Российский Университет Дружбы народов

Факультет физико-математических и естественных наук

Отчет по лабораторной работе №10

“Работа с файлами средствами NASM “

Студентка: Богомолова Полина Петровна

Группа: НКАбд-01-25

Москва, 2025 год

Оглавление

Цель

Приобретение навыков написания программ для работы с файлами.

Теоретическое введение

Права доступа к файлам

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

chown [ключи] [:новая_группа]

или

chgrp [ключи] < новая_группа >

Набор прав доступа задается тройками битов и состоит из прав на чтение, запись и исполнение файла. В символьном представлении он имеет вид строк rwx, где вместо любого символа может стоять дефис. Всего возможно 8 комбинаций, приведенных в таблице 10.1. Буква означает наличие права (установлен в единицу второй бит триады r — чтение, первый бит w — запись, нулевой бит x — исполнение), а дефис означает отсутствие права (нулевое значение соответствующего бита). Также права доступа могут быть представлены как восьмеричное число. Так, права доступа rw- (чтение и запись, без исполнения) понимаются как три двоичные цифры 110 или как восьмеричная цифра 6.

Двоичный, буквенный и восьмеричный способ записи триады прав доступа представлен в таблице 1.

Таблица 1

Двоичный	Буквенный	Восмеричный
111	rwx	7
110	rw-	6
101	r-x	5
100	r--	4
011	-wx	3
010	-w-	2
001	--x	1
000	---	0

Полная строка прав доступа в символьном представлении имеет вид:

<права_владельца> <права_группы > <права_остальных >

Так, например, права rwx r-x --x выглядят как двоичное число 111 101 001, или восьмеричное 751. Свойства (атрибуты) файлов и каталогов можно вывести на терминал с помощью команды ls с ключом -l. Так например, чтобы узнать права доступа к файлу README можно узнать с помощью следующей команды:

$ls -l /home/debugger/README

-rwxr-xr-- 1 debugger users 0 Feb 14 19:08 /home/debugger/README

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:

$chmod 640 README # 110 100 000 == 640 == rw-r-----

$ls -l README

-rw-r 1 debugger users 0 Feb 14 19:08 /home/debugger/README

В символьном представлении есть возможность явно указывать какой группе какие права необходимо добавить, отнять или присвоить. Например, чтобы добавить право на исполнение файла README группе и всем остальным:

$chmod go+x README

$ls -l README

-rw-r-x--x 1 debugger users 0 Feb 14 19:08 /home/debugger/README

Формат символьного режима:

chmod <категория><действие><набор_прав><файл>

Возможные значения аргументов команды представлены в таблице 2.

Таблица 2

Категория	Обозначение	Значение
Принадлежность	u	Владелец
	g	Группа владельца
	o	Прочие пользователи
	a	Все пользователи, то есть «a» эквивалентно «ugo»
Действие	+	Добавить набор прав
	-	Отменить набор прав
	=	Назначить набор прав
Право	r	Право на чтение
	w	Право на запись
	x	Право на исполнение

Работа с файлами средствами NASM

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.

В таблице 3 приведены системные вызовы для обработки файлов.

Таблица 3

Имя системного вызова	eax	ebx	ecx	edx
sys_read	3	дескриптор файла	адрес в памяти	количество байтов
sys_write	4	дескриптор файла	строка	количество байтов
sys_open	5	имя файла	режим доступа к файлу	права доступа к файлу
sys_close	6	дескриптор файла	—	—
sys_creat	8	имя файла	права доступа к файлу	—
sys_unlink	10	имя файла	—	—
sys_lseek	19	имя файла	значение смещения в байтах	позиция для смещения

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 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. Среди режимов доступа к файлам чаще всего используются:

• (0) – O_RDONLY (открыть файл в режиме только для чтения);

• (1) – O_WRONLY – (открыть файл в режиме только записи);

• (2) – 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 получить 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 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. Значение смещения можно задавать в байтах. Значения обозначающие исходную позиции могут быть следующими:

• (0) – SEEK_SET (начало файла);

• (1) – SEEK_CUR (текущая позиция);

• (2) – SEEK_END (конец файла). В случае ошибки, системный вызов возвращает код ошибки в регистр EAX.

Удаление файла

Удаление файла осуществляется системным вызовом sys_unlink, который использует один аргумент – имя файла в регистре EBX.

Выполнение лабораторной работы

1. Создадим каталог для программ лабораторной работы №10, перейдем в него и создадим файлы lab10-1.asm, readme-1.txt и readme-2.txt:

   mkdir ~/work/arch-pc/lab09

   cd ~/work/arch-pc/lab09

   touch lab10-1.asm readme-1.txt readme-2.txt

Во время выполнения лабораторной работы будем использовать Midnight Commander для работы с файлами, запускаем его с помощью команды mc.

