Отчет по лабораторной работе №10

Дисциплина архитектура компьютера

Царёв Максим Александрович

Содержание

1 Цель работы

Приобретение навыков написания программ для работы с файлами.

2 Задание

1. Ввести текст программы
2. Изменить права доступа к исполняемому файлу,запретив его выполнение
3. Изменить права доступа к файлу с исходным текстом программы,добавив права на исполнение
4. 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
5. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

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] [:новая_группа]

или

chgrp [ключи] < новая_группа >

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 x — 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:

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

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.

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

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

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

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 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

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 введеноой 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 lab10,перехожу в него и создаю файлы lab10-1.asm, readme-1.txt и readme-2.txt:

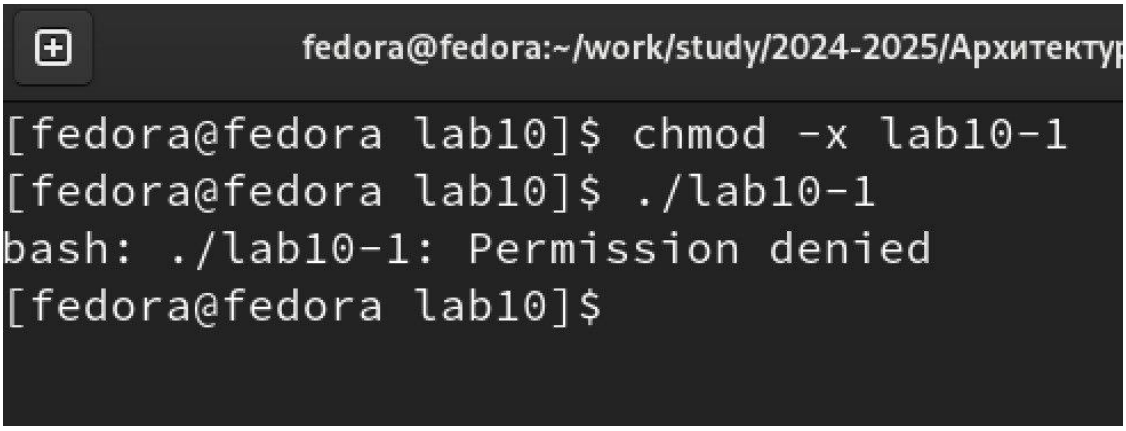
```
fedora@fedora:~/work/study/2024-2025/Архитектура компьютеров/study_2024-2025_ar-h-pc/lab10
[fedora@fedora lab10]$ touch lab10-1.asm readme-1.txt readme-2.txt
[fedora@fedora lab10]$ ls
lab10-1.asm  readme-1.txt  readme-2.txt
[fedora@fedora lab10]$
```

Ввожу в файл lab10-1.asm текст программы из листинга 10.1. Создаю исполняемый файл и проверьте его работу.

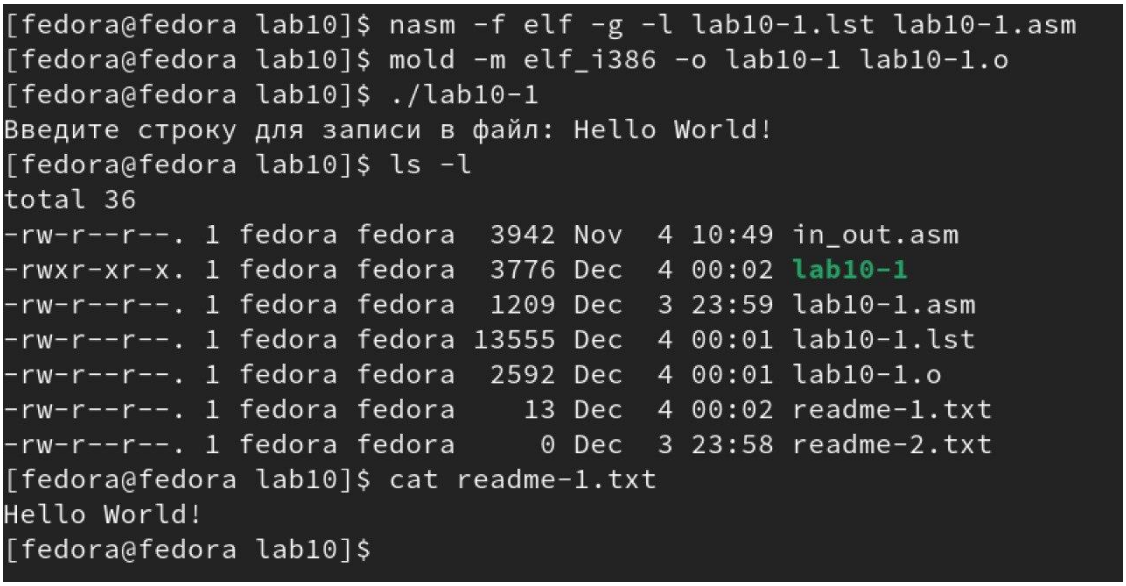
```
fedora@fedora:~/work/study/2024-2025/Архитектура компьютеров/study_2024-2025_ar-h-pc/lab10
  GNU nano 7.2                          lab10-1.asm
%include 'in_out.asm'

SECTION .data
filename db 'readme-1.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
```

