Отчёт по лабораторной работе №10

Дисциплина: Архитектура компьютера

София Андреевна Кудякова

Содержание

Список иллюстраций

1 Цель работы

Цель данной лабораторной работы - научиться писать программы для работы с файлами.

2 Задания

1. Написание программ для работы с файлами
2. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

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] [: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. 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. 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. 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. 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). 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. 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. 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 EDX, значение смещения в байтах в ECX, файловый дескриптор в EBX и номер системного вызова sys_lseek (19) в EAX. Удаление файла осуществляется системным вызовом sys_unlink, который использует один аргумент – имя файла в регистре EBX.

4 Выполнение лабораторной работы

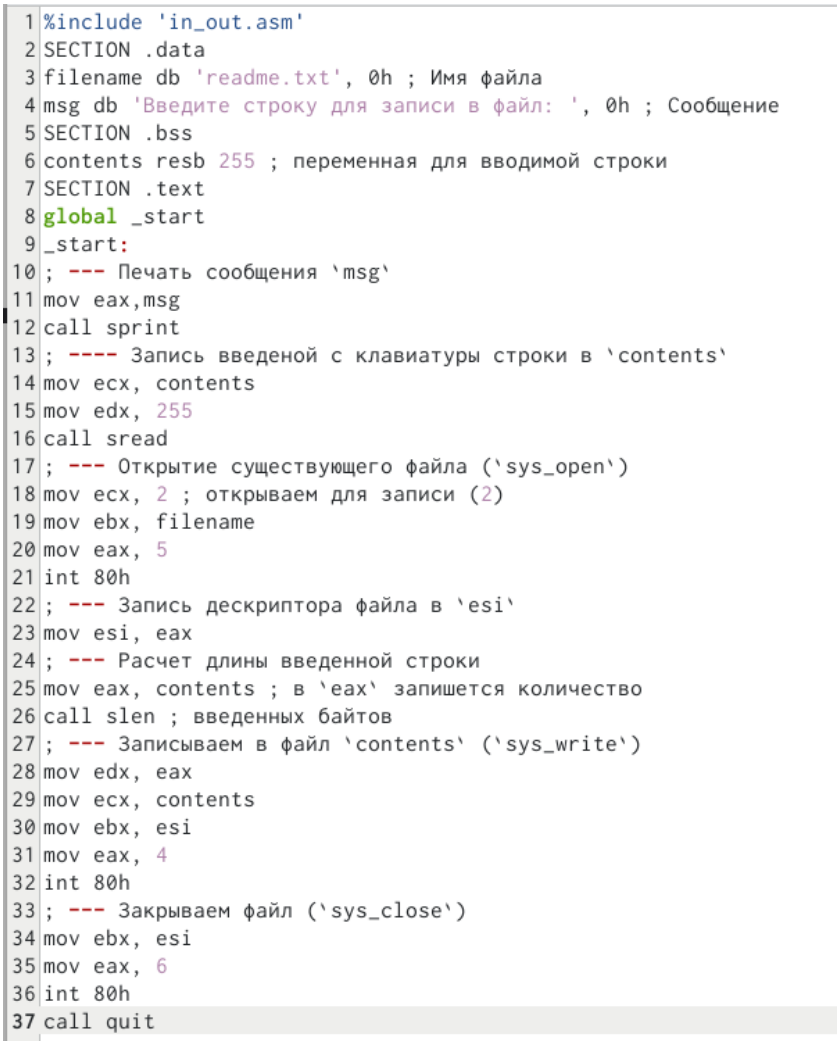
4.1 Написание программ для работы с файлами

Ввожу команду mkdir, с помощью которой создаю директорию, в которой буду создавать файлы. Перехожу в нее. С помощью команды touch создаю файлы lab09-1.asm, readme-1.txt и readme-2.txt. (рис. 4.1).

```
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ touch lab10-1.asm readme-1.txt readme-2.txt
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $
```

Рис. 4.1: Создание директории и файлов

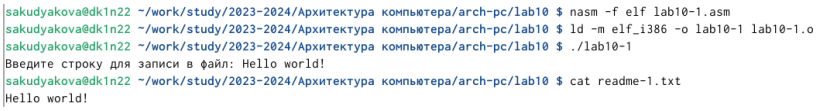
Ввожу в файл lab10-1.asm текст программы из листинга 10.1. (рис. 4.2).



```
%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
```

Рис. 4.2: Редактирование файла

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



```
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ nasm -f elf lab10-1.asm
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ ./lab10-1
Введите строку для записи в файл: Hello world!
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ cat readme-1.txt
Hello world!
```

Рис. 4.3: Запуск программы файла

С помощью команды chmod u-x изменяю права доступа к исполняемому файлу lab10-1,n запретив его выполнение. Попытаюсь выполнить файл. (рис. 4.4).

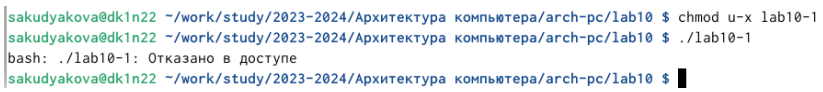


Рис. 4.4: Запрет на выполнение команды

Программа выдает “Отказано в доступе”, так как в команде указано u-x(“u” - владелец(я), “-” - удалить разрешение, “x” - разрешение на выполнение)

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

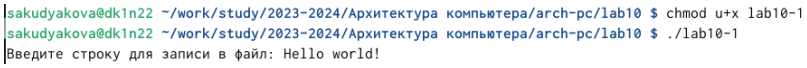


Рис. 4.5: Добавления прав на исполнение

Программа срабатывает, так как в команде указано u+x(“u” - владелец(я), “+” - добавить разрешение, “x” - разрешение на выполнение)

В соответствии с вариантом(мой вариант - 14) 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.(рис. 4.6).

Предоставление прав доступа в символьном виде и в двоичной системе

Рис. 4.6: Предоставление прав доступа в символьном виде и в двоичной системе

4.2 Выполнение заданий для самостоятельной работы

Пишу код программы, которая работает по 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. (рис. 4.7).

```
%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 sprintLF

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
```

Рис. 4.7: Редактирование файла

Создаю исполняемый файл, запускаю его и проверяю его работу. (рис. 4.8).

```
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ touch lab10-2.asm
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ nasm -f elf lab10-2.asm
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ ld -m elf_i386 -o lab10-2 lab10-2.o
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ ./lab10-2
Как Вас зовут?
Кудякова София
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ ls
in_out.asm  lab10-1  lab10-1.asm  lab10-1.o  lab10-2  lab10-2.asm  lab10-2.o  name.txt  readme-1.txt  readme-2.txt
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $ cat name.txt
Меня зовут Кудякова София
sakudyakova@dk1n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab10 $
```

Рис. 4.8: Запуск программы файла

Программа отработала корректно.

Листинг 4.2.1.

```
%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 sprintLF

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

В ходе данной лабораторной работы я научилась писать программы для работы с файлами.

Список литературы

Архитектура ЭВМ