

Лабораторная работа №5

Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux

Чекмарев Александр Дмитриевич | группа: НПИбд 02-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Ознакомление с Midnight Commander	6
2.2	Подключение внешнего файла in_out.asm	11
3	Самостоятельная работа	16
4	Выводы	21

Список иллюстраций

2.1	Рис 2.1.1: Демонстрация ввода команды <code>mc</code>	6
2.2	Рис 2.1.2: Демонстрация <code>mc</code>	6
2.3	Рис 2.1.3: Переход в каталог	7
2.4	Рис 2.1.4: Создание папки <code>lab05</code>	7
2.5	Рис 2.1.5: Демонстрация перехода в каталог <code>lab05</code>	8
2.6	Рис 2.1.6: Демонстрация ввода команды <code>touch</code>	8
2.7	Рис 2.1.7: Создание файла <code>.asm</code>	8
2.8	Рис 2.1.8: Демонстрация текста в файле	9
2.9	Рис 2.1.9: Сохранение	9
2.10	Рис 2.1.10: Проверка содержимого текста в файле	10
2.11	Рис 2.1.11: Демонстрация ввода команд для оттрансляции текста .	10
2.12	Рис 2.2.1: Копирование скаченного файла в каталог <code>lab05</code>	12
2.13	Рис 2.2.2: Демонстрация каталога после копирования	12
2.14	Рис 2.2.3: Создание копии файла с новым именем	13
2.15	Рис 2.2.4: Демонстрация каталога после создания копии	13
2.16	Рис 2.2.5: Демонстрация текста в файле	14
2.17	Рис 2.2.6: Демонстрация ввода команд для оттрансляции текста .	14
2.18	Рис 2.2.7: Проверка работы файлы	14
2.19	Рис 2.2.8: Демонстрация измененного текста в файле	15
2.20	Рис 2.2.9: Демонстрация повторного ввода команд для оттрансля- ции текста и проверка работы файла	15
3.1	Рис 3.1.1: Демонстрация созданной папки	16
3.2	Рис 3.1.2: Создание копии файла	17
3.3	Рис 3.1.3: Редактирование программы(кода)	17
3.4	Рис 3.2.1: Компиляция и обработка файла	18
3.5	Рис 3.2.2: Проверка программы	18
3.6	Рис 3.3.1: Копирование <code>int_out.asm</code>	18
3.7	Рис 3.3.2: Копирование <code>lab5-2.asm</code>	19
3.8	Рис 3.3.3: Редактирование программы(кода)	19
3.9	Рис 3.4.1: Компиляция и обработка файла	20
3.10	Рис 3.4.2: Проверка программы	20

Список таблиц

1 Цель работы

Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`.

