

Лабораторная работа №5

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

Перфилов Александр Константинович | 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	12
3	Самостоятельная работа	17
4	Выводы	22

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>	7
2.3	Рис 2.1.3: Переход в каталог	7
2.4	Рис 2.1.4: Создание папки <code>lab05</code>	8
2.5	Рис 2.1.5: Демонстрация ввода команды <code>touch</code>	8
2.6	Рис 2.1.6: Демонстрация текста в файле	9
2.7	Рис 2.1.7: Проверка содержимого текста в файле	11
2.8	Рис 2.1.8: Демонстрация ввода команд для оттрансляции текста .	12
2.9	Рис 2.1.9: Проверка программы	12
2.10	Рис 2.2.1: Копирование скаченного файла в каталог <code>lab05</code>	14
2.11	Рис 2.2.2: Создание копии файла с новым именем	14
2.12	Рис 2.2.4: Демонстрация текста в файле	15
2.13	Рис 2.2.5: Демонстрация ввода команд для оттрансляции текста и проверка работы программы	15
2.14	Рис 2.2.6: Демонстрация измененного текста в файле	16
2.15	Рис 2.2.7: Демонстрация повторного ввода команд для оттрансля- ции текста и проверка работы файла	16
3.1	Рис 3.1.1: Создание копии файла	17
3.2	Рис 3.1.2: Редактирование программы	18
3.3	Рис 3.2.1: Оттранслирование и компоновка файла	19
3.4	Рис 3.2.2: Проверка программы	19
3.5	Рис 3.3.1: Копирование файлов	20
3.6	Рис 3.3.2: Редактирование программы(кода)	21
3.7	Рис 3.4.1: Оттранслирование и компоновка файла	21
3.8	Рис 3.4.2: Проверка программы	21

Список таблиц

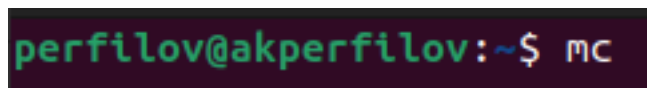
1 Цель работы

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

A terminal window with a dark background. The prompt 'perfilov@akperfilov:~\$' is shown in green text, followed by the command 'mc' in red text.

```
perfilov@akperfilov:~$ mc
```

Рис. 2.1: Рис 2.1.1: Демонстрация ввода команды mc

Левая панель				Правая панель			
Имя	Размер	Время	правки	Имя	Размер	Время	правки
..	-ВВЕРХ-	ноя 4	21:03	..	-ВВЕРХ-	ноя 4	21:03
/.cache	4096	ноя 5	22:17	/.cache	4096	ноя 5	22:17
/.config	4096	ноя 6	00:47	/.config	4096	ноя 6	00:47
/.gnupg	4096	ноя 5	22:41	/.gnupg	4096	ноя 5	22:41
/.local	4096	ноя 4	21:13	/.local	4096	ноя 4	21:13
/.mozilla	4096	ноя 5	22:17	/.mozilla	4096	ноя 5	22:17
/.ssh	4096	ноя 4	23:56	/.ssh	4096	ноя 4	23:56
/.texlive2021	4096	ноя 5	23:51	/.texlive2021	4096	ноя 5	23:51
/.thunderbird	4096	ноя 5	22:17	/.thunderbird	4096	ноя 5	22:17
/snap	4096	ноя 5	22:05	/snap	4096	ноя 5	22:05
/tmp	4096	ноя 5	22:03	/tmp	4096	ноя 5	22:03
/work	4096	ноя 6	02:01	/work	4096	ноя 6	02:01
/Видео	4096	ноя 4	21:17	/Видео	4096	ноя 4	21:17
/Документы	4096	ноя 5	00:13	/Документы	4096	ноя 5	00:13
/Загрузки	4096	ноя 11	17:44	/Загрузки	4096	ноя 11	17:44
/Изображения	4096	ноя 6	01:26	/Изображения	4096	ноя 6	01:26
/Музыка	4096	ноя 4	21:17	/Музыка	4096	ноя 4	21:17
/Общедоступные	4096	ноя 4	21:17	/Общедоступные	4096	ноя 4	21:17
/Рабочий стол	4096	ноя 4	21:17	/Рабочий стол	4096	ноя 4	21:17
/Шаблоны	4096	ноя 4	21:17	/Шаблоны	4096	ноя 4	21:17
.bash_history	9232	ноя 11	17:35	.bash_history	9232	ноя 11	17:35
.bash_logout	220	ноя 4	21:03	.bash_logout	220	ноя 4	21:03
.bashrc	3771	ноя 4	21:03	.bashrc	3771	ноя 4	21:03
.gitconfig	158	ноя 4	22:51	.gitconfig	158	ноя 4	22:51
.pam_environment	354	ноя 4	21:16	.pam_environment	354	ноя 4	21:16
.profile	807	ноя 4	21:03	.profile	807	ноя 4	21:03
.python_history	5	ноя 5	23:52	.python_history	5	ноя 5	23:52
.wget-hsts	172	ноя 5	22:00	.wget-hsts	172	ноя 5	22:00
work.zip	11815K	ноя 5	22:54	work.zip	11815K	ноя 5	22:54
Л02_Перфилов А.К_Н-бд-02-23_отчет.pdf	1596666	ноя 5	00:07	Л02_Перфилов А.К_Н-бд-02-23_отчет.pdf	1596666	ноя 5	00:07
-ВВЕРХ-				-ВВЕРХ-			
7698М/29G (26%)				7698М/29G (26%)			

