

Отчёт по лабораторной работе №10

Дисциплина: архитектура компьютеров и операционные системы

Закиров Нурислам Дамирович

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	17
6	Список литературы	18

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	11
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	13

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

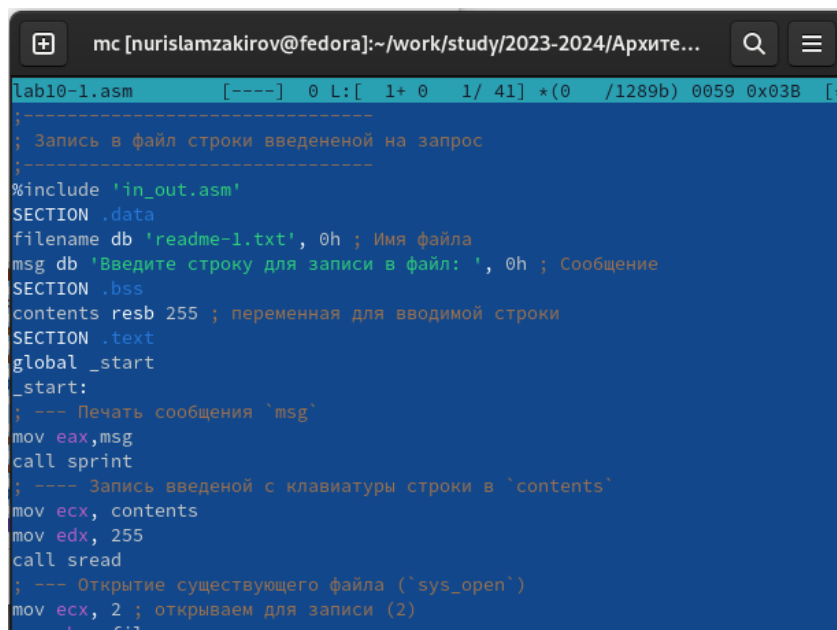
4.1 Написание программ для работы с файлами

Создаю каталог для программ лабораторной работы, перехожу в него и создаю файлы lab10-1.asm, readme-1.txt и readme-2.txt. (рис. 4.1)

```
[nurislamzakirov@fedora study_2023-2024_arh-pc]$ mkdir lab10  
[nurislamzakirov@fedora study_2023-2024_arh-pc]$ cd lab10  
[nurislamzakirov@fedora lab10]$ touch lab10-1.asm readme-1.txt readme-2.txt  
[nurislamzakirov@fedora lab10]$
```

Рис. 4.1: Создание файлов для лабораторной работы

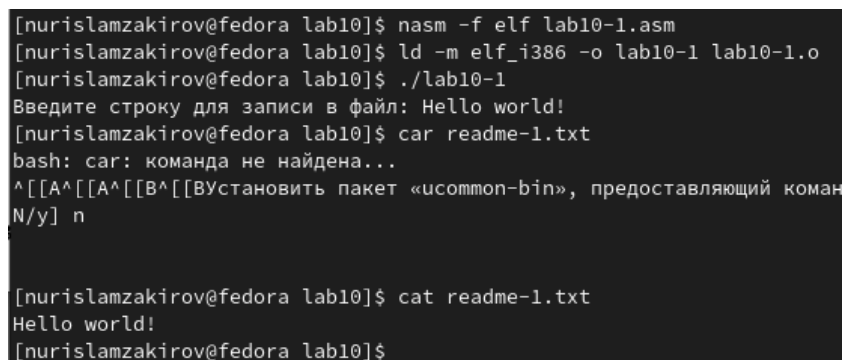
Ввожу в файл lab10-1.asm текст программы. (рис. 4.2)



```
lab10-1.asm [----] 0 L: [ 1+ 0 1/ 41] *(0 /1289b) 0059 0x03B [+
;
; -----
; Запись в файл строки введенной на запрос
; -----
%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
```

Рис. 4.2: Ввод текста программы из листинга 10.1

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



```
[nurislamzakirov@fedora lab10]$ nasm -f elf lab10-1.asm
[nurislamzakirov@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[nurislamzakirov@fedora lab10]$ ./lab10-1
Введите строку для записи в файл: Hello world!
[nurislamzakirov@fedora lab10]$ cat readme-1.txt
bash: cat: команда не найдена...
^[[A^[[A^[[B^[[BУстановить пакет «uccommon-bin», предоставляющий коман
N/y] n

[nurislamzakirov@fedora lab10]$ cat readme-1.txt
Hello world!
[nurislamzakirov@fedora lab10]$
```

Рис. 4.3: Запуск исполняемого файла

Далее я изменяю права доступа к файлу lab10-1, запретив его выполнение и пытаюсь его выполнить с помощью команды chmod u-x. (рис. 4.4)

```
[nurislamzakirov@fedora lab10]$ chmod u-x lab10-1
[nurislamzakirov@fedora lab10]$ ./lab10-1
bash: ./lab10-1: Отказано в доступе
[nurislamzakirov@fedora lab10]$
```

Рис. 4.4: Запрет на выполнение файла

В команде я указал «u» для владельца (себя), «-» для отмены набора прав и «x» для исполнения, поэтому файл не выполняется.

