

Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Мустафина Аделя Юрисовна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	10.2.1. Права доступа к файлам	7
3.2	10.2.2. Работа с файлами средствами Nasm	9
3.2.1	10.2.2.1. Открытие и создание файла	9
3.2.2	10.2.2.2. Запись в файл	10
3.2.3	10.2.2.3. Чтение файла	11
3.2.4	10.2.2.4. Заккрытие файла	12
3.2.5	10.2.2.5. Изменение содержимого файла	12
3.2.6	10.2.2.6. Удаление файла	13
4	Выполнение лабораторной работы	16
4.1	Самостоятельная работа	18
5	Выводы	21
6	Список литературы	22

Список иллюстраций

4.1	Создание каталога	16
4.2	Листинг	16
4.3	Проверка программы	17
4.4	Добавление команд	17
4.5	Добавление команд	17
4.6	Команда	18
4.7	Запуск	20

Список таблиц

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 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 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. Полная строка прав доступа в символьном представлении имеет вид:

<права_владельца> <права_группы> <права_остальных>

Так, например, права *gwx 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: 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>
```

3.2 10.2.2. 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 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. 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 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).

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 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 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. Заккрытие 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 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 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, записывает в него сообщение и закрывает файл.

- Листинг 10.1. Программа записи в файл сообщения.*

```

;-----
; 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:~$
```

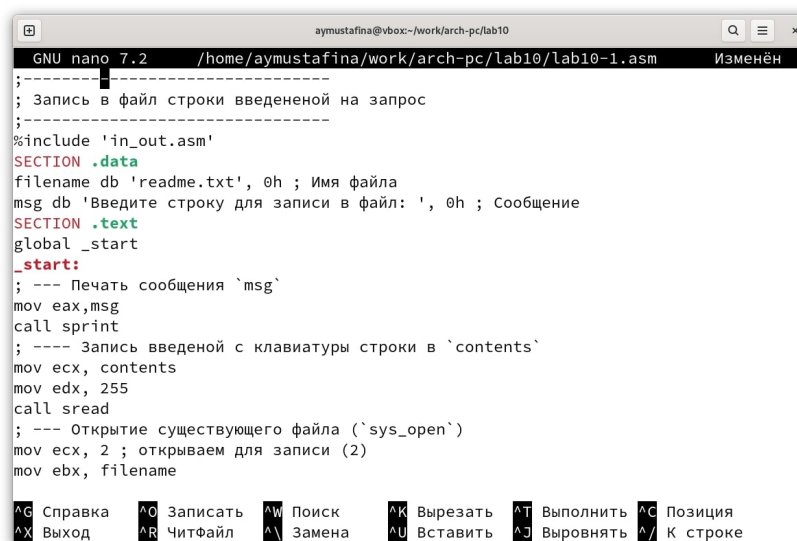
4 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы №10(рис. 4.1).

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

Рис. 4.1: Создание каталога

Ввожу текст из листинга (рис. 4.2).



```
GNU nano 7.2 /home/aymustafina/work/arch-pc/lab10/lab10-1.asm
;-----
; 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 .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
```

Рис. 4.2: Листинг

И проверяю его работу (рис. 4.3).


```

aymustafina@vbox:~/work/arch-pc/lab10$ nasm -f elf -g -l lab10-1.lst lab10-1.asm
aymustafina@vbox:~/work/arch-pc/lab10$ ld -m elf_i386 -o lab10-1 lab10-1.o
aymustafina@vbox:~/work/arch-pc/lab10$ ./lab10-1
Введите строку для записи в файл: Hello world!
aymustafina@vbox:~/work/arch-pc/lab10$ ls -l
итого 40
-rw-r--r--. 1 aymustafina aymustafina 3942 ноя  3 23:47 in_out.asm
-rwxr-xr-x. 1 aymustafina aymustafina 9740 дек 14 19:06 lab10-1
-rw-r--r--. 1 aymustafina aymustafina 1287 дек 14 19:04 lab10-1.asm
-rw-r--r--. 1 aymustafina aymustafina 13753 дек 14 19:05 lab10-1.lst
-rw-r--r--. 1 aymustafina aymustafina 2528 дек 14 19:05 lab10-1.o
-rw-r--r--. 1 aymustafina aymustafina  0 дек 14 18:13 readme-1.txt
-rw-r--r--. 1 aymustafina aymustafina  0 дек 14 18:13 readme-2.txt
aymustafina@vbox:~/work/arch-pc/lab10$ █

```

Рис. 4.3: 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. 4.4).

```

aymustafina@vbox:~/work/arch-pc/lab10$ chmod u-x lab10-1
aymustafina@vbox:~/work/arch-pc/lab10$ ./lab10-1
bash: ./lab10-1: Отказано в доступе
aymustafina@vbox:~/work/arch-pc/lab10$ █

```

Рис. 4.4: 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. 4.5).

```

aymustafina@vbox:~/work/arch-pc/lab10$ chmod +x lab10-1.asm
aymustafina@vbox:~/work/arch-pc/lab10$ ./lab10-1
bash: ./lab10-1: Отказано в доступе
aymustafina@vbox:~/work/arch-pc/lab10$

```

Рис. 4.5: Добавление команд

По 20 варианту предоставляю права доступа к файлу readme-1.txt представленные в символьном виде, а для файла readme-2.txt – в двоичном виде. Проверить правильность выполнения с помощью команды `ls -l` (рис. 4.6).

```
aymustafina@vbox:~/work/arch-pc/lab10$ chmod u=g, rw, o=w readme-1.txt
aymustafina@vbox:~/work/arch-pc/lab10$ chmod 011 111 readme-2.txt
chmod: невозможно получить доступ к '111': Нет такого файла или каталога
aymustafina@vbox:~/work/arch-pc/lab10$ chmod 111 readme-2.txt
aymustafina@vbox:~/work/arch-pc/lab10$ ls -l readme-1.txt readme-2.txt
----rw--w-. 1 aymustafina aymustafina 0 дек 14 18:13 readme-1.txt
---x--x--x. 1 aymustafina aymustafina 0 дек 14 18:13 readme-2.txt
aymustafina@vbox:~/work/arch-pc/lab10$
```

Рис. 4.6: 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. 4.7).

```
aymustafina@vbox:~/work/arch-pc/lab10$ nasm -f elf -g -l lab10-2.lst lab10-2.a
aymustafina@vbox:~/work/arch-pc/lab10$ ld -m elf_i386 -o lab10-2 lab10-2.o
aymustafina@vbox:~/work/arch-pc/lab10$ ./lab10-2
Как Вас зовут?Adelya
aymustafina@vbox:~/work/arch-pc/lab10$ ls
in_out.asm  lab10-1.asm  lab10-1.o  lab10-2.asm  lab10-2.o  readme-1.txt
lab10-1     lab10-1.lst  lab10-2   lab10-2.lst  name.txt   readme-2.txt
aymustafina@vbox:~/work/arch-pc/lab10$ cat name.txt
Меня зовут Adelya
aymustafina@vbox:~/work/arch-pc/lab10$ █
```

Рис. 4.7: 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