Совет: Вы сможете видеть скрытые файлы .*, установив опцию в меню Конфигурация.

Рис. 2.2: Рис 2.1.2: Демонстрация mc

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

Имя	Размер	Время	правки
..	-ВВЕРХ-	ноя 6	02:01
/lab04	4096	ноя 11	17:17

Рис. 2.3: Рис 2.1.3: Переход в каталог

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

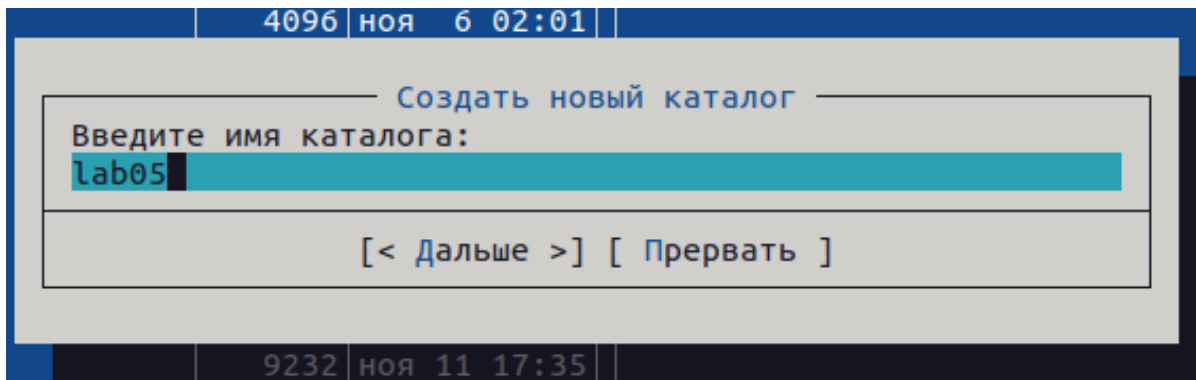


Рис. 2.4: Рис 2.1.4: Создание папки lab05

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

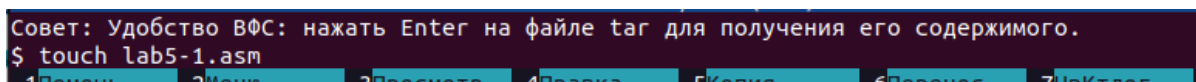


Рис. 2.5: Рис 2.1.5: Демонстрация ввода команды touch

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


```

/home/perfilov/work/arch-pc/lab5-1.asm [----] 0 L:[ 1+ 0 1/ 46] *
;-----
; 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 ;Descriptor 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)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