Результат представлен на рисунках 1-2

Рис. 1

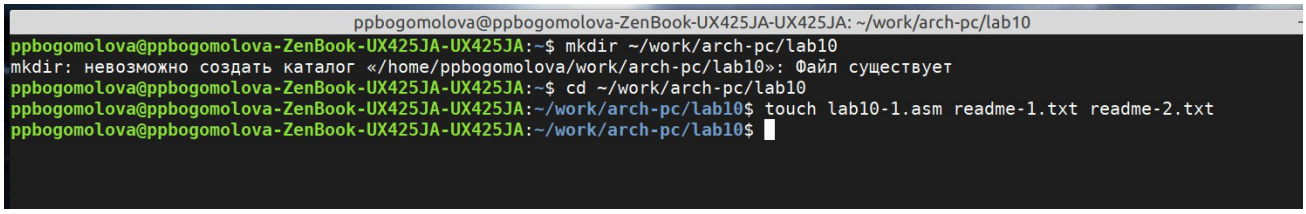
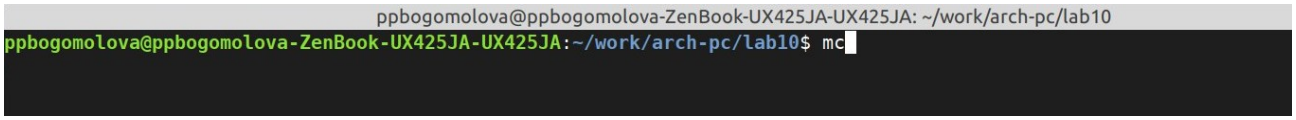


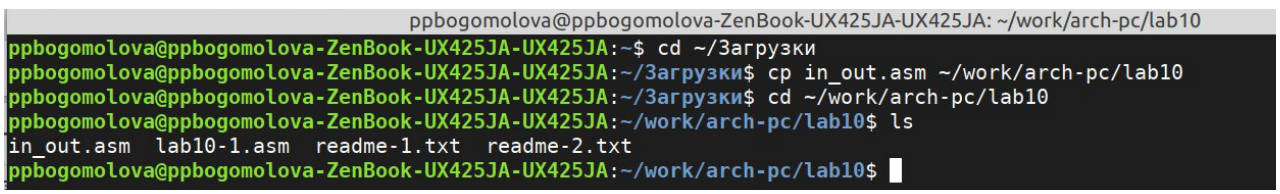
Рис. 2



Подготовимся к выполнению лабораторной работы. Скопируем файл in_out.asm в каталог лабораторной работы номер 10 с помощью команды cp, перемещаться между каталогами будем с помощью команды cd. С помощью команды ls убедимся в том, что файл был скопирован в нужном каталоге.

Результат представлен на рисунке 3.

Рис. 3



2. Введем в файл lab10-1.asm текст программы из листинга 10.1 (Программа записи в файл сообщения). Создадим исполняемый файл и проверим его работу. Для этого будем использовать команды nasm –f elf –g –l, ld –m elf_i386 –o, ./. Команду ls -l используем для вывода списка файлов и папок в текущей директории в подробном виде. Она показывает не только имена файлов, но и дополнительную информацию о них, такую как права доступа, владелец, размер и время последнего изменения. Команду cat используем для просмотра содержимого текстового файла.

   Результат представлен на рисунках 4-6.

