Отчёт по лабораторной работе №10

Дисциплина: Архитектура компьютера

Осина Виктория Александровна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ для работы с файлами.

# 2 Задание

1. Выполнение лабораторной работы
2. Выполнение задания для самостоятельной работы

# 3 Теоретическое введение

**Права доступа к файлам**

Права доступа определяют набор действий (чтение (r), запись (r), выполнение (x)), разрешённых для выполнения пользователям системы над файлами, дефис (-) - право не дано. Для каждого файла пользователь может входить в одну из трех групп: владелец (u), член группы владельца (g), все остальные (o). Для каждой из этих групп может быть установлен свой набор прав доступа.Для предоставления прав доступа другому пользователю или другой группе командой

chown [ключи] <новый\_пользователь>[:новая\_группа] <файл>

или

chgrp [ключи] < новая\_группа > <файл>

Права доступа rw- (чтение и запись, без исполнения) понимаются как три двоичные цифры 110 или как восьмеричная цифра 6

Полная строка прав доступа в символьном представлении имеет вид:

<права\_владельца> <права\_группы> <права\_остальных>

Свойства (атрибуты) файлов и каталогов можно вывести на терминал с помощью команды ls с ключом -l.

Тип файла определяется первой позицией, это может быть: каталог — d, обычный файл — дефис (-) или символьная ссылка на другой файл — l.

Для изменения прав доступа служит команда chmod, которая понимает как символьное, так и числовое указание прав.

Формат символьного режима:

chmod <категория><действие><набор\_прав><файл>

**Работа с файлами средствами Nasm**

Обработка файлов в операционной системе Linux осуществляется за счет использования определенных системных вызовов. Для корректной работы и доступа к файлу при его от- крытии или создании, файлу присваивается уникальный номер (16-битное целое число) – дескриптор файла.

Общий алгоритм работы с системными вызовами в Nasm можно представить в следующем виде:

1. Поместить номер системного вызова в регистр EAX;
2. Поместить аргументы системного вызова в регистрах EBX, ECX и EDX;
3. Вызов прерывания (int 80h);
4. Результат обычно возвращается в регистр EAX.

**Открытие и создание файла**

Для создания и открытия файла служит системный вызов sys\_creat, который использует следующие аргументы: права доступа к файлу в регистре ECX, имя файла в EBX и номер системного вызова sys\_creat (8) в EAX.

Для открытия существующего файла служит системный вызов sys\_open, который исполь- зует следующие аргументы: права доступа к файлу в регистре EDX, режим доступа к файлу в регистр ECX, имя файла в EBX и номер системного вызова sys\_open (5) в EAX. Среди режимов доступа к файлам чаще всего используются:

* 1. – O\_RDONLY (открыть файл в режиме только для чтения);
  2. – O\_WRONLY – (открыть файл в режиме только записи);
  3. – O\_RDWR – (открыть файл в режиме чтения и записи).

Системный вызов возвращает файловый дескриптор открытого файла в регистр EAX. В случае ошибки, код ошибки также будет находиться в регистре EAX.

**Запись в файл**

Для записи в файл служит системный вызов sys\_write, который использует следующие аргументы: количество байтов для записи в регистре EDX, строку содержимого для записи ECX, файловый дескриптор в EBX и номер системного вызова sys\_write (4) в EAX. Системный вызов возвращает фактическое количество записанных байтов в регистр EAX. В случае ошибки, код ошибки также будет находиться в регистре EAX. Прежде чем записывать в файл, его необходимо создать или открыть, что позволит полу- чить дескриптор файла.

**Чтение файла**

Для чтения данных из файла служит системный вызов sys\_read, который использует следующие аргументы: количество байтов для чтения в регистре EDX, адрес в памяти для записи прочитанных данных в ECX, файловый дескриптор в EBX и номер системного вызова sys\_read (3) в EAX. Как и для записи, прежде чем читать из файла, его необходимо открыть, что позволит получить дескриптор файла.

