

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №6**  
**по дисциплине «Операционные системы»**  
**Тема: Построение модуля динамической структуры**

Студентка гр. 7383

Маркова А.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

### **Постановка задачи.**

Исследовать возможность построения загрузочного модуля динамической структуры. В отличие от предыдущих лабораторных работ в этой работе рассматривается приложение, состоящее из нескольких модулей, а не из одного модуля простой структуры. В этом случае разумно предположить, что все модули приложения находятся в одном каталоге и полный путь в этот каталог можно взять из среды, как это делалось в работе 2. Понятно, что такое приложение должно запускаться в соответствии со стандартными ОС.

Также нужно исследовать интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция 4B00h прерывания int 21h. Все загрузочные модули находятся в одном каталоге. Необходимо обеспечить возможность запуска модуля динамической структуры из любого каталога.

Описание процедур, используемых в работе:

- `LINE_OUTPUT` – выводит сообщение на экран;
- `TETR_TO_HEX` – вспомогательная функция, которая переводит из двоичной в шестнадцатеричную систему счисления, используется для работы функции `BYTE_TO_HEX`;
- `BYTE_TO_HEX` – переводит байтовое число из регистра `AL` в шестнадцатеричную систему счисления;
- `FREE_MEMORY` – освобождает место в памяти, используя функцию 4Ah прерывания int 21h;
- `PROCESSING` – запуск вызываемого модуля `LR2.COM`;
- `CREATE_BP` – создание блока параметров;
- `RETURN_CODE` – вывод кода завершения программы;
- `NOT_LOADED_ERROR` – обработка ошибок, если программа не была выполнена;
- `BEGINNING` – главная процедура.

Таблица 1 – Описание структуры данных:

Название	Тип	Назначение
Mem_7	db	Строка – сообщение об ошибке освобождения памяти: 7 – разрушен
Mem_8	db	управляющий блок памяти; 8 –
Mem_9	db	недостаточно памяти для выполнения функции; 9 – неверный адрес блока памяти
Err_11	db	Строка – сообщение об ошибке загрузки вызываемой программы: 1 – если номер функции неверен; 2 – если файл не найден; 5 – при ошибке диска; 8 – при недостаточном объеме памяти; 10 – при неправильной строке среды; 11 – если неверен формат
Err_10	db	
Err_1	db	
Err_2	db	
Err_5	db	
Err_8	db	
End_0	db	Строка – сообщение, содержащая причину завершения вызываемой программы: 0 – нормальное завершение; 1 – завершение по Ctrl-Break; 2 – завершение по ошибке устройства; 3 – завершение по функции 31h, оставляющей программу резидентной
End_1	db	
End_2	db	
End_3	db	
PATH	db	Строка, содержащая название вызываемого модуля
KEEP_SS	dw	Переменная для сохранение содержимого регистра SS
KEEP_SP	dw	Переменная для сохранение содержимого регистра SP

END_CODE	db	Строка-сообщение для вывода информации о коде завершения вызываемого модуля
ParameterBlock	dw/dd	Блок параметров для хранения информации о: сегментном адресе среды, сегменте и смещении командной строки, сегменте и смещении первого и второго FCB

### Ход работы.

1. Был написан и отлажен программный модуль .EXE.
2. Смонтирован виртуальный диск k с каталогом tasm.
3. Было произведено транслирование программы с помощью строки tasm «имя файла», в последствии чего создался объектный файл.
4. Линковка загрузочного модуля с помощью команды tlink.

Результат представлен на рис. 1.

```
Z:\>mount k d:/7383/LR6/TASM
Drive K is mounted as local directory d:/7383/LR6/TASM\

Z:\>k:

K:\>cd LAB6

K:\LAB6>tasm lr6
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file:   lr6.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  469k

K:\LAB6>tlink lr6
Turbo Link Version 5.1 Copyright (c) 1992 Borland International
```

Рисунок 1 – Результат сборки программного модуля

5. Получен LR6.EXE модуль из исходного кода для .EXE модуля.
6. Был изменен код программы LR2.COM, перед завершением работы ожидается ввод символа с клавиатуры.
7. Запуск отлаженной программы, когда текущей каталог является каталогом с разработанными модулями. Выведенная информация показана на рис. 2.

```

K:\LAB6>tlink/t lr2
Turbo Link  Version 5.1 Copyright (c) 1992 Borland International

K:\LAB6>lr6
Segment address of inaccessible memory: 9FFF
Segment address of the environment: 1191
Command-line tail: Empty

The contents of the environment area:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Loadable module path:
:\LAB6\LR2.COM _

```

Рисунок 2 – Итог работы программы lr2.com

Программа вызывает LR2.COM, которая останавливается, ожидая символ с клавиатуры.