Рис. 4

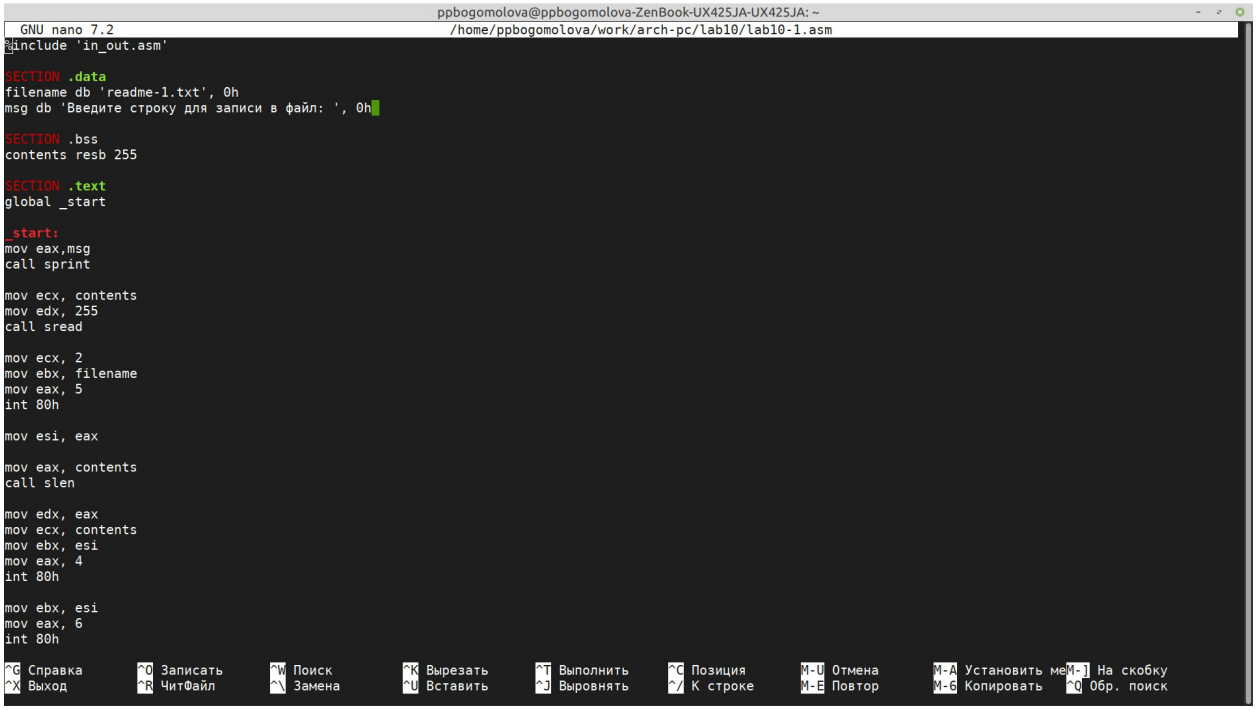


Рис. 5

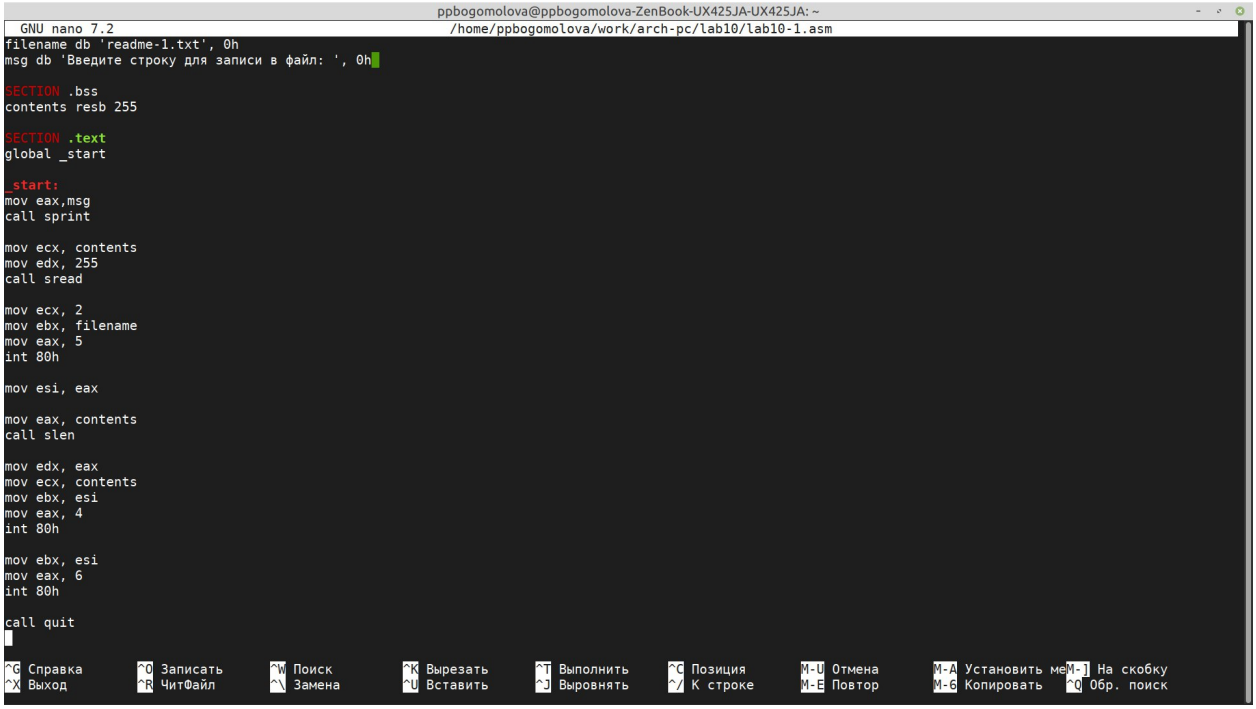
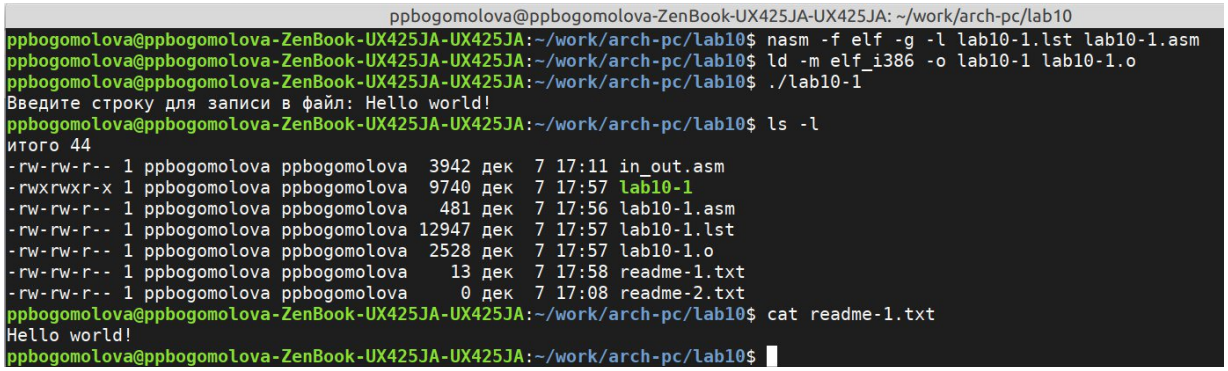


