РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10

дисциплина: Архитектура компьютера

Студент: Ибрагимов Гаджимурад Шамильевич

Группа: НКАбд-02-25

МОСКВА

2025 г.

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

Приобретение навыков написания программ для работы с файлами.

2. ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

2.1. 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. Владельцем файла является его создатель. Для предоставления прав доступа другому пользователю или другой группе командой

chown [ключи] <новый_пользователь>[: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 rwx, где вместо любого символа может стоять дефис. Всего возможно 8 комбинаций, приведенных в таблице. 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.

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

2.2. 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 удаленияфайла,предоставление прав доступа

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.

2.2.1 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.

2.2.2 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

2.2.3 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.

2.2.4 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)

2.2.5. Удаление файла

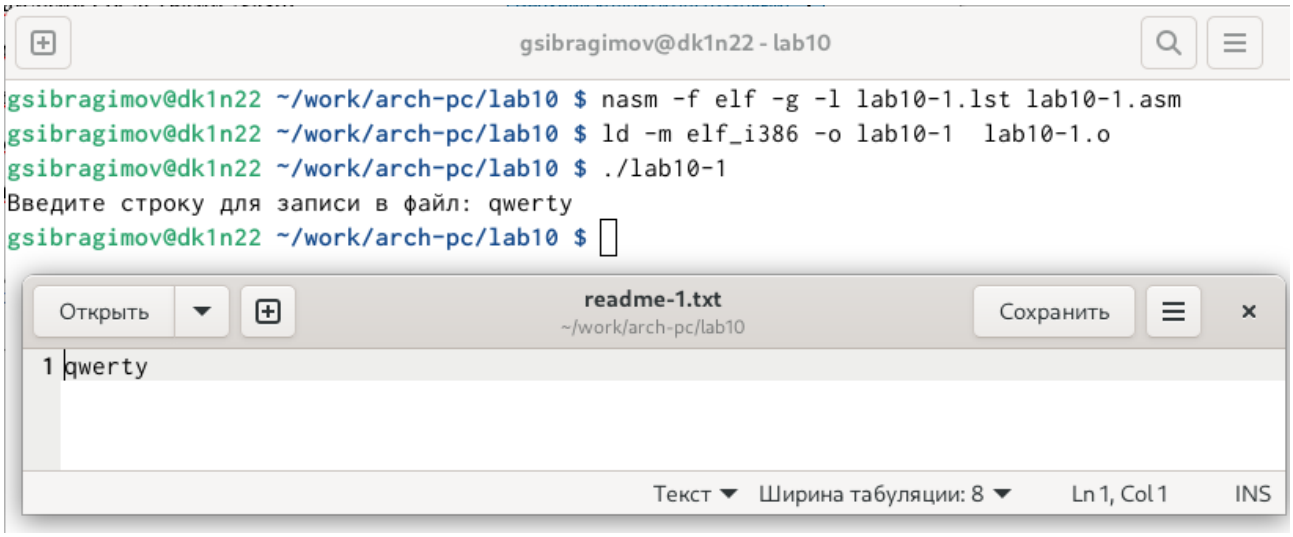
Удаление файла осуществляется системным вызовом sys_unlink, который использует один аргумент – имя файла в регистре EBX.

3. ВЫПОЛНЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

3.1.1 Создаем каталог для программам лабораторной работы № 10, перейдите в него и создайте файлы lab10-1.asm, readme-1.txt и readme-2.txt:

```
gsibragimov@dk1n22 ~/work/arch-pc/lab10 $ touch lab10-1.asm readme-1.txt readme-2.txt
gsibragimov@dk1n22 ~/work/arch-pc/lab10 $ ls
lab10-1.asm  readme-1.txt  readme-2.txt
gsibragimov@dk1n22 ~/work/arch-pc/lab10 $
```

3.1.2 Введите в файл lab10-1.asm текст программы (Программа записи в файл сообщения). Создаем исполняемый файл и проверьте его работу.



