Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Мустафина Аделя Юрисовна

Содержание

# 1 Цель работы

Приобретение навыков написания программ для работы с файлами.

# 2 Задание

1. Выполнение лабораторной работы
2. Выполнение самостоятельной работы

# 3 Теоретическое введение

## 3.1 10.2.1. Права доступа к файлам

ОС GNU/Linux является многопользовательской операционной системой. И для обеспече- ния защиты данных одного пользователя от действий других пользователей существуют специальные механизмы разграничения доступа к файлам. Кроме ограничения доступа, дан- ный 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>

или

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

Набор прав доступа задается тройками битов и состоит из прав на чтение, запись и ис- полнение 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 х — исполнение), а дефис означает отсутствие права (нулевое значение соответствующего бита). Также права доступа могут быть представлены как вось- меричное число. Так, 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>

Так, например, права rwx r-x –x выглядят как двоичное число 111 101 001, или вось- меричное 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: 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 исполне- ние 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>

## 3.2 10.2.2. Работа с файлами средствами Nasm

В операционной системе Linux существуют различные методы управления файлами, на- пример, такие как создание и открытие файла, только для чтения или для чтения и записи, добавления в существующий файл, закрытия и удаления файла, предоставление прав досту- па. 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.

### 3.2.1 10.2.2.1. 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 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: • (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). С другими режимами доступа можно ознакомиться в https://man7.org/. 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 10.2.2.2. 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 полу- чить 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 10.2.2.3. 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 10.2.2.4. 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 10.2.2.5. 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: • (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 10.2.2.6. 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, запи- сывает в него сообщение и закрывает файл.

* Листинг 10.1. Программа записи в файл сообщения.\*

;--------------------------------  
; Запись в файл строки введененой 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 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

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

Создаю каталог для программ лабораторной работы №10(рис. 1).

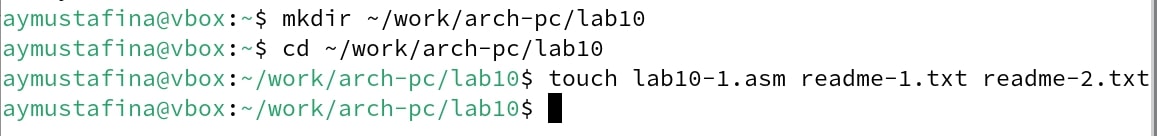


Рис. 1: Создание каталога

Ввожу текст из листинга (рис. 2).

