

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

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

Студент гр. 8383

Степанов В.Д.

Преподаватель

Губкин А.Ф.

Санкт-Петербург

2020

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры.

Выполнение работы.

1. Был написан программный модуль типа .EXE, который выполняет следующий функции:

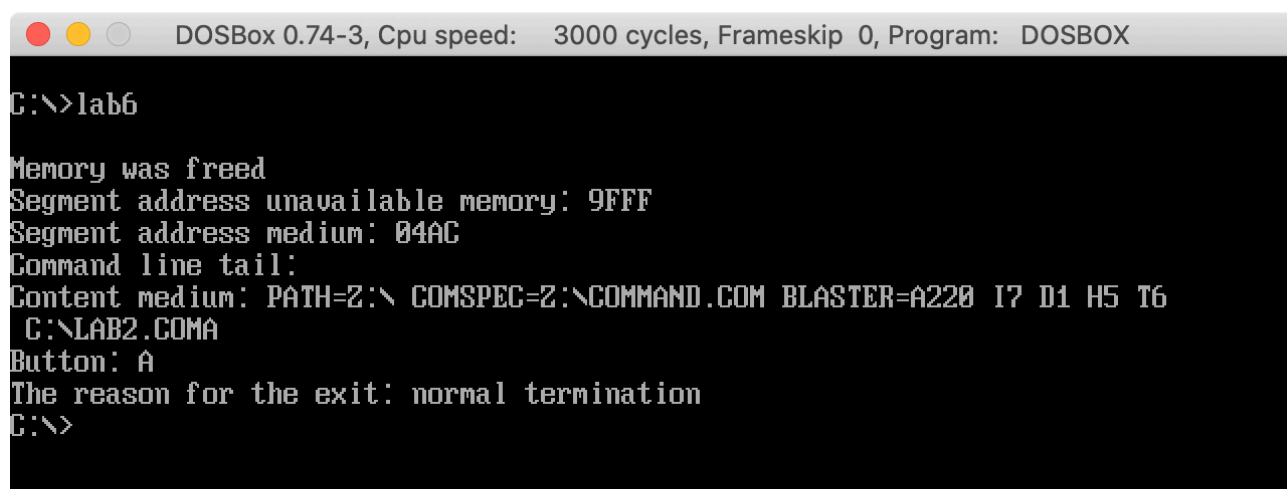
1) Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится он сам. Вызываемому модулю передается новая среда, созданная вызывающим модулем и новая командная строка.

2) Вызываемый модуль запускается с помощью загрузчика.

3) После запуска проверяется выполнение загрузчика, а затем результат вызываемой программы.

В качестве вызываемой программы была взята программа ЛР 2, которая распечатывает среду и командную строку. Данная программа была модифицирована, теперь перед выходом просит ввести символ.