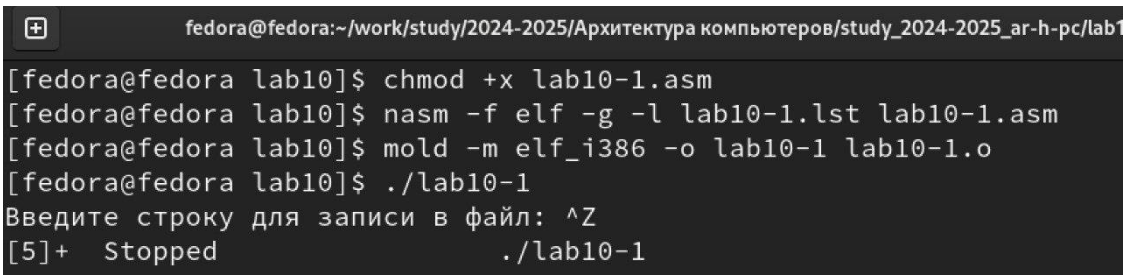
Проверяю работоспособность программы, программа работает корректно



С помощью команды chmod изменяю права доступа к исполняемому файлу lab10-1, запретив его выполнение.Пытаюсь выполнить файл,в результате выполнения отказано в доступе. Ошибка возникает, потому что права на выполнение файла были удалены. Когда права на выполнение (execute) отсутствуют, операционная система запрещает запускать файл, даже если это действительный исполняемый файл.



С помощью команды chmod изменяю права доступа к файлу lab10-1.asm с исходным текстом программы, добавив права на исполнение.Выполняю его.



В соответствии с вариантом в таблице 10.4, мой вариант - 12, предоставляю права доступа к файлу readme- 1.txt представленные в символьном виде, а для файла readme-2.txt – в двочном виде. Проверяю правильность выполнения с помощью команды ls -l.

```
[fedora@fedora lab10]$ chmod u=---,g=rw,o=w readme-1.txt
[fedora@fedora lab10]$ nasm -f elf -g -l lab10-1.lst lab10-1.asm
[fedora@fedora lab10]$ mold -m elf_i386 -o lab10-1 lab10-1.o
[fedora@fedora lab10]$ ./lab10-1
Введите строку для записи в файл: Hello World!
[fedora@fedora lab10]$ ls -l readme-1.txt readme-2.txt
----rw--w-. 1 fedora fedora 13 Dec  4 00:02 readme-1.txt
-rw-r--r--. 1 fedora fedora  0 Dec  3 23:58 readme-2.txt
[fedora@fedora lab10]$
```

