

Отчёт по лабораторной работе №10

Дисциплина: Архитектура компьютера

Долгаев Евгений Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Права доступа к файлам	7
3.2	Работа с файлами средствами Nasm	10
3.2.1	Открытие и создание файла	12
3.2.2	Запись в файл	13
3.2.3	Чтение файла	13
3.2.4	Закрытие файла	14
3.2.5	Изменение содержимого файла	14
3.2.6	Удаление файла	15
4	Выполнение лабораторной работы	18
4.1	Задание для самостоятельной работы	20
5	Выводы	21
	Список литературы	22

Список иллюстраций

3.1	Структура	9
4.1	Подготовка рабочего пространства	18
4.2	Текст программы	18
4.3	Создание и работа исполняемого файла	19
4.4	Содержимое текстового файла	19
4.5	Изменение прав доступа	19
4.6	Изменение прав доступа	19
4.7	Изменение прав доступа	19
4.8	Текст программы	20
4.9	Создание и работа исполняемого файла	20

Список таблиц

3.1	Двоичный, буквенный и восьмеричный способ записи триады прав доступа	8
3.2	Возможные значения аргументов команды <code>chmod</code>	10
3.3	Системные вызовы для обработки файлов	11

1 Цель работы

Приобретение навыков написания программ для работы с файлами.

2 Задание

- 1) Порядок выполнения лабораторной работы
- 2) Задание для самостоятельной работы

3 Теоретическое введение

3.1 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. Владелцем файла является его создатель. Для предоставления прав доступа другому пользователю или другой группе командой

```
chown [ключи] <новый_пользователь>[:новая_группа] <файл>
```

или

```
chgrp [ключи] < новая_группа > <файл>
```

Набор прав доступа задается тройками битов и состоит из прав на чтение, запись и исполнение файла. В символьном представлении он имеет вид строк `gwx`, где вместо любого символа может стоять дефис. Всего возможно 8 комбинаций, приведенных в таблице 3.1. 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.

Таблица 3.1: Двоичный, буквенный и восьмеричный способ записи триады прав доступа

Двоичный	Буквенный	Восьмиричный
111	<i>rwX</i>	7
110	<i>rw-</i>	6
101	<i>r-X</i>	5
100	<i>r-</i>	4
011	<i>-wX</i>	3
010	<i>-w-</i>	2
001	<i>-X</i>	1
000	<i>-</i>	0

Полная строка прав доступа в символьном представлении имеет вид:

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
```


В первой колонке показаны текущие права доступа, далее указан владелец файла и группа:

-	rwX	r-X	r-X	l debugger user
тип	владелец	группа	остальные	
type	owner	group	others	

Рис. 3.1: Структура

Тип файла определяется первой позицией, это может быть: каталог — d, обычный файл — дефис (-) или символическая 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 файла 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 3.2.

Таблица 3.2: Возможные значения аргументов команды `chmod`

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

3.2 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 3.3 приведены системные вызовы для обработки файлов.

Таблица 3.3: Системные вызовы для обработки файлов

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.

3.2.1 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`.

```
mov ecx, 0777o ; установка прав доступа
mov ebx, filename ; имя создаваемого файла
mov eax, 8 ; номер системного вызова `sys_creat`
int 80h ; вызов ядра
```

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`.

```
mov ecx, 0 ; режим доступа (0 - только чтение)
mov ebx, filename ; имя открываемого файла
mov eax, 5 ; номер системного вызова `sys_open`
int 80h ; вызов ядра
```

3.2.2 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.

```
mov ecx, 0777o ; Создание файла.  
mov ebx, filename ; в случае успешного создания файла,  
mov eax, 8 ; в регистр eax запишется дескриптор файла  
int 80h  
mov edx, 12 ; количество байтов для записи  
mov ecx, msg ; адрес строки для записи в файл  
mov ebx, eax ; дескриптор файла  
mov eax, 4 ; номер системного вызова 'sys_write'  
int 80h ; вызов ядра
```

3.2.3 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.

```
mov ecx, 0 ; Открытие файла.  
mov ebx, filename ; в случае успешного открытия файла,
```

```

mov eax, 5 ; в регистр EAX запишется дескриптор файла
int 80h
mov edx, 12 ; количество байтов для чтения
mov ecx, fileCont ; адрес в памяти для записи прочитанных данных
mov ebx, eax ; дескриптор файла
mov eax, 3 ; номер системного вызова `sys_read`
int 80h ; вызов ядра

```

3.2.4 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.

