

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Адмиральская Александра Андреевна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выводы	20
	Список литературы	21

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Ввод текста программы	7
3.3	Создание и запуск исполняемого файла	8
3.4	Изменение текста программы	8
3.5	Создание и запуск исполняемого файла	8
3.6	Изменение текста программы	9
3.7	Запуск исполняемого файла	9
3.8	Создание файла и переход в него	9
3.9	Ввод текста программы	10
3.10	Проверка работы исполняемого файла	11
3.11	Создание файла листинга	11
3.12	Файл листинга	12
3.13	Удаление одного операнда	13
3.14	Выполнение трансляции	13
3.15	Ошибка в листинге файла	14
3.16	Создание файла	14
3.17	Ввод текста программы	15
3.18	Проверка работы исполняемого файла	16
3.19	Создание файла	16
3.20	Написание программы	17
3.21	Создание и проверка работы исполняемого файла	18
3.22	Проверка работы исполняемого файла	18

Список таблиц

2.1	Описание некоторых каталогов файловой системы GNU Linux . . .	6
-----	---	---

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 2.1 приведено краткое описание стандартных каталогов Unix.

Таблица 2.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

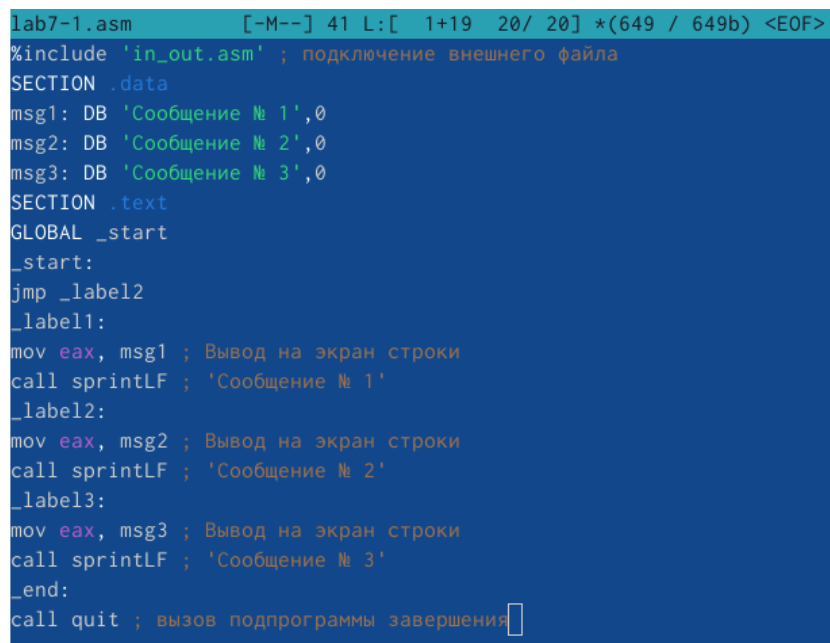
3 Выполнение лабораторной работы

Для начала создаем каталог для программ лабораторной работы № 7, переходим в него и создаем файл lab7-1.asm (рис. 3.1).

```
aaadmiraljskaya@dk3n55 ~ $ mkdir ~/work/arch-pc/lab07
aaadmiraljskaya@dk3n55 ~ $ cd ~/work/arch-pc/lab07
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 3.1: Создание каталога и файла

Открываем созданный файл и вводим в него текст программы из листинга 7.1. Программа выводит 2, 3 (рис. 3.2).



```
lab7-1.asm      [-M--] 41 L:[ 1+19 20/ 20] *(649 / 649b) <EOF>
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Ввод текста программы

Создаем исполняемый файл и запускаем его (рис. 3.3).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ █

```

Рис. 3.3: Создание и запуск исполняемого файла

Заново открываем файл lab7-1.asm и изменяем текст программы, вывод стал 2, 1 (рис. 3.4).

```

lab7-1.asm      [-M--] 41 L:[ 1+21 22/ 22] *(670 / 670b) <EOF>
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения █

```

Рис. 3.4: Изменение текста программы

Создаем исполняемый файл и запускаем его (рис. 3.5).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ █

```

Рис. 3.5: Создание и запуск исполняемого файла

Затем изменяем текст программы в этом же файле, чтобы вывод программы был 3, 2, 1 (рис. 3.6).


```

lab7-1.asm      [-M--] 41 L:[ 1+22 23/ 23] *(682 / 682b) <EOF>
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 3.6: Изменение текста программы

Создаем и запускаем исполняемый файл (рис. 3.7).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 3.7: Запуск исполняемого файла

Далее создаем файл lab7-2.asm и открываем его (рис. 3.8).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-2.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ mc