Рис. 6

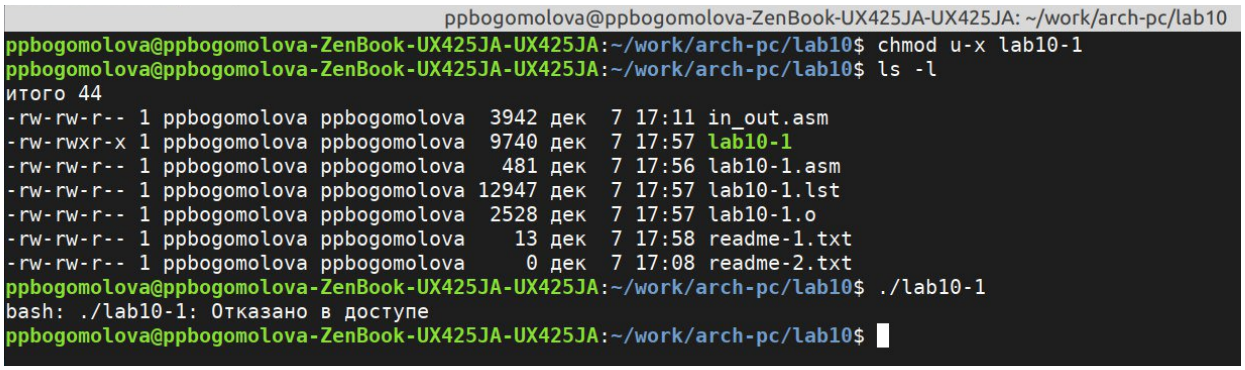


3. С помощью команды chmod изменим права доступа к исполняемому файлу lab10-1, запретив его выполнение. Попытаемся выполнить файл. **Объясните результат.**

При выполнении команды chmod u-x lab10-1 доступ к файлу был запрещен его владельцу, поэтому в результате мы получили сообщение ‘Отказано в доступе’ при попытке запустить файл. Команду ls -l используем для вывода списка файлов и папок в текущей директории в подробном виде.

Результат предствлен на рисунке 7.

