

Отчёт по лабораторной работе №10

Дисциплина: архитектура компьютеров и операционные системы

Черная София Витальевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Написание программ для работы с файлами	9
4.2	Задание для самостоятельной работы	12
5	Выводы	16
6	Список литературы	17

Список иллюстраций

4.1	Создание файлов для лабораторной работы	9
4.2	Ввод текста программы из листинга 10.1	10
4.3	Запуск исполняемого файла	10
4.4	Запрет на выполнение файла	10
4.5	Добавление прав на исполнение	11
4.6	Предоставление прав доступа в символьном и двоичном виде . .	11
4.7	Написание текста программы	12
4.8	Запуск исполняемого файла и проверка его работы	12

Список таблиц

1 Цель работы

Приобретение навыков написания программ для работы с файлами.

2 Задание

1. Написание программ для работы с файлами.
2. Задание для самостоятельной работы.

3 Теоретическое введение

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.

Для изменения прав доступа служит команда `chmod`, которая понимает как символьное, так и числовое указание прав.

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.

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`.

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`.

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.

Для чтения данных из файла служит системный вызов `sys_read`, который использует следующие аргументы: количество байтов для чтения в регистре EDI, 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.

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.

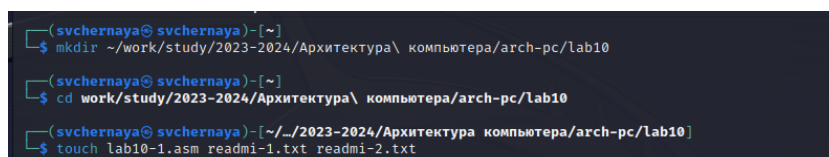
Для изменения содержимого файла служит системный вызов `sys_lseek`, который использует следующие аргументы: исходная позиция для смещения EDI, значение смещения в байтах в ECX, файловый дескриптор в EBX и номер системного вызова `sys_lseek` (19) в EAX. Значение смещения можно задавать в байтах.

Удаление файла осуществляется системным вызовом `sys_unlink`, который использует один аргумент – имя файла в регистре EBX.

4 Выполнение лабораторной работы

4.1 Написание программ для работы с файлами

Создаю каталог для программ лабораторной работы № 10, перехожу в него и создаю файлы lab10-1.asm, readme-1.txt и readme-2.txt. (рис. 4.1)



```
(svchernaya@svchernaya)~  
$ mkdir ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab10  
(svchernaya@svchernaya)~  
$ cd work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab10  
(svchernaya@svchernaya)~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10  
$ touch lab10-1.asm readme-1.txt readme-2.txt
```

Рис. 4.1: Создание файлов для лабораторной работы

Ввожу в файл lab10-1.asm текст программы, записывающей в файл сообщения, из листинга 10.1. (рис. 4.2)

```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10/lab10-1.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2 SECTION .data
3 filename db 'readme.txt', 0h ; Имя файла
4 msg db 'Введите строку для записи в файл: ', 0h ; Сообщение
5 SECTION .bss
6 contents resb 255 ; переменная для вводимой строки
7 SECTION .text
8 global _start
9 _start:
10 ; --- Печать сообщения `msg`
11 mov eax,msg
12 call sprint
13 ; --- Запись введенной с клавиатуры строки в `contents`
14 mov ecx, contents
15 mov edx, 255
16 call sread
17 ; --- Открытие существующего файла (`sys_open`)
18 mov ecx, 2 ; открываем для записи (2)
19 mov ebx, filename
20 mov eax, 5
21 int 80h
22 ; --- Запись дескриптора файла в `esi`
23 mov esi, eax
24 ; --- Расчет длины введенной строки
25 mov eax, contents ; в `eax` запишется количество
26 call slen ; введенных байтов
27 ; --- Записываем в файл `contents` (`sys_write`)
28 mov edx, eax
29 mov ecx, contents
30 mov ebx, esi
31 mov eax, 4
```

Рис. 4.2: Ввод текста программы из листинга 10.1

Создаю исполняемый файл и проверяю его работу. (рис. 4.3)

```
(svchernaya@svchernaya)-[~/./2023-2024/Архитектура компьютера/arch-pc/lab10]
$ nasm -f elf lab10-1.asm

(svchernaya@svchernaya)-[~/./2023-2024/Архитектура компьютера/arch-pc/lab10]
$ ld -m elf_i386 -o lab10-1 lab10-1.o

(svchernaya@svchernaya)-[~/./2023-2024/Архитектура компьютера/arch-pc/lab10]
$ ./lab10-1
Введите строку для записи в файл: Hola amigo!

(svchernaya@svchernaya)-[~/./2023-2024/Архитектура компьютера/arch-pc/lab10]
$ cat readme-1.txt
Hola amigo!
```