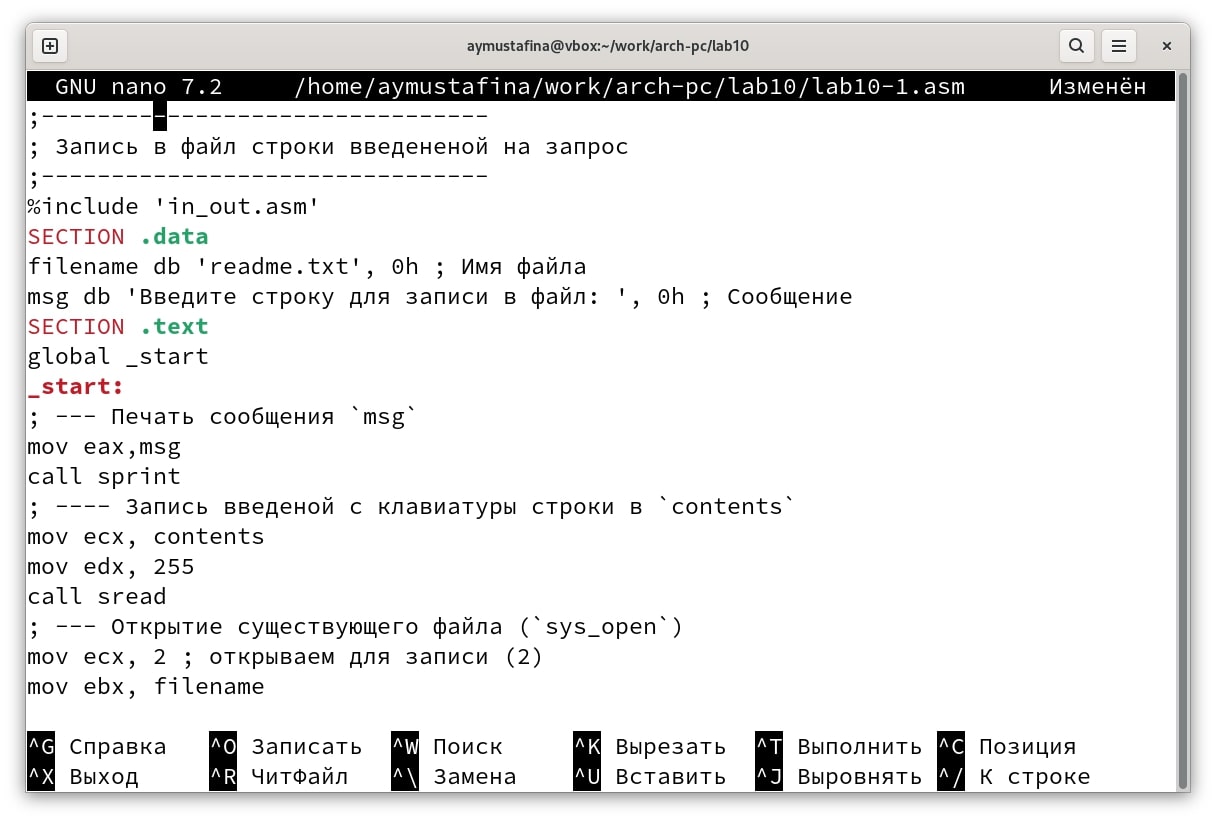


Рис. 2: Листинг

И проверяю его работу (рис. 3).

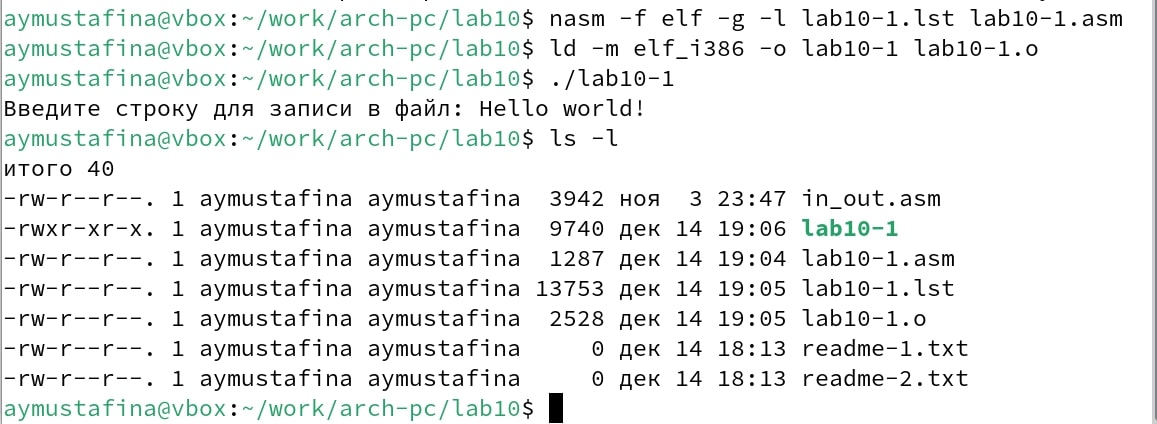


Рис. 3: Проверка программы

С помощью команды chmod меняю права доступа к исполняемому файлу lab10-1, запретив его выполнение.Причина, по которой я получаю сообщение (Доступ запрещен), заключается в том, что я убрала права на выполнение файла для текущего пользователя (владельца). В Unix-подобных системах файлы имеют три типа прав: чтение (r), запись (w) и выполнение (x). Если у пользователя нет права на выполнение файла, операционная система не позволит ему запустить этот файл (рис. 4).

