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»

Студент гр. 7381 Минуллин М.А.

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 лабораторной работе №2. Понятно, что такое приложение должно запускаться в соответствии с со стандартами ОС.

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

Необходимые сведения для составления программы.

Для загрузки и выполнения

Ход работы.

Был написан программный .EXE модуль, выполняющий следующие функции:

1. Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором он находится сам. Вызываемому модулю передаётся 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.

Был выполнен запуск отлаженной программы, когда текущим каталогом является каталог с разработанными модулями. Для проверки работы программы вводился случайный символ от A до Z.

Был выполнен запуск отлаженной программы, когда текущим каталогом является какой-либо другой каталог, отличный от того, в котором содержатся разработанные ранее программные модули (см. рис. 1).

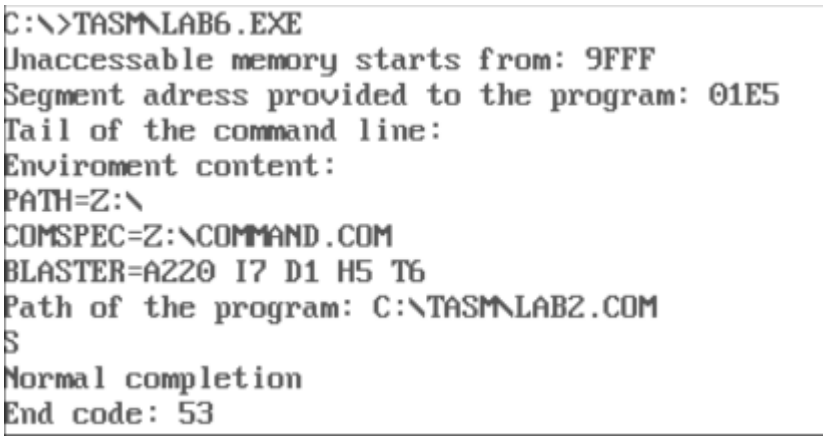

Рисунок 1 – Программные модули находятся в одной директории.

Был проведён запуск отлаженной программы, когда программные модули находятся в разных каталогах. В результате выполнения на экран было выведено соответствующее сообщение об ошибке (см. рис. 2).

Рисунок 2 – Программные модули находятся в разных директориях.

Контрольные вопросы.

В: Как реализовано прерывание Ctrl-C?

О: При нажатии комбинации клавиш Ctrl+C вызывается прерывание 23h. Адрес в этом векторе 0000:008Ch, по которому передаётся управление. Обычная системная обработка Ctrl+C сводится к немедленному снятию программы.

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

О: Вызываемая программа заканчивается при выполнении функции 4Ch прерывания 21h.

В: В какой точке заканчивается вызываемая программа по прерыванию Ctrc+C?

О: Вызываемая программа заканчивается в месте ожидания нажатия клавиши на функции 01h прерывания 21h.

Выводы.