2 Выполнение лабораторной работы

2.1 Ознакомление с Midnight Commander

Откроем *Midnight Commander*

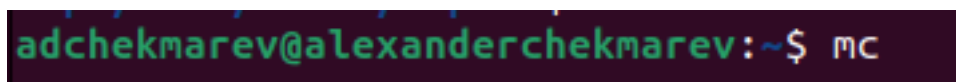


Рис. 2.1: Рис 2.1.1: Демонстрация ввода команды mc

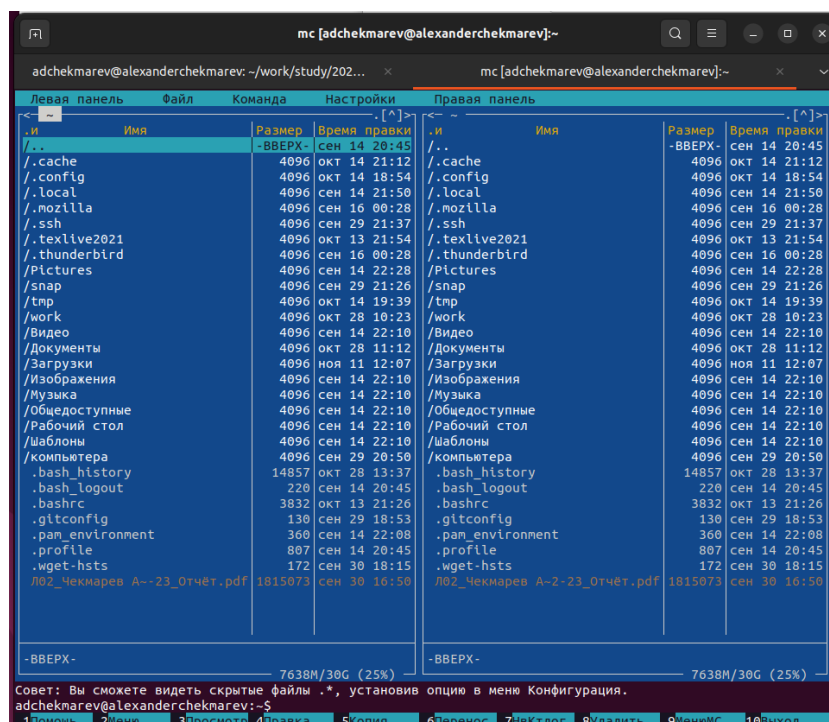


Рис. 2.2: Рис 2.1.2: Демонстрация mc

Перейдем в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №4

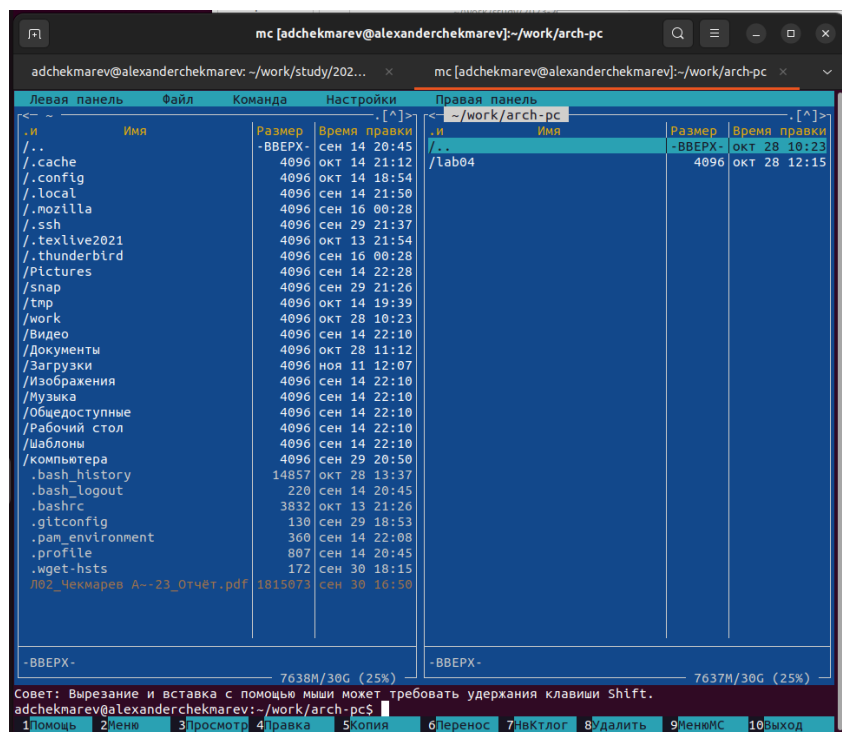


Рис. 2.3: Рис 2.1.3: Переход в каталог

Создадим папку lab05 с помощью фнкциональной клавиши F7 и перейдем в этот каталог

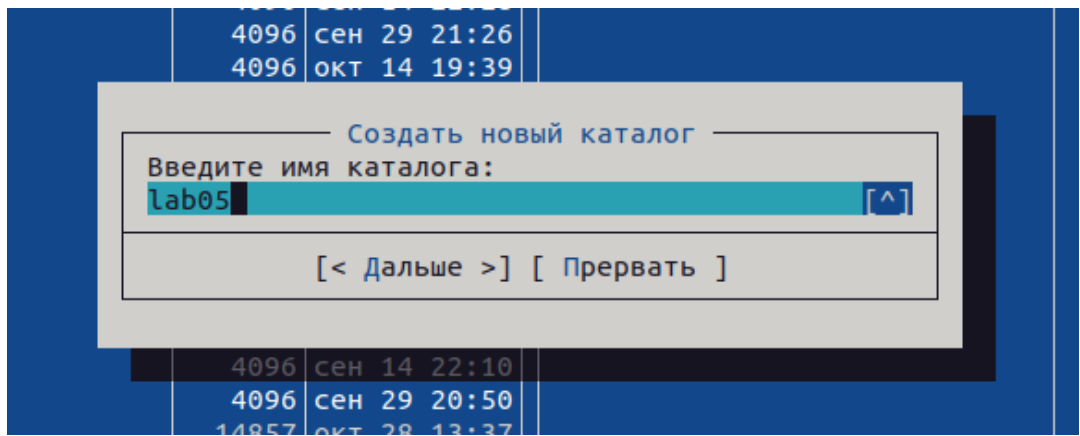


Рис. 2.4: Рис 2.1.4: Создание папки lab05

Правая панель

```
< ~/work/arch-pc/lab05 .[^]>
```

.и	Имя	Размер	Время правки
/..		-ВВЕРХ-	ноя 11 12:20

Рис. 2.5: Рис 2.1.5: Демонстрация перехода в каталог lab05

Пользуясь строкой ввода и командой *touch* создадим файл *lab5-1.asm*

Совет: Автодополнение работает во всех строках ввода. Просто нажмите M-Tab.
 adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05\$ touch lab5-1.asm

