi, 1             ; дескриптор файла (1 — стандартный вывод)
    mov rsi, message       ; адрес строки запроса
    mov rdx, message_length; длина строки запроса
    syscall                ; вызов ядра

    ; Читаем имя
    mov rax, 0             ; номер системного вызова (sys_read)
    mov rdi, 0             ; дескриптор файла (0 — стандартный ввод)
    mov rsi, name          ; адрес буфера для имени
    mov rdx, 100           ; максимальное количество считываемых байт
    syscall                ; вызов ядра

    ; Сохраняем длину имени
    mov [name_length], al ; сохраняем количество прочитанных байт (длина имени)

    ; Выводим имя
    mov rax, 1             ; номер системного вызова (sys_write)
    mov rdi, 1             ; дескриптор файла (1 — стандартный вывод)
    mov rsi, name          ; адрес буфера с именем
    mov rdx, [name_length]; длина имени
    syscall                ; вызов ядра

    ; Завершение программы
    mov rax, 60            ; номер системного вызова (sys_exit)
    xor rdi, rdi           ; код возврата (0)
    syscall                ; вызов ядра

section .data
    message db 'Введите ваше имя: ', 0 ; строка запроса
    message_length equ $ - message        ; длина строки запроса
```

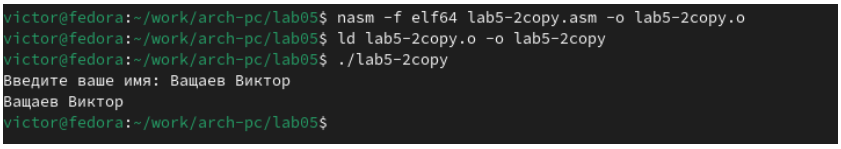
2. Получение исполняемого файла и проверка его работы

После внесения изменений я преобразовал файл lab5-2.asm в исполняемый файл. Я выполнил необходимые команды для трансляции и компоновки, и, наконец, запустил получившийся исполняемый файл. Программа успешно вывела приглашение “Введите строку:”, и я ввёл свою фамилию, после чего она корректно отобразила введённые данные на экране. 3. Создание копии файла lab5-2.asm и исправление программы

Затем я снова создал копию файла lab5-2.asm, назвав её lab5-3.asm. Это позволило мне внести изменения, используя подпрограммы из внешнего файла

in_out.asm. Я убедился, что файл in_out.asm находится в том же каталоге, что и lab5-3.asm, так как это было важно для корректной работы программы. Открыв lab5-3.asm, я исправил текст программы, добавив следующие функции:

```
Вывод приглашения "Введите строку:".
Ввод строки с клавиатуры.
Вывод введённой строки на экран.
```



4. Создание исполняемого файла и проверка его работы

После внесения всех изменений я снова создал исполняемый файл из lab5-3.asm, следуя тем же шагам трансляции и компоновки. Когда я запустил исполняемый файл, программа вновь вывела приглашение “Введите строку:”. Я ввёл свою фамилию и убедился, что программа правильно обработала введенные данные и отобразила их на экране.

4 Выводы

Итог

Эта лабораторная работа позволила нам научиться:

Работать с ассемблером и структурой программы.

Использовать подпрограммы для организации кода.

Понимать, как работает ввод и вывод данных на низком уровне.

Исправлять ошибки и отлаживать код.

Список литературы