2. Был запущен модуль программный модуль, после был введен символ 'А'. Результат запуска представлен на рисунке 1.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>lab6
Memory was freed
Segment address unavailable memory: 9FFF
Segment address medium: 04AC
Command line tail:
Content medium: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
C:\LAB2.COM
Button: A
The reason for the exit: normal termination
C:\>
```

Рисунок 1 – Результат работы программы с вводом 'А'

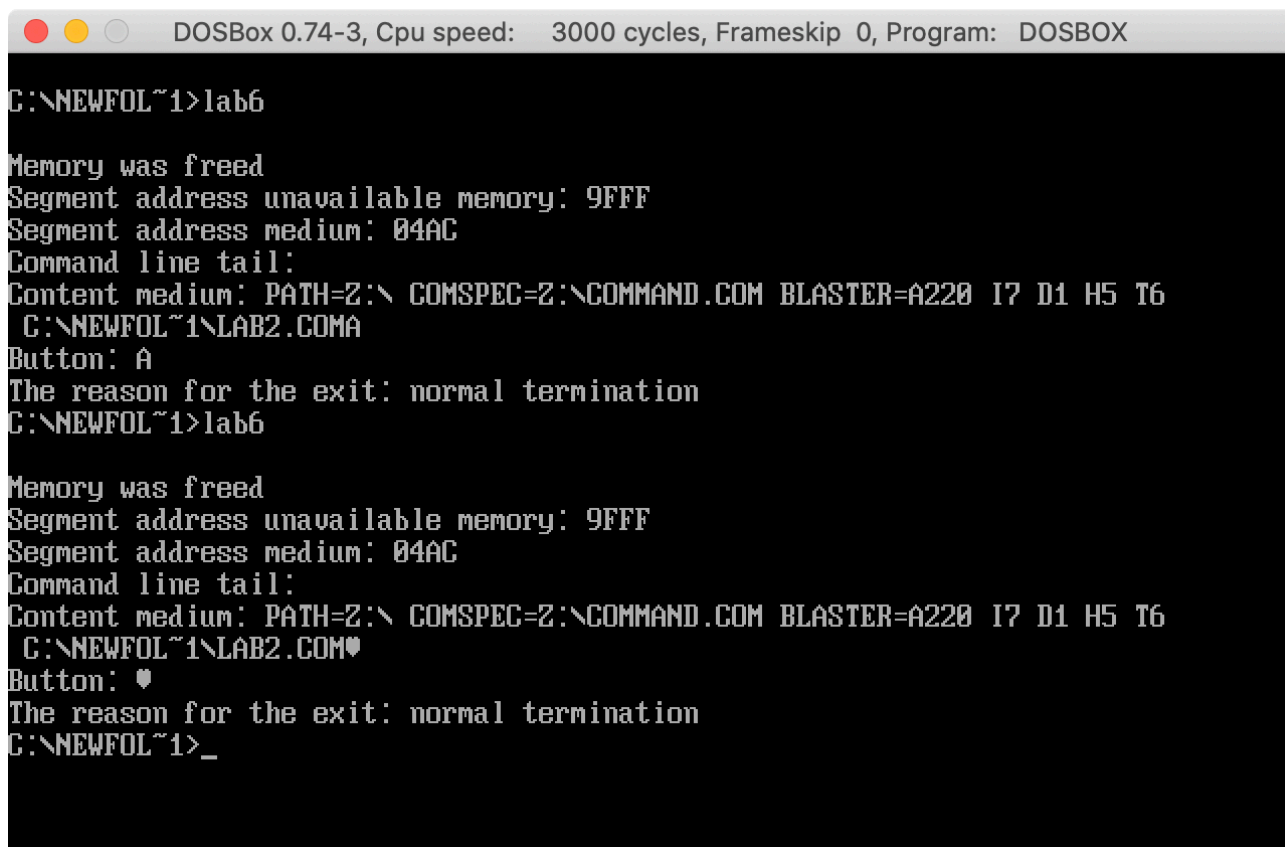
3. Был запущен программный модуль, после была введена комбинация символов "Ctrl+C". Результат запуска представлен на рисунке 2.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>lab6
Memory was freed
Segment address unavailable memory: 9FFF
Segment address medium: 04AC
Command line tail:
Content medium: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
C:\LAB2.COM
Button:
The reason for the exit: normal termination
C:\>_
```

Рисунок 2 – Результат работы программы с вводом Ctrl+C

4. Программы были перемещены в другой каталог. После были повторены шаги 2 и 3. Результат работы представлен на рисунке 3.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\NEWFOL~1>lab6
Memory was freed
Segment address unavailable memory: 9FFF
Segment address medium: 04AC
Command line tail:
Content medium: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
C:\NEWFOL~1\LAB2.COM
Button: A
The reason for the exit: normal termination
C:\NEWFOL~1>lab6
Memory was freed
Segment address unavailable memory: 9FFF
Segment address medium: 04AC
Command line tail:
Content medium: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
C:\NEWFOL~1\LAB2.COM
Button:
The reason for the exit: normal termination
C:\NEWFOL~1>_
```

Рисунок 3 – Результат работы программы в новом каталоге

5. Был запущен программный модуль в каталоге, без вызываемой программ ЛР 2. Результат запуска представлен на рисунке 4.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

C:\>lab6

Memory was freed
Program was not loaded
File not found
C:\>
```

Рисунок 4 – Запуск программы в каталоге без вызываемой программой

Контрольные вопросы.

1) Как реализовано прерывание Ctrl+C?