Рис. 7

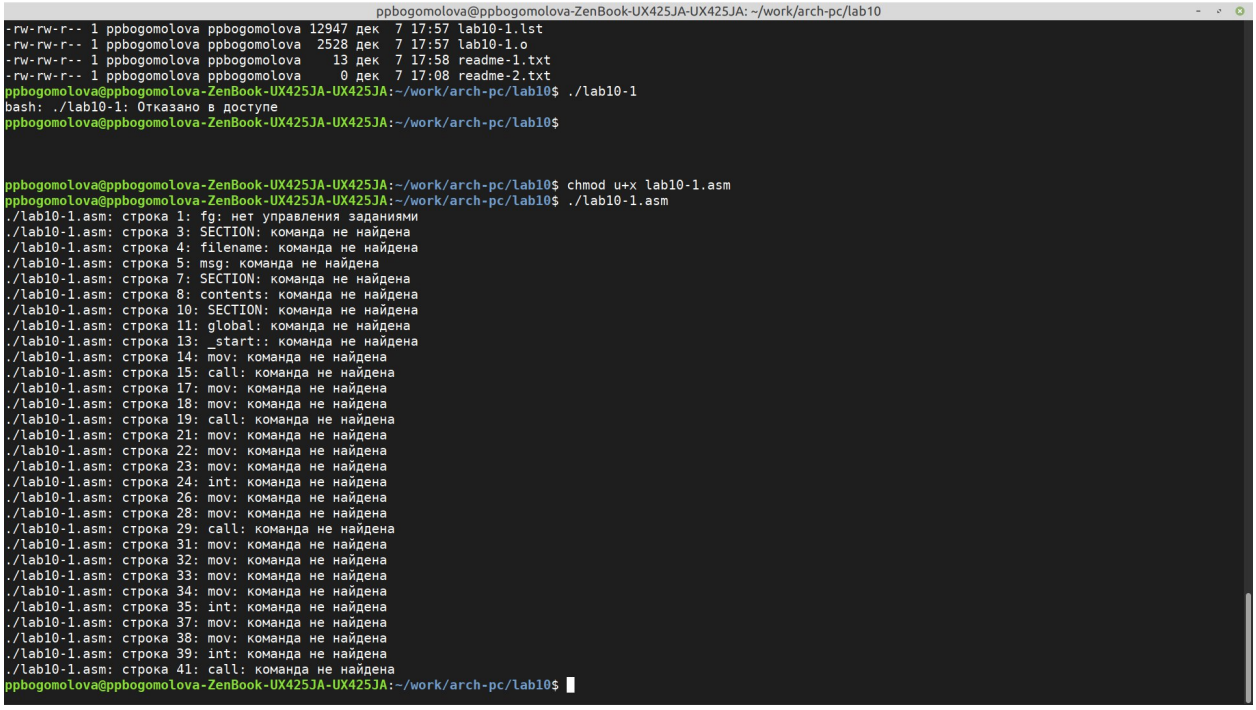


4. С помощью команды chmod изменим права доступа к файлу lab10-1.asm с исходным текстом программы, добавив права на исполнение. Попытаемся выполнить его с помощью команды ./

Команда chmod u+x lab10-1 добавляет право на выполнение файла для его владельца. Это означает, что теперь файл можно запускать как программу. Однако права на выполнение не изменяют содержимое файла - если внутри файла нет корректного исполняемого кода или команд, оболочка Bash не сможет его выполнить. При попытке запуска такого файла появляется сообщение “команда не найдена”, потому что Bash интерпретирует строки файла как команды, а содержимое файла не соответствует допустимому синтаксису Bash. Таким образом, chmod u+x делает файл исполняемым с точки зрения системы, но успешный запуск возможен только если файл действительно содержит действительный код.

Результат представлен на рисунке 8.

Рис. 8



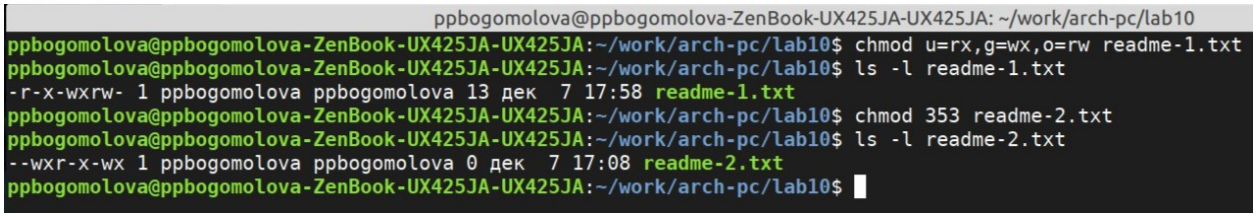
5. В соответствии с вариантом (3) в таблице предоставим права доступа к файлу readme1.txt представленные в символьном виде, а для файла readme-2.txt – в двоичном виде. Проверим правильность выполнения с помощью команды ls -l. Результат представлен на рисунке 9.

   В Linux права доступа к файлам определяют, кто и какие действия может выполнять с файлом или директорией. Права задаются для трёх категорий пользователей: владельца файла, группы и остальных. Существует несколько способов отображения и задания прав: символьный вид, двоичный вид и восьмеричный вид. В символьном виде каждое право обозначается отдельной буквой: r - чтение, w - запись, x - выполнение, а отсутствие права - символ -. Для файла readme1.txt права были заданы как r-x для владельца, -wx для группы и rw- для остальных, что означает, что владелец может читать и выполнять файл, группа может записывать и выполнять, а остальные пользователи могут читать и писать. Для файла readme-2.txt права задавались в двоичном виде как 011, 101 и 011. В двоичном виде каждый бит обозначает наличие или отсутствие конкретного права: первый бит - чтение, второй - запись, третий - выполнение, где 1 означает разрешение, а 0 - отсутствие права. Чтобы использовать такие права с командой chmod, их переводят в восьмеричный вид. Для этого каждому

праву присваивается числовое значение: чтение r = 4, запись w = 2, выполнение x = 1, и затем суммируются включённые права. Например, двоичная запись 011 для владельца файла означает отсутствие чтения (0), наличие записи (2) и выполнения (1), в сумме получается 3 в восьмеричной системе. Аналогично вычисляются права для группы 101 (чтение + выполнение = 4 + 1 = 5) и для остальных 011 (запись + выполнение = 2 + 1 = 3). Таким образом, итоговая восьмеричная запись прав доступа для файла readme-2.txt - 353.

