

Отчет по лабораторной работе №10

Дисциплина архитектура компьютера

Ахатов Эмиль Эрнстович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	12
5	Выполнение заданий для самостоятельной работы	15
6	Выводы	17

Список иллюстраций

4.1	Создание файлов	12
4.2	Редактирование файла	12
4.3	Запуск исполняемого файла	13
4.4	Запуск исполняемого файла	13
4.5	Запуск исполняемого файла	13
4.6	Запуск исполняемого файла	14
4.7	Запуск исполняемого файла	14
5.1	Написание программы	15
5.2	Запуск исполняемого файла	15
5.3	Изменения в тексте файла	16

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 обеспечения защиты данных одного пользователя от действий других пользователей существуют специальные механизмы ограничения 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] [: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 `gwx`, 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 `г` — 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 `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 файла 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`

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.

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

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

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 ecx, 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 ecx, 5 int 80h
mov edx, 2 ; значение смещения – конец файла mov ecx, 0 ; смещение на 0 байт
mov ebx, eax ; дескриптор файла mov ecx, 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 ecx, 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 ecx, 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 ; Сооб-

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

```
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ touch lab10-1.asm r
eadme-1.txt readme-2.txt
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ls
lab10-1.asm  readme-1.txt  readme-2.txt
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$
```

Рис. 4.1: Создание файлов

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

```
%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 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
```

Рис. 4.2: Редактирование файла

Проверяю работоспособность программы, программа работает корректно

```

emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ nasm -f elf -g -l lab10-1.lst lab10-1.asm
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ld -m elf_i386 -o lab10-1 lab10-1.o
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ./lab10-1
Введите строку для записи в файл: Hello world!
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ls -l
итого 44
-rw-r--r--. 1 emil emil 3942 окт 26 11:58 in_out.asm
-rwxr-xr-x. 1 emil emil 9760 окт 29 11:13 lab10-1
-rw-r--r--. 1 emil emil 1209 окт 29 11:12 lab10-1.asm
-rw-r--r--. 1 emil emil 13555 окт 29 11:13 lab10-1.lst
-rw-r--r--. 1 emil emil 2544 окт 29 11:13 lab10-1.o
-rw-r--r--. 1 emil emil 13 окт 29 11:14 readme-1.txt
-rw-r--r--. 1 emil emil 0 окт 29 10:54 readme-2.txt
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ cat readme-1.txt
Hello world!

```

Рис. 4.3: Запуск исполняемого файла

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.

```

emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ chmod -x lab10-1
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ./lab10-1
bash: ./lab10-1: Отказано в доступе

```

Рис. 4.4: Запуск исполняемого файла

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

```

emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ chmod +x lab10-1.asm
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ nasm -f elf -g -l lab10-1.lst lab10-1.asm
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ld -m elf_i386 -o lab10-1 lab10-1.o
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ./lab10-1
Введите строку для записи в файл:

```

Рис. 4.5: Запуск исполняемого файла

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 двоичном виде. Проверяю правильность выполнения с помощью команды `ls -l`.

```

emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ chmod u=x,g=wx,o=rx readme-1.txt
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ nasm -f elf -g -l lab10-1.lst lab10-1.asm
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ld -m elf_i386 -o lab10-1 lab10-1.o
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ./lab10-1
Введите строку для записи в файл: Hello world!
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ls -l readme-1.txt readme-2.txt
---x-wxr-x. 1 emil emil 13 окт 29 11:37 readme-1.txt
-rw-r--r--. 1 emil emil 0 окт 29 10:54 readme-2.txt
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$

```

Рис. 4.6: Запуск исполняемого файла

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`

```

emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ chmod 122 readme-2.txt
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ls -l readme-1.txt readme-2.txt
---x-wxr-x. 1 emil emil 13 окт 29 11:37 readme-1.txt
---x-w--w-. 1 emil emil 0 окт 29 10:54 readme-2.txt

```

Рис. 4.7: Запуск исполняемого файла

Права доступа выданы верно.

5 Выполнение заданий для самостоятельной работы

Создаю файл lab10.asm, пишу программу по заданию.

```
SECTION .data
prompt db "Как Вас зовут?", 10 ; Сообщение приглашения с новой строкой
prompt_len equ $ - prompt ; Длина сообщения

output_msg db "Меня зовут ", 0 ; Сообщение для записи в файл
output_msg_len equ $ - output_msg ; Длина сообщения

filename db "name.txt", 0 ; Имя файла для создания

SECTION .bss
name resb 64 ; Буфер для хранения введенного имени (64 байта)

SECTION .text
global _start

_start:
; Вывод приглашения "Как Вас зовут?"
mov eax, 4 ; syscall: sys_write
mov ebx, 1 ; дескриптор: stdout
mov ecx, prompt ; адрес сообщения
mov edx, prompt_len ; длина сообщения
int 0x80 ; вызов ядра
```

Рис. 5.1: Написание программы

создаю объектный файл и запускаю программу.

```
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ nasm -f elf32 lab10-2.asm -o lab10-2.o
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ld -m elf_i386 -o lab10-2 lab10-2.o
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$ ./lab10-2
Как Вас зовут?
Ахатов Эмиль
emil@fedora:~/study_2024-2025_arhpc/labs/lab10/report/lab10$
```

Рис. 5.2: Запуск исполняемого файла

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

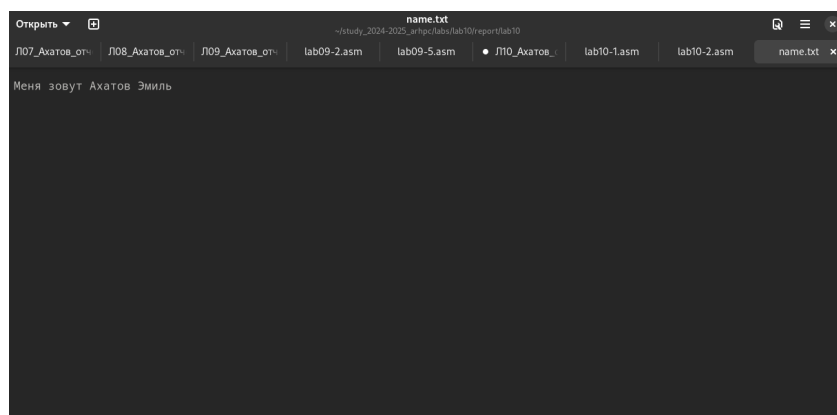


Рис. 5.3: Изменения в тексте файла

6 Выводы

Я приобрел навыки написания программ для работы с файлами.