Рис. 4.3: Запуск исполняемого файла

Далее с помощью команды `chmod` у-х изменяю права доступа к исполняемому файлу `lab10-1`, запретив его выполнение и пытаюсь выполнить файл. (рис. 4.4)

```
(svchernaya@svchernaya)-[~/./2023-2024/Архитектура компьютера/arch-pc/lab10]
$ chmod u-x lab10-1

(svchernaya@svchernaya)-[~/./2023-2024/Архитектура компьютера/arch-pc/lab10]
$ ./lab10-1
zsh: permission denied: ./lab10-1
```

Рис. 4.4: Запрет на выполнение файла

Файл не выполняется, т.к в команде я указала “u” - владелец (себя), “-” - отменить набор прав, “x” - право на исполнение.

С помощью команды `chmod u+x` изменяю права доступа к файлу `lab10-1.asm` с исходным текстом программы, добавив права на исполнение, и пытаюсь выполнить его. (рис. 4.5)

```
(svchernaya@svchernaya)~[~/2023-2024/Архитектура компьютера/arch-pc/lab10]
$ chmod u+x lab10-1.asm

(svchernaya@svchernaya)~[~/2023-2024/Архитектура компьютера/arch-pc/lab10]
$ ./lab10-1.asm
./lab10-1.asm: 1: %include: not found
./lab10-1.asm: 2: SECTION: not found
./lab10-1.asm: 3: filename: not found
./lab10-1.asm: 3: Имя: not found
./lab10-1.asm: 4: msg: not found
./lab10-1.asm: 4: Сообщение: not found
./lab10-1.asm: 5: SECTION: not found
./lab10-1.asm: 6: contents: not found
./lab10-1.asm: 6: переменная: not found
./lab10-1.asm: 7: SECTION: not found
./lab10-1.asm: 8: global: not found
./lab10-1.asm: 9: _start:: not found
./lab10-1.asm: 10: Syntax error: ";" unexpected
```

Рис. 4.5: Добавление прав на исполнение

Текстовый файл начинает исполнение, но не исполняется, т.к не содержит в себе команд для терминала.

В соответствии со своим вариантом (10) в таблице 10.4 предоставляю права доступа к файлу `readme1.txt` представленные в символьном виде, а для файла `readme-2.txt` – в двоичном виде:

`r-- r-- rwx, 001 100 010`

И проверяю правильность выполнения с помощью команды `ls -l`. (рис. 4.6)

```
(svchernaya@svchernaya)~[~/2023-2024/Архитектура компьютера/arch-pc/lab10]
$ chmod 640 readme-1.txt # r-- r-- rwx

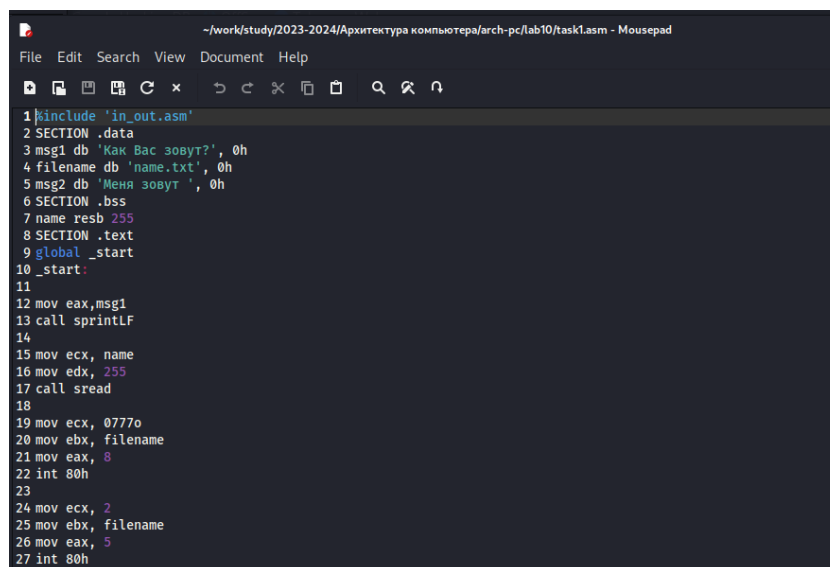
(svchernaya@svchernaya)~[~/2023-2024/Архитектура компьютера/arch-pc/lab10]
$ chmod 640 readme-2.txt # 001 100 010

(svchernaya@svchernaya)~[~/2023-2024/Архитектура компьютера/arch-pc/lab10]
$ ls -l
total 28
-rw-r--r-- 1 svchernaya svchernaya 3942 Dec 16 18:33 in_out.asm
-rw-r--xr-x 1 svchernaya svchernaya 9164 Dec 16 18:37 lab10-1
-rwxr--r-- 1 svchernaya svchernaya 1142 Dec 16 18:37 lab10-1.asm
-rw-r--r-- 1 svchernaya svchernaya 1472 Dec 16 18:37 lab10-1.o
-rw-r--r-- 1 svchernaya svchernaya 12 Dec 16 18:37 readme-1.txt
-rw-r--r-- 1 svchernaya svchernaya 0 Dec 16 18:26 readme-2.txt
```