```

Рис. 2.6: Рис 2.1.6: Демонстрация текста в файле

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

```

/home/perfilov/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)
mov ebx,0                     ; Выход с кодом возврата 0 (без ошибок)
int 80h                       ; Вызов ядра

```

Рис. 2.7: Рис 2.1.7: 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 'Перфилов Александр Константинович'

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

Рис. 2.8: Рис 2.1.8: Демонстрация ввода команд для оттрансляции текста

```
perfilov@akperfilov:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Перфилов Александр Константинович
```

Рис. 2.9: Рис 2.1.9: Проверка программы

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 *mc* 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*.

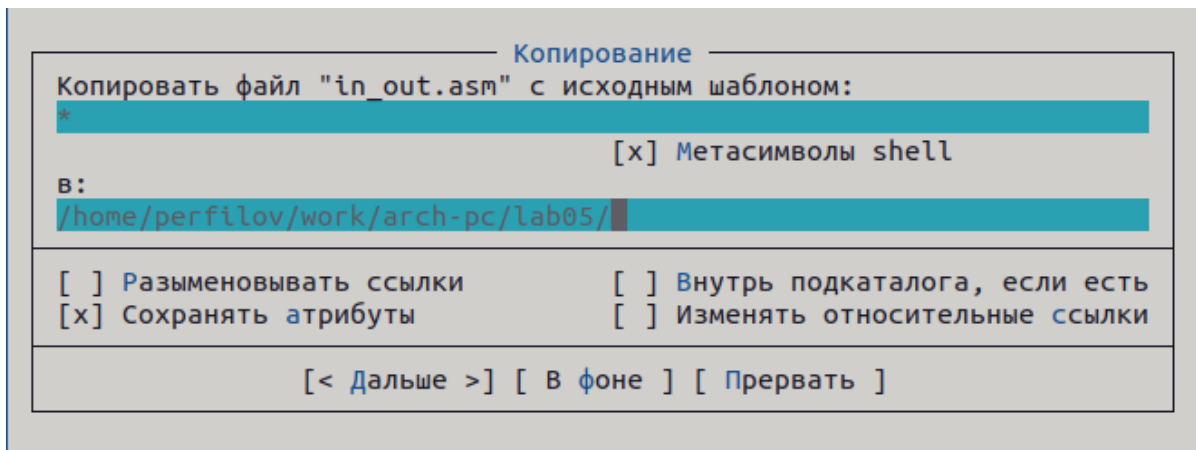


Рис. 2.10: Рис 2.2.1: Копирование скаченного файла в каталог lab05

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*.