8. Был введён произвольный символ «F» из числа допустимых A-Z.

Результат представлен на рис. 3.

```

K:\LAB6>tlink/t lr2
Turbo Link  Version 5.1 Copyright (c) 1992 Borland International

K:\LAB6>lr6
Segment address of inaccessible memory: 9FFF
Segment address of the environment: 1191
Command-line tail: Empty

The contents of the environment area:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Loadable module path:
:\LAB6\LR2.COM F
Normal completion!
Exit code: 46

```

Рисунок 3 – Результат запуска программы lr6.exe

Причиной завершения является нормальное завершение работы программы. Код ошибки – ASCII-код «F» в шестнадцатеричной системе счисления.

9. Была запущена отлаженная программа ещё один раз с теми же условиями и введена комбинация Ctrl-C. Выведенная информация показана на рис. 4.

```
K:\LAB6>lr6
Segment address of inaccessible memory: 9FFF
Segment address of the environment: 1191
Command-line tail: Empty

The contents of the environment area:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Loadable module path:
:\LAB6\LR2.COM ♥
Normal completion!
Exit code: 03
```

Рисунок 4 – Повторный запуск программы для тестирования Ctrl-C

Причиной завершения является нормальное завершение работы программы, код завершения – 03. В данном случае, причиной должно было бы являться прерывание по Ctrl-Break, но в DOSBOX игнорируется это прерывание.

10. Изменена текущая директория на каталог, в котором находится папка с нужными файлами. Расположение модулей представлено на рис. 5.

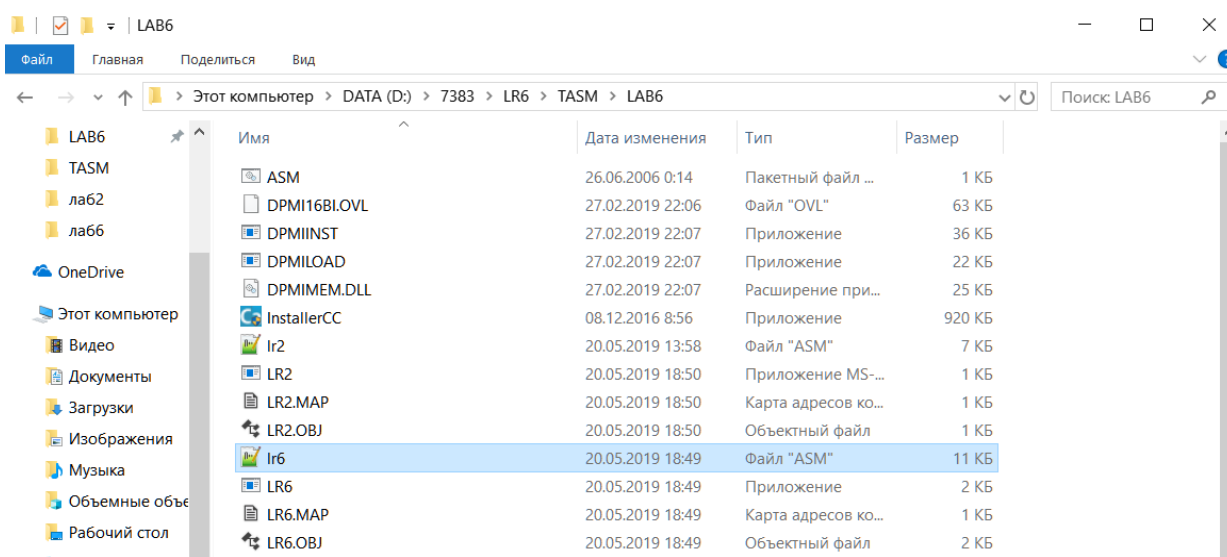


Рисунок 5 – Размещение модулей программы

11. Был запущен программный модуль LR6.EXE и введен символ «F».  
Результат показан на рис. 6.

```
K:\LAB6>cd ..  
  
K:\>\LAB6\lr6  
Segment address of inaccessible memory: 9FFF  
Segment address of the environment: 1191  
Command-line tail: Empty  
  
The contents of the environment area:  
PATH=Z:\  
COMSPEC=Z:\COMMAND.COM  
BLASTER=A220 I7 D1 H5 T6  
  
Loadable module path:  
:\LAB6\LR2.COM F  
Normal completion!  
Exit code: 46
```