```

mov ecx, 0 ; Открытие файла.
mov ebx, filename ; в случае успешного открытия файла,
mov eax, 5 ; в регистр EAX запишется дескриптор файла
int 80h
mov ebx, eax ; дескриптор файла
mov eax, 6 ; номер системного вызова `sys_close`
int 80h ; вызов ядра

```

3.2.5 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.

```
mov ecx, 1 ; Открытие файла (1 - для записи).
mov ebx, filename
mov eax, 5
int 80h
mov edx, 2 ; значение смещения -- конец файла
mov ecx, 0 ; смещение на 0 байт
mov ebx, eax ; дескриптор файла
mov eax, 19 ; номер системного вызова 'sys_lseek'
int 80h ; вызов ядра
mov edx, 9 ; Запись в конец файла
mov ecx, msg ; строки из переменной 'msg'
mov eax, 4
int 80h
```

3.2.6 Удаление файла

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

```
mov ebx, filename ; имя файла
mov eax, 10 ; номер системного вызова 'sys_unlink'
int 80h ; вызов ядра
```

В качестве примера приведем программу, которая открывает существующий файл, записывает в него сообщение и закрывает файл.

```

;-----
; Запись в файл строки введенной на запрос
;-----
#include 'in_out.asm'

SECTION .data
filename db 'readme.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 введенной 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

```

Результат работы программы:

```

user@dk4n31:~$ nasm -f elf -g -l main.lst main.asm
user@dk4n31:~$ ld -m elf_i386 -o main main.o
user@dk4n31:~$ ./main
Введите строку для записи в файл: Hello world!
user@dk4n31:~$ ls -l
-rwxrwxrwx 1 user user 20 Jul 2 13:06 readme.txt
-rwxrwxrwx 1 user user 11152 Jul 2 13:05 main
-rwxrwxrwx 1 user user 1785 Jul 2 13:03 main.asm
-rwxrwxrwx 1 user user 22656 Jul 2 13:05 main.lst
-rwxrwxrwx 1 user user 4592 Jul 2 13:05 main.o
user@dk4n31:~$ cat readme.txt
Hello world!
user@dk4n31:~$

```

4 Выполнение лабораторной работы

Создим каталог для программ лабораторной работы № 10, перейдём в него и создадим файлы lab10-1.asm, readme-1.txt и readme-2.txt(рис. 4.1):

```
esdolgaev@esdolgaev-VirtualBox:~$ mkdir ~/work/arch-pc/lab10
esdolgaev@esdolgaev-VirtualBox:~$ cd ~/work/arch-pc/lab10
esdolgaev@esdolgaev-VirtualBox:~/work/arch-pc/lab10$ touch lab10-1.asm readme-1.
txt readme-2.txt
esdolgaev@esdolgaev-VirtualBox:~/work/arch-pc/lab10$ ls
lab10-1.asm readme-1.txt readme-2.txt
esdolgaev@esdolgaev-VirtualBox:~/work/arch-pc/lab10$
```

Рис. 4.1: Подготовка рабочего пространства

Введём в файл lab10-1.asm текст программы (Программа записи в файл сообщения). Создадим исполняемый файл и проверим его работу(рис. 4.2, 4.3, 4.4).

```
/home/esdolgaev/work/arch-pc/lab10/lab10-1.asm [----] 9 L: [ 8+14 22/ 40] *(792 /1286b) 0102 0x066 [*][X]
SECTION .bss
contents resb 255 ; переменная для вводимой строки
SECTION .text
global _start
_start:
; --- Печать сообщения 'msg'
mov eax,msg
call sprint
; ---- Запись введенной 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

```

Рис. 4.2: Текст программы

```
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ nasm -f elf -g -l lab10-1.lst lab10-1.asm
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ ld -m elf_i386 -o lab10-1 lab10-1.o
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ ./lab10-1
Введите строку для записи в файл: 12345
```

Рис. 4.3: Создание и работа исполняемого файла

```
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ cat readme.txt
12345
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$
```

Рис. 4.4: Содержимое текстового файла

С помощью команды `chmod` изменим права доступа к исполняемому файлу `lab10-1`, запретив его выполнение(рис. 4.5).