Рис. 4.6: Предоставление прав доступа в символьном и двоичном виде

4.2 Задание для самостоятельной работы

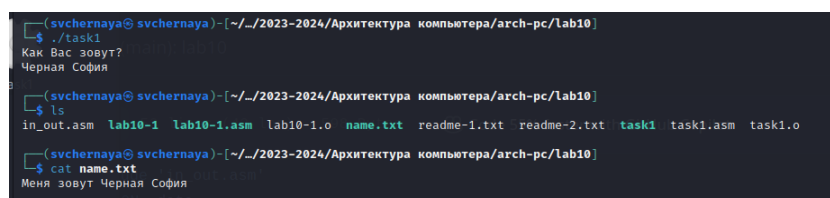
Пишу код программы, выводящей приглашения “Как Вас зовут?”, считывающей с клавиатуры фамилию и имя и создающую файл, в который записывается сообщение “Меня зовут”ФИ”. (рис. 4.7)



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Как Вас зовут?', 0h
4 filename db 'name.txt', 0h
5 msg2 db 'Меня зовут ', 0h
6 SECTION .bss
7 name resb 255
8 SECTION .text
9 global _start
10 _start:
11
12 mov eax, msg1
13 call sprintf
14
15 mov ecx, name
16 mov edx, 255
17 call sread
18
19 mov ecx, 0777o
20 mov ebx, filename
21 mov eax, 8
22 int 80h
23
24 mov ecx, 2
25 mov ebx, filename
26 mov eax, 5
27 int 80h
28
```

Рис. 4.7: Написание текста программы

Создаю исполняемый файл и проверяю его работу. Проверяю наличие файла и его содержимое с помощью команд ls и cat. (рис. 4.8)



```
(svchernaya@svchernaya)~[~/2023-2024/Архитектура компьютера/arch-pc/lab10]
$ ./task1
Как Вас зовут?
Черная София

(svchernaya@svchernaya)~[~/2023-2024/Архитектура компьютера/arch-pc/lab10]
$ ls
in_out.asm  lab10-1  lab10-1.asm  lab10-1.o  name.txt  readme-1.txt  readme-2.txt  task1  task1.asm  task1.o

(svchernaya@svchernaya)~[~/2023-2024/Архитектура компьютера/arch-pc/lab10]
$ cat name.txt
Меня зовут Черная София
```

Рис. 4.8: Запуск исполняемого файла и проверка его работы

Программа работает корректно.

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg1 db 'Как Вас зовут?', 0h
```

```
filename db 'name.txt', 0h
```

```
msg2 db 'Меня зовут ', 0h
```

```
SECTION .bss
```

```
name resb 255
```

```
SECTION .text
```

```
global _start
```

```
_start:
```

```
mov eax,msg1
```

```
call sprintf
```

```
mov ecx, name
```

```
mov edx, 255
```

```
call sread
```

```
mov ecx, 0777o
```

```
mov ebx, filename
```

```
mov eax, 8
```

```
int 80h
```

```
mov ecx, 2
```

```
mov ebx, filename
```

```
mov eax, 5
```

```
int 80h
```

```
mov esi, eax
```

```
mov eax, msg2
```

```
call slen
```

```
mov edx, eax
```

```
mov ecx, msg2
```

```
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
```

5 Выводы

Благодаря данной лабораторной работе я приобрела навыки написания программ для работы с файлами.

6 Список литературы

- [illegible]