Рис. 2.6: Рис 2.1.6: Демонстрация ввода команды touch

Правая панель

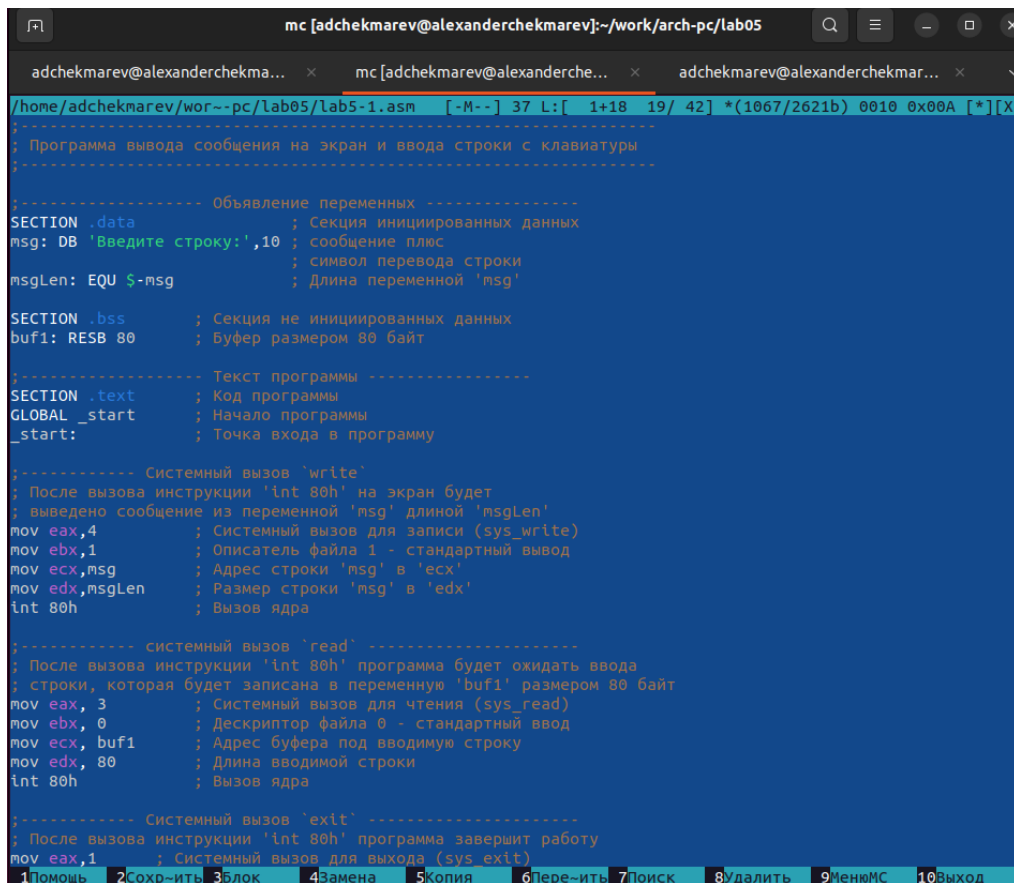
```
< ~/work/arch-pc/lab05 .[^]>
```

.и	Имя	Размер	Время правки
/..		-ВВЕРХ-	ноя 11 12:20
lab5-1.asm		0	ноя 11 12:21

Рис. 2.7: Рис 2.1.7: Создание файла .asm

С помощью функциональной клавиши **F4** откроем файл *lab5-1.asm* для редактирования во встроенном редакторе mcedit.

Введем текст программы из листинга 5.1 (взятый из лаб.№5), сохраним изменения и закроем файл



```
mc [adchekmarev@alexanderchekmarev]:~/work/arch-pc/lab05
adchekmarev@alexanderchekma... x mc [adchekmarev@alexanderche... x adchekmarev@alexanderchekmar... x
/home/adchekmarev/work/arch-pc/lab05/lab5-1.asm [-M--] 37 L: [ 1+18 19/ 42] *(1067/2621b) 0010 0x00A [*][X]
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'

SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт

;----- Текст программы -----
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

;----- Системный вызов 'write'
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра

;----- системный вызов 'read' -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
mov eax,3 ; Системный вызов для чтения (sys_read)
mov ebx,0 ; Дескриптор файла 0 - стандартный ввод
mov ecx,buf1 ; Адрес буфера под вводимую строку
mov edx,80 ; Длина вводимой строки
int 80h ; Вызов ядра

;----- Системный вызов 'exit' -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Пере-ить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 2.8: Рис 2.1.8: Демонстрация текста в файле

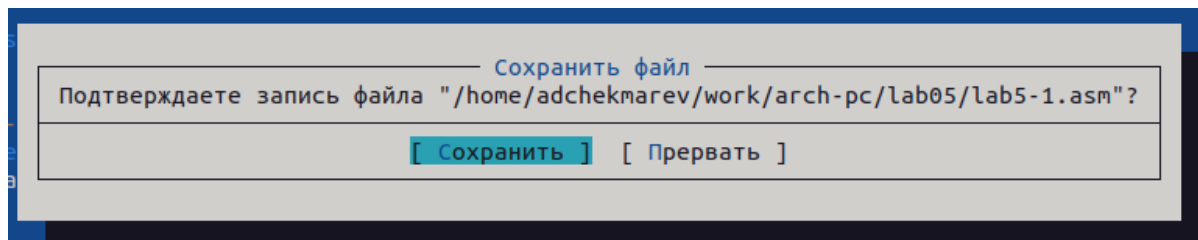


Рис. 2.9: Рис 2.1.9: Сохранение

С помощью функциональной клавиши **F3** откроем файл *lab5-1.asm* для просмотра. Убедимся, что файл содержит текст программы.