```
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ chmod a-x lab10-1
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ ls -l
итого 40
-rw-rw-r-- 1 esdolgaev esdolgaev 3942 дек 13 16:45 in_out.asm
-rw-rw-r-- 1 esdolgaev esdolgaev 9736 дек 13 17:12 lab10-1
-rw-rw-r-- 1 esdolgaev esdolgaev 1286 дек 13 17:12 lab10-1.asm
-rw-rw-r-- 1 esdolgaev esdolgaev 13713 дек 13 17:12 lab10-1.lst
-rw-rw-r-- 1 esdolgaev esdolgaev 2528 дек 13 17:12 lab10-1.o
-rw-rw-r-- 1 esdolgaev esdolgaev 0 дек 13 16:41 readme-1.txt
-rw-rw-r-- 1 esdolgaev esdolgaev 0 дек 13 16:41 readme-2.txt
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ ./lab10-1
bash: ./lab10-1: Отказано в доступе
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$
```

Рис. 4.5: Изменение прав доступа

Команда `chmod a-x` запрещает выполняться файлу `lab10-1`.

С помощью команды `chmod` изменим права доступа к файлу `lab10-1.asm` с исходным текстом программы, добавив права на исполнение(рис. 4.6).

```
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ chmod a+x lab10-1
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ ./lab10-1
Введите строку для записи в файл: 123
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$
```

Рис. 4.6: Изменение прав доступа

Команда `chmod a+x` запрещает выполняться файлу `lab10-1`.

Предоставим права доступа к файлу `readme-1.txt` представленные в символьном виде, а для файла `readme-2.txt` – в двоичном виде. Проверим правильность выполнения с помощью команды `ls -l`(рис. 4.7).

```
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ chmod 631 readme-1.txt
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ chmod 210 readme-2.txt
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ ls -l readme-1.txt
-rw--wx--x 1 esdolgaev esdolgaev 0 дек 13 16:41 readme-1.txt
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$ ls -l readme-2.txt
--w---x--- 1 esdolgaev esdolgaev 0 дек 13 16:41 readme-2.txt
esdolgaev@esdolgaev-VirtualBox: ~/work/arch-pc/lab10$
```

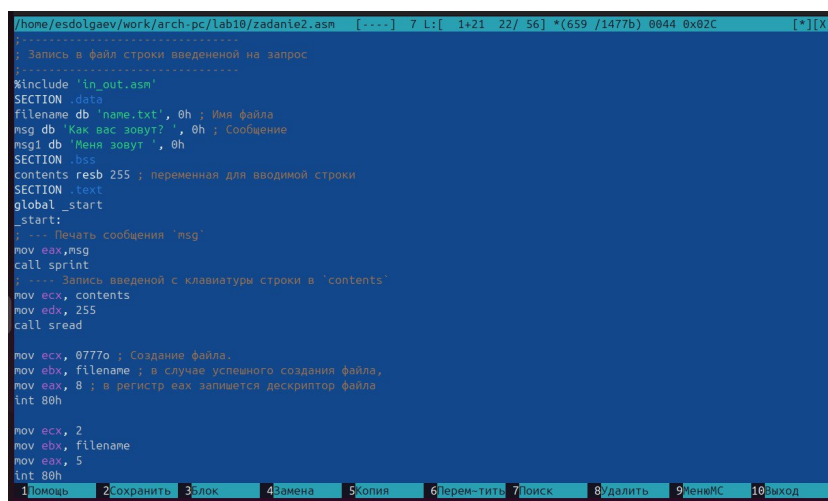
Рис. 4.7: Изменение прав доступа

4.1 Задание для самостоятельной работы

Напишем 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

Создим исполняемый файл и проверим его работу. Проверим наличие файла и его содержимое с помощью команд `ls` и `cat`(рис. 4.8, 4.9).

A screenshot of a text editor showing an assembly program. The code includes comments in Russian and assembly instructions. It defines a filename 'name.txt', a message 'Как вас зовут?', and a buffer 'contents' of size 255. The program prints the message, reads input from the user into 'contents', and then attempts to create the file. At the bottom, there is a menu bar with icons and labels: 1Помощь, 2Сохранить, 3Блок, 4Замена, 5Копия, 6Перем-тить, 7Поиск, 8Удалить, 9МенюМС, 10Выход.

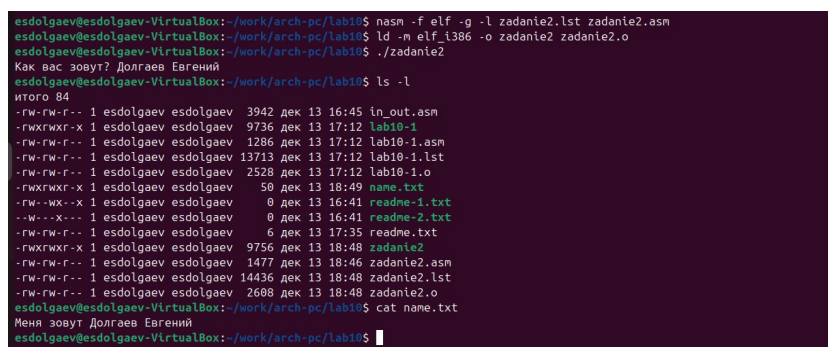
```
/home/esdolgaev/work/arch-pc/lab10/zadanie2.asm [----] 7 L: 1+21 22/ 56 *(659 /1477b) 0044 0x02C [*][X]
;-----
; Запись в файл строки введенной на запрос
;-----
%include 'in_out.asm'
SECTION .data
filename db 'name.txt', 0h ; Имя файла
msg db 'Как вас зовут? ', 0h ; Сообщение
msg1 db 'Меня зовут ', 0h
SECTION .bss
contents resb 255 ; переменная для вводимой строки
SECTION .text
global _start
_start:
; --- Печать сообщения 'msg'
mov eax,msg
call sprint
; --- Запись введенной с клавиатуры строки в 'contents'
mov ecx, contents
mov edx, 255
call sread