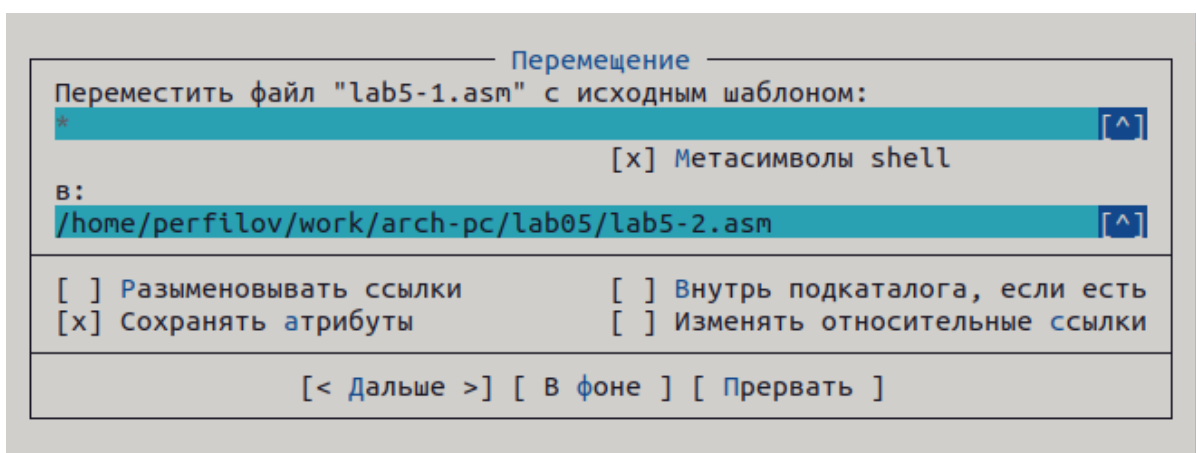


Рис. 2.11: Рис 2.2.2: Создание копии файла с новым именем

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/perfilov/work/arch-pc/lab05/lab5-2.asm [----] 48 L:[ 1+24 25/ 26]
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
%include 'in_out.asm' ; подключение внешнего файла

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

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, 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. 2.12: Рис 2.2.4: Демонстрация текста в файле

```

perfilov@akperfilov:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
perfilov@akperfilov:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
perfilov@akperfilov:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Перфилов Александр

```

Рис. 2.13: Рис 2.2.5: Демонстрация ввода команд для оттрансляции текста и проверка работы программы

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

```

/home/perfilov/work/arch-pc/lab05/lab5-2.asm [-M--] 48 L:[ 1+24 25/ 26]
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
%include 'in_out.asm' ; подключение внешнего файла

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

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, 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. 2.14: Рис 2.2.6: Демонстрация измененного текста в файле

```

perfilov@akperfilov:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
perfilov@akperfilov:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
perfilov@akperfilov:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Перфилов Александр
perfilov@akperfilov:~/work/arch-pc/lab05$

```

Рис. 2.15: Рис 2.2.7: Демонстрация повторного ввода команд для оттрансляции текста и проверка работы файла

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 `tasks` в каталоге `lab05`

Создадим копию файла `lab5-1.asm` в `mc` с помощью `F5`

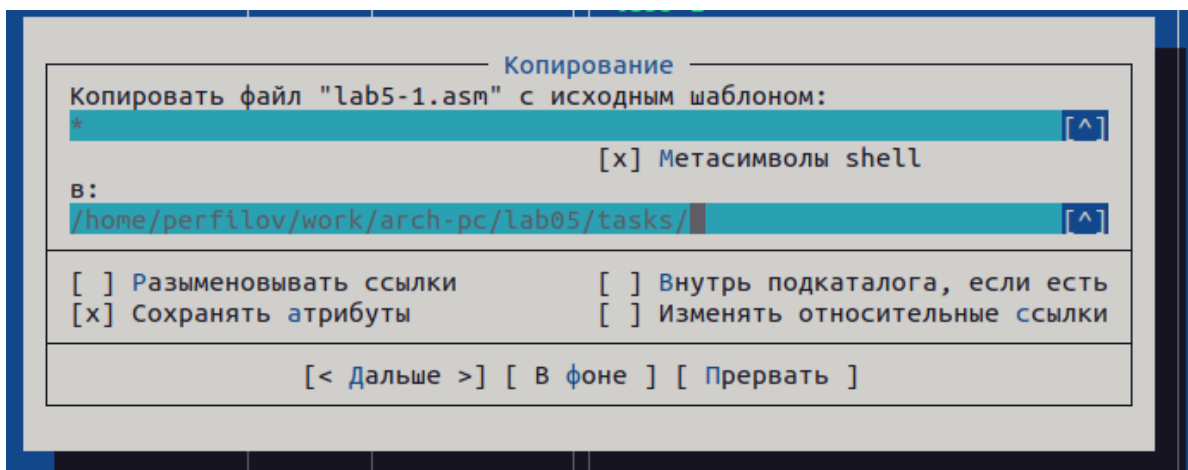


Рис. 3.1: Рис 3.1.1: Создание копии файла

Изменим программу в `mc` под условие задачи

```

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

SECTION .bss                 ; Секция не инициированных 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

mov eax, 4
mov ebx, 1
mov ecx, buf1
mov edx, 80
int 80h

```

Рис. 3.2: Рис 3.1.2: Редактирование программы

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

Оттранслируем текст программы в объектный файл. Выполним компоновку объектного файла

