

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

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

Студентка гр. 7381

\_\_\_\_\_

Кортев Ю.В.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2019

### **Цель работы.**

Исследование возможности 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.

В работе исследуется 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 функций:**

Print	Процедура вызова прерывания, печатающего строку
MAIN	Главная функция программы

### **Описание структур данных:**

Parameter_block	Строки, оповещающие о различных ошибках
error1 – error9	Строки, содержащие причины завершения дочерней программы

finishcode	Строка, оповещающая, что далее следует код завершения
PARMBLOCK	Указатель на блок параметров
finishednormally, ctrlfinish, deverrfinish, funcfinish	Строки, содержащие информацию о статусе/причине завершения
KEEP_SS KEEP_SP	Переменные для сохранения регистров SS и SP перед вызовом модуля

### **Выполнение работы.**

Был написан программный модуль .EXE, который выполняет следующие функции:

1. Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится он сам. Вызываемому модулю передается новая среда, созданная вызывающим модулем и новая командная строка.
2. Вызываемый модуль запускается с использованием загрузчика.
3. После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Необходимо проверять причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

### **Тестирование**

- 1) Запуск отлаженной программы, когда текущим каталогом является каталог с разработанными модулями и ввод произвольного символа из числа A-Z.

```
C:\>LAB6.EXE
Inaccessible memory segment begins at address: 9FFF
Environment segment address: 0823
No tail

Environment contents:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Module path:
C:\LAB2.COM1
Program finished with code 1
Finished normally.
```

- 2) Запуск отлаженной программы, когда текущим каталогом является другой каталог, отличный от того, где содержатся программные модули и ввод комбинация клавиш.

```
C:\>lab6/lab6.exe
Inaccessible memory segment begins at address: 9FFF
Environment segment address: 0843
No tail

Environment contents:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Module path:
C:\LAB2.COMz
Program finished with code z
Finished normally.
```

3) Запуск отлаженной программы, когда модули находятся в разных каталогах.

```
C:\>lab6/lab6.exe  
The file wasn't found.
```

## **Выводы**

В результате выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

## **Ответы на контрольные вопросы**

1. Как реализовано прерывание Ctrl-C?

Ответ: При нажатии сочетания клавиш Ctrl-C или Ctrl-Break вызывается прерывание int 23h, которое завершает текущий процесс, при этом управление передается по адресу 0000:008c.

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

Ответ: Если код причины завершения 0, то вызываемая программа заканчивается в месте вызова функции 4Ch прерываний int 21h.

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

Ответ: При нажатии сочетания клавиш Ctrl+C программа завершает работу в том месте, где программа ожидала ввода символа, т.е. в точке вызова функции 01h прерывания int 21h.



## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД

```
AStack SEGMENT STACK
    DW 100h DUP(?)
AStack ENDS
```

#### DATA SEGMENT

```
error1 db 'Memory control block destroyed.', 0dh, 0ah, '$'
error2 db 'Adress of memory block is incorrect.', 0dh, 0ah, '$'
error3 db 'Not enough memory for function.', 0dh, 0ah, '$'
error4 db 'File wasnt found.', 0dh, 0ah, '$'
error5 db 'Disk error.', 0dh, 0ah, '$'
error6 db 'Incorrect number of function.', 0dh, 0ah, '$'
error7 db 'Not enough memory.', 0dh, 0ah, '$'
error8 db 'Incorrect environment string.', 0dh, 0ah, '$'
error9 db 'Incorrect format.', 0dh, 0ah, '$'
```

```
finishcode db 0dh, 0ah, 'Program finished with code # ', 0dh, 0ah, '$'
finishednorm db 'Finished normally.', 0dh, 0ah, '$'
ctrlfinish db 'Finished by Ctrl-Break', 0dh, 0ah, '$'
deverrfinish db 'Finished by device error.', 0dh, 0ah, '$'
funcfinish db 'Finished by 31h fun.', 0dh, 0ah, '$'
DATA ENDS
```

#### CODE SEGMENT

```
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:AStack
```

```
KEEP_SS dw ?
KEEP_SP dw ?
```

```
par_block db 14 dup(0)
filepath db 70 dup(0)
position dw 0
```