mov ecx, 0777o ; Создание файла.
mov ebx, filename ; в случае успешного создания файла,
mov ebx, 0 ; в регистр eax запишется дескриптор файла
int 80h

mov ecx, 2
mov ebx, filename
mov eax, 5
int 80h

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перем-тить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 4.8: Текст программы

A screenshot of a terminal window showing the compilation and execution of the assembly program. The user runs `nasm -f elf -g -l zadanie2.lst zadanie2.asm`, `ld -r elf i386 -o zadanie2 zadanie2.o`, and `./zadanie2`. The output shows the file `name.txt` being created and its contents. Then, the user runs `ls -l` showing the file permissions and `cat name.txt` showing the output.

```
esdolgaev@esdolgaev-VirtualBox: /work/arch-pc/lab10$ nasm -f elf -g -l zadanie2.lst zadanie2.asm
esdolgaev@esdolgaev-VirtualBox: /work/arch-pc/lab10$ ld -r elf i386 -o zadanie2 zadanie2.o
esdolgaev@esdolgaev-VirtualBox: /work/arch-pc/lab10$ ./zadanie2
Как вас зовут? Долгаев Евгений
esdolgaev@esdolgaev-VirtualBox: /work/arch-pc/lab10$ ls -l
итого 84
-rw-rw-r-- 1 esdolgaev esdolgaev 3942 дек 13 16:45 in_out.asm
-rwxrwxr-x 1 esdolgaev esdolgaev 9736 дек 13 17:12 lab10-1
-rw-rw-r-- 1 esdolgaev esdolgaev 1286 дек 13 17:12 lab10-1.asm
-rw-rw-r-- 1 esdolgaev esdolgaev 13713 дек 13 17:12 lab10-1.lst
-rw-rw-r-- 1 esdolgaev esdolgaev 2528 дек 13 17:12 lab10-1.o
-rwxrwxr-x 1 esdolgaev esdolgaev 50 дек 13 18:49 name.txt
-rw-rw-r-- 1 esdolgaev esdolgaev 0 дек 13 16:41 readme-1.txt
-rw-rw-r-- 1 esdolgaev esdolgaev 0 дек 13 16:41 readme-2.txt
-rw-rw-r-- 1 esdolgaev esdolgaev 6 дек 13 17:35 readme.txt
-rwxrwxr-x 1 esdolgaev esdolgaev 9756 дек 13 18:48 zadanie2
-rw-rw-r-- 1 esdolgaev esdolgaev 1477 дек 13 18:46 zadanie2.asm
-rw-rw-r-- 1 esdolgaev esdolgaev 14436 дек 13 18:48 zadanie2.lst
-rw-rw-r-- 1 esdolgaev esdolgaev 2608 дек 13 18:48 zadanie2.o
esdolgaev@esdolgaev-VirtualBox: /work/arch-pc/lab10$ cat name.txt
Меня зовут Долгаев Евгений
esdolgaev@esdolgaev-VirtualBox: /work/arch-pc/lab10$
```

Рис. 4.9: Создание и работа исполняемого файла

5 Выводы

В ходе выполнения лабораторной работы я приобрёл навыки написания программ для работы с файлами.

Список литературы