Рисунок 6 – Тестирование ввода символа, когда модули находятся не в текущем каталоге

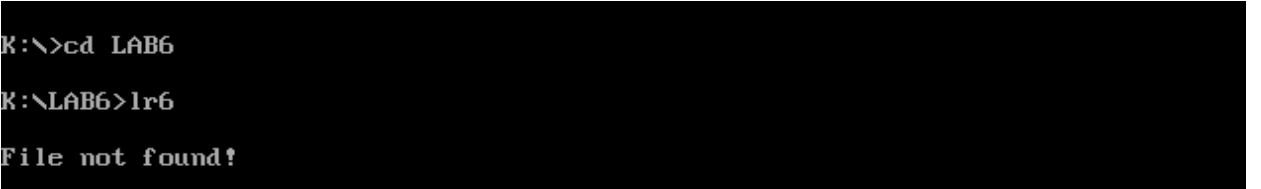
12. Была запущена отлаженная программа ещё один раз с теми же условиями и введена комбинация Ctrl-C. Выведенная информация показана на рис. 7.

```
K:\>\LAB6\lr6  
Segment address of inaccessible memory: 9FFF  
Segment address of the environment: 1191  
Command-line tail: Empty  
  
The contents of the environment area:  
PATH=Z:\  
COMSPEC=Z:\COMMAND.COM  
BLASTER=A220 I7 D1 H5 T6  
  
Loadable module path:  
:\LAB6\LR2.COM ♥  
Normal completion!  
Exit code: 03
```

Рисунок 7 – Тестирование ввода комбинации клавиш Ctrl-C, когда модули находятся не в текущем каталоге

13. Запуск отлаженной программы, когда модули находятся в разных каталогах (LR6.EXE – в K:\LAB6, а LR2.COM – K:\). Выведенное сообщение представлено на рис. 8.





```
K:\>cd LAB6  
K:\LAB6>lr6  
File not found!
```

Рисунок 8 – Запуск программы, когда модули находятся в разных каталогах

### **Выводы.**

В ходе данной лабораторной работы была изучена возможность построения загрузочного модуля динамической структуры, а также модифицирован ранее построенный программный модуль. Было реализовано и исследовано взаимодействие между вызывающим и вызываемым модулями. Код программы представлен в приложении А.

## **Ответы на контрольные вопросы.**

### **1. Как реализовано прерывание Ctrl-C?**

Если были нажаты комбинации клавиш Ctrl-C или Ctrl-Break, то вызывается прерывание int 23h. Когда DOS распознает, что пользователь нажал Ctrl-Break или Ctrl-C, управление передается по адресу – 0000:008Ch. Адрес по вектору int 23h копируется в поле PSP Ctrl-Break Address функциями DOS 26h (создать PSP) и 4Ch (EXEC). Исходное значение адреса обработчика Ctrl-Break восстанавливается из PSP при завершении программы. Таким образом, по завершении порожденного процесса будет восстановлен адрес обработчика Ctrl-Break из родительского процесса.

### **2. В какой точке заканчивается вызываемая программа, если код причины завершения 0?**

Если код причины завершения 0, то вызываемая программа заканчивается в точке вызова функции 4Ch прерывания int 21h.

### **3. В какой точке заканчивается вызываемая программа по прерыванию Ctrl-C?**

В месте, где программа ожидала ввода символа: в точке вызова функции 01h прерывания int 21h.