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

```
[nurislamzakirov@fedora lab10]$ chmod u+x lab10-1.asm
[nurislamzakirov@fedora lab10]$ ./lab10-1.asm
./lab10-1.asm: строка 1: синтаксическая ошибка рядом с неожиданным маркером «;»
./lab10-1.asm: строка 1: `;-----'
[nurislamzakirov@fedora lab10]$
```

Рис. 4.5: Добавление прав на исполнение

Текстовый файл начинает исполнение, но не выполняется из-за отсутствия команд терминала.

Так как мой вариант 10, я 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

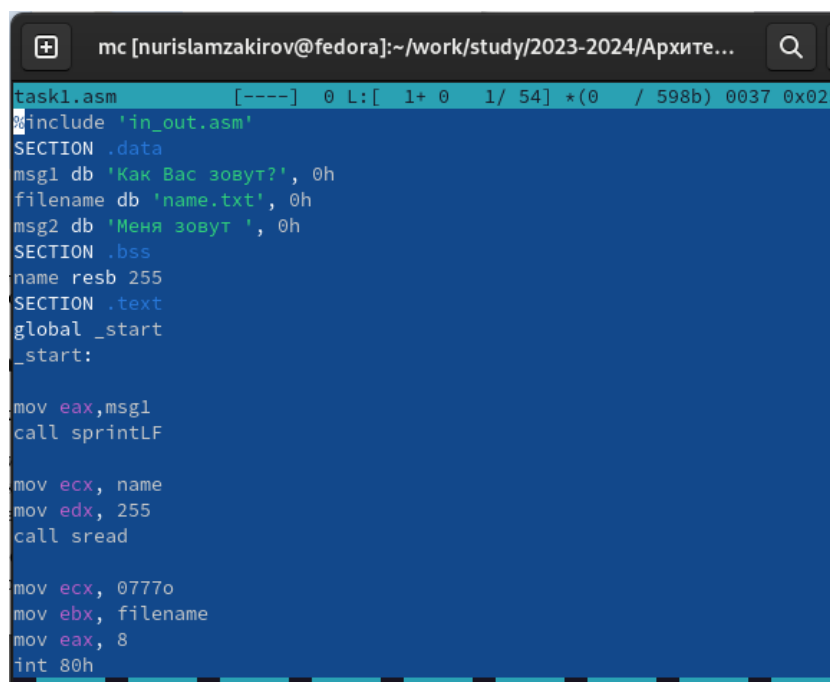
И проверяю корректность выполнения с помощью команды ls -l. (рис. 4.6)

```
[nurislamzakirov@fedora lab10]$ chmod 640 readme-1.txt # r-- r-- rwx
[nurislamzakirov@fedora lab10]$ chmod 640 readme-2.txt # 001 100 010
[nurislamzakirov@fedora lab10]$ ls -l
итого 20
-rw-r--r--. 1 nurislamzakirov nurislamzakirov 3773 ноя 15 14:22 in_out.asm
-rw-r-xr-x. 1 nurislamzakirov nurislamzakirov 1524 дек 16 01:33 lab10-1
-rwxr--r--. 1 nurislamzakirov nurislamzakirov 1289 дек 16 01:32 lab10-1.asm
-rw-r--r--. 1 nurislamzakirov nurislamzakirov 1472 дек 16 01:33 lab10-1.o
-rw-r-----. 1 nurislamzakirov nurislamzakirov 13 дек 16 01:33 readme-1.txt
-rw-r-----. 1 nurislamzakirov nurislamzakirov 0 дек 16 01:30 readme-2.txt
[nurislamzakirov@fedora lab10]$
```

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”Ф И”. (рис. 4.7)



```
task1.asm [----] 0 L:[ 1+ 0 1/ 54] *(0 / 598b) 0037 0x025
%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 printf

mov ecx, name
mov edx, 255
call sread

mov ecx, 0777o
mov ebx, filename
mov eax, 8
int 80h
```

Рис. 4.7: Написание текста программы

Я создаю файл для выполнения и проверяю его работу. Я использую команды ls и cat для проверки наличия файла и его содержимого. (рис. 4.8)

```

[nurislamzakirov@fedora lab10]$ touch task1.asm
[nurislamzakirov@fedora lab10]$ mc

[nurislamzakirov@fedora lab10]$ nasm -f elf task1.asm
[nurislamzakirov@fedora lab10]$ ld -m elf_i386 -o task1 task1.o
[nurislamzakirov@fedora lab10]$ ls
in_out.asm  lab10-1.asm  readme-1.txt  task1        task1.o
lab10-1     lab10-1.o    readme-2.txt  task1.asm
[nurislamzakirov@fedora lab10]$ ./task1
Как Вас зовут?
Закиров Нурислам
[nurislamzakirov@fedora lab10]$ ls
in_out.asm  lab10-1.asm  name.txt      readme-2.txt  task1.asm
lab10-1     lab10-1.o    readme-1.txt  task1        task1.o
[nurislamzakirov@fedora lab10]$ cat name.txt
Меня зовут Закиров Нурислам
[nurislamzakirov@fedora lab10]$ █

```

Рис. 4.8: Запуск исполняемого файла и проверка его работы

Программа работает правильно.

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

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005 — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).