Лабораторная работа №10

Работа с файлами средствами Nasm

Хорошева Алёна Евгеньевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобрести навыки работы с файлами с помощью команд в терминале, а также с помощью системных вызовов внутри программ.

2 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
  Создать исполняемый файл и проверить его работу. Проверить наличие файла и его содержимое с помощью команд ls и cat.

3 Теоретическое введение

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] [: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. Б уква 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:

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

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:

3. Поместить номер системного вызова в регистр EAX;

4. Поместить аргументы системного вызова в регистрах EBX, ECX и EDX;

5. Вызов прерывания (int 80h);

6. 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.

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`

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

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

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 записСоздать исполняемый файл и проверить его работу. Проверить наличие файла и его содержимое с помощью команд ls и cати прочитанных данных

mov ebx, eax ; дескриптор файла

mov eax, 3 ; номер системного вызова sys_read

int 80h ; вызов ядра

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

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

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.

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

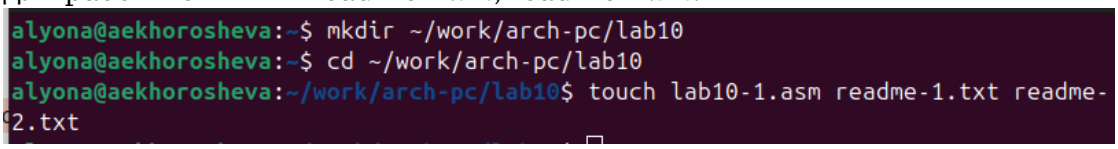
user@dk4n31:~$ cat readme.txt

Hello world!

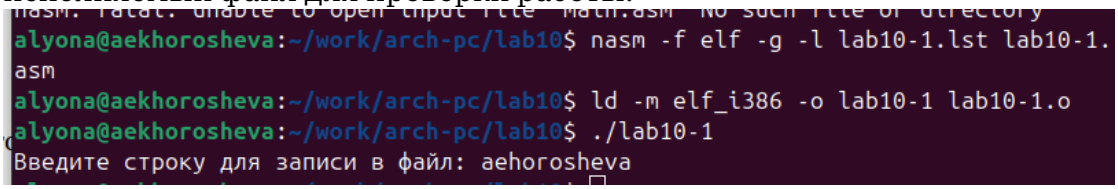
user@dk4n31:~$

4 Выполнение лабораторной работы

1. Создаем файл для написания программы - lab10-1.asm и текстовые файлы для работы с ними - readme-1.txt, readme-2.txt.



2. Вводим в файл программу из листинга 10.1 для записи в существующий файл сообщения, которое мы введем с клавиатуры в терминале. Создаём исполняемый файл для проверки работы:



Теперь проверим, что данный текст находится в файле readme-1.txt. Для этого используем команду cat.

```
alyona@aekhorosheva:~/work/arch-pc/lab10$ ./lab10-1
Введите строку для записи в файл: aehorosheva
alyona@aekhorosheva:~/work/arch-pc/lab10$ ls -l
итого 44
-rw-rw-r-- 1 alyona alyona  3942 ноя  9 18:27 in_out.asm
-rwxrwxr-x 1 alyona alyona  9736 дек 14 14:11 lab10-1
-rw-rw-r-- 1 alyona alyona  1299 дек 14 14:10 lab10-1.asm
-rw-rw-r-- 1 alyona alyona 14205 дек 14 14:11 lab10-1.lst
-rw-rw-r-- 1 alyona alyona  2512 дек 14 14:11 lab10-1.o
-rw-rw-r-- 1 alyona alyona    12 дек 14 14:11 readme-1.txt
-rw-rw-r-- 1 alyona alyona     0 дек 14 14:00 readme-2.txt
alyona@aekhorosheva:~/work/arch-pc/lab10$ cat readme-1.txt
aehorosheva
```

3. Далее командой chmod(“change mod”) меняем права доступа к исполняемому файлу. Для этого в аргументе указываем u-x, что значит владелец(u)-убрать право доступа(-)-право на исполнение(x). Далее проверяем, сможем ли мы запустить этот исполняемый файл - терминал выдаёт ошибку “Отказано в доступе”.