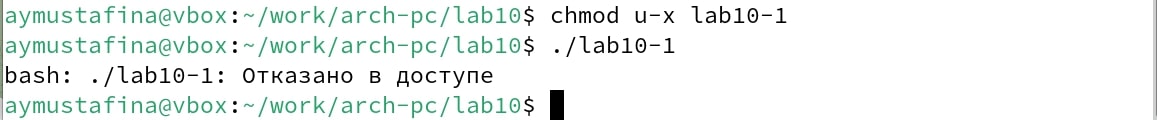


Рис. 4: Добавление команд

С помощью команды chmod меняю права доступа к файлу lab10-1.asm с исходным текстом программы, добавив права на исполнение. Пытаюсь выполнить его и снова получаю ошибку. Причина, по которой я получаю сообщение об ошибке, заключается в том, что файл lab10-1.asm является исходным текстом программы на ассемблере, а не исполняемым файлом. Даже если я добавлю права на выполнение, операционная система не сможет запустить файл, так как он не является скомпилированным исполняемым файлом.(рис. 5).

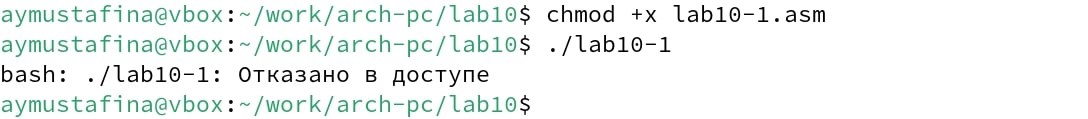


Рис. 5: Добавление команд

По 20 варианту предоставляю 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 (рис. 6).



Рис. 6: Команда

## 4.1 Самостоятельная работа

* Листинг для самостоятельной работы \*

%include 'in\_out.asm'  
  
SECTION .data  
filename db 'name.txt', 0  
prompt db 'Как Вас зовут?', 0  
intro db 'Меня зовут ', 0  
  
SECTION .bss  
name resb 255  
  
SECTION .text  
global \_start  
\_start:  
mov eax, prompt  
call sprint  
  
mov ecx, name  
mov edx, 255  
call sread  
  
mov eax, 8  
mov ebx, filename  
mov ecx, 0744o  
int 80h  
  
mov esi, eax  
  
mov eax, intro  
call slen  
mov edx, eax  
mov ecx, intro  
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

Запуск файла (рис. 7).

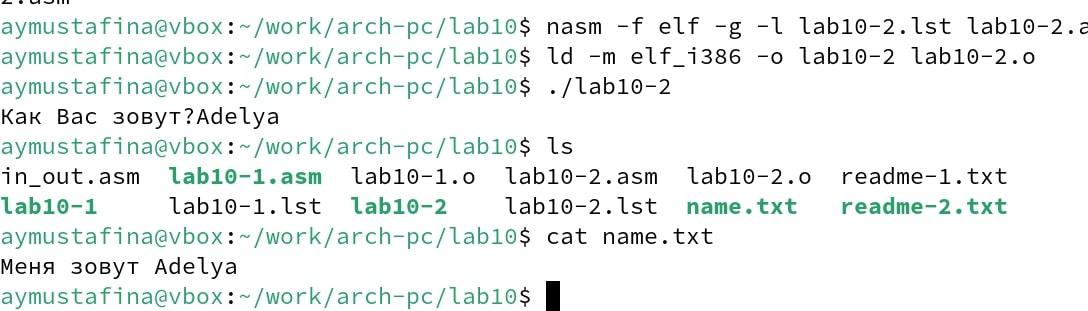


Рис. 7: Запуск

# 5 Выводы

В процессе выполнения лабораторной работы я научилась писать программы для работы с файлами и редактировать права для файлов.

# 6 Список литературы

1. [Лабораторная работа №10](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)