**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №6**

**по дисциплине «Операционные системы»**

Тема: Построение модуля динамической структуры

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6381 |  | Лопатина А.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи**

**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. Понятно, что такое приложение должно запускаться в соответствии со стандартами ОС.

В работе исследуется интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция 4В00h прерывания int 21h. Все загрузочные модули находятся в одном каталоге. Необходимо обеспечить возможность запуска модуля динамической структуры из любого каталога.

**Сведения о функциях и структурах данных управляющей программы.**

Сведения о функциях:

|  |  |
| --- | --- |
| Название | Назначение |
| OUTPUT\_PROC | Вывод на экран |
| ERROR\_PROC | Выявление ошибок, если программа не была загружена |
| COMPLETION\_PROC | Выявление причины завершения |

Сведения о структурах данных:

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| env\_addr | dw | Сегментный адрес среды |
| cmd | dd | Сегмент и смещение командной строки |
| seg\_of\_1FCB | dd | Сегмент и смещение первого FCB |
| seg\_of\_2FCB | dd | Сегмент и смещение второго FCB |
| keep\_sp | dw | Переменная для хранения sp |
| keep\_ss | dw | Переменная для хранения ss |
| path\_str | db | Путь к лаб2 |
| EOL | db | Переход на новую строку |
| err\_1\_7 | db | Разрушен управляющий блок памяти |
| err\_1\_8 | db | Недостаточно памяти для выполнения функции |
| err\_1\_9 | db | Неверный адрес блока памяти |
| finish | db | Программа завершена |
| finish\_0 | db | Нормальное завершение |
| finish­\_1 | db | Завершение по Ctrl+Break |
| finish\_2 | db | Завершение по ошибке устройства |
| finish\_3 | db | Завершение по функции 31h |
| err\_2 | db | Программа не загружена |
| err\_2\_1 | db | Номер функции неверен |
| err\_2\_2 | db | Файл не найден |
| err\_2\_5 | db | Ошибка диска |
| err\_2\_8 | db | Недостаточный объем памяти |
| err\_2\_10 | db | Неправильная строка среды |
| err\_2\_11 | db | Неверен формат |

**Последовательность действий, выполняемых программой.**

Модуль типа .EXE 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.

В качестве вызываемой программы взята программа lab2, которая распечатывает среду и командную строку.

**Результат работы программы.**

1. Написали и отладили программный модуль .EXE. Запустили отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры.

Введен произвольный символ m.

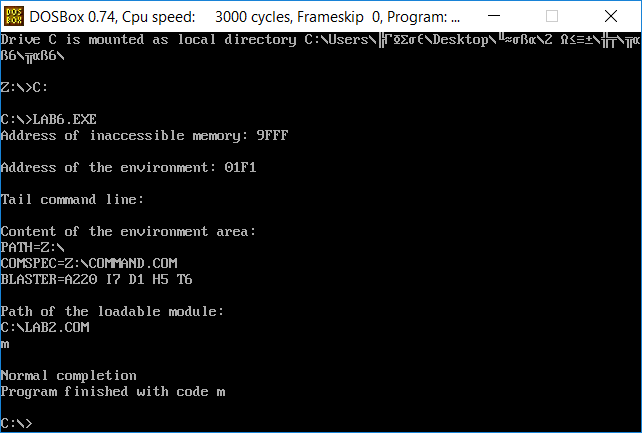


рис. 1

1. Запустили отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры.

Введена комбинация Ctrl-C.

