11. Лабораторная работа №11. Работа с файлами средствами Nasm

11.1. Цель работы

Приобретение навыков написания программ для работы с файлами.

11.2. Теоретическое введение

11.2.1. 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] <новый_пользователь>[:новая_группа] <файл>
```

или

```
chgrp [ключи] < новая_группа > <файл>
```

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, приведенных в таблице 11.1. 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 восьмеричное число. Так, права доступа `rw-` (чтение и запись, без исполнения) понимаются как три двоичные цифры 110 или как восьмеричная цифра 6.

Таблица 11.1. Двоичный, буквенный и восмеричный способ записи триады прав доступа

Двоичный	Буквенный	Восмеричный
111	rwx	7
110	rw-	6
101	r-x	5
100	r--	4
011	-wx	3
010	-w-	2
001	--x	1
000	---	0

Полная строка прав доступа в символьном представлении имеет вид:

```
<права_владельца> <права_группы> <права_остальных>
```

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
```

В первой колонке показаны текущие права доступа, далее указан владелец файла и группа:

```
-         rwx         r-x         r-x         1 debugger user
|         |           |           |
тип     владелец    группа    остальные
type     owner      group      others
```

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 <категория><действие><набор_прав><файл>
```

Возможные значения аргументов команды представлены в таблице 11.2.

Таблица 11.2. Возможные значения аргументов команды `chmod`

Категория	Обозначение	Значение
Принадлежность	u	Владелец
	g	Группа владельца
	o	Прочие пользователи
	a	Все пользователи, то есть «a» эквивалентно «ugo»
Действие	+	Добавить набор прав
	-	Отменить набор прав
	=	Назначить набор прав
Право	r	Право на чтение
	w	Право на запись
	x	Право на исполнение

11.2.2. 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.

В таблице 11.3 приведены системные вызовы для обработки файлов.

Таблица 11.3. Системные вызовы для обработки файлов

Имя системного вызова	eax	ebx	ecx	edx
sys_read	3	дескриптор файла	адрес в памяти	количество байтов
sys_write	4	дескриптор файла	строка	количество байтов
sys_open	5	имя файла	режим доступа к файлу	права доступа к файлу
sys_close	6	дескриптор файла	—	—
sys_creat	8	имя файла	права доступа к файлу	—
sys_unlink	10	имя файла	—	—

Имя системного вызова	eax	ebx	ecx	edx
sys_lseek	19	имя файла	значение смещения в байтах	позиция для смещения

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.

11.2.2.1. 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
```

11.2.2.2. 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
```

11.2.2.3. 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 записи прочитанных
 ↪  данных
mov  ebx, eax         ; дескриптор файла
mov  eax, 3           ; номер системного вызова `sys_read`
int  80h              ; вызов ядра
```

11.2.2.4. 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
```

11.2.2.5. 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
```

11.2.2.6. 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.

Листинг 11.1. Программа записи в файл сообщения.

```
;------------------------------
; Запись в файл строки введеноей 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:~$
```

11.3. Порядок выполнения лабораторной работы

1. Создайте каталог для программам лабораторной работы № 11, перейдите в него и создайте файл `lab11-1.asm` и `readme.txt`:

```
mkdir  ~/work/arch-pc/lab09
cd  ~/work/arch-pc/lab09
touch lab11-1.asm readme.txt
```

2. Введите в файл `lab11-1.asm` текст программы из листинга 11.1 (Программа записи в файл сообщения). Создайте исполняемый файл и проверьте его работу.

3. С помощью команды `chmod` измените права доступа к исполняемому файлу `lab11-1`, запретив его выполнение. Попытайтесь выполнить файл. Объясните результат.

4. С помощью команды `chmod` измените права доступа к файлу `lab11-1.asm` с исходным текстом программы, добавив права на исполнение. Попытайтесь выполнить его и объясните результат.

5. Предоставить права доступа к файлу `readme.txt` в соответствии с вариантом в таблице 11.4. Проверить правильность выполнения с помощью команды `ls -l`.

11.4. Варианты заданий

Таблица 11.4. Наборы прав доступа для задания №2.

Номер варианта	В символьном виде	В двочной системе
1	`--x -wx rwx`	000 110 010
2	`rwx rwx --x`	110 111 101
3	`r-x -wx rw-`	011 101 011
4	`-w- --- -w-`	001 011 110

Номер варианта	В символьном виде	В двочной системе
5	`--x -w- r-x`	001 101 010
6	`-w- r-x -w-`	011 001 111
7	`rw- rwx rw-`	101 111 111
8	`rw- -wx --x`	010 001 000
9	`--x -w- -w-`	001 011 101
10	`r-- r-- rwx`	001 100 010
11	`--x r-- -w-`	000 100 111
12	`--x -wx r-x`	001 010 010
13	`-w- --x ---`	110 011 001
14	`r-x rwx rwx`	110 111 110
15	`-wx --x rwx`	010 101 010
16	`--x r-x -w-`	001 010 101
17	`r-x -wx rw-`	010 000 010
18	`-wx r-x -wx`	101 011 110
19	`rw- rwx r-x`	111 111 001
20	`--- rw- -w-`	001 011 111

11.5. Задание для самостоятельной работы

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`.

11.6. Содержание отчёта

Отчёт должен включать:

- Титульный лист с указанием номера лабораторной работы и ФИО студента.
- Формулировка цели работы.
- Описание результатов выполнения лабораторной работы:
  - описание выполняемого задания;
  - скриншоты (снимки экрана), фиксирующие выполнение заданий лабораторной работы;
  - комментарии и выводы по результатам выполнения заданий.
- Описание результатов выполнения заданий для самостоятельной работы:
  - описание выполняемого задания;
  - скриншоты (снимки экрана), фиксирующие выполнение заданий;
  - комментарии и выводы по результатам выполнения заданий;
  - листинги написанных программ (текст программ).
- Выводы, согласованные с целью работы.

Отчёт по выполнению лабораторной работы оформляется в формате `Markdown`. В качестве отчёта необходимо предоставить отчёты в 3 форматах: `pdf`, `docx` и `md`. А также файлы с исходными текстами написанных при выполнении лабораторной работы программ (файлы `*.asm`). Файлы необходимо загрузить на странице курса в ТУИС в задание к соответствующей лабораторной работе и загрузить на Github.

11.7. Вопросы для самопроверки

1. Каким образом в Unix-подобных ОС определяются права доступа к файлу?
2. Как ОС определяет, является ли файл исполняемым? Как регулировать права на чтение и запись?
3. Как разграничить права доступа для различных категорий пользователей?
4. Какой номер имеют системные вызовы `sys_read`, `sys_write`, `sys_open`, `sys_close`, `sys_creat`.
5. Какие регистры и как используют системные вызовы `sys_read`, `sys_write`, `sys_open`, `sys_close`, `sys_creat`.
6. Что такое дескриптор файла?