**Закрытие файла**

Для правильного закрытия файла служит системный вызов sys\_close, который использует один аргумент – дескриптор файла в регистре EBX. После вызова ядра происходит удаление дескриптора файла, а в случае ошибки, системный вызов возвращает код ошибки в регистр EAX.

**Изменение содержимого файла**

Для изменения содержимого файла служит системный вызов sys\_lseek, который исполь- зует следующие аргументы: исходная позиция для смещения EDX, значение смещения в байтах в ECX, файловый дескриптор в EBX и номер системного вызова sys\_lseek (19) в EAX.

Значение смещения можно задавать в байтах. Значения обозначающие исходную позиции могут быть следующими:

* 1. – SEEK\_SET (начало файла)
  2. – SEEK\_CUR (текущая позиция);
  3. – SEEK\_END (конец файла).

В случае ошибки, системный вызов возвращает код ошибки в регистр EAX.

**Удаление файла**

Удаление файла осуществляется системным вызовом sys\_unlink(10), который использует один аргумент – имя файла в регистре EBX.

# 4 Выполнение лабораторной работы

Создаю каталог lab10 для программ лабораторной работы №10, перехожу в него и создаю файл lab10-1.asm, readme-1.txt и readme-2.txt (рис. [1](#fig:001))

Figure 1: Создание каталога lab010 и файлов в нем

Figure 1: Создание каталога lab010 и файлов в нем

Перед работой с программами копирую файл in\_out.asm в каталог и проверяю, что файл находится в нужном каталоге (рис. [2](#fig:002)) (рис. [3](#fig:003))

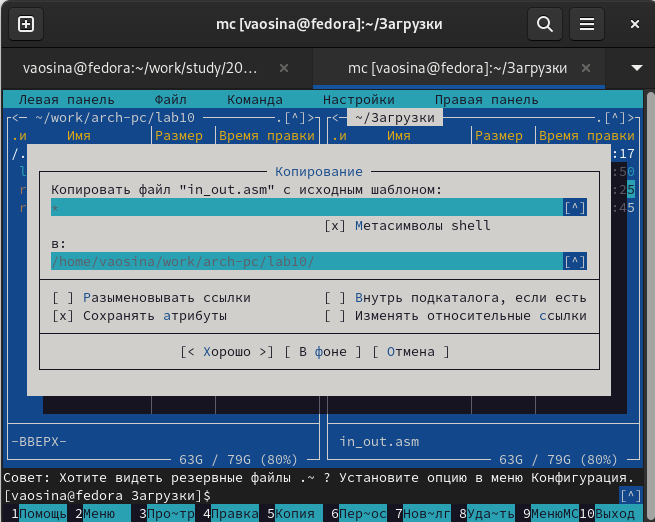


Figure 2: Копирование файла in\_out.asm

Figure 3: Проверка, что файл находится в нужном каталоге

Figure 3: Проверка, что файл находится в нужном каталоге

Открываю lab10-1.asm в редакторе и ввожу в него текст программы записи в файл сообщения (рис. [4](#fig:004)) и (рис. [5](#fig:005)).