```

Рис. 3.8: Создание файла и переход в него

Вводим в него текст программы из листинга 7.3 (рис. 3.9).

```

lab7-2.asm      [-M--] 23 L: [ 1+24 25/ 49] *(706 /1743b) 0010 0x00A
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'

```

Рис. 3.9: Ввод текста программы

Создаем исполняемый файл и проверяем его работу для разных значений B (рис. 3.10).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 100
Наибольшее число: 100
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 1
Наибольшее число: 50
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 38
Наибольшее число: 50
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ 

```

Рис. 3.10: Проверка работы исполняемого файла

Затем создаем файл листинга для программы из файла lab7-2.asm (рис. 3.11).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst

```

Рис. 3.11: Создание файла листинга

Открываем этот файл и изучаем его (рис. 3.12).

```

lab7-2.lst      [-----] 55 L:[ 1+30 31/225] *(1980/14458b) 0010 0x00A
1               %include 'in_out.asm'
1               <1> ;----- slen -----
2               <1> ; Функция вычисления длины сообщения
3               <1> slen:
4 00000000 53    <1> push ebx
5 00000001 89C3  <1> mov ebx, eax
6               <1> ....
7               <1> nextchar:
8 00000003 803800 <1> cmp byte [eax], 0
9 00000006 7403   <1> jz finished
10 00000008 40    <1> inc eax
11 00000009 EBF8  <1> jmp nextchar
12              <1> ....
13              <1> finished:
14 0000000B 29D8  <1> sub eax, ebx
15 0000000D 5B    <1> pop ebx
16 0000000E C3    <1> ret
17              <1> .
18              <1> .
19              <1> ;----- sprint -----
20              <1> ; Функция печати сообщения
21              <1> ; входные данные: mov eax, <message>
22              <1> sprint:
23 0000000F 52    <1> push edx
24 00000010 51    <1> push ecx
25 00000011 53    <1> push ebx
26 00000012 50    <1> push eax
27 00000013 E8E8FFFF <1> call slen
28              <1> ....
29 00000018 89C2  <1> mov edx, eax
30 0000001A 58    <1> pop eax
31              <1> ....
32 0000001B 89C1  <1> mov ecx, eax
33 0000001D BB010000 <1> mov ebx, 1
34 00000022 B8040000 <1> mov eax, 4
35 00000027 CD80  <1> int 80h
36              <1> .

```

Рис. 3.12: Файл листинга

Затем открываем файл с программой lab7-2.asm и удаляем один операнд (рис. 3.13).

```

lab7-2.asm      [----]  7  L:[  1+13  14/
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите В: '
mov eax,
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread

```

Рис. 3.13: Удаление одного операнда

Выполняем трансляцию с получением файла листинга (рис. 3.14).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands

```

Рис. 3.14: Выполнение трансляции

Открываем файл lab7-2.lst и проверяем на наличие ошибки (рис. 3.15).

```

lab7-2.lst      [B---] 90 L:[181+ 9 190/226] *(11682/14544b) 0010 0x00A
6 00000039 35300000      C dd '50'
7                                section .bss
8 00000000 <res Ah>      max resb 10
9 0000000A <res Ah>      B resb 10
10                               section .text
11                               global _start
12                               _start:
13                               ; ----- Вывод сообщения 'Введите B: '
14                               mov eax
14 ***** error: invalid combination of opcode and operands
15 000000E8 E822FFFFFF      call sprint
16                               ; ----- Ввод 'B'
17 000000ED B9[0A000000]    mov ecx,B
18 000000F2 BA0A000000      mov edx,10
19 000000F7 E847FFFFFF      call sread
20                               ; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000]    mov eax,B
22 00000101 E896FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
23 00000106 A3[0A000000]    mov [B],eax ; запись преобразованного числа в 'B'
24                               ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000]    mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000]    mov [max],ecx ; 'max = A'
27                               ; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 0000011D 7F0C            jg check_B ; если 'A>C', то переход на метку 'check_B',
30 0000011F 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx = C'
31 00000125 890D[00000000]    mov [max],ecx ; 'max = C'
32                               ; ----- Преобразование 'max(A,C)' из символа в число

```

Рис. 3.15: Ошибка в листинге файла

Перейдем к выполнению заданий для самостоятельной работы. Создаем файл lab7-3.asm (рис. 3.16).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-3.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $

```

Рис. 3.16: Создание файла

В файл вводим программу для нахождения наименьшей из 3 целочисленных переменных. Значение переменных берем из 7 варианта (рис. 3.17).

```

#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '45'
B dd '67'
C dd '15'
section .bss
min resb 10
section .text
global _start
_start:
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'min'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Рис. 3.17: Ввод текста программы

Создаем исполняемый файл и проверяем его работу. Все верно: из переменных 45, 67 и 15 - 15 является наименьшей, и на выводе мы получаем именно ее (рис. 3.18).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 15