Функции ввода/вывода в своём большинстве проверяют код 03h в кольцевом буфере клавиатуры и при его обнаружении вызывают прерывание **Int 23h**. Системный обработчик этого прерывания завершает текущую программу вызовом известной нам функции **DOS 4Ch**.

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

В точке вызова функции 4Ch прерывания int 21h

3) В какой точке заканчивается вызываемая программа по прерыванию Ctrl+C?

В точке вызова функции 01h прерывания int 21h

Выводы.

В ходе лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ

```
DATA SEGMENT

PSP_SEG dw 0
POS_OF_LINE dw 0

PARAMETR_BLOCK dw 0 ;сегментный адрес среды
                  dd 0 ;сегмент и смещение командной строки
                  dd 0 ;сегмент и смещение первого FCB
                  dd 0 ;сегмент и смещение второго FCB

LR2 db "LAB2.COM",0
COMMAND_LINE db 1h, 0Dh
PROG_PATH db 50h dup(0)
ERROR_FREE_MEM db 13, 10, "Memory was not freed$", 13, 10
ERRIR_FREE_MEM_7 db 13, 10, "The control memory block was
destroyed$"
ERRIR_FREE_MEM_8 db 13, 10, "Not enough memory to execute$"
ERRIR_FREE_MEM_9 db 13, 10, "Invalid address of the memory
block$"
SUCCES_FREED db 13, 10, "Memory was freed$"

ERROR_LOAD_PROG db 13, 10, "Program was not loaded$"
SUCCES_LOAD_PROG db 13, 10, "Program was loaded$"

ERROR_LOAD_PROG_1 db 13, 10, "The function number is
invalid$"
ERROR_LOAD_PROG_2 db 13, 10, "File not found$"
ERROR_LOAD_PROG_5 db 13, 10, "Disk error$"
ERROR_LOAD_PROG_8 db 13, 10, "Insufficient memory$"
ERROR_LOAD_PROG_10 db 13, 10, "Invalid environment string $"
ERROR_LOAD_PROG_11 db 13, 10, "Incorrect format$"
STR_PROGRAM_END db 13, 10, "The reason for the exit: $"

EXIT_CODE_0 db "normal termination$"
EXIT_CODE_1 db "completion by Ctrl+Break$"
EXIT_CODE_2 db "completion of the device error$"
EXIT_CODE_3 db "completion by function 31h, which leaves the
program resident$"
FREE_FLAGG db 0
```

```

    STR_BUTTON db 13, 10, "Button: $"
    END_DATA db 0
DATA ENDS

```

```

LAB6STACK SEGMENT STACK
    dw 100h dup(0)
LAB6STACK ENDS

```

```

CODE SEGMENT
SAVE_SS dw 0
SAVE_SP dw 0

```

```

    ASSUME CS:CODE, DS:DATA, SS:LAB6STACK

```

```

;-----
MEM_FREE PROC NEAR
    push ax
    push bx
    push cx
    push dx

    mov bx, offset END_PROG
    mov ax, offset END_DATA
    add bx, ax
    add bx, 30Fh

    mov cl, 4
    shr bx, cl
    mov ax, 4A00h
    int 21h

    jnc FREE_MEM_OK
    mov FREE_FLAGG, 0
    mov dx, offset ERROR_FREE_MEM
    call PRINT

    cmp ax, 7
    je ERR_7
    cmp ax, 8
    je ERR_8
    cmp ax, 9
    je ERR_9

```

```

ERR_7:
    mov dx, offset ERRIR_FREE_MEM_7
    jmp END_FREE
ERR_8:
    mov dx, offset ERRIR_FREE_MEM_8
    jmp END_FREE
ERR_9:
    mov dx, offset ERRIR_FREE_MEM_9
    jmp END_FREE
FREE_MEM_OK:
    mov FREE_FLAGG, 1
    mov dx, offset SUCCES_FREED

END_FREE:
    call PRINT
    pop dx
    pop cx
    pop bx
    pop ax
    ret
MEM_FREE ENDP
;-----

SET_PROG PROC NEAR
    push ax
    push si
    push di
    push es

    mov ax, PSP_SEG
    mov es, ax
    mov es, es:[2Ch]
    mov si, 0

FIND00:
    mov ax, es:[si]
    inc si
    cmp ax, 0
    jne FIND00
    add si, 3
    mov di, 0
WRITE:
    mov al, es:[si]