Происходит запись введеной строки в мой текстовый файл.

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

```
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ ls
in_out.asm  lab10-1.asm  lab10-1.o     readme-2.txt
lab10-1     lab10-1.lst  readme-1.txt
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ chmod a-x lab10-1
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ ./lab10-1
bash: ./lab10-1: Отказано в доступе
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $
```

По итогу запустить исполняемый файл мы не можем

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

```
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ ls
in_out.asm  lab10-1.asm  lab10-1.o     readme-2.txt
lab10-1     lab10-1.lst  readme-1.txt
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ chmod a+x lab10-1.asm
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ ./lab10-1.asm
./lab10-1.asm: строка 1: fg: нет управления заданиями
./lab10-1.asm: строка 2: SECTION: команда не найдена
./lab10-1.asm: строка 3: filename: команда не найдена
./lab10-1.asm: строка 3: Имя: команда не найдена
./lab10-1.asm: строка 4: msg: команда не найдена
./lab10-1.asm: строка 4: Сообщение: команда не найдена
./lab10-1.asm: строка 6: SECTION: команда не найдена
./lab10-1.asm: строка 7: contents: команда не найдена
./lab10-1.asm: строка 7: переменная: команда не найдена
./lab10-1.asm: строка 8: SECTION: команда не найдена
./lab10-1.asm: строка 9: global: команда не найдена
./lab10-1.asm: строка 10: _start:: команда не найдена
./lab10-1.asm: строка 11: синтаксическая ошибка рядом с неожиданным маркером «;»
./lab10-1.asm: строка 11: `; --- Печать сообщения `msg`'
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $
```

3.1.5 В соответствии с вариантом в таблице 10.4 предоставить права доступа к файлу readme-1.txt представленные в символьном виде, а для файла readme-2.txt – в двочном виде. Проверить правильность выполнения с помощью команды ls -l

13	-w- --x ---	110 011 001

Переведем обе команды в восьмеричную систему для удобства

1. Первый случай: -w- --x --- = `010 001 000 = 210`

2. Второй случай: 110 011 001 = rw- -wx -xx = 631

```
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ ls
in_out.asm  lab10-1  lab10-1.asm  lab10-1.lst  lab10-1.o  readme-1.txt  readme-2.txt
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ chmod 631 readme-2.txt
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ ls -l
итого 34
-rw-r--r-- 1 gsibragimov studsci  3942 ноя 14 17:50 in_out.asm
-rw-r--r-- 1 gsibragimov studsci  9768 дек  9 11:47 lab10-1
-rwxr-xr-x 1 gsibragimov studsci  1143 дек  9 11:44 lab10-1.asm
-rw-r--r-- 1 gsibragimov studsci 13489 дек  9 11:47 lab10-1.lst
-rw-r--r-- 1 gsibragimov studsci  2544 дек  9 11:47 lab10-1.o
-rw-r--r-- 1 gsibragimov studsci     7 дек  9 11:47 readme-1.txt
-rw--wx--x 1 gsibragimov studsci     0 дек  9 11:23 readme-2.txt
```

```
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ chmod 210 readme-1.txt
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ ls -l
итого 34
-rw-r--r-- 1 gsibragimov studsci  3942 ноя 14 17:50 in_out.asm
-rw-r--r-- 1 gsibragimov studsci  9768 дек  9 11:47 lab10-1
-rwxr-xr-x 1 gsibragimov studsci  1143 дек  9 11:44 lab10-1.asm
-rw-r--r-- 1 gsibragimov studsci 13489 дек  9 11:47 lab10-1.lst
-rw-r--r-- 1 gsibragimov studsci  2544 дек  9 11:47 lab10-1.o
--w---x--- 1 gsibragimov studsci     7 дек  9 11:47 readme-1.txt
-rw--wx--x 1 gsibragimov studsci     0 дек  9 11:23 readme-2.txt
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $
```