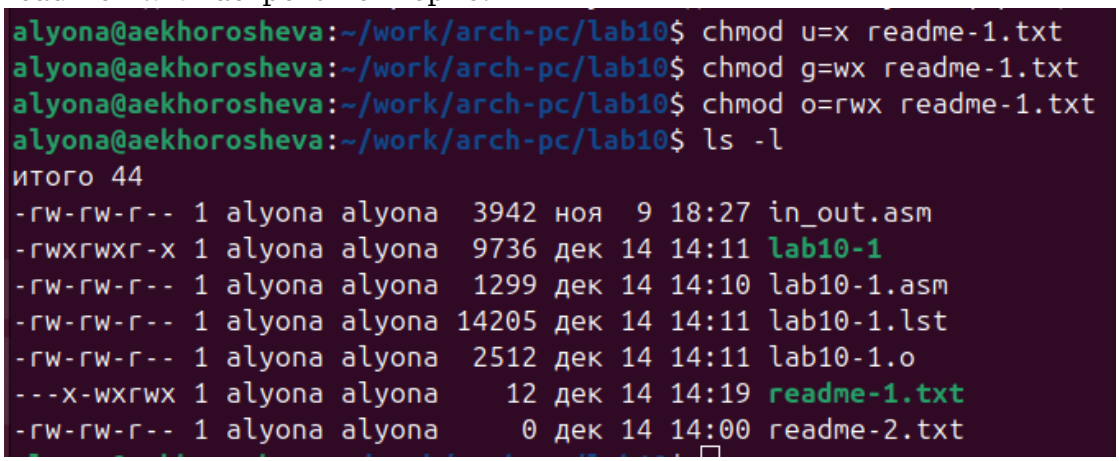
   Вернём обратно право на исполнение командой chmod u+x, она работает аналогично, только + означает здесь “добавить права доступа”. После этой настройки запускаем исполняемый файл - программа запускается, значит доступ на исполнение файла открыт.

```
alyona@aekhorosheva:~/work/arch-pc/lab10$ chmod u-x lab10-1
alyona@aekhorosheva:~/work/arch-pc/lab10$ ./lab10-1
bash: ./lab10-1: Отказано в доступе
alyona@aekhorosheva:~/work/arch-pc/lab10$ chmod u+x lab10-1
alyona@aekhorosheva:~/work/arch-pc/lab10$ ./lab10-1
Введите строку для записи в файл: hello
```

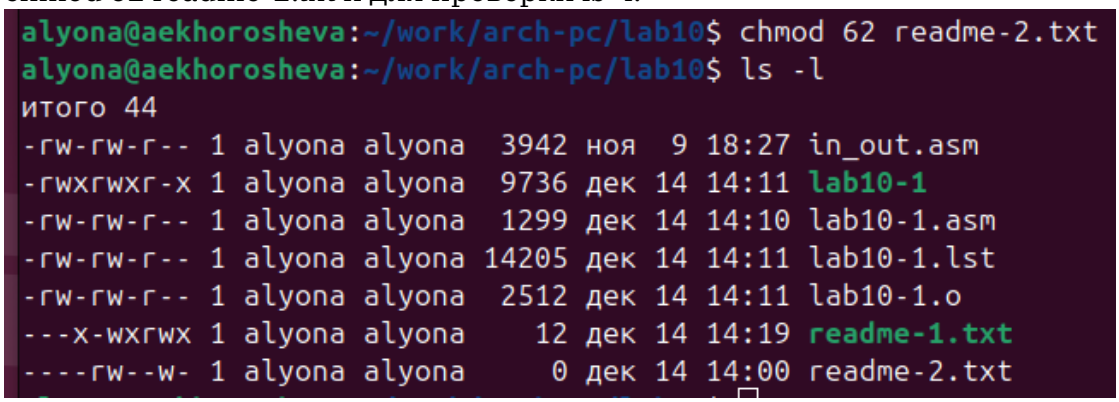
4. Теперь наша задача предоставить права доступа к файлу readme-1.txt в соответствии с 1-м вариантом (–x -wx rwx) таблицы 10.4. Делаем это в символьном виде: назначаем(=) соотвествующие права владельцу(u), группе пользователей(g), остальным пользователям(o). Право на чтение -

это r, на запись - w, на исполнение - x.