```

/home/adchekmarev/work/arch-pc/lab05/lab5-1.asm
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----

;----- Объявление переменных -----
SECTION .data                ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
                                ; символ перевода строки
msgLen: EQU $-msg            ; Длина переменной 'msg'

SECTION .bss                 ; Секция не инициализированных данных
buf1: RESB 80                ; Буфер размером 80 байт

;----- Текст программы -----
SECTION .text                ; Код программы
GLOBAL _start               ; Начало программы
_start:                      ; Точка входа в программу

;----- Системный вызов `write` -----
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
mov eax,4                    ; Системный вызов для записи (sys_write)
mov ebx,1                    ; Описатель файла 1 - стандартный вывод
mov ecx,msg                  ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen               ; Размер строки 'msg' в 'edx'
int 80h                      ; Вызов ядра

;----- системный вызов `read` -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
mov eax,3                    ; Системный вызов для чтения (sys_read)
mov ebx,0                    ; Дескриптор файла 0 - стандартный ввод
mov ecx,buf1                 ; Адрес буфера под вводимую строку
mov edx,80                   ; Длина вводимой строки
int 80h                      ; Вызов ядра

;----- Системный вызов `exit` -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1                    ; Системный вызов для выхода (sys_exit)

```

Рис. 2.10: Рис 2.1.10: Проверка содержимого текста в файле

Оттранслируем текст программы lab5-1.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос введем мое ФИО 'Чекмарев Александр Дмитриевич'

