

**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. 8382

\_\_\_\_\_

Кузина А.М.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

## Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. Исследование интерфейса между вызывающим и вызываемым модулями по управлению и по данным.

## Ход выполнения работы.

Была написана программа, исходный код которой приведет в приложении А, выполняющая следующие функции:

- Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится она сама. Вызываемому модулю передаются новые среда и командная строка.
- Вызываемый модуль запускается с использованием загрузчика.
- После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Проверяется причина завершения и, в зависимости от значения, на экран выводится соответствующее сообщение.

В качестве вызываемой программы была взята программа из второй лабораторной работы, которая распечатывает среду и командную строку. Также она была модифицирована таким образом, что в конце работы запрашивает ввод символа с клавиатуры. На рисунке 1 представлен результат работы программы, при нахождении обоих модулей в одной директории и введении символа е.

Рисунок 1 — Результат работы программы при введении е.



```
C:\>16
Memory successfully freed
Launching program: C:\12.COM
  Unavailable memory address: 9FFF
  Environment address: 0200
  Command line tail:
  Enviroment contents:
    PATH=Z:\
    COMSPEC=Z:\COMMAND.COM
    BLASTER=A220 I7 D1 H5 T6
  Load module path: C:\12.COM
e
Normal exit with code: e
```

На рисунке 2 представлен результат работы программы, при нахождении обоих модулей в одной директории и введении комбинации Ctrl-C.

Рисунок 2 — Результат работы программы при введении Ctrl-C.

```
C:\>I6
Memory successfully freed
Launching program: C:\I2.COM
  Unavailable memory address: 9FFF
  Environment address: 0200
  Command line tail:
  Enviroment contents:
    PATH=Z:\
    COMSPEC=Z:\COMMAND.COM
    BLASTER=A220 I7 D1 H5 T6
  Load module path: C:\I2.COM
♥
Normal exit with code: ♥
```

На рисунке 3 представлен результат работы программы, при нахождении обоих модулей в одной директории, не являющейся текущей, и введении символа e.

Рисунок 3 — Результат работы программы при расположении модулей не в текущем каталоге и введении символа e.

```
C:\>test\I6
Memory successfully freed
Launching program: C:\TEST\I2.COM
  Unavailable memory address: 9FFF
  Environment address: 0200
  Command line tail:
  Enviroment contents:
    PATH=Z:\
    COMSPEC=Z:\COMMAND.COM
    BLASTER=A220 I7 D1 H5 T6
  Load module path: C:\TEST\I2.COM
e
Normal exit with code: e
```

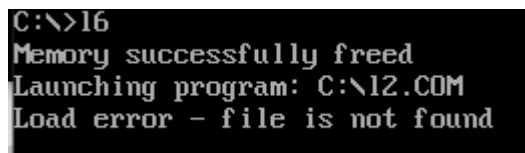
На рисунке 4 представлен результат работы программы, при нахождении обоих модулей в одной директории, не являющейся текущей, и введении комбинации Ctrl-C.

Рисунок 4 — Результат работы программы при расположении модулей не в текущем каталоге и введении комбинации Ctrl-C .

```
C:\>test\I6
Memory successfully freed
Launching program: C:\TEST\I2.COM
  Unavailable memory address: 9FFF
  Environment address: 0200
  Command line tail:
  Enviroment contents:
    PATH=Z:\
    COMSPEC=Z:\COMMAND.COM
    BLASTER=A220 I7 D1 H5 T6
  Load module path: C:\TEST\I2.COM
♥
Normal exit with code: ♥
```

На рисунке 5 представлен результат работы программы при нахождении модулей в разных директориях.

Рисунок 5 — Результат работы программы при отсутствии вызываемого модуля в соответствующей директории.



```
C:\>I6
Memory successfully freed
Launching program: C:\I2.COM
Load error - file is not found
```

### Контрольные вопросы

- Как реализовано прерывание Ctrl-C?

При распознавании нажатия сочетания клавиш Ctrl-C выполняется прерывание int 23h. Обычная системная обработка Ctrl-Break вызывает немедленное завершение работы программы.