```
perfilov@akperfilov:~/work/arch-pc/lab05/tasks$ nasm -f elf lab5-1.asm
perfilov@akperfilov:~/work/arch-pc/lab05/tasks$ ld -m elf_i386 -o lab5-1 lab5-1.o
```

Рис. 3.3: Рис 3.2.1: Оттранслирование и компоновка файла

Проверим работоспособность файла(программы)

```
perfilov@akperfilov:~/work/arch-pc/lab05/tasks$ ./lab5-1
Введите строку:
Перфилов
Перфилов
```

Рис. 3.4: 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

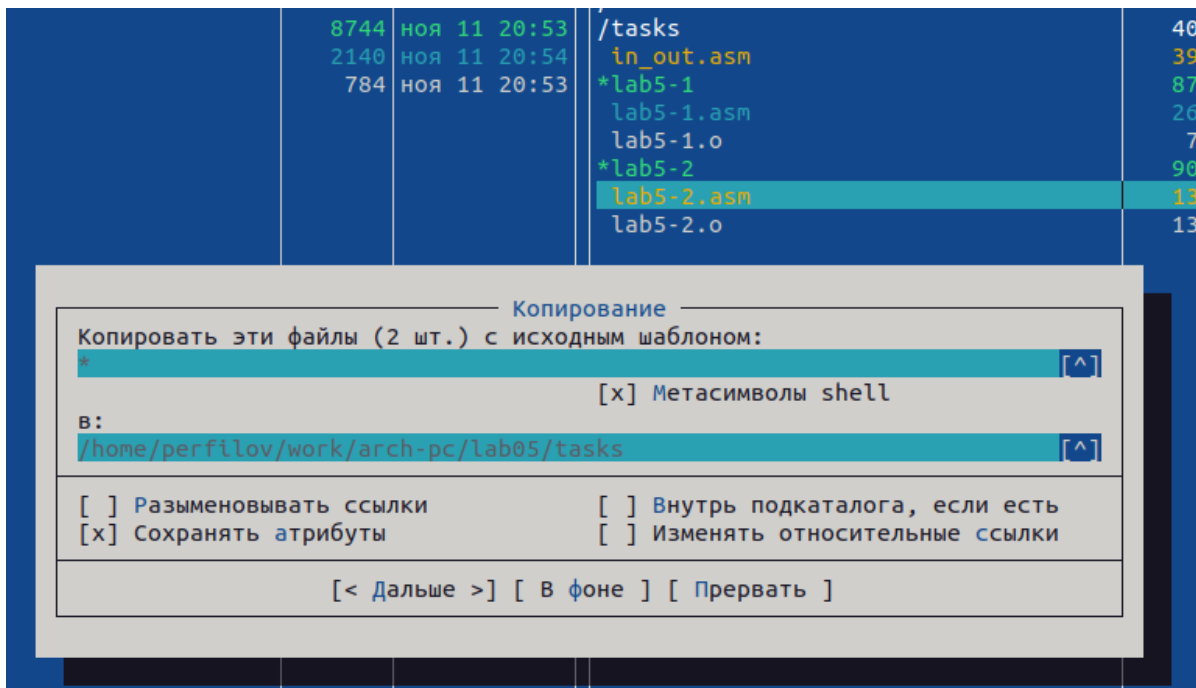


Рис. 3.5: Рис 3.3.1: Копирование файлов

Изменим программу в tc под условие задачи

```

;-----
; 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 sprintf                ; 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 sprintf

call quit                  ; вызов подпрограммы завершения

```

Рис. 3.6: Рис 3.3.2: Редактирование программы(кода)

Задание№4 Создайте исполняемый файл и проверьте его работу.

Оттранслируем текст программы в объектный файл. Выполним компоновку объектного файла

```

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

```

Рис. 3.7: Рис 3.4.1: Оттранслирование и компоновка файла

Проверим работоспособность программы

```

perfilov@akperfilov:~/work/arch-pc/lab05/tasks$ ./lab5-2
Введите строку:
Перфилов
Перфилов

```

Рис. 3.8: Рис 3.4.2: Проверка программы

По завершению лаб. работы файлы были загружены на github

4 Выводы

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