```

adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Чекмарев Александр Дмитриевич
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ █

```

Рис. 2.11: Рис 2.1.11: Демонстрация ввода команд для оттрансляции текста

2.2 Подключение внешнего файла in_out.asm

Для упрощения написания программ часто встречающиеся одинаковые участки кода (такие как, например, вывод строки на экран или выход из программы) можно оформить в виде подпрограмм и сохранить в отдельные файлы, а во всех нужных местах поставить вызов нужной подпрограммы. Это позволяет сделать основную программу более удобной для написания и чтения. NASM позволяет подключать внешние файлы с помощью директивы `%include`, которая предписывает ассемблеру заменить эту директиву содержимым файла. Подключаемые файлы также написаны на языке ассемблера. Важно отметить, что директива `%include` в тексте программы должна стоять раньше, чем встречаются вызовы подпрограмм из подключаемого файла. Для вызова подпрограммы из внешнего файла используется инструкция `call`, которая имеет следующий вид

`call`

где `function` имя подпрограммы.

Для выполнения лабораторных работ используется файл `in_out.asm1`, который содержит следующие подпрограммы [4]:

- `slen` – вычисление длины строки (используется в подпрограммах печати сообщения для определения количества выводимых байтов);
- `sprint` – вывод сообщения на экран, перед вызовом `sprint` в регистр `eax` необходимо записать выводимое сообщение (`mov eax,;`);
- `sprintLF` – работает аналогично `sprint`, но при выводе на экран добавляет к сообщению символ перевода строки;
- `sread` – ввод сообщения с клавиатуры, перед вызовом `sread` в регистр `eax` необходимо записать адрес переменной в которую введенное сообщение будет записано (`moveax,;`), в регистр `ebx` – длину вводимой строки (`mov ebx,;`);
- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax,;`);
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки;

- `atoi` – функция преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`moveax,;`);

- `quit` – завершение программы

Скачаем файл `in_out.asm` со страницы курса в ТУИС.

Подключаемый файл `in_out.asm` должен лежать в том же каталоге, что и файл с программой, в которой он используется. В одной из панелей `ms` откроем каталог с файлом `lab5-1.asm`. В другой панели каталог со скачанным файлом `in_out.asm`. Скопируем файл `in_out.asm` в каталог с файлом `lab5-1.asm` с помощью функциональной клавиши **F5**.

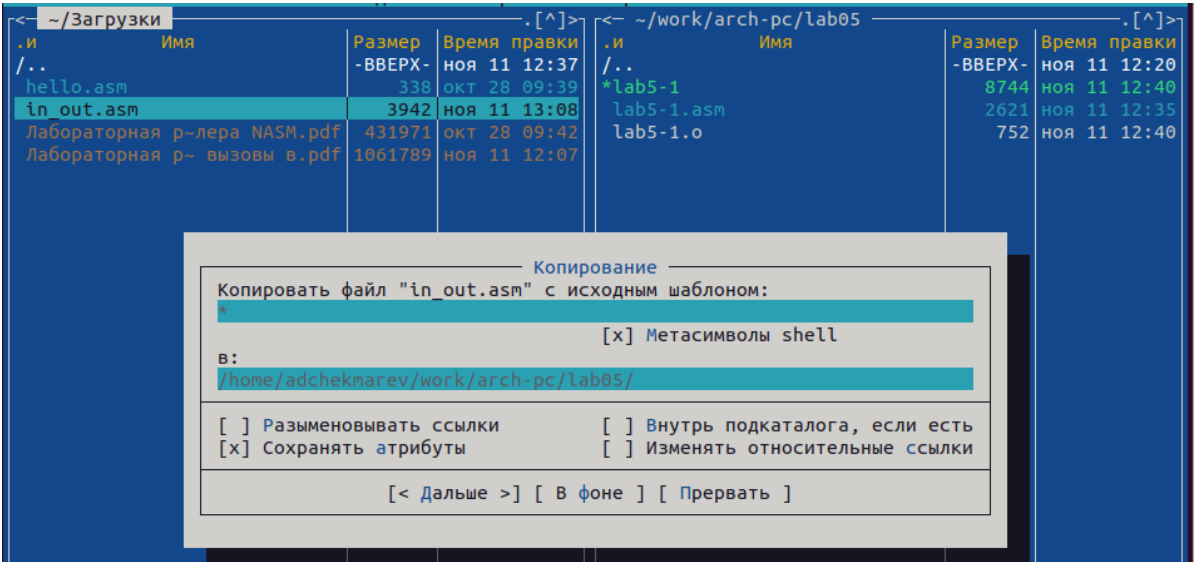


Рис. 2.12: Рис 2.2.1: Копирование скаченного файла в каталог lab05

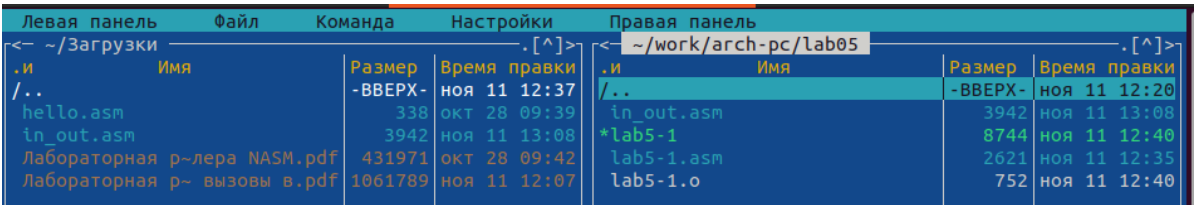


Рис. 2.13: Рис 2.2.2: Демонстрация каталога после копирования

С помощью функциональной клавиши **F6** создадим копию файла *lab5-1.asm* с именем *lab5-2.asm*. Выделим файл *lab5-1.asm*, нажмем клавишу **F6**, введем имя файла *lab5-2.asm* и нажмем клавишу **Enter**.

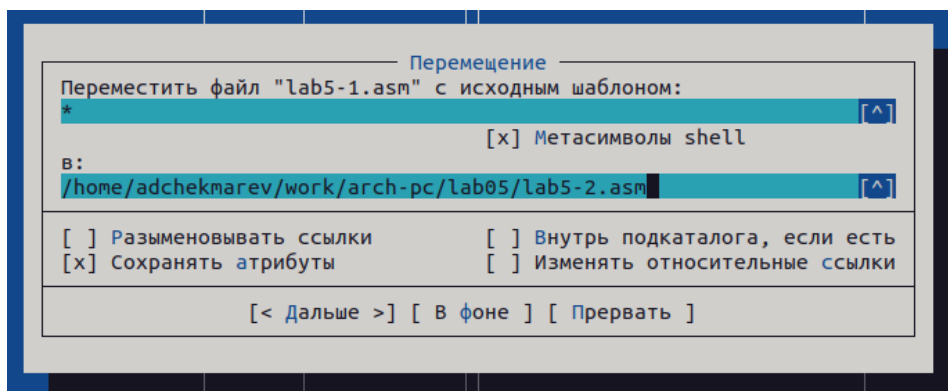


Рис. 2.14: Рис 2.2.3: Создание копии файла с новым именем

Имя		Размер	Время правки
-ВВЕРХ-			ноя 11 12:20
./			ноя 11 13:08
in_out.asm		3942	ноя 11 12:40
*lab5-1		8744	ноя 11 12:40
lab5-1.o		752	ноя 11 12:35
lab5-2.asm		2621	

Рис. 2.15: Рис 2.2.4: Демонстрация каталога после создания копии

Исправим текст программы в файле *lab5-2.asm* с использованием подпрограмм из внешнего файла *in_out.asm* (используем подпрограммы `sprintLF`, `sread` и `quit`) в соответствии с листингом 5.2. Создадим исполняемый файл и проверим его работу

