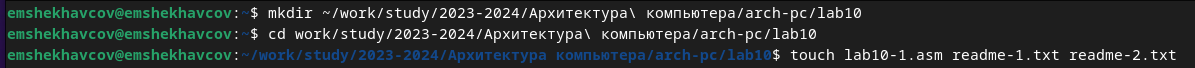
# 1 Отчёт по лабораторной работе №10

## 1.1 Дисциплина: архитектура компьютеров и операционные системы

### 1.1.1 Шехавцов Евгений Михайлович

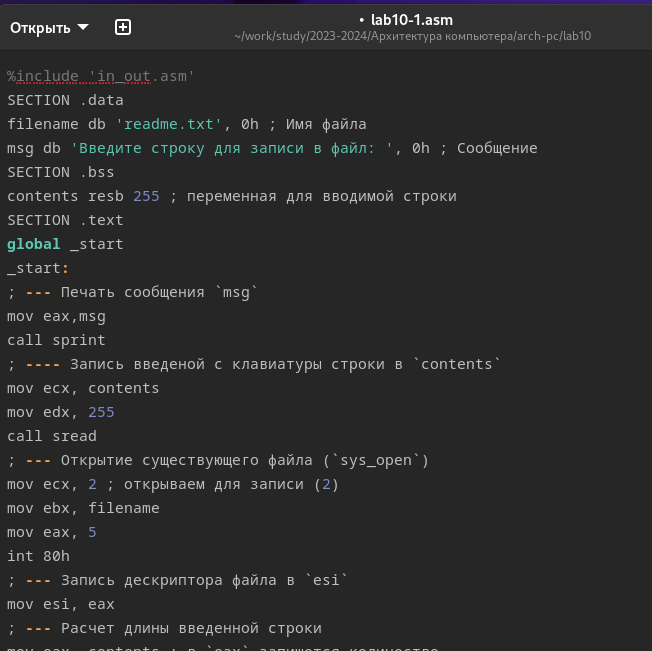
### 1.1.2 Содержание

1 Цель работы 1 2 Задание 1 3 Теоретическое введение 1 4 Выполнение лабораторной работы 2 4.1 Написание программ для работы с файлами 2 4.2 Задание для самостоятельной работы 5 5 Выводы 8 6 Список литературы 8 ## 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 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. 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)