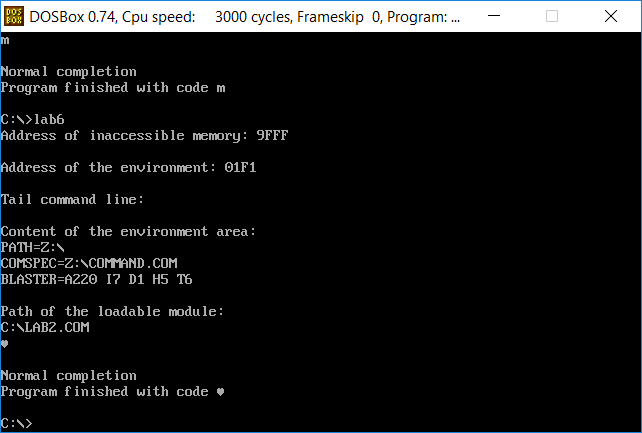


рис. 2

Причиной завершения является нормальное завершение программы, так как DOSbox игнорирует прерывание по Ctrl+C.

1. Запустили отлаженную программу, когда текущим каталогом является какой-нибудь другой каталог, отличный от того, в котором содержатся разработанные программные модули.

Повторили ввод комбинации клавиш.

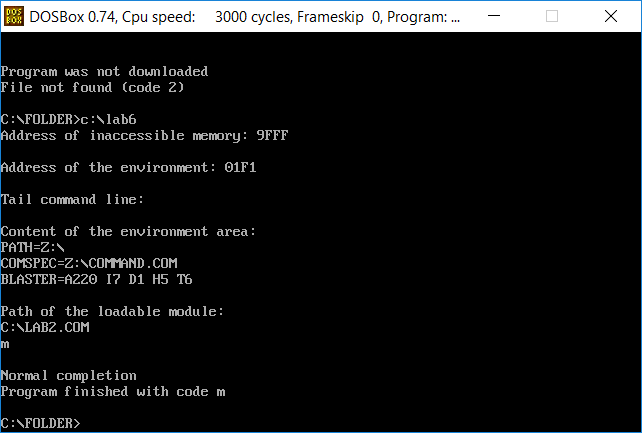


рис. 3 Ввод символа ‘m’

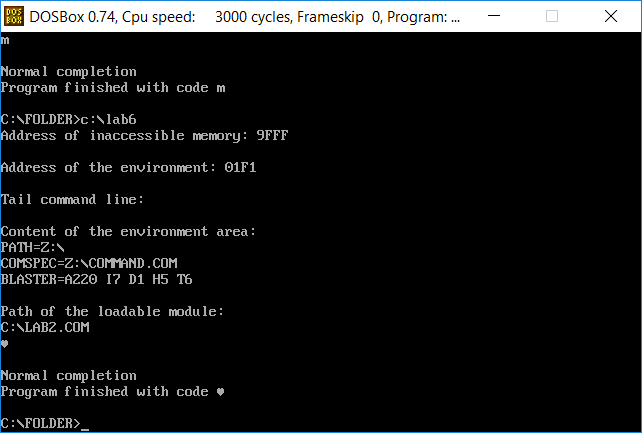


рис. 4 Ввод комбинации Ctrl+C

1. Запустили отлаженную программу, когда модули находятся в разных каталогах. lab6.exe располагалось в папке folder, а lab2.com в каталоге на уровень выше.

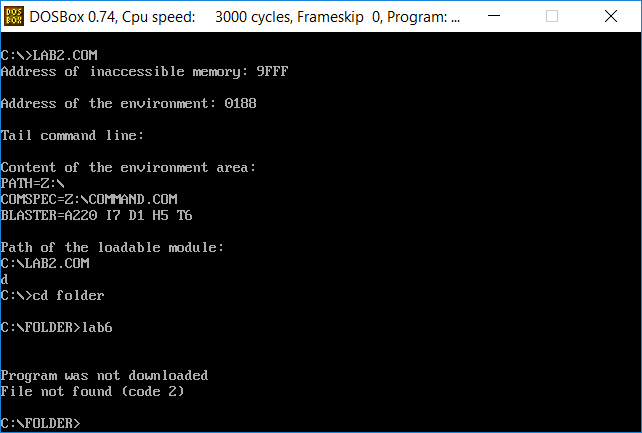


рис. 5

**Описание результатов исследования проблем, поставленных в лабораторной работе**

1. Как реализовано прерывание CTRL+C?