```

/home/adchekmarev/work-h-pc/lab05/lab5-2.asm [----] 17 L: [ 1+24 25/
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение

SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

mov eax, msg ; запись адреса выводимого сообщения в 'EAX'
call sprintLF ; вызов подпрограммы печати сообщения

mov ecx, buf1 ; запись адреса переменной в 'EAX'
mov edx, 80 ; запись длины вводимого сообщения в 'EBX'

call sread ; вызов подпрограммы ввода сообщения

call quit ; вызов подпрограммы завершения

```

Рис. 2.16: Рис 2.2.5: Демонстрация текста в файле

```

adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o

```

Рис. 2.17: Рис 2.2.6: Демонстрация ввода команд для оттрансляции текста

```

adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Чекмарев Александр Дмитриевич
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$

```

Рис. 2.18: Рис 2.2.7: Проверка работы файлы

В файле *lab5-2.asm* заменим подпрограмму *sprintLF* на *sprint*. Создадим исполняемый файл и проверим его работу.

```

/home/adchekmarev/work-h-pc/lab05/lab5-2.asm  [-M--] 17 L:[ 1+17
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----

%include 'in_out.asm'          ; подключение внешнего файла

SECTION .data                  ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение

SECTION .bss                   ; Секция не инициализированных данных
buf1: RESB 80                  ; Буфер размером 80 байт

SECTION .text                   ; Код программы
GLOBAL _start                  ; Начало программы
_start:                         ; Точка входа в программу

mov eax, msg                   ; запись адреса выводимого сообщения в `EAX`
call sprint                    ; вызов подпрограммы печати сообщения

mov ecx, buf1                  ; запись адреса переменной в `EAX`
mov edx, 80                    ; запись длины вводимого сообщения в `EBX`

call sread                     ; вызов подпрограммы ввода сообщения

call quit                      ; вызов подпрограммы завершения

```

Рис. 2.19: Рис 2.2.8: Демонстрация измененного текста в файле

```

adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Чекмарев Александр Дмитриевич
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$

```

Рис. 2.20: Рис 2.2.9: Демонстрация повторного ввода команд для оттрансляции текста и проверка работы файла

В чем разница?

В случае *sprintLF* мы вводим сообщение в след строке, в случае *sprint* воод сообщения происходит в той же строке, где нас просят ввести сообщение после :

3 Самостоятельная работа

Задание №1 Создайте копию файла *lab5-1.asm*. Внесите изменения в программу (без использования внешнего файла *in_out.asm*), так чтобы она работала по следующему алгоритму:

- вывести приглашение типа “Введите строку:”
- ввести строку с клавиатуры
- вывести введённую строку на экран

Для удобства создадим новую папку в каталоге *lab05* для самостоятельной работы

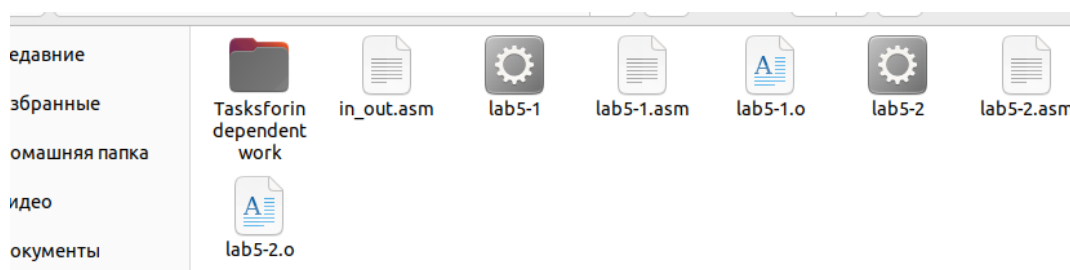


Рис. 3.1: Рис 3.1.1: Демонстрация созданной папки

Создадим копию файла *lab5-1.asm* в *ms* с помощью **F5**

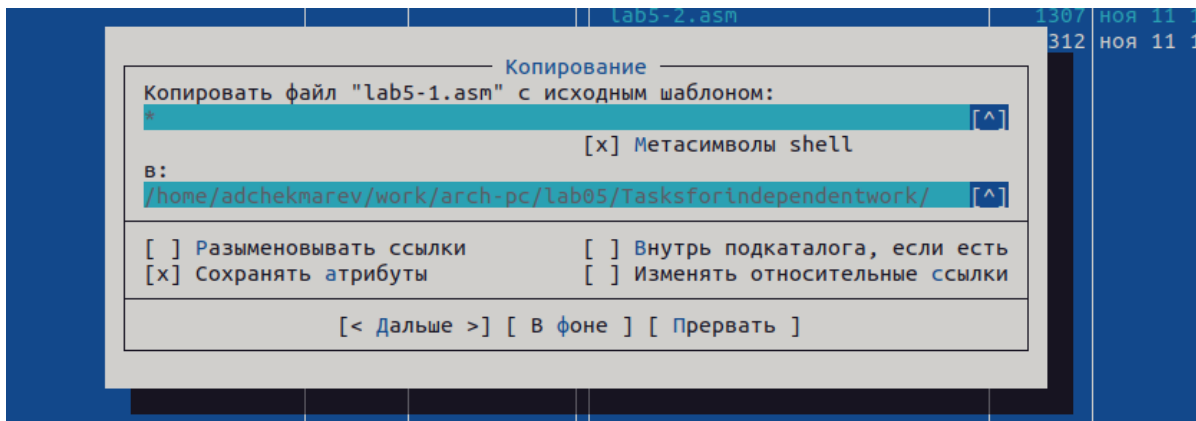


Рис. 3.2: Рис 3.1.2: Создание копии файла

Изменим программу в тс под условие задания