```

```

    cmp al, 0
    je WRITE_PROG
    cmp al, '\'
    jne ADD_SYM
    mov POS_OF_LINE, di
ADD_SYM:
    mov BYTE PTR [PROG_PATH + di], AL
    inc si
    inc di
    jmp WRITE

WRITE_PROG:
    cld
    mov di, POS_OF_LINE
    inc di
    add di, offset PROG_PATH
    mov si, offset LR2
    mov ax, ds
    mov es, ax

REWRITE_NAME_SYMB:
    lodsb
    stosb
    cmp AL, 0
    jne REWRITE_NAME_SYMB

    pop es
    pop di
    pop si
    pop ax
    ret
SET_PROG ENDP
;-----
LOAD_PROG PROC NEAR
    push ax
    push bx
    push dx
    push ds
    push es

    mov SAVE_SP, sp
    mov SAVE_SS, ss
    mov ax, DATA

```



```

mov es, ax
mov bx, offset PARAMETR_BLOCK
mov dx, offset COMMAND_LINE
mov [bx + 2], dx
mov [bx + 4], ds
mov dx, offset PROG_PATH
mov ax, 4B00h
int 21h
mov ss, CS:SAVE_SS
mov sp, CS:SAVE_SP
pop ES
pop ds

jnc LOAD_SUCCESS
mov dx, offset ERROR_LOAD_PROG
call PRINT
cmp ax, 1
je L_ERR_1
cmp ax, 2
je L_ERR_2
cmp ax, 5
je L_ERR_5
cmp ax, 8
je L_ERR_8
cmp ax, 10
je L_ERR_10
cmp ax, 11
je L_ERR_11
L_ERR_1:
mov dx, offset ERROR_LOAD_PROG_1
call PRINT
jmp LOAD_END
L_ERR_2:
mov dx, offset ERROR_LOAD_PROG_2
call PRINT
jmp LOAD_END
L_ERR_5:
mov dx, offset ERROR_LOAD_PROG_5
call PRINT
jmp LOAD_END
L_ERR_8:
mov dx, offset ERROR_LOAD_PROG_8
call PRINT

```

```

    jmp LOAD_END
L_ERR_10:
    mov dx, offset ERROR_LOAD_PROG_10
    call PRINT
    jmp LOAD_END
L_ERR_11:
    mov dx, offset ERROR_LOAD_PROG_11
    call PRINT
    jmp LOAD_END
LOAD_SUCCESS:
    mov ax, 4D00h
    int 21h
    mov di, offset STR_BUTTON
    mov [di + 10], AL
    mov dx, offset STR_BUTTON
    call PRINT

    mov dx, offset STR_PROGRAM_END
    call PRINT
    cmp AH, 0
    je END0
    cmp AH, 1
    je END1
    cmp AH, 2
    je END2
    cmp AH, 3
    je END3
END0:
    mov dx, offset EXIT_CODE_0
    call PRINT
    jmp LOAD_END
END1:
    mov dx, offset EXIT_CODE_1
    call PRINT
    jmp LOAD_END
END2:
    mov dx, offset EXIT_CODE_2
    call PRINT
    jmp LOAD_END
END3:
    mov dx, offset EXIT_CODE_3
    call PRINT

```

```

LOAD_END:
    pop dx
    pop bx
    pop ax
    ret
LOAD_PROG ENDP
;-----

PRINT PROC near

    push ax
    sub ax, ax
    mov ah, 9h
    int 21h
    pop ax

    ret
PRINT ENDP
;-----

MAIN:

    mov bx, ds
    mov ax, DATA
    mov ds, ax
    mov PSP_SEG, bx

    call MEM_FREE

    cmp FREE_FLAGG, 1
    jne END_MAIN

    call SET_PROG
    call LOAD_PROG

END_MAIN:
    xor ax, ax
    mov ah, 4Ch
    int 21h
END_PROG:
CODE ENDS
END MAIN

```