Ответ: прерывание Ctrl-C – прерывание с номером 23h. Если было нажато сочетание клавиш ctrl+c, то управление передаётся по адресу 0000:008Ch. Этот адрес копируется в PSP функциями 26h и 4Ch и восстанавливается из него при завершении программы;

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

Ответ: если код завершения 0, то программа завершается при выполнении функции 4Ch прерывания int 21h;

1. В какой точке заканчивается вызываемая программа по прерывания Ctrl+C?

Ответ: если во время выполнения программы было нажато Ctrl+C, то программа завершится непосредственно в том месте, в котором произошло нажатие сочетания клавиш (то есть в месте ожидания нажатия клавиши: 01h вектора прерывания 21h);

**Заключение**

В результате выполнения лабораторной работы были изучены возможности построения загрузочного модуля динамической структуры и разработано приложение в возможностью запуска модуля динамической структуры из любого каталога.

Приложения

**Код программы** **lab6**.**asm**

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

AStack SEGMENT STACK

DW 64 DUP(?)

AStack ENDS

DATA SEGMENT

env\_addr dw ? ; сегментный адрес среды

cmd dd ? ; сегмент и смещение командной строки

seg\_of\_1FCB dd ? ; сегмент и смещение первого FCB

seg\_of\_2FCB dd ? ; сегмент и смещение второго FCB

keep\_sp dw ?

keep\_ss dw ?

path\_str db ' ',0dH,0ah,'$',0h

EOL db ' ',0dh,0ah,'$'

err\_1\_7 db 'The memory block is destroyed (code 7)', 0dh, 0ah, '$'

err\_1\_8 db 'Not enought memory to perform the function (code 8)', 0dh, 0ah, '$'

err\_1\_9 db 'Wrong adress of memory block (code 9)', 0dh, 0ah, '$'

DATA ENDS

CODE SEGMENT

;----------------------------

OUTPUT\_PROC PROC NEAR ;Вывод на экран сообщения

push ax

mov ah, 09h

int 21h

pop ax

ret

OUTPUT\_PROC ENDP

;----------------------------

COMPLETION\_PROC PROC NEAR

jmp begin\_f

finish db 'Program finished with code ', 0dh, 0ah, '$'

finish\_0 db 'Normal completion', 0dh, 0ah, '$'

finish\_1 db 'Completion by Ctrl-Break', 0dh, 0ah, '$'

finish\_2 db 'Completion by device error', 0dh, 0ah, '$'

finish\_3 db 'Completion by 31h function', 0dh, 0ah, '$'

begin\_f:

push ds

push ax

push dx

push bx

mov ax, 4D00h

int 21h

push ax

mov ax, SEG finish

mov ds, ax

pop ax

lea dx, finish\_0

cmp ah, 0h

je output\_err\_1

lea dx, finish\_1

cmp ah, 1h

je output\_err\_1

lea dx, finish\_2

cmp ah, 2h

je output\_err\_1

lea dx, finish\_3

cmp ah, 3h

output\_err\_1:

call OUTPUT\_PROC

lea dx, finish

mov bx, dx

add bx, 1Bh

mov byte ptr [bx], al

call OUTPUT\_PROC

pop bx

pop dx

pop ax

pop ds

ret

COMPLETION\_PROC ENDP

;----------------------------

ERROR\_PROC PROC NEAR

jmp begin2

err\_2 db 'Program was not downloaded', 0dh, 0ah, '$'

err\_2\_1 db 'Wrong number of the function (code 1)', 0dh, 0ah, '$'

err\_2\_2 db 'File not found (code 2)', 0dh, 0ah, '$'

err\_2\_5 db 'Disk error (code 5)', 0dh, 0ah, '$'

