

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. 9382

Сорочина М.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры.

Ход работы.

Для выполнения лабораторной работы был написан и отлажен программный модуль типа .exe, который выполняет функции:

- 1) подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором сам находится. 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. Проверяется причина завершения и в зависимости от значения выводится соответствующее сообщение.

В качестве вызываемой программы была использована программа, реализованная во время выполнения второй лабораторной работы. Программа была немного изменена, а именно: перед выходом добавлено обращение к функции ввода символа с клавиатуры.

На рис.1 представлен вывод программы b.exe с введенным символом “w”, когда оба модуля находились в текущем каталоге.

```
C:\>6.EXE
Segment address of memory:9FFF
Segment address of environment:1179
Command line tail:
No tail
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Loadable module path:
C:\Z.COM
Normal termination
End code: 77
C:\>
```

Рис. 1.

На рис.2 представлен вывод программы 6.exe с введенной комбинацией Ctrl+C.

```
C:\>6.EXE
Segment address of memory:9FFF
Segment address of environment:1179
Command line tail:
No tail
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Loadable module path:
C:\Z.COM
Normal termination
End code: 03
C:\>
```

Рис. 2.

На рис. 3 представлен вывод программы при запуске из другой директории и вводе символа “@”.

```

C:\LAB6>cd ../
C:\>LAB6\6.EXE
Segment address of memory:9FFF
Segment address of environment:1179
Command line tail:
No tail
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Loadable module path:
C:\LAB6\Z.COM
Normal termination
End code: 40
C:\>

```

Рис. 3.

На рис. 4 представлен вывод программы в случае, когда модули находятся в разных директориях.

```

C:\>LAB6\6.EXE
File not found
C:\>

```

Рис. 4.

По рисунку видно, что файл не был найден и, соответственно, не был запущен.

Ответы на контрольные вопросы.

1. Как реализовано прерывание прерывание Ctrl-C?

Вызывается прерывание 23H, которое завершает текущий процесс и передает управление порождаемому процессу.

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 выполнения данной работы была изучена возможность построения загрузочного модуля динамической структуры, а также реализован интерфейс вызывающего и вызываемого модуля, где в качестве последнего была использована программа, написанная при выполнении второй лабораторной, с небольшими изменениями.

ПРИЛОЖЕНИЕ А.

Исходный код программы.

```

AStack SEGMENT STACK
        DW 128 DUP(0)
AStack ENDS

DATA SEGMENT
        param          dw      ?
        dd      ?
        dd      ?
        dd      ?
        memErr7        db      0DH,0AH,'MCB destroyed ',0DH,0AH,'$'
        memErr8        db      0DH,0AH,'Not enough memory
        ',0DH,0AH,'$'
        memErr9        db      0DH,0AH,'Wrong adress',0DH,0AH,'$'
        loadErr1        db      0DH,0AH,'Number of function is
        wrong',0DH,0AH,'$'
        loadErr2        db      0DH,0AH,'File not
        found',0DH,0AH,'$'
        loadErr5        db      0DH,0AH,'Disk error',0DH,0AH,'$'
        loadErr8        db      0DH,0AH,'Insufficient value of
        memory',0DH,0AH,'$'
        loadErr10       db      0DH,0AH,'Incorrect enviroment
        string',0DH,0AH,'$'
        loadErr11       db      0DH,0AH,'Wrong format',0DH,0AH,'$'
        norm           db      0DH,0AH,'Normal
        termination',0DH,0AH,'$'
        ctrl           db      0DH,0AH,'Ended by
        ctrl-break',0DH,0AH,'$'
        deviceErr       db      0DH,0AH,'Ended with device
        error',0DH,0AH,'$'
        end31h         db      0DH,0AH,'Ended by
        31h',0DH,0AH,'$'
        path           db      '
        ',0DH,0AH,'$'
        keep_ss         dw      0
        keep_sp         dw      0
        endCode         db      'End code:      ',0DH,0AH,'$'

DATA ENDS

seg1 segment
seg1 ends

CODE SEGMENT
        ASSUME CS:CODE, DS:DATA, SS:AStack
; Процедуры
;-----