- В какой точке завершается вызываемая программа, если код причины завершения 0?

Если код завершения равен 0, значит программа завершилась нормально, вызовом функции 4Ch прерывания int 21h.

- В какой точке завершается вызываемая программа по прерыванию Ctrl-C?

При считывании комбинации Ctrl-C функцией 01h прерывания int 21h, управление передается прерыванию int 23h, которое заканчивает работу программы.

### Выводы

В ходе лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры. Был исследован интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Была написана и протестирована в различных условиях программа, вызывающая другую программу и анализирующая результаты ее работы.

## ПРИЛОЖЕНИЕ А

### Исходный код программы l6.asm

```
ASTACK SEGMENT STACK
    dw 100 DUP(0)
ASTACK ENDS

DATA SEGMENT
MemN db 'Memory successfully freed', 13, 10, '$'
MemErr7 db 'Memory error - memory block was destroyed', 13, 10, '$'
MemErr8 db 'Memory error - not enough memory for function', 13, 10, '$'
MemErr9 db 'Memory error - incorrect memory block address', 13, 10, '$'
LoadErr1 db 'Load error - incorrect function number', 13, 10, '$'
LoadErr2 db 'Load error - file is not found', 13, 10, '$'
LoadErr5 db 'Load error - disc error', 13, 10, '$'
LoadErr8 db 'Load error - not enough memory', 13, 10, '$'
LoadErr10 db 'Load error - incorrect env string', 13, 10, '$'
LoadErr11 db 'Load error - incorrect format', 13, 10, '$'
ExitReas0 db 13, 10, 'Normal exit ', '$'
ExitReas1 db 13, 10, 'Ctrl-break exit ', 13, 10, '$'
ExitReas2 db 13, 10, 'Device error exit ', 13, 10, '$'
ExitReas3 db 13, 10, 'Exit with int 31h - resident ', 13, 10, '$'
ExitCode db 'with code: ', 13, 10, '$'

ParameterBlock db 0
                dd 0
                dd 0
                dd 0

Path db 128 DUP(0)
SPath db 'Launching program: ', 13, 10, '$'
KeepSS dw 0
KeepSP dw 0
KeepDS dw 0
DataEnd dw 0

DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:NOTHING, SS:ASTACK

Preparation PROC

FreeMem:
    push ax
    push bx

    ;определяем сколько памяти нужно оставить нашей программе
    push dx
    mov dx, offset DataEnd
    mov bx, offset ProgEnd
    add bx, dx
    push cx
    mov cl, 4
    shr bx, cl ;переводим в параграфы
    add bx, 28h
    pop cx
    mov ah, 4Ah ;всю остальную освобождаем
    int 21h
    jnc MRET
```

```

cmp ax, 7
    je MERR7
cmp ax, 8
    je MERR8
cmp ax, 9
    je MERR9

```

MERR7:

```

mov dx, offset MemErr7
mov ah, 09h
int 21h
mov ah, 4Ch
int 21h

```

MERR8:

```

mov dx, offset MemErr8
mov ah, 09h
int 21h
mov ah, 4Ch
int 21h

```

MERR9:

```

mov dx, offset MemErr9
mov ah, 09h
int 21h
mov ah, 4Ch
int 21h

```

MRET:

```

mov dx, offset MEMN
mov ah, 09h
int 21h
pop dx
pop bx
pop ax

```

ParametrBlock:

```

;настраиваем блок параметров
push ax
push bx
push cx

mov bx, offset ParameterBlock
mov cx, es

mov ax, 0
mov [bx], ax ;сегментный адрес среды
;сегментный адрес среды 0 -> вызываемая программа наследует среду данной
mov [bx+2], cx
mov ax, 80h
mov [bx+4], ax ;сегмент и смещение командной строки

mov [bx+6], cx
mov ax, 5Ch
mov [bx+8], ax; сегмент и смещение первого FCB

mov [bx+10], cx
mov ax, 6Ch
mov [bx+12], ax; сегмент и смещение второго FCB

pop cx
pop bx
pop ax