err\_2\_8 db 'Not enought memory (code 8)', 0dh, 0ah, '$'

err\_2\_10 db 'Wrong enviroment string (code 10)', 0dh, 0ah, '$'

err\_2\_11 db 'Wrong format (code 11)', 0dh, 0ah, '$'

begin2:

push ds

push ax

push dx

push ax

mov ax, SEG err\_2

mov ds, ax

pop ax

lea dx, err\_2

call OUTPUT\_PROC

lea dx, err\_2\_1

cmp ax, 1h

je output\_err\_2

lea dx, err\_2\_2

cmp ax, 2h

je output\_err\_2

lea dx, err\_2\_5

cmp ax, 5h

je output\_err\_2

lea dx, err\_2\_8

cmp ax, 8h

je output\_err\_2

lea dx, err\_2\_10

cmp ax, 10h

je output\_err\_2

lea dx, err\_2\_11

cmp ax, 11h

output\_err\_2:

call OUTPUT\_PROC

pop dx

pop ax

pop ds

ret

ERROR\_PROC ENDP

;----------------------------

Main PROC FAR

mov ax, DATA

mov ds, ax

;Освобождение памяти

lea bx, ENDPROG

mov cl,4h

shr bx,cl

add bx,30h ;размер памяти, необходимый для лаб6

mov ah,4ah

int 21h

jnc success ; CF=0

lea dx, err\_1\_7

cmp ax, 7h

je output\_err

lea dx, err\_1\_8

cmp ax, 8h

je output\_err

lea dx, err\_1\_9

output\_err:

call OUTPUT\_PROC

jmp quit

success:

;Заполнение блока параметров

mov env\_addr, 00h

mov ax, es

mov word ptr cmd, ax

mov word ptr cmd+2, 0080h

mov word ptr seg\_of\_1FCB, ax

mov word ptr seg\_of\_2FCB+2, 005ch

mov word ptr seg\_of\_2FCB,ax

mov word ptr seg\_of\_1FCB, 006ch

;Подготовка среды, содержащей имя и путь вызываемой программмы

;push es

;push dx

;push bx

mov es, es:[2Ch]; сегментный адрес среды, передаваемый программе

mov si, 0

env:

mov dl, es:[si]

cmp dl, 00h ; конец строки?

je EOL\_

inc si

jmp env

EOL\_:

inc si

mov dl, es:[si]

cmp dl, 00h ;конец среды?

jne env

add si, 03h ; si указывает на начало маршрута

push di

lea di, path\_str

path\_:

mov dl, es:[si]

cmp dl, 00h ;конец маршрута?

je EOL2

mov [di], dl

inc di

inc si

jmp path\_

EOL2:

sub di, 05h

mov [di], byte ptr '2'

mov [di+2], byte ptr 'C'

mov [di+3], byte ptr 'O'

mov [di+4], byte ptr 'M'

mov [di+5], byte ptr 0h

pop di

;pop bx

;pop ds

;pop es

;Сохраняем содержимое регистров SS и SP переменных

push ds

mov keep\_sp, sp

mov keep\_ss, ss

mov ax, DATA

mov es, ax ;es:bx должен указывать на блок параметров

lea bx, env\_addr

mov ds, ax ; ds:dx должен указывать на подготовленную строку

lea dx, path\_str

mov ax, 4B00h ; Вызываем загрузчик OS

int 21h

;Восстанавливаем параметры

pop ds

mov ss, keep\_ss

mov sp, keep\_sp

push ax

push dx

push ds

mov ax, DATA

mov ds, ax

lea dx, EOL

call OUTPUT\_PROC

call OUTPUT\_PROC

pop ds

pop dx

pop ax

jnc null\_ ; CF=0

call ERROR\_PROC

jmp quit

null\_:

call COMPLETION\_PROC

quit:

xor al, al

mov ah, 4ch

int 21h

Main ENDP

ENDPROG:

CODE ENDS

END Main