```

/home/adchekmarev/work/~ependentwork/lab5-1.asm [----] 33 L:[ 1
; lab5-1.asm
SECTION .data                ; Секция инициированных данных
    msg: DB 'Введите строку:',10 ; сообщение плюс
                                ; символ перевода строки
    msgLen: EQU $-msg         ; Длина переменной 'lab5-1'

SECTION .bss                 ; Секция не инициированных данных
    buf1: RESB 80             ; Буфер размером 80 байт

SECTION .text                 ; Код программы
GLOBAL _start                 ; Начало программы

_start:                       ; Точка входа в программу

    mov eax,4                 ; Системный вызов для записи (sys_write)
    mov ebx,1                 ; Описатель файла 1 - стандартный вывод
    mov ecx,msg               ; Адрес строки 'msg' в 'ecx'
    mov edx,msgLen            ; Размер строки 'msg' в 'edx'
    int 80h                   ; Вызов ядра

    mov eax, 3                 ; Системный вызов для чтения (sys_read)
    mov ebx, 0                 ; Дескриптор файла 0 - стандартный ввод
    mov ecx, buf1              ; Адрес буфера под вводимую строку
    mov edx, 80                ; Длина вводимой строки
    int 80h                   ; Вызов ядра

    mov eax,4                 ; Системный вызов для записи (sys_write)
    mov ebx,1                 ; Описатель файла 1 - стандартный вывод
    mov ecx,buf1              ; Адрес строки 'msg' в 'ecx'
    mov edx,80                ; Размер строки 'msg' в 'edx'
    int 80h                   ; Вызов ядра

```

Рис. 3.3: Рис 3.1.3: Редактирование программы(кода)

Задание №2 Получите исполняемый файл и проверьте его работу. На приглашение ввести строку введите свою фамилию.

Скомпилируем и отправим файл на обработку компоновщику

```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05/TaskstoforIndependentwork$ nasm -f elf lab5-1.asm
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05/TaskstoforIndependentwork$ ld -m elf_i386 -o lab5-1 lab5-1.o
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05/TaskstoforIndependentwork$
```

Рис. 3.4: Рис 3.2.1: Компиляция и обработка файла

Проверим работоспособность файла(программы)