Затем командой ls с ключом -l проверяем, что права доступа к файлу readme-1.txt настроились верно:



5. Есть второй способ настройки прав доступа - в виде двоичной/восьмеричной записи. В таблице дана двооичная система(000 110 010), а в переводе в 8-чную мы получим число 62. Далее аналогично используем команду chmod 62 readme-2.txt и для проверки ls -l:



6. Напишем программу в новом файле lab10-2.asm, которая будет спрашивать, как зовут пользователя и затем введённое имя запишет в текстовый файл name.txt. Перед введенным именем также должен добавиться текст: “Меня зовут”.

   Воспользуемся предыдущей программой и внесем некоторые дополнения:

...

SECTION .data

...

name db 'Меня зовут', 0h ; для записи в файл "Меня зовут"

Далее нужно посчитать память, которую займет строка "Меня зовут", затем записать её в файл(sys_write):

; — Расчёт длины строки "Меня зовут"

mov eax, name

call slen

; — Записываем в файл name (sys_write)

mov edx, eax

mov ecx, name

mov ebx, esi

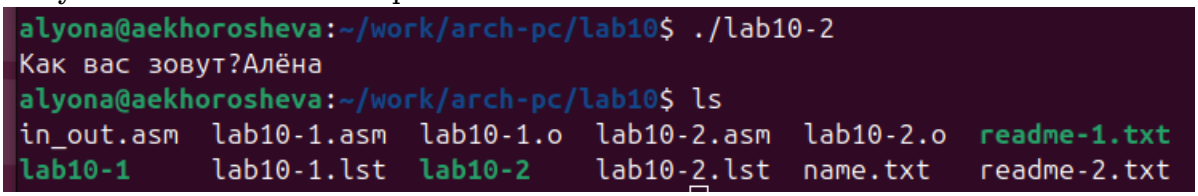
mov eax, 4

int 80h

xor ecx, ecx ; очищаем регистр ecx

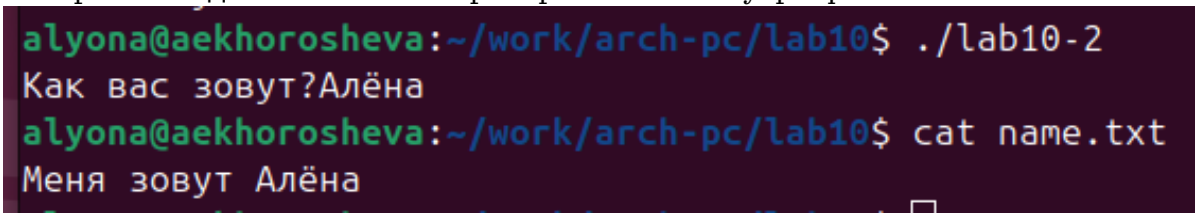
Остальной текст программы аналогичен lab10-1.asm.

Запускаем исполняемый файл:



Командой ls мы проверили, какие файлы находятся в каталоге lab10.

Теперь командой cat name.txt проверяем текст внутри файла:



Программа работает корректно и записывает в файл "Меня зовут ".

5 Выводы

В результате выполнения лабораторной работы были получены практические навыки работы с файлами через системные вызовы - sys_write и т.п., которые позволяют записывать текст в файл, открывать/закрывать существующий файл, изменять его содержимое.
Также были освоены настройки прав доступа к файлам командой chmod и различными её аргументами, как в символьном виде, так и в двоичном/восьмеричном.

Список литературы