```

```

        push dx
        mov si, offset SPath
        add si, 19
PrepPath:
        mov es, es:[2Ch]
        mov bx, 0

EnvLoop:
        mov dl, es:[bx]
        cmp dl, 0
        je      EnvEnd
        inc bx
        jmp EnvLoop

EnvEnd:
        inc bx
        mov dl, es:[bx]
        cmp dl, 0
        jne EnvLoop
        add bx, 3

        mov di, offset Path
Ploop:
        mov dl, es:[bx]
        cmp dl, 0
        je      PEdn
        mov [di], dl
        mov [si], dl
        inc si
        inc di
        inc bx
        jmp Ploop

PEdn:
        sub di, 6
        sub si, 6
        mov [di], byte ptr 'l'
        mov [di+1], byte ptr '2'
        mov [di+2], byte ptr '.'
        mov [di+3], byte ptr 'C'
        mov [di+4], byte ptr 'O'
        mov [di+5], byte ptr 'M'
        mov [di+6], byte ptr 0
        mov [si], byte ptr 'l'
        mov [si+1], byte ptr '2'
        mov [si+2], byte ptr '.'
        mov [si+3], byte ptr 'C'
        mov [si+4], byte ptr 'O'
        mov [si+5], byte ptr 'M'
        mov [si+6], byte ptr 0

        push ax
        mov dx, offset SPath
        mov ah, 09h
        int 21h
        pop ax
        pop dx
        ;берем путь нашей программы, заменяем ее имя на имя нужного исполняемого файла
        ret
Preparation ENDP

```

```

CallIt PROC
    mov KeepDS, ds
    mov KeepSS, ss
    mov KeepSP, sp
    mov ax, ds
    mov es, ax
    mov bx, offset ParameterBlock
    mov dx, offset Path

    mov ax, 4B00h
    int 21h
    mov ds, KeepDS
    mov ss, KeepSS
    mov sp, KeepSP

    jnc LOK ;CF = 0

    cmp ax, 1
        je LERR1
    cmp ax, 2
        je LERR2
    cmp ax, 5
        je LERR5
    cmp ax, 8
        je LERR8
    cmp ax, 10
        je LERR10
    cmp ax, 11
        je LERR11
LERR1:
    mov dx, offset LoadErr1
    mov ah, 09h
    int 21h
    mov ah, 4Ch
    int 21h
LERR2:
    mov dx, offset LoadErr2
    mov ah, 09h
    int 21h
    mov ah, 4Ch
    int 21h
LERR5:
    mov dx, offset LoadErr5
    mov ah, 09h
    int 21h
    mov ah, 4Ch
    int 21h
LERR8:
    mov dx, offset LoadErr8
    mov ah, 09h
    int 21h
    mov ah, 4Ch
    int 21h
LERR10:
    mov dx, offset LoadErr10
    mov ah, 09h
    int 21h
    mov ah, 4Ch
    int 21h
LERR11:
    mov dx, offset LoadErr11
    mov ah, 09h

```



```

        int 21h
        mov ah, 4Ch
        int 21h

LOK:
        mov ah, 4Dh
        int 21h

        cmp ah, 0
            je T0
        cmp ah, 1
            je T1
        cmp ah, 2
            je T2
        cmp ah, 3
            je T3
T0:
        mov dx, offset ExitReas0
        mov ah, 09h
        int 21h
        mov di, offset ExitCode
        add di, 11
        mov [di], al
        mov [di+1], ah
        mov dx, offset ExitCode
        mov ah, 09h
        int 21h
        ret
T1:
        mov dx, offset ExitReas1
        jmp r
T2:
        mov dx, offset ExitReas2
        jmp r
T3:
        mov dx, offset ExitReas3
        jmp r
r:
        mov ah, 09h
        int 21h
        ret

CallIt ENDP

```

```

MAIN PROC
        mov ax, DATA
        mov ds, ax

        call Preparation
        call CallIt

        mov ah, 4Ch
        int 21h

        MAIN ENDP
ProgEnd:
CODE ENDS

END MAIN

```