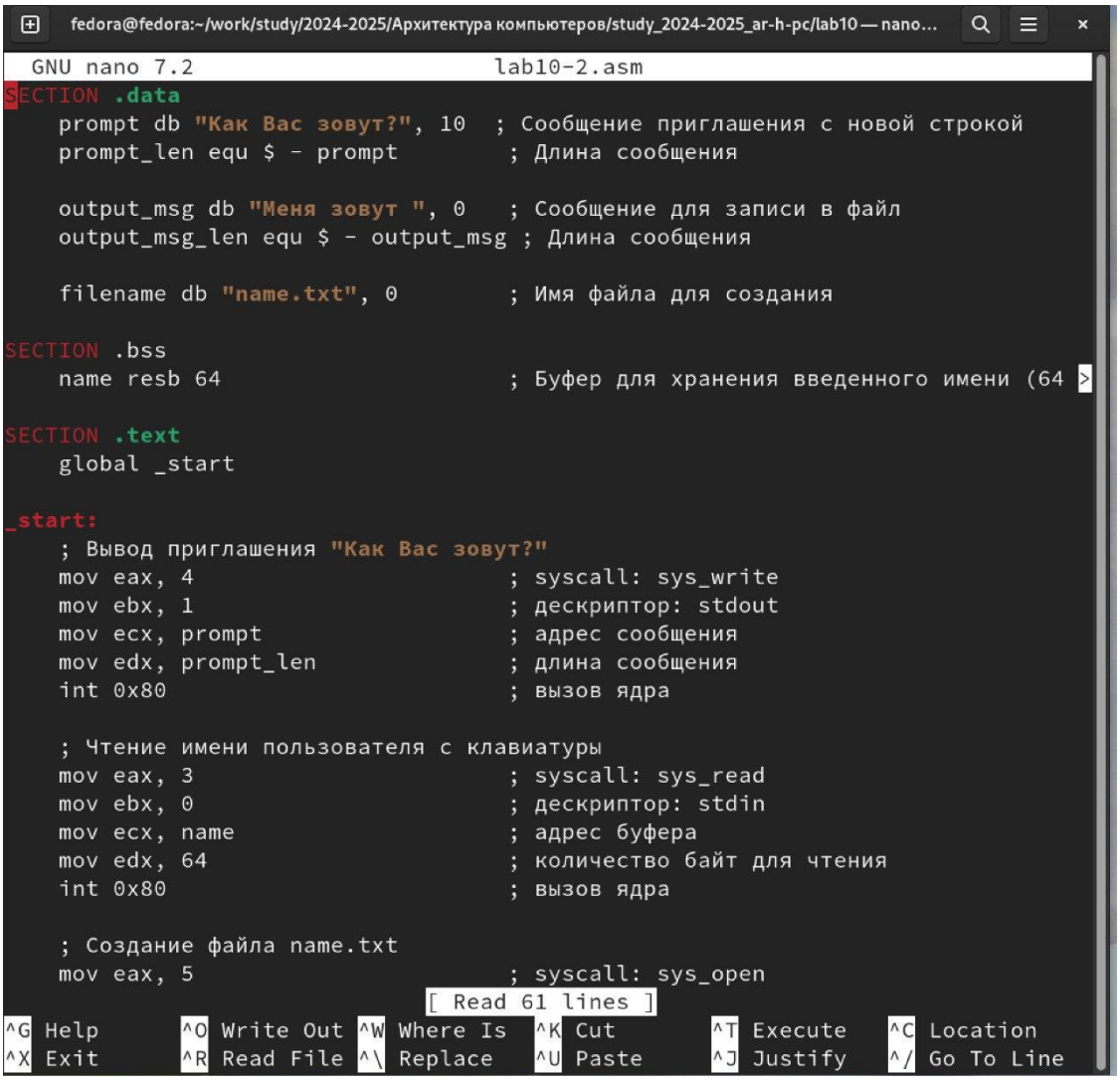
Права в двоичной системе для readme-2.txt Для файла readme-2.txt права в двоичном виде: 001 010 010. Эти права в восьмеричном представлении будут выглядеть как 122: Владелец (user): 001 — только выполнение. Группа (group): 010 — только запись. Остальные (others): 010 — только запись. Устанавливаю эти права с помощью команды chmod

```
[fedora@fedora lab10]$ chmod 137 readme-2.txt
[fedora@fedora lab10]$ ls -l readme-1.txt readme-2.txt
----rw--w-. 1 fedora fedora 13 Dec  4 00:02 readme-1.txt
---x-wxrwx. 1 fedora fedora  0 Dec  3 23:58 readme-2.txt
[fedora@fedora lab10]$
```

Права доступа выданы верно.

5 Выполнение заданий для самостоятельной работы

Создаю фалй lab10.asm,пишу программу по заданию.



создаю объектный файл и запускаю программу.

```
[fedora@fedora lab10]$ touch lab10-2.asm
[fedora@fedora lab10]$ nano lab10-2.asm
[fedora@fedora lab10]$ nasm -f elf32 lab10-2.asm -o lab10-2.o
[fedora@fedora lab10]$ mold -m elf_i386 -o lab10-2 lab10-.o
mold: fatal: cannot open lab10-.o: No such file or directory
[fedora@fedora lab10]$ mold -m elf_i386 -o lab10-2 lab10-2.o
[fedora@fedora lab10]$ ./lab10-2
Как Вас зовут?
Max Tsarev
```

В файл записалась строка с моим именем и фамилией.Программа работает корректно.

6 Выводы

Я приобретел навыки написания программ для работы с файлами.