В ходе выполнения лабораторной работы был модифицирован ранее построенный программный модуль лабораторной работы №2, а также построен загрузочный модуль динамической структуры.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ ТЕКСТ .EXE МОДУЛЯ

```
DATA segment
    parameter_block dw ? ;сегментный адрес среды
                    dd ? ;сегмент и смещение командной строки
                    dd ? ;сегмент и смещение первого FCB
                    dd ? ;сегмент и смещение второго FCB

    error1_7        db 'Memory control block destroyed', 10, 13, '$'
    error1_8        db 'Not enough memory to perform the function',
10, 13, '$'
    error1_9        db 'Wrong memory address', 10, 13, '$'

    error2_1        db 'Number of function is incorrect', 10, 13, '$'
    error2_2        db 'File not found', 10, 13, '$'
    error2_5        db 'Disk error', 10, 13, '$'
    error2_8        db 'Insufficient memory', 10, 13, '$'
    error2_10       db 'Incorrect environment string', 10, 13, '$'
    error2_11       db 'Wrong format', 10, 13, '$'

    end0            db 'Normal completion', 10, 13, '$'
    end1            db 'Completion by Ctrl-Break', 10, 13, '$'
    end2            db 'Completion by device error', 10, 13, '$'
    end3            db 'Completion by function 31h', 10, 13, '$'

    endl            db ' ', 10, 13, '$'

    output_code     db 'End code: $'

    path            db 20h dup (0)

    keep_ss         dw 0
    keep_sp         dw 0
DATA ends

ASTACK segment STACK
    dw 100 dup (?)
ASTACK ends

CODE segment
    assume CS:CODE, DS:DATA, ES:DATA, SS:ASTACK

TETR_TO_HEX     proc    near
    and     al, 0Fh
    cmp     al, 09
    jbe     NEXT
    add     al, 07
```

```
NEXT:
    add     al, 30h
    ret
TETR_TO_HEX     endp

BYTE_TO_HEX     proc    near
    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

PRINT   proc    near
    push    ax
    mov     ah, 09h
    int     21h
    pop     ax
    ret
PRINT   endp

ERROR_PROCESSING    proc    near
    cmp     ax, 7
    mov     dx, offset error1_7
    je      write_massage
    cmp     ax, 8
    mov     dx, offset error1_8
    je      write_massage
    cmp     ax, 9
    mov     dx, offset error1_9
    je      write_massage

write_massage:
    call     PRINT
    ret
ERROR_PROCESSING    endp

CLEAR_MEMORY    proc    near
    mov     ax, ASTACK
    mov     bx, es
    sub     ax, bx
    add     ax, 10h
    mov     bx, ax
    mov     ah, 4Ah
    int     21h
    jnc     end_clear
```

```
    call    ERROR_PROCESSING

end_clear:
    ret
CLEAR_MEMORY    endp

CREATION_PARAMETER_BLOCK    proc    near
    mov     ax, es:[2Ch]
    mov     parameter_block, ax
    mov     parameter_block + 2, es
    mov     parameter_block + 4, 80h
    ret
CREATION_PARAMETER_BLOCK endp

ERR_PROCESSING  proc    near
    cmp     ax, 1
    mov     dx, offset error2_1
    je      write_message2
    cmp     ax, 2
    mov     dx, offset error2_2
    je      write_message2
    cmp     ax, 5
    mov     dx, offset error2_5
    je      write_message2
    cmp     ax, 8
    mov     dx, offset error2_8
    je      write_message2
    cmp     ax, 10
    mov     dx, offset error2_10
    je      write_message2
    cmp     ax, 11
    mov     dx, offset error2_11

write_message2:
    call    PRINT
    ret
ERR_PROCESSING  endp

COMPLETION_PROCESSING   proc    near
    mov     dx, offset endl
    call    PRINT
    cmp     ah, 0
    je      normal
    cmp     ah, 1
    mov     dx, offset end1
    je      write_message3
    cmp     ah, 2
    mov     dx, offset end2
    je      write_message3
    cmp     ah, 3
    mov     dx, offset end3
```

```
normal:
    mov     dx, offset  end0
    call    PRINT
    mov     dx, offset output_code
    call    PRINT
    call    BYTE_TO_HEX
    push    ax
    mov     ah, 02h
    mov     dl, al
    int     21h
    pop     ax
    xchg    ah, al
    mov     ah, 02h
    mov     dl, al
    int     21h
    jmp     exit
write_message3:
    call    PRINT
exit:
    ret
COMPLETION_PROCESSING   endp

BASE_PROCESS    proc    near
    mov     es, es:[2ch]
    mov     si, 0

m1:
    mov     dl, es:[si]
    cmp     dl, 0
    je      m2
    inc     si
    jmp     m1

m2:
    inc     si
    mov     dl, es:[si]
    cmp     dl, 0
    jne     m1
    add     si, 3
    lea     di, path

m3:
    mov     dl, es:[si]
    cmp     dl, 0
    je      m4
    mov     [di], dl
    inc     di
    inc     si
    jmp     m3

m4:
```

```
    sub     di, 8

    mov     [di], byte ptr 'L'
    mov     [di + 1], byte ptr 'A'
    mov     [di + 2], byte ptr 'B'
    mov     [di + 3], byte ptr '2'
    mov     [di + 4], byte ptr '.'
    mov     [di + 5], byte ptr 'C'
    mov     [di + 6], byte ptr 'O'
    mov     [di + 7], byte ptr 'M'
    mov     dx, offset path

    push    ds
    pop     es
    mov     bx, offset parameter_block

    mov     keep_sp, SP
    mov     keep_ss, SS

    mov     ax, 4b00h
    int     21h
    jnc     success

    push    ax
    mov     ax, DATA
    mov     ds, ax
    pop     ax
    mov     ss, keep_ss
    mov     sp, keep_sp

error:
    call    ERR_PROCESSING
    ret

success:
    mov     ax, 4d00h
    int     21h

    call    COMPLETION_PROCESSING
    ret
BASE_PROCESS endp

MAIN proc far
    mov     ax, data
    mov     ds, ax

    call    CLEAR_MEMORY
    call    CREATION_PARAMETER_BLOCK
    call    BASE_PROCESS

    xor     al, al
```

```
    mov     ah, 4Ch
    int     21h
MAIN endp

CODE ends

end MAIN
```