# ПРИЛОЖЕНИЕ А

## LR6.ASM

```

EOFLine EQU '$' ;определение символьной константы
; $ - "конец строки"

;-----
ASTACK SEGMENT STACK
    DW 64 DUP (?)
ASTACK ENDS
;ДАННЫЕ
;-----
DATA SEGMENT
ParameterBlock dw 0000h ;сегментный адрес среды
                dd 0000h ;сегмент и смещение командной строки
                dd 0000h ;сегмент и смещение первого FCB (File Control Block)
                dd 0000h ;сегмент и смещение второго FCB
Mem_8          DB 0DH, 0AH, 'Not enough memory to perform the function!', 0DH, 0AH, EOFLine
PATH           DB ' ', 0DH, 0AH, EOFLine, 0
End_2          DB 0DH, 0AH, 'The completion of the device error!', 0DH, 0AH, EOFLine
Mem_9          DB 0DH, 0AH, 'Wrong address of the memory block!', 0DH, 0AH, EOFLine
Err_1          DB 0DH, 0AH, 'The number of function is wrong!', 0DH, 0AH, EOFLine
Mem_7          DB 0DH, 0AH, 'Memory control unit destroyed!', 0DH, 0AH, EOFLine
Err_10         DB 0DH, 0AH, 'Incorrect environment string!', 0DH, 0AH, EOFLine
End_3          DB 0DH, 0AH, 'Completion by function 31h!', 0DH, 0AH, EOFLine
Err_8          DB 0DH, 0AH, 'Insufficient memory!', 0DH, 0AH, EOFLine
End_0          DB 0DH, 0AH, 'Normal completion!', 0DH, 0AH, EOFLine
End_1          DB 0DH, 0AH, 'End by Ctrl-Break!', 0DH, 0AH, EOFLine
Err_2          DB 0DH, 0AH, 'File not found!', 0DH, 0AH, EOFLine
Err_11         DB 0DH, 0AH, 'Wrong format!', 0DH, 0AH, EOFLine
Err_5          DB 0DH, 0AH, 'Disk error!', 0DH, 0AH, EOFLine
END_CODE       DB 'Exit code: ', 0DH, 0AH, EOFLine
KEEP_SS        DW 0000h
KEEP_SP        DW 0000h
DATA ENDS
;-----
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK
START: JMP BEGINNING ;переход на метку
;ПРОЦЕДУРЫ
;-----
LINE_OUTPUT PROC NEAR ;вывод строки
    mov AH, 0009h
    int 21h
    ret
LINE_OUTPUT ENDP
;-----
TETR_TO_HEX PROC near ;из двоичной в шестнадцатеричную ss
    and AL, 0Fh ;PROC near - вызывается в том же сегменте, в котором определена
    cmp AL, 09
    jbe NEXT
    add AL, 07
NEXT: add AL, 30h
    ret
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC near ;байтовое число в шестнадцатеричную ss
    push CX
    mov AH, AL
    call TETR_TO_HEX

```

```

    xchg AL, AH
    mov CL, 4
    shr AL, CL
    call TETR_TO_HEX
    pop CX
    ret
BYTE_TO_HEX ENDP
;-----
FREE_MEMORY PROC NEAR
    mov BX, offset LAST_BYTE
    mov AX, ES
    sub BX, AX
    mov CL, 0004h
    shr BX, CL
    mov AH, 4Ah
    int 21h
    jnc WITHOUT_ERROR
    jmp WITH_ERROR

    WITHOUT_ERROR:
        ret

    WITH_ERROR:
    MEM_7_ERROR:
        cmp AX, 0007h
        jne MEM_8_ERROR
        mov DX, offset Mem_7
        jmp END_ERROR

    MEM_8_ERROR:
        cmp AX, 0008h
        jne MEM_9_ERROR
        mov DX, offset Mem_8
        jmp END_ERROR

    MEM_9_ERROR:
        mov DX, offset Mem_9

    END_ERROR:
        call LINE_OUTPUT
        xor AL, AL
        mov AH, 4Ch
        int 21h
FREE_MEMORY ENDP

PROCESSING PROC NEAR
    mov ES, ES:[2Ch]
    mov SI, 0000h
    cycle:
        mov DL, ES:[SI]
        cmp DL, 0000h
        je end_cycle
        inc SI
        jmp cycle
    end_cycle:
        inc SI
        mov DL, ES:[SI]
        cmp DL, 0000h

```

```

;освобождение места в памяти
;перед вызовом функции надо определить объём памяти, необходимый Lr6
;кладем в BX адрес конца программы
;ES - начало программы
;BX = BX - ES, число параграфов, которые будут выделяться программе
;переводим в параграфы
;функция сжатия блока памяти
;флаг CF = 0, если нет ошибки
;обработка ошибок CF=1 AX = код ошибки, если CF установлен

```

```

;разрушен управляющий блок памяти

```

```

;недостаточно памяти для выполнения функции

```

```

;неверный адрес блока памяти

```

```

;сегментный адрес среды, передаваемый программе

```

```

;проверка: конец строки?