Мы можем заметить, что результат и предполагаемые права доступа совпадают

4. ВЫПОЛНЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

4.1. Выполнить задание

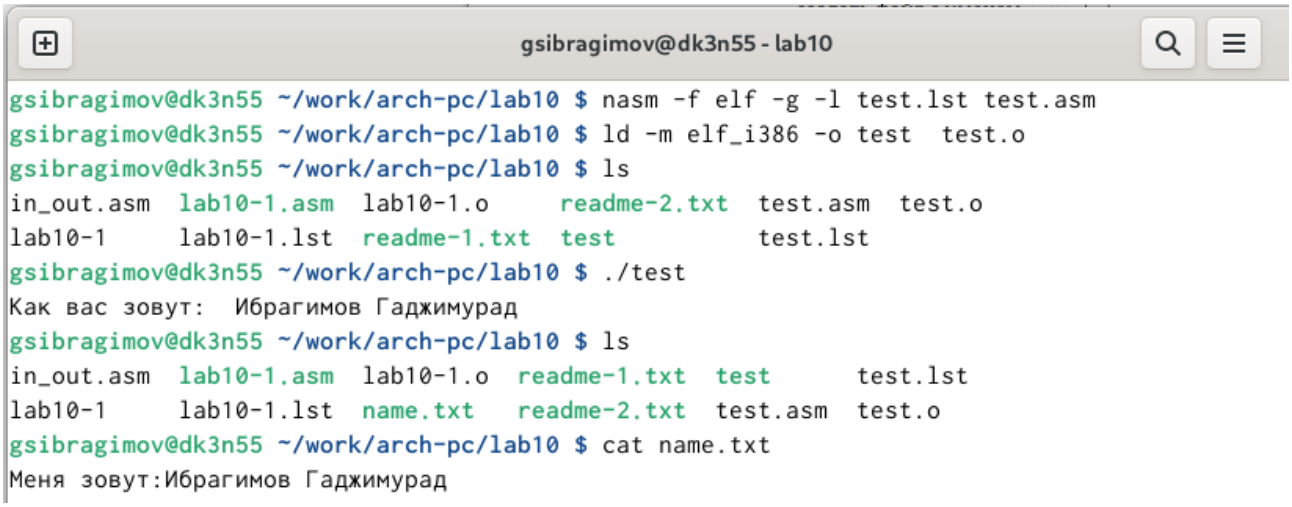
1. Напишите программу работающую по следующему алгоритму:

   - 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

Создаем файл test.asm. В нем будет текст моей программы.

```
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ ls
in_out.asm  lab10-1  lab10-1.asm  lab10-1.lst  lab10-1.o  readme-1.txt  readme-2.txt
gsibragimov@dk3n55 ~/work/arch-pc/lab10 $ touch test.asm
```

Создаем исполняемый файл и запускаем его. Проверяем содержимое файла name.txt



Текст моей программы

```
%include 'in_out.asm'
SECTION .data
msg db "Как вас зовут:  ",0h
filename db "name.txt", 0h
msg2 db "Меня зовут: ", 0h
section .bss
name resb 255

SECTION .text
global _start
_start:
mov eax, msg ; Вывод сообщения на экран "Как вас зовут?"
call sprint
mov ecx, name ; Ввод имени
mov edx, 55
call sread

mov ecx, 0777o ; Создание файла
mov ebx, filename
mov eax, 8
int 80h
mov edx, 20
mov ecx, msg2
mov ebx, eax
mov eax,4
int 80h

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, 5 ; смещение на 5 байт
mov ebx, eax ; дескриптор файла
mov eax, 19 ; номер системного вызова `sys_lseek`
int 80h ; вызов ядра

mov eax, name ; в `eax` запишется количество
call slen ; введенных байтов

mov edx, eax; Запись в конец файла по длине веденой строки
mov ecx, name;
mov eax, 4
int 80h

call quit
```

5. ВЫВОД

В ходе выполнения лабораторной работы я научился регулировать права доступа к файлам, а также научился работать с файлами средствами NASM

6. ИСТОЧНИКИ

1. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.

2. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science)