Figure 4: Открытие файла в редакторе

Figure 4: Открытие файла в редакторе

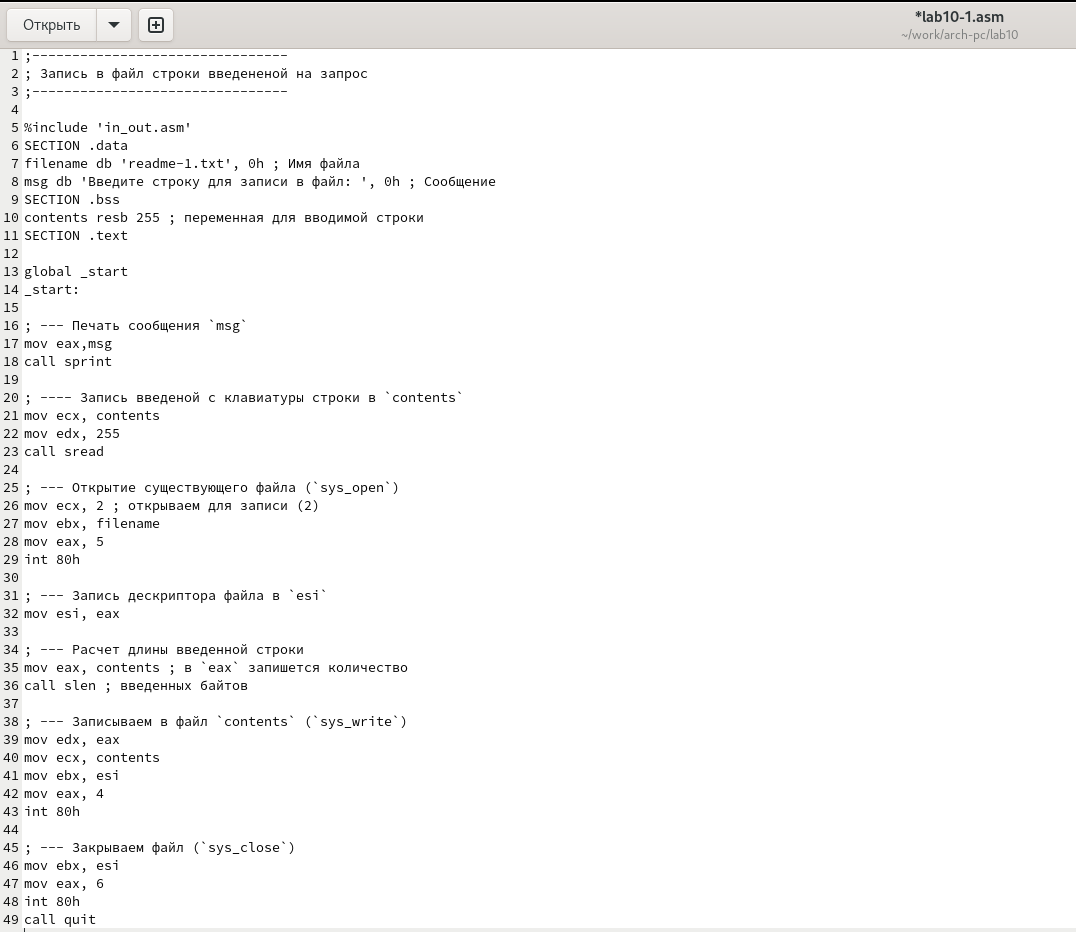


Figure 5: Ввод текста программы в файл

Создаю исполняемый файл и запускаю его (рис. [6](#fig:006)). Проверяю, записано ли сообщение в нужный файл, и вижу, что сообщение, введённое с клавиатуры, теперь действительно находится в файле readme-1.txt

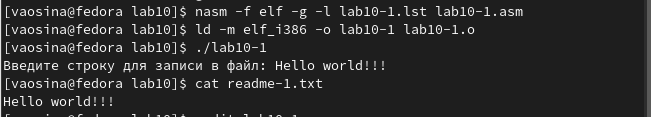


Figure 6: Создание исполняемого файла и его запуск

С помощью команды chmod a-x изменяю права доступа к исполняемому файлу lab10-1, запретив его выполнение. Пытаюсь выполнить файл, после чего выводится надпись “Отказано в доступе”. Так выходит, потому что введённая мной команда, где “а” - “все пользователи”, “-” - отмена набора прав, а “х” - право на исполнение, и подразумевает собой для всех пользователей запрет на исполнение файла, поэтому я и не могу выполнить файл. (рис. [7](#fig:007))