Команда chmod 353 readme-2.txt позволяет задать все права для владельца, группы и остальных пользователей без необходимости писать длинные символьные комбинации. Проверка правильности заданных прав выполняется командой ls -l, которая отображает текущие права доступа в символьном виде и позволяет убедиться, что права установлены корректно.

Рис. 9



Задания для самостоятельной работы

1. Напишите программу, работающую 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

   Создать исполняемый файл и проверить его работу. Проверить наличие файла и его содержимое с помощью команд ls и cat.

   Создадим исполняемый файл с помощью команды touch. Создадим исполняемый файл и проверим его работу с помощью команд nasm –f elf, ld –m elf_i386, ./. Введем фамилию и имя. Проверку правильности заданных прав выполним командой ls -l, которая отображает текущие права доступа в символьном виде и позволяет убедиться, что права установлены корректно. С помощью команды cat проверим содержимое текстового файла name и увидим, что в нем содержится ‘Меня зовут Полина Богомолова’, значит программа работает верно.

   Результат представлен на рисунках 10-13.

Рис. 10

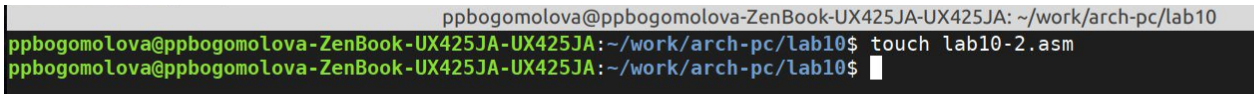


Рис. 11

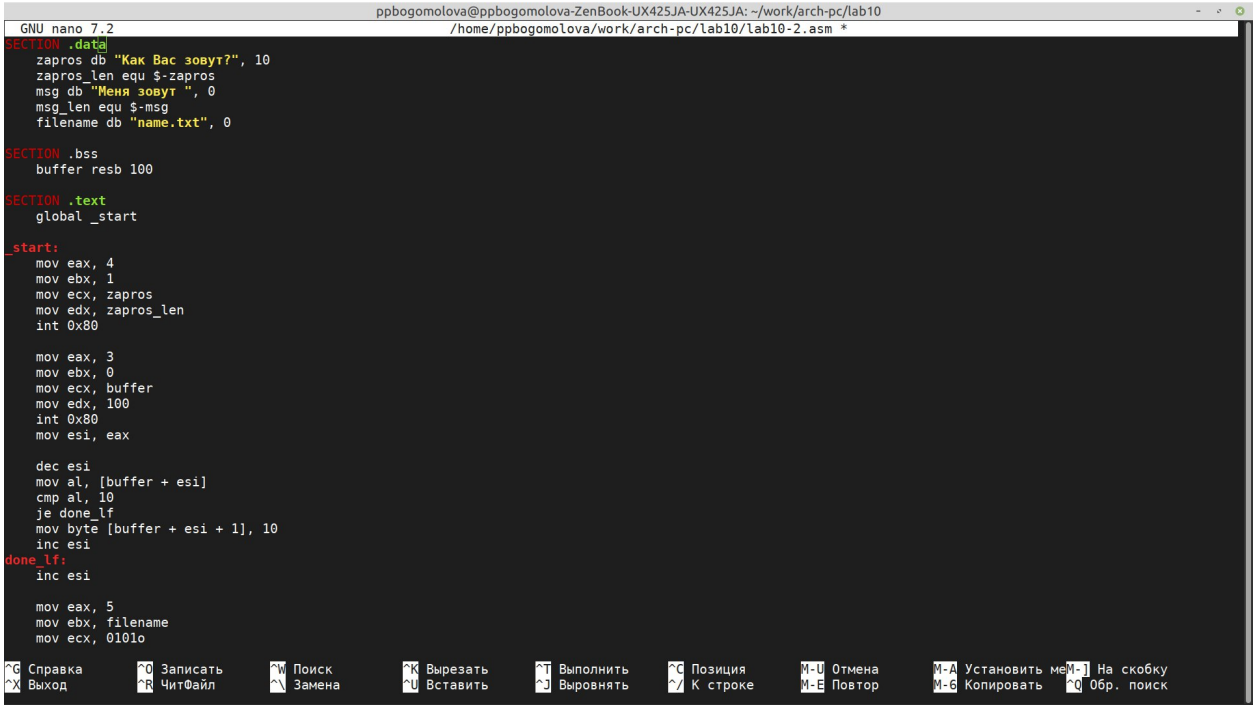

ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~/work/arch-pc/lab10
GNU nano 7.2 /home/ppbogomolova/work/arch-pc/lab10/lab10-2.asm *
SECTION .data
zapros db "Как Вас зовут?", 10
zapros_len equ $-zapros
msg db "Меня зовут ", 0
msg_len equ $-msg
filename db "name.txt", 0
SECTION .bss
buffer resb 100
SECTION .text
global _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, zapros
mov edx, zapros_len
int 0x80
mov eax, 3
mov ebx, 0
mov ecx, buffer
mov edx, 100
int 0x80
mov esi, eax
dec esi
mov al, [buffer + esi]
cmp al, 10
je done_lf
mov byte [buffer + esi + 1], 10
inc esi
done_lf:
inc esi
mov eax, 5
mov ebx, filename
mov ecx, 0101o
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция M-U Отмена M-A Установить ме M-] На скобку
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/ К строке M-E Повтор M-6 Копировать ^Q Обр. поиск

Рис. 12

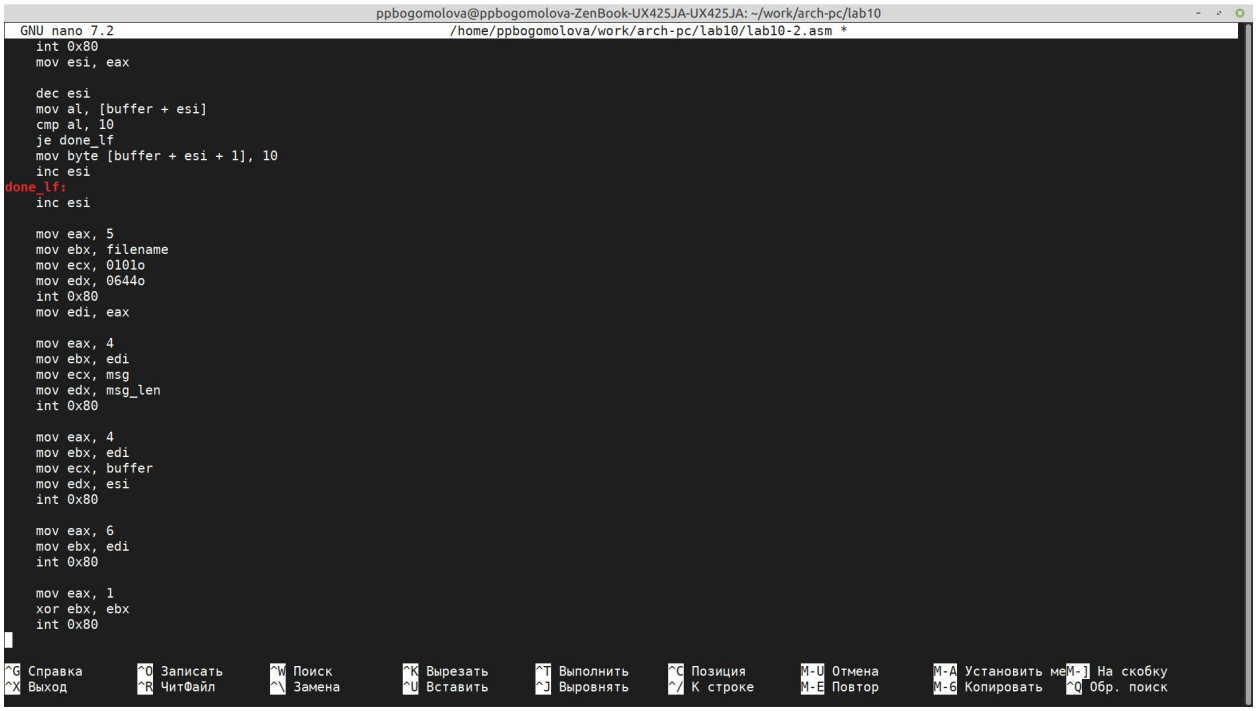
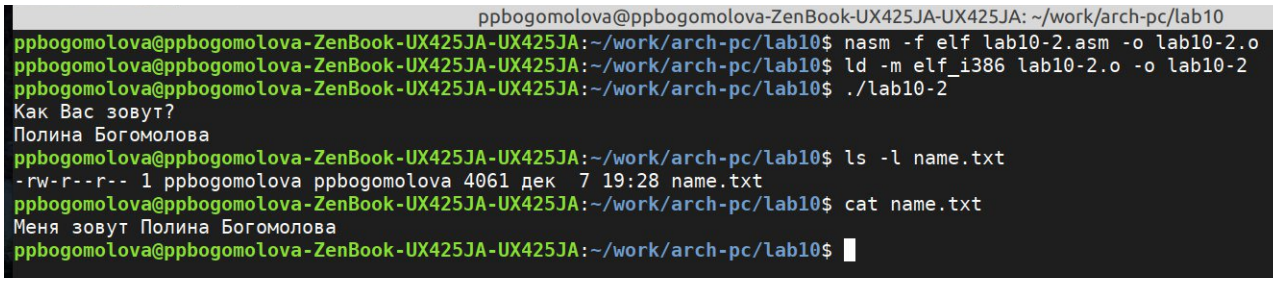