#### Print PROC near

```
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
```

#### Print ENDP

#### MAIN PROC

```
    mov ax, DATA
    mov ds, ax
    mov ax, END CB
    mov bx, es
    sub ax, bx
    mov cx, 0004h
    shl ax, cl
    mov bx, ax ; в регистр bx-число параграфов, которые будут выделяться в
программе
    mov ax, 4A00h ; сначала нужно освободить место в памяти
```

```

    int 21h    ; эта функция позволяет уменьшить отведенный программе блок
памяти
    jnc success ; если не может быть выполнена то выставится флаг CF=1 и в ax
вынесется код ошибки
    cmp ax,07h ; разрушен управляющий блок памяти
    je error1j
    cmp ax,08h ; недостаточно памяти для выполнения функции
    je error3j
    cmp ax,09h ; неверный адрес блока памяти
    je error2j
error1j:
    lea dx,error1
    call Print
    jmp ending
error2j:
    lea dx,error2
    call Print
    jmp ending
error3j:
    lea dx,error3
    call Print
    jmp ending

success:
    mov byte ptr [par_block],00h
    mov es,es:[2Ch]
    mov si,00h
is_zero:
    mov ax,es:[si]
    inc si
    cmp ax,0000h
    jne is_zero
    add si,03h
    mov di,00h
write_path:
    mov cl,es:[si]
    cmp cl,00h
    je flagn
    cmp cl,'\'
    jne not_yet
    mov position,di
not_yet:
    mov byte ptr [filepath+DI],cl
    inc si
    inc di
    jmp write_path
flagn:
    mov bx,position
    inc bx
    mov byte ptr [filepath+BX],\'
    inc bx
    mov byte ptr [filepath+BX],\'a\'
    inc bx
    mov byte ptr [filepath+BX],\'b\'
    inc bx

```



```

mov byte ptr [filepath+BX],'2'
inc bx
mov byte ptr [filepath+BX], '.'
inc bx
mov byte ptr [filepath+BX], 'c'
inc bx
mov byte ptr [filepath+BX], 'o'
inc bx
mov byte ptr [filepath+BX], 'm'
inc bx
mov byte ptr [filepath+BX], '$'
push ds
push es
mov KEEP_SP, sp
mov KEEP_SS, ss
mov sp, 0FEh
mov ax, CODE
mov ds, ax
mov es, ax
lea bx, par_block
lea dx, filepath
mov ax, 4B00h ; загрузчик OC
int 21h
mov ss, cs:KEEP_SS
mov sp, cs:KEEP_SP
pop es
pop ds

```

```

jnc is_performed_4Bh
cmp ax, 01h
je error6j
cmp ax, 02h
je error4j
cmp ax, 05h
je error5j
cmp ax, 08h
je error3j_4Bh
cmp ax, 0Ah
je error8j
cmp ax, 0Bh
je error9j
error6j:
    lea dx, error6
    call Print
    jmp ending
error4j:
    lea dx, error4
    call Print
    jmp ending
error5j:
    lea dx, error5
    call Print
    jmp ending
error3j_4Bh:

```

```

        lea dx,error7
        call Print
        jmp ending
error8j:
        lea dx,error8
        call Print
        jmp ending
error9j:
        lea dx,error9
        call Print
        jmp ending
is_performed_4Bh:
        mov ax,4D00h ; обработка завершения - в AH причина, в AL код завершения
        int 21h
        mov bx,ax
        add bh,30h
        lea di,finishcode
        mov [di+29],bl
        lea dx,finishcode
        call Print
        cmp ah,00h
        je finish_0
        cmp ah,01h
        je finish_1
        cmp ah,02h
        je finish_2
        cmp ah,03h
        je finish_3
finish_0:
        lea dx,finishednorm
        call Print
        jmp ending
finish_1:
        lea dx,ctrlfinish
        call Print
        jmp ending
finish_2:
        lea dx,deverrfinish
        call Print
        jmp ending
finish_3:
        lea dx,funcfinish
        call Print
ending:
        mov ah,4Ch
        int 21h
MAIN ENDP
CODE ENDS

ENDCB SEGMENT
ENDCB ENDS

END MAIN

```