Figure 7: Изменение прав доступа к исполняемому файлу

Figure 7: Изменение прав доступа к исполняемому файлу

С помощью команды chmod a+x изменяю права доступа к файлу lab10-1.asm с исходным текстом программы, добавив права на исполнение, и пытаюсь исполнить его. В этом случае также выдается ошибка, т.к. в файле нет команд для терминала. (рис. [8](#fig:008))

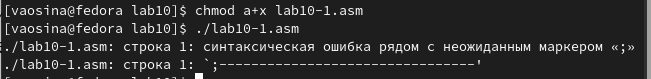


Figure 8: Изменение прав доступа к файлу

В соответствии со своим вариантом (вариант 7) предоставляю права доступа rw- rwx rw- к файлу readme-1.txt, представленные в символьном виде. Проверяю правильность выполнения при помощи команды ls -l. Права предоставлены корректно.(рис. [9](#fig:009))

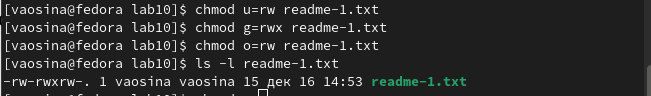


Figure 9: Предоставление прав для файла readme-1.txt

Для файла readme-2.txt предоставляю права доступа 101 111 111, но т.к. я не разобралась, как их предоставить, используя двоичное представление, то я использую восьмеричное представление, т.е. 577. Проверяю правильность выполнения при помощи команды ls -l. Права предоставлены корректно. (рис. [10](#fig:010))

Figure 10: Предоставление прав для файла readme-2.txt

Figure 10: Предоставление прав для файла readme-2.txt

# 5 Выполнение задания для самостоятельной работы

Создаю файл lab10-2.asm и ввожу в него текст программы (рис. [11](#fig:011)), работающей по следующему алгоритму:

* Вывод приглашения “Как Вас зовут?”
* ввести с клавиатуры свои фамилию и имя
* создать файл с именем name.txt
* записать в файл сообщение “Меня зовут”
* дописать в файл строку введенную с клавиатуры
* закрыть файл