```

Рис. 3.18: Проверка работы исполняемого файла

Теперь создаем файл lab7-4.asm (рис. 3.19).

```

aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-4.asm
aaadmiraljskaya@dk3n55 ~/work/arch-pc/lab07 $ mc

```

Рис. 3.19: Создание файла

В файле пишем программу, которая для введенных с клавиатуры значений x и y вычисляет значение заданной функции $f(x, y)$ и выводит результат вычислений. Функцию берем из 7 варианта (рис. 3.20).


```

lab7-4.asm      [----]  5 L:[ 1+26 27/ 47] *(512 / 735b) 0097
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1 db 'Введите значение переменной x: ',0h
msg2 db 'Введите значение переменной a: ',0h
msg3 db "Результат: ",0h
SECTION .bss
fin resb 10
A resb 10
X resb 10
SECTION .text
GLOBAL _start
_start:
mov eax, msg1
call sprint
mov ecx, A
mov edx, 10
call sread
mov eax, A
call atoi
mov [A], eax
mov eax, msg2
call sprint
mov ecx, X
mov edx, 10
call sread
mov eax, X
call atoi
mov [X], eax
mov ecx, [X]
cmp ecx, [A]
jne func2
mov ax, [A]
mov bx, 6
mul bx
mov [fin], ax
jmp final
func2:
mov ax, [A]
add ax, [X]
mov [fin], ax
jmp final

```

Рис. 3.20: Написание программы

Создаем исполняемый файл и проверяем его работу для значений а,х из варианта №7: x=1, a=1 (рис. 3.21).

```

aaadmiraljskaya@dk3n61 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
aaadmiraljskaya@dk3n61 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
aaadmiraljskaya@dk3n61 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной x: 1
Введите значение переменной a: 1
Результат: 6

```

Рис. 3.21: Создание и проверка работы исполняемого файла

Теперь проверяем его работу для значений $x=2$, $a=1$. Все верно (рис. 3.22).

```

aaadmiraljskaya@dk3n61 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной x: 2
Введите значение переменной a: 1
Результат: 3

```

Рис. 3.22: Проверка работы исполняемого файла

Текст программы первого задания самостоятельной работы:

```

%include 'in_out.asm' section .data msg1 db 'Введите B:',0h msg2 db "Наимень-
шее число:",0h A dd '45' B dd '67' C dd '15' section .bss min resb 10 section .text
global _start _start: mov eax,B call atoi ; Вызов подпрограммы перевода символа в
число mov [B],eax ; запись преобразованного числа в 'B' ; ----- Записываем 'A' в
переменную 'min' mov ecx,[A] ; 'ecx = A' mov [min],ecx ; 'min = A' ; ----- Сравнива-
ем 'A' и 'C' (как символы) cmp ecx,[C] ; Сравниваем 'A' и 'C' jl check_B ; если 'A<C',
то переход на метку 'check_B', mov ecx,[C] ; иначе 'ecx = C' mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число check_B: mov eax,min call
atoi ; Вызов подпрограммы перевода символа в число mov [min],eax ; запись пре-
образованного числа в min ; ----- Сравниваем 'min(A,C)' и 'B' (как числа) mov
ecx,[min] cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B' jl fin ; если 'min(A,C)<B', то пере-
ход на 'fin', mov ecx,[B] ; иначе 'ecx = B' mov [min],ecx ; ----- Вывод результата fin:
mov eax, msg2 call sprint ; Вывод сообщения 'Наименьшее число:' mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)' call quit ; Выход

```

Текст программы второго задания самостоятельной работы:

```

%include 'in_out.asm' ; подключение внешнего файла SECTION .data msg1 db
'Введите значение переменной x:',0h msg2 db 'Введите значение переменной

```

```

a:',0h msg3 db "Результат:",0h SECTION .bss fin resb 10 A resb 10 X resb 10 SECTION
.text GLOBAL _start _start: mov eax, msg1 call sprint mov ecx, A mov edx, 10 call sread
mov eax,A call atoi mov [A], eax mov eax,msg2 call sprint mov ecx, X mov edx, 10 call
sread mov eax, X call atoi mov [X], eax mov ecx,[X] cmp ecx,[A] jne func2 mov ax,[A]
mov bx,6 mul bx mov [fin],ax jmp final func2: mov ax,[A] add ax,[X] mov [fin],ax jmp
final final: mov eax,msg3 call sprint mov eax,[fin] call iprintLF call quit

```

4 Выводы

В процессе выполнения лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и познакомилась с назначением и структурой файла листинга.

Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.