```

```

;проверка: конец среды?

```

```

jne cycle
add SI, 0003h
push DI
lea DI, PATH
loop_:
mov DL, ES:[SI]
cmp DL, 0000h
je end_loop
mov [DI], DL
inc DI
inc SI
jmp loop_
end_loop:
sub DI, 0007h
mov [DI], byte ptr 'L'
mov [DI + 0001h], byte ptr 'R'
mov [DI + 0002h], byte ptr '2'
mov [DI + 0003h], byte ptr '.'
mov [DI + 0004h], byte ptr 'C'
mov [DI + 0005h], byte ptr 'O'
mov [DI + 0006h], byte ptr 'M'
mov [DI + 0007h], byte ptr 0h
pop DI
;
mov KEEP_SP, SP
mov KEEP_SS, SS
push DS
pop ES
mov BX, offset ParameterBlock
mov DX, offset PATH
mov AX, 4B00h
int 21h
jnc is_loaded
push AX
mov AX, DATA
mov DS, AX
pop AX
mov SS, KEEP_SS; восстановление DS, SS, SP
mov SP, KEEP_SP
call NOT_LOADED_ERROR
is_loaded:
mov AX, 4d00h
int 21h
call RETURN_CODE
ret
PROCESSING ENDP

CREATE_BP PROC NEAR
mov AX, ES:[2Ch]
mov ParameterBlock, AX
mov ParameterBlock + 0002h, ES
mov ParameterBlock + 0004h, 0080h
ret
CREATE_BP ENDP

RETURN_CODE PROC NEAR

```

*;SI указывает на начало маршрута*

*;проверка: конец маршрута?*

*;сохраняем содержимое регистров SS и SP*

*;вызываем загрузчик OS*

*;если вызываемая программа не была загружена,  
то устанавливается флаг переноса CF=1 и в AX заносится код ошибки*

*;в AH - причина, в AL - код завершения*

*;сегментный адрес параметров командной строки  
смещение параметров командной строки*

```

cmp AH, 0000h ;нормальное завершение
mov DX, offset End_0
je EXIT_CODE
cmp AH, 0001h ;завершение по Ctrl-Break
mov DX, offset End_1
je EXIT_CODE
cmp AH, 0002h ;завершение по ошибке устройства
mov DX, offset End_2
je EXIT_CODE
cmp AH, 0003h ;завершение по функции 31h, оставляющей программу резидентной
mov DX, offset End_3

EXIT_CODE:
call LINE_OUTPUT ;выводим код завершения на экран
mov DI, offset END_CODE
call BYTE_TO_HEX
add DI, 000Bh
mov [DI], AL
add DI, 0001h
xchg AH, AL
mov [DI], AL
mov DX, offset END_CODE
call LINE_OUTPUT
xor AL, AL
mov AH, 4Ch
int 21h
RETURN_CODE ENDP

NOT_LOADED_ERROR PROC NEAR ;обработка ошибок, если программа не была выполнена
;если номер функции неверен
cmp AX, 0001h
mov DX, offset Err_1
je NOT_LOADED
cmp AX, 0002h ;если файл не найден
mov DX, offset Err_2
je NOT_LOADED
cmp AX, 0005h ;при ошибке диска
mov DX, offset Err_5
je NOT_LOADED
cmp AX, 0008h ;при недостаточном объёме памяти
mov DX, offset Err_8
je NOT_LOADED
cmp AX, 000Ah ;при неправильной строке среды
mov DX, offset Err_10
je NOT_LOADED
cmp AX, 000Bh ;если неверен формат
mov DX, offset Err_11

NOT_LOADED:
call LINE_OUTPUT
xor AL, AL
mov AH, 4Ch
int 21h
NOT_LOADED_ERROR ENDP
;
-----
BEGINNING:
mov AX, DATA
mov DS, AX
call FREE_MEMORY
call CREATE_BP
call PROCESSING
xor AL, AL
mov AH, 4Ch ;выход
int 21h
LAST_BYTE:
CODE ENDS
END START

```