```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05/TaskstoforIndependentwork$ ./lab5-1
Введите строку:
Чекмарев
Чекмарев
```

Рис. 3.5: Рис 3.2.2: Проверка программы

Задание №3 Создайте копию файла *lab5-2.asm*. Исправьте текст программы с использованием подпрограмм из внешнего файла *in_out.asm*, так чтобы она работала по следующему алгоритму:

- вывести приглашение типа “Введите строку:”
- ввести строку с клавиатуры
- вывести введенную строку на экран

Скопируем файлы *in_out.asm* и *lab5-2.asm* в отдельную папку

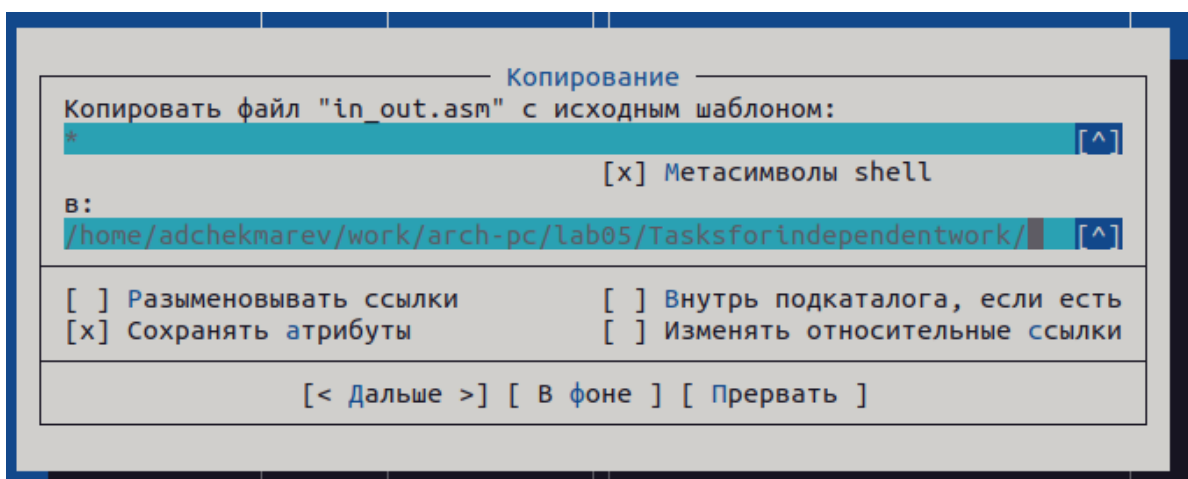


Рис. 3.6: Рис 3.3.1: Копирование *in_out.asm*

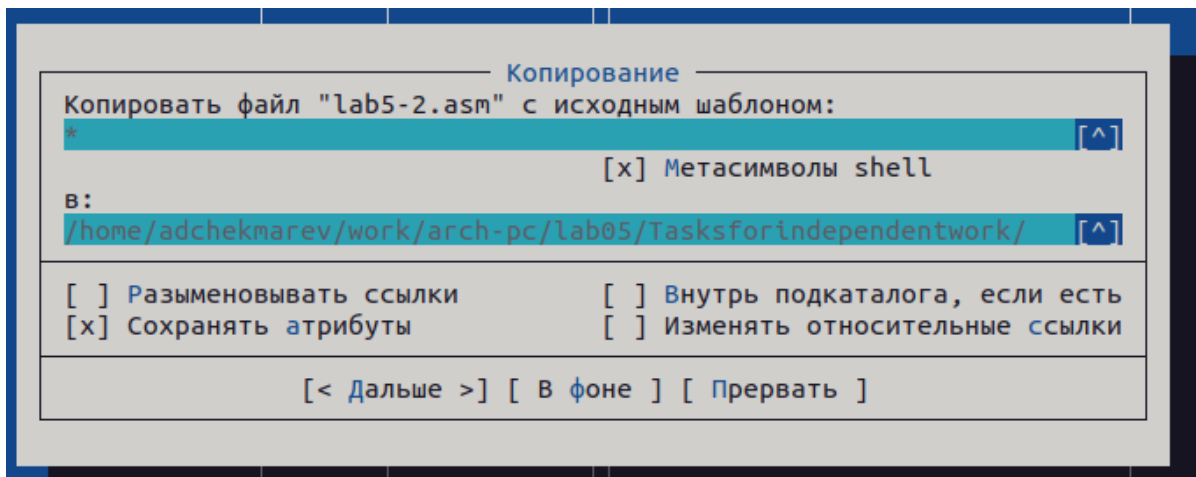


Рис. 3.7: Рис 3.3.2: Копирование lab5-2.asm

Изменим программу в тмс под условие задания

```

/home/adchekmarev/work/~ependentwork/lab5-2.asm [----] 11 L:[ 1+13 14/
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение

SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call sprint ; вызов подпрограммы печати сообщения

mov ecx, buf1 ; запись адреса переменной в `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`

call sread ; вызов подпрограммы ввода сообщения

mov eax, buf1
call sprint

call quit ; вызов подпрограммы завершения

```

Рис. 3.8: Рис 3.3.3: Редактирование программы(кода)

Задание№4 Создайте исполняемый файл и проверьте его работу.

Скомпилируем и отправим файл на обработку

```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05/Task sforIndependentwork$ nasm -f elf lab5-2.asm  
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05/Task sforIndependentwork$ ld -m elf_i386 -o lab5-2 lab5-2.o
```

Рис. 3.9: Рис 3.4.1: Компиляция и обработка файла

Проверим работоспособность программы

```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05/Task sforIndependentwork$ ./lab5-2  
Введите строку: Чекмарев  
Чекмарев
```

Рис. 3.10: Рис 3.4.2: Проверка программы

Загрузим все файлы на github, как всегда

4 Выводы

Я приобрел практические навыки работы в Midnight Commander. Освоил инструкции языка ассемблера `mov` и `int`.