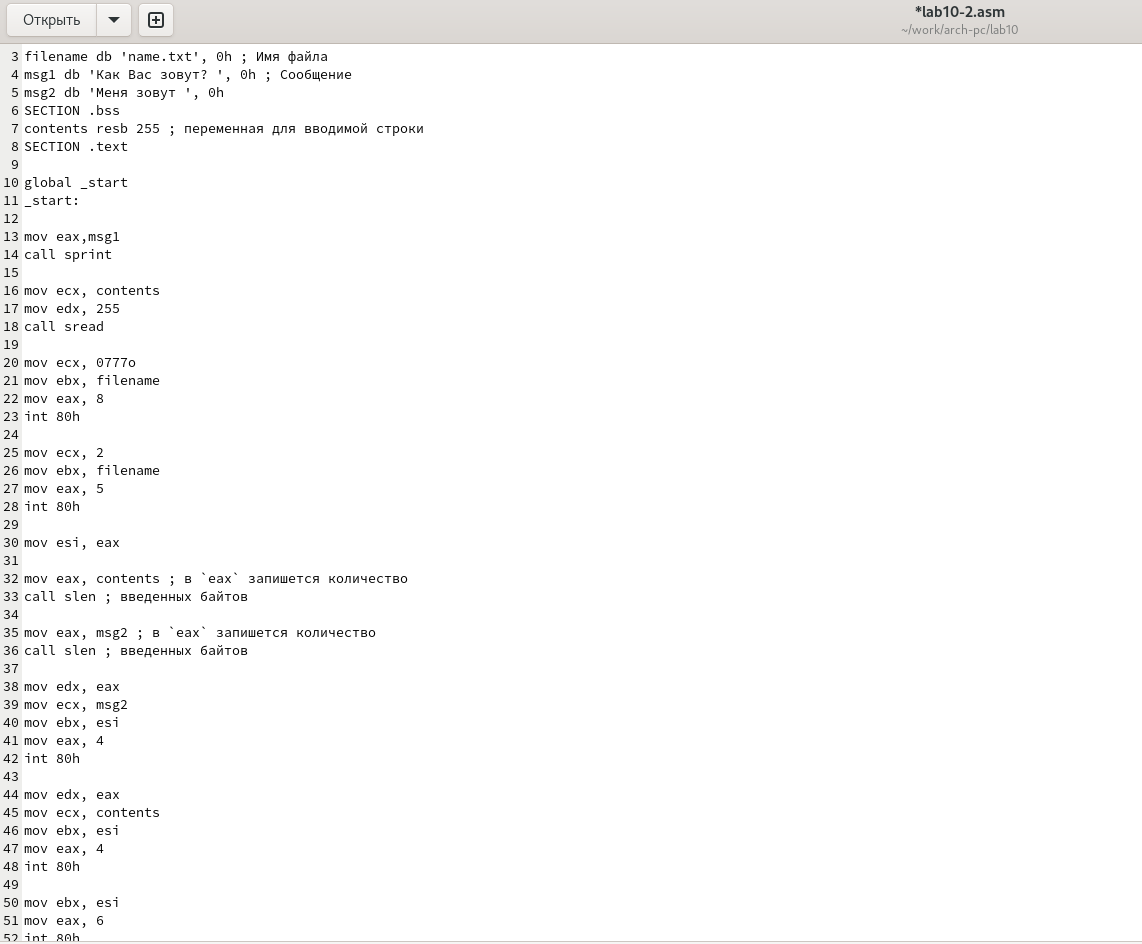


Figure 11: Ввод текста программы в файл

Создаю исполняемый файл и запускаю его. Проверяю наличие созданного при исполнении программы файла и вижу, что он действительно создан и находится в текущем каталоге. После этого проверяю содержимое файла при помощи команды cat. Программа сработала корректно. (рис. [12](#fig:012)).

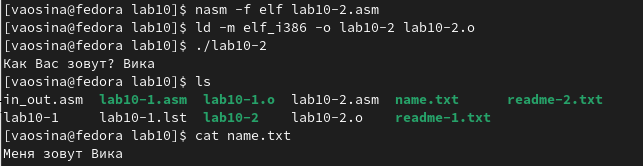


Figure 12: Выполнение программы, проверка наличия файла и его содержимого

Текст программы:

%include 'in\_out.asm'  
SECTION .data  
filename db 'name.txt', 0h ; Имя файла  
msg1 db 'Как Вас зовут? ', 0h ; Сообщение  
msg2 db 'Меня зовут ', 0h  
SECTION .bss  
contents resb 255 ; переменная для вводимой строки  
SECTION .text  
  
global \_start  
\_start:  
  
mov eax,msg1  
call sprint  
  
mov ecx, contents  
mov edx, 255  
call sread  
  
mov ecx, 0777o  
mov ebx, filename  
mov eax, 8  
int 80h  
  
mov ecx, 2  
mov ebx, filename  
mov eax, 5  
int 80h  
  
mov esi, eax  
  
mov eax, contents  
call slen   
  
mov eax, msg2   
call slen   
  
mov edx, eax  
mov ecx, msg2  
mov ebx, esi  
mov eax, 4  
int 80h  
  
mov edx, eax  
mov ecx, contents  
mov ebx, esi  
mov eax, 4  
int 80h  
  
mov ebx, esi  
mov eax, 6  
int 80h  
call quit

# 6 Выводы

При выполнении данной лабораторной работы я приобрела навыки написания программ для работы с файлами.

# 7 Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089097/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%9610.%20%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%D1%81%20%D1%84%D0%B0%D0%B9%D0%BB%D0%B0%D0%BC%D0%B8%20%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0%D0%BC%D0%B8%20Nasm.pdf)