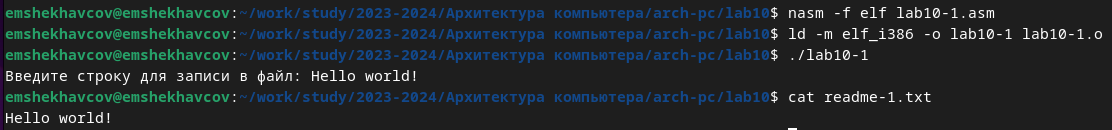
Создание файлов для лабораторной работы

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



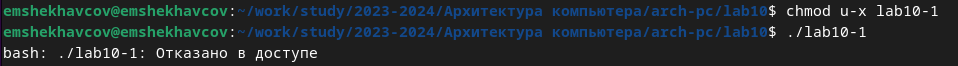
Ввод текста программы из листинга 10.1

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



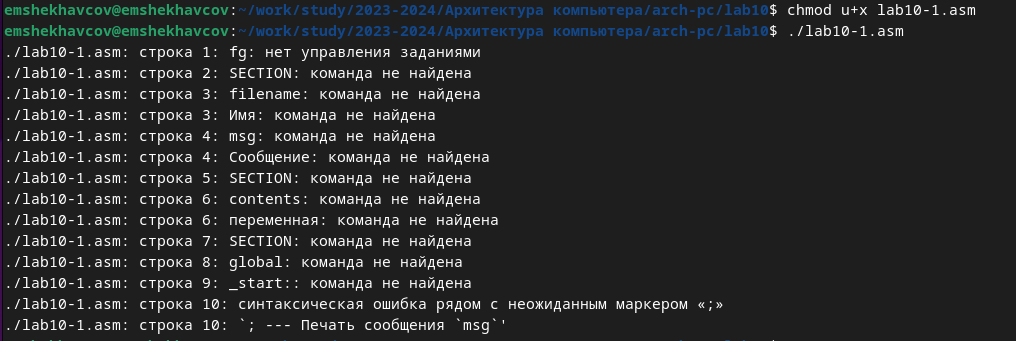
Запуск исполняемого файла

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



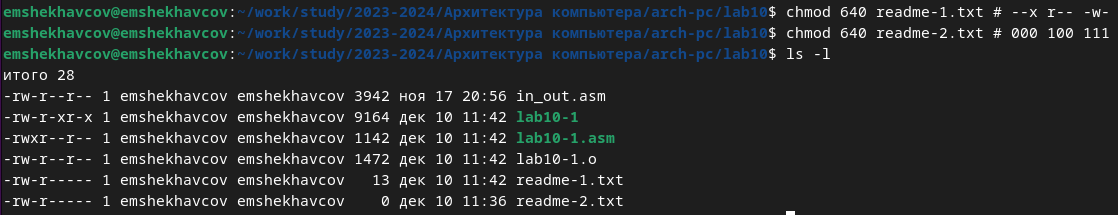
Запрет на выполнение файла

Файл не выполняется, т.к в команде я указал “u” - владелец (себя), “-” - отменить набор прав, “х” - право на исполнение. С помощью команды chmod u+х изменяю права доступа к файлу lab10-1.asm с исходным текстом программы, добавив права на исполнение, и пытаюсь выполнить его. (рис.4.5)



Добавление прав на исполнение

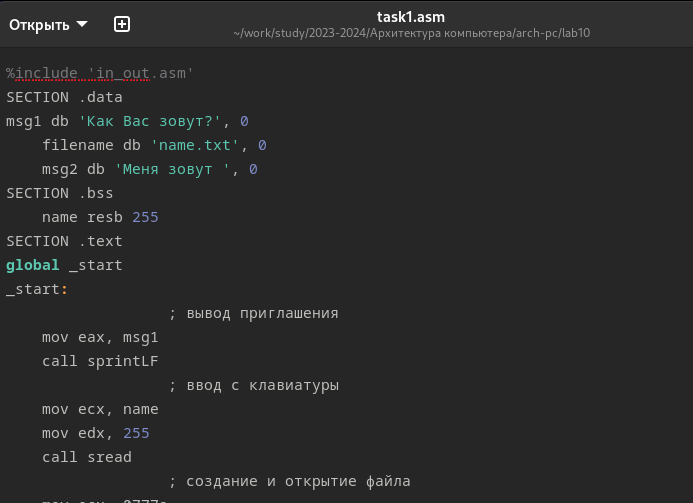
Текстовый файл начинает исполнение, но не исполняется, т.к не содержит в себе команд для терминала. В соответствии со своим вариантом (11) в таблице 10.4 предоставляю права доступа к файлу readme1.txt представленные в символьном виде, а для файла readme-2.txt – в двочном виде: –x r– -w-, 000 100 111 И проверяю правильность выполнения с помощью команды ls -l. (рис.4.6)



Предоставление прав доступа в символьном и двоичном виде

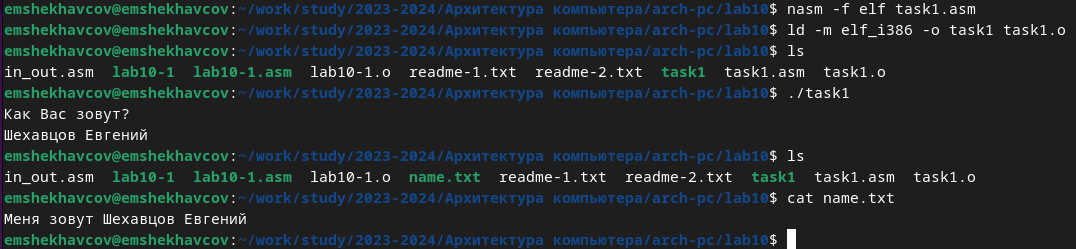
### 1.1.3 4.2 Задание для самостоятельной работы

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)



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)



Запуск исполняемого файла и проверка его работы

Программа работает корректно. Код программы: %include ‘in\_out.asm’ SECTION .data msg1 db ‘Как Вас зовут?’, 0 filename db ‘name.txt’, 0 msg2 db ‘Меня зовут’, 0 SECTION .bss name resb 255 SECTION .text global \_start \_start: ; вывод приглашения mov eax, msg1 call sprintLF ; ввод с клавиатуры mov ecx, name mov edx, 255 call sread ; создание и открытие файла mov ecx, 0777o mov ebx, filename mov eax, 8 int 0x80 mov esi, eax ; вывод строки “Меня зовут” mov eax, msg2 call slen mov edx, eax mov ecx, msg2 mov ebx, esi mov eax, 4 int 0x80 ; вывод введенного имени mov eax, name call slen mov edx, eax mov ecx, name mov ebx, esi mov eax, 4 int 0x80 ; закрытие файла mov ebx, esi mov eax, 6 int 0x80 call quit ## 5 Выводы Благодаря данной лабораторной работе я приобрел 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).