ppbogomolova@ppbogomolova-ZenBook-UX425JA-UX425JA: ~/work/arch-pc/lab10
GNU nano 7.2 /home/ppbogomolova/work/arch-pc/lab10/lab10-2.asm *
int 0x80
mov esi, eax
dec esi
mov al, [buffer + esi]
cmp al, 10
je done_lf
mov byte [buffer + esi + 1], 10
inc esi
done_lf:
inc esi
mov eax, 5
mov ebx, filename
mov ecx, 0101o
mov edx, 0644o
int 0x80
mov edi, eax
mov eax, 4
mov ebx, edi
mov ecx, msg
mov edx, msg_len
int 0x80
mov eax, 4
mov ebx, edi
mov ecx, buffer
mov edx, esi
int 0x80
mov eax, 6
mov ebx, edi
int 0x80
mov eax, 1
xor ebx, ebx
int 0x80
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция M-U Отмена M-A Установить ме M-] На скобку
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/ К строке M-E Повтор M-6 Копировать ^Q Обр. поиск

Рис. 13



Ссылка на мой репозиторий в GitHub:

https://github.com/bogomolova-pp/study_2025-2026_arh-pc

Вывод

В результате выполнения лабораторной работы я приобрела навыки написания программ для работы с файлами.

Листинги

Листинг 10.1

```
%include 'in_out.asm'
SECTION .data
filename db 'readme-1.txt', 0h ; Имя файла
msg db 'Введите строку для записи в файл: ', 0h ; Сообщение
SECTION .bss
contents resb 255 ; переменная для вводимой строки
SECTION .text
global _start
_start:
; --- Печать сообщения `msg`
mov eax,msg
call sprint
; ---- Запись введеной с клавиатуры строки в `contents`
mov ecx, contents
mov edx, 255
call sread
; --- Открытие существующего файла (`sys_open`)
mov ecx, 2 ; открываем для записи (2)
mov ebx, filename
mov eax, 5
int 80h
; --- Запись дескриптора файла в `esi`
mov esi, eax
; --- Расчет длины введенной строки
mov eax, contents ; в `eax` запишется количество
call slen ; введенных байтов
; --- Записываем в файл `contents` (`sys_write`)
mov edx, eax
```

```
mov ecx, contents
mov ebx, esi
mov eax, 4
int 80h
; --- Закрываем файл (`sys_close`)
mov ebx, esi
mov eax, 6
int 80h
call                                                    quit
```

Листинг 10.2

```
SECTION .data
    zapros db "Как Вас зовут?", 10
    zapros_len equ $-zapros
    msg db "Меня зовут ", 0
    msg_len equ $-msg
    filename db "name.txt", 0

SECTION .bss
    buffer resb 100

SECTION .text
    global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, zapros
    mov edx, zapros_len
    int 0x80
```

```
    mov eax, 3
    mov ebx, 0
    mov ecx, buffer
    mov edx, 100
    int 0x80
    mov esi, eax

    dec esi
    mov al, [buffer + esi]
    cmp al, 10
    je done_lf
    mov byte [buffer + esi + 1], 10
    inc esi
done_lf:
    inc esi

    mov eax, 5
    mov ebx, filename
    mov ecx, 0101o
    mov edx, 0644o
    int 0x80
    mov edi, eax

    mov eax, 4
    mov ebx, edi
    mov ecx, msg
    mov edx, msg_len
    int 0x80

    mov eax, 4
    mov ebx, edi
```

```
mov ecx, buffer
mov edx, esi
int 0x80

mov eax, 6
mov ebx, edi
int 0x80

mov eax, 1
xor ebx, ebx
int 0x80
```

Список литературы

1. Демидова А.В – Лабораторная работа №10. Работа с файлами средствами NASM.