Отчёт по лабораторной работе №10

Дисциплина: Архитектура компьютера

Карпова Анастасия Александровна

Содержание

[1 Цель работы 1](#_Toc153371414)

[2 Задание 1](#_Toc153371415)

[3 Теоретическое введение 1](#_Toc153371416)

[4 Выполнение лабораторной работы 2](#_Toc153371417)

[5 Выводы 5](#_Toc153371418)

[Список литературы 6](#_Toc153371419)

# 1 Цель работы

Приобретение навыков написания программ для работы с файлами.

# 2 Задание

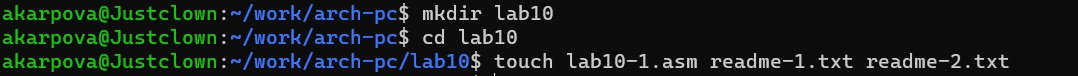
1. Написание программы для работы с файлами
2. Выполнение заданий для лабораторной работы

# 3 Теоретическое введение

ОС GNU/Linux является многопользовательской операционной системой. И для обеспечения защиты данных одного пользователя от действий других пользователей существуют специальные механизмы разграничения доступа к файлам. Кроме ограничения доступа, данный механизм позволяет разрешить другим пользователям доступ данным для совместной работы. Права доступа определяют набор действий (чтение, запись, выполнение), разрешённых для выполнения пользователям системы над файлами. Для каждого файла пользователь может входить в одну из трех групп: владелец, член группы владельца, все остальные. Для каждой из этих групп может быть установлен свой набор прав доступа. Владельцем файла является его создатель. Для предоставления прав доступа другому пользователю или другой группе командой 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. Следующие 3 набора по 3 символа определяют конкретные права для конкретных групп: r — разрешено чтение файла, w — разрешена запись в файл; x — разрешено исполнение файл и дефис (-) — право не дано. Для изменения прав доступа служит команда chmod, которая понимает как символьное, так и числовое указание прав. Для того чтобы назначить файлу /home/debugger/README права rw-r, то есть разрешить владельцу чтение и запись, группе только чтение, остальным пользователям — ничего. В символьном представлении есть возможность явно указывать какой группе какие права необходимо добавить, отнять или присвоить. В операционной системе Linux существуют различные методы управления файлами, например, такие как создание и открытие файла, только для чтения или для чтения и записи, добавления в существующий файл, закрытия и удаления файла, предоставление прав доступа. Обработка файлов в операционной системе Linux осуществляется за счет использования определенных системных вызовов. Для корректной работы и доступа к файлу при его открытии или создании, файлу присваивается уникальный номер (16-битное целое число) – дескриптор 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 использует 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. Системный вызов возвращает файловый дескриптор открытого файла в регистр EAX. В случае ошибки, код ошибки также будет находиться в регистре EAX 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 получить дескриптор 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. 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. Для изменения содержимого файла служит системный вызов sys\_lseek, который использует 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. Удаление файла осуществляется системным вызовом sys\_unlink, который использует один аргумент – имя файла в регистре EBX.

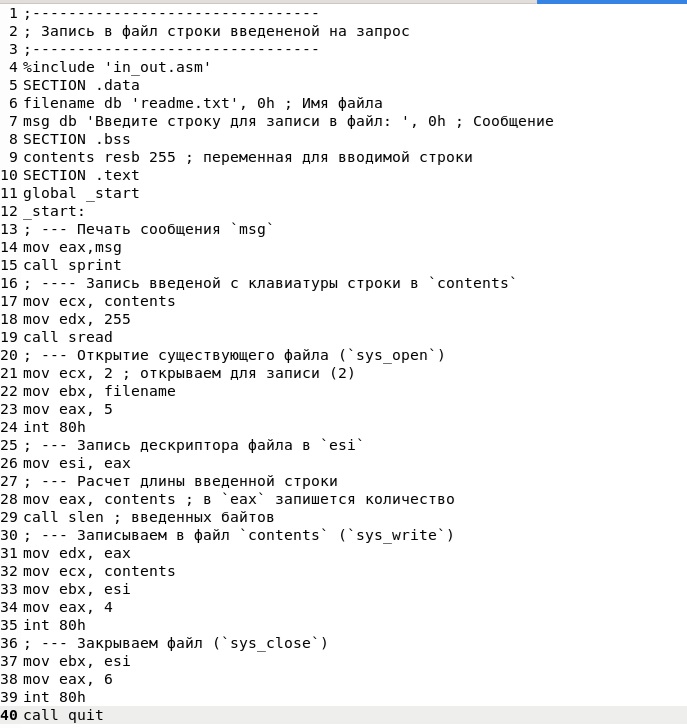
# 4 Выполнение лабораторной работы

Создаю каталог для программам лабораторной работы № 10, перехожу в него и создаю файлы lab10-1.asm, readme-1.txt и readme-2.txt (рис. 1).