```

```

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
;байт в AL переводится в два символа шест. числа в AX
    push CX
    mov AH, AL
    call TETR_TO_HEX
    xchg AL, AH
    mov CL, 4
    shr AL, CL
    call TETR_TO_HEX ;в AL старшая цифра
    pop CX           ;в AH младшая
    ret
BYTE_TO_HEX ENDP
;-----
PRINT proc near
push ax
    mov ah, 09h
    int 21h
pop
    ax
    ret
PRINT endp
;-----
FREE PROC
    mov bx, offset seg1
    mov ax, es
    sub bx, ax
    mov cl, 4h
    shr bx, cl
    mov ah, 4ah
    int 21h
    jnc good
    cmp ax, 7
    mov dx, offset memErr7
    je err
    cmp ax, 8
    mov dx, offset memErr8
    je err
    cmp ax, 9
    mov dx, offset memErr9

err:
    call print
    xor al, al
    mov ah, 4ch

```

```

                int            21h

good:
    mov     ax, es
    mov     param, 0
    mov     param+2, ax
    mov     param+4, 80h
    mov     param+6, ax
    mov     param+8, 5ch
    mov     param+10, ax
    mov     param+12, 6ch
    ret

FREE  ENDP
;-----
RUN_P PROC      NEAR
    mov     es, es:[2ch]
    mov     si, 0

env:
    mov     dl, es:[si]
    cmp     dl, 00h
    je      eol
    inc     SI
    jmp     env

eol:
    inc     si
    mov     dl, es:[si]
    cmp     dl, 00h
    jne     env
    add     si, 03H
    push    di
    lea     di, path

g_path:
    mov     dl, es:[si]
    cmp     dl, 00h
    je      eol2
    mov     [di], dl
    inc     di
    inc     si
    jmp     g_path

eol2:
    sub     di, 5
    mov     [di], byte ptr '2'
    mov     [di+1], byte ptr '.'
    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

```



```

mov    keep_sp, sp
mov    keep_ss, ss
push   ds
pop     es
mov    bx, offset param
mov    dx, offset path
mov    ax, 4b00h
int     21h
jnc     isloaded
push   ax
mov    ax, data
mov    ds, ax
pop     ax
mov    sp, keep_sp
mov    ss, keep_ss

cmp    ax, 1
mov    dx, offset loadErr1
je      endError
cmp    ax, 2
mov    dx, offset loadErr2
je      endError
cmp    ax, 5
mov    dx, offset loadErr5
je      endError
cmp    ax, 8
mov    dx, offset loadErr8
je      endError
cmp    ax, 10
mov    dx, offset loadErr10
je      endError
cmp    ax, 11
mov    dx, offset loadErr11
je      endError

endError:
call    print
xor     al, al
mov     ah, 4ch
int     21h

isloaded:
mov     ax, 4d00h
int     21h
cmp     ah, 0
mov     dx, offset norm
je      endrun
cmp     ah, 1
mov     dx, offset ctrl
je      endrun
cmp     ah, 2
mov     dx, offset deviceErr
je      endrun

```

```

        cmp     ah, 3
        mov     dx, offset end31h

endrun:
        call    print
        mov     di, offset endCode
        call    byte_to_hex
        add     di, 0ah
        mov     [di], al
        add     di, 1
        xchg    ah, al
        mov     [di], al
        mov     dx, offset endCode
        call    print
        ret

RUN_P ENDP
;-----
; Код
MAIN PROC
        push    DS           ;\ Сохранение адреса начала PSP в стеке
        sub     AX,AX        ; > для последующего восстановления по
        push    AX           ;/ команде ret, завершающей процедуру.
        mov     AX,DATA      ; Загрузка сегментного
        mov     DS,AX        ; регистра данных.

        call    free
        call    run_p

exit:
; Выход в DOS
        xor     AL,AL
        mov     AH,4Ch
        int     21H

MAIN ENDP
CODE ENDS

        END MAIN

```