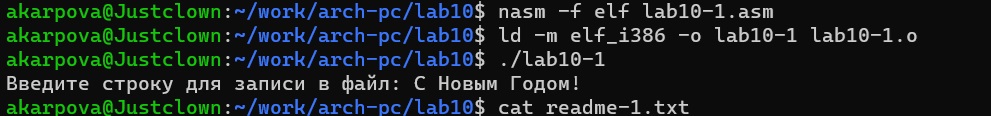
Создание каталога и файлов

Ввожу в файл lab10-1.asm текст программы из листинга 10.1 (рис.2)



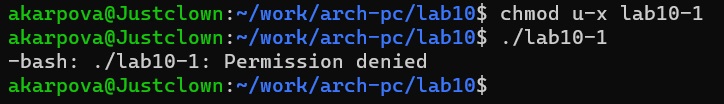
Ввод программы

Создаю и запускаю исполняемый и проверяю егор работу (рис. 3)



Создание и запуск исполняемого файла + проверка работы

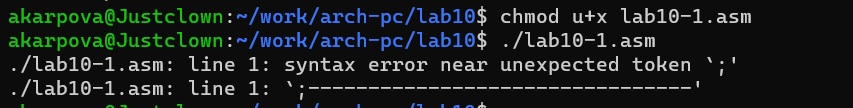
Теперь с помощью команды chmod u-x изменяю права доступа к исполняемому файлу lab10-1, запретив его выполнение и теперь пытаюсь выполнить файл (рис. 4)



Запрет на выполнение файла

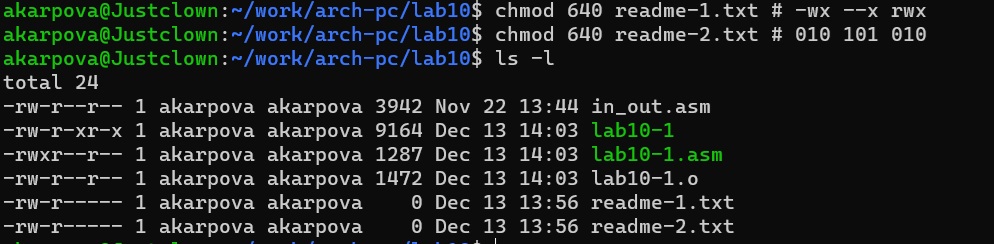
Файл не выполняется, так как в команде я указала u (себя) и “-” - отменить набор прав, а x - это право на исполнение

Далее, с помощью команды chmod+x я изменяю права доступа к файлу с исходным текстом программы, добавив права на исполнение, и снова выполняю файл (рис. 5)



Добавление права на исполнение

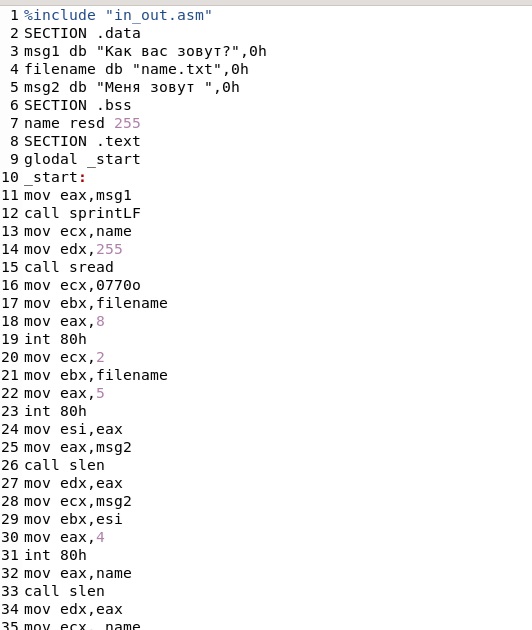
Текстовый файл начинает исполнение, но не исполняется, тк содержит в себе команду терминала. Мой вариант - 15. В соответствии с ним предоставляю права доступа к файлу readme-1.txt представленные в символьном виде, а для файла readme-2.txt в двоичном коде: -wx –x rwx 010 101 010 (рис. 6)



Предоставление прав доступа в символьном и двоичном видах

Выполнение заданий для самостоятельной работы

Пишу программу, выводящую приглашение “Как вас зовут?”, которая считывает с клавиатуры фамилию и имя, создает файл в который записывается сообщение “Меня зовут ФИ” (рис. 7)



Написание программы

Создаю исполняемый и запускаю его. Проверяю наличие файло при помощи ls и сat. (рис. 7)

![Создание и запуск исполняемого файла + проверка]](image/8.jpg){#fig:008 width=70%}

Код программы:

%include “in\_out.asm”

SECTION .data msg1

db “Как вас зовут?”,0h

filename db “name.txt”,0h

msg2 db “Меня зовут”,0h

SECTION .bss

name resd 255

SECTION .text

global \_start

\_start:

mov eax,msg1

call sprintLF

mov ecx,name

mov edx, 255

call sread

mov ecx, 0770o

mov ebx,filename

mov eax, 8

int 80h

mov ecx,

2 mov ebx,filename

mov eax, 5

int 80h

mov esi,eax

mov eax,msg2

call slen

mov edx,eax

mov ecx,msg2

mov ebx,esi

mov eax,4

int 80h

mov eax,name

call slen

mov edx,eax

mov ecx, name

mov ebx,esi

mov eax,4

int 80h

mov ebx,esi

mov eax,6

int 80h

call quit

# 5 Выводы

В ходе лабораторной работы я приобрела навыки написания программ для работы с файлами

# Список литературы

1. Архитектура ЭВМ