

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. 8382

Рочева А.К.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

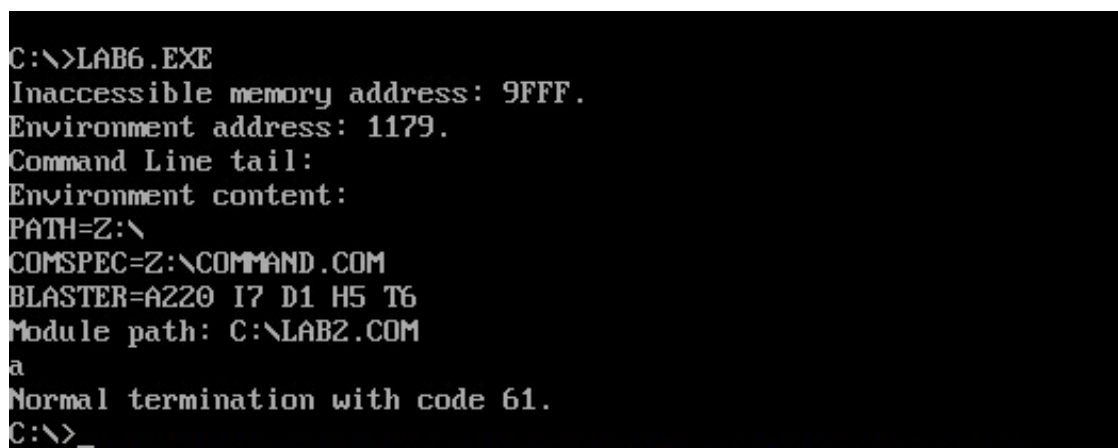
Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры.

Ход выполнения.

Для выполнения работы была написана программа (приложение А), которая вызывает загрузочный модуль lab2.com из того же диалога, в котором она находится. Вызываемому модулю передается новая среда, созданная вызывающим модулем и новая командная строка. Вызываемый модуль запускается с использованием загрузчика. После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы.

На рис. 1 представлена работа программы при вводе символа «а», когда текущим каталогом является каталог с разработанными модулями.



```
C:\>LAB6.EXE
Inaccessible memory address: 9FFF.
Environment address: 1179.
Command Line tail:
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Module path: C:\LAB2.COM
a
Normal termination with code 61.
C:\>_
```

Рис.1 — работа программы в текущем каталоге

На рис. 2 представлена работа программы при вводе Ctrl-C, когда текущим каталогом является каталог с разработанными модулями.

```

C:\>LAB6.EXE
Inaccessible memory address: 9FFF.
Environment address: 1179.
Command Line tail:
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Module path: C:\LAB2.COM
♥
Normal termination with code 03.
C:\>_

```

Рис.2 — работа программы в текущем каталоге

На рис. 3 и 4 представлена работа программы при вводе символа «а» и Ctrl-C, когда текущим каталогом является другой каталог, а модули находятся в каталоге os2.

```

C:\OS2>LAB6.EXE
Inaccessible memory address: 9FFF.
Environment address: 1179.
Command Line tail:
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Module path: C:\OS2\LAB2.COM
a
Normal termination with code 61.
C:\OS2>

```

Рис.3 — работа программы в каталоге /os2

```

C:\OS2>LAB6.EXE
Inaccessible memory address: 9FFF.
Environment address: 1179.
Command Line tail:
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Module path: C:\OS2\LAB2.COM
♥
Normal termination with code 03.
C:\OS2>

```

Рис.4 — работа программы в каталоге /os2

На рис.5 представлена работа программы, когда модули находятся в разных каталогах (в os2 находится модуль lab6.exe).



```
C:\OS2>LAB6.EXE
File not found.
C:\OS2>
```

Рис. 5 — работа программы в каталоге /os2

Ответы на вопросы:

1. Как реализовано прерывание Ctrl-C?

При нажатии комбинации клавиш 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 12h.

3. В какой точке заканчивается программа по прерыванию Ctrl-C?

В точке вызова функции 01h прерывания int12h.

Выводы

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД LAB6.ASM

```

AStack SEGMENT STACK
    dw 100h dup(?)
AStack ENDS

DATA SEGMENT
    MESSAGE_DAMAGE_CONTROL db 'The control block of memory is destroyed.',
0DH,0AH, '$'
    MESSAGE_NO_MEMORY db 'Not enough memory to complete the function.',
0DH,0AH, '$'
    MESSAGE_INVALID_ADDRESS db 'Invalid memory block address.', 0DH,0AH, '$'
    PARAMETER_BLOCK dw ?
                                dd ?
                                dd ?
                                dd ?

    KEEP_SP dw ?
    KEEP_SS dw ?
    MESSAGE_INVALID_NUM_FUNC db 'Function number is invalid.', 0DH,0AH, '$'
    MESSAGE_FILE_NOT_FOUND db 'File not found.', 0DH,0AH, '$'
    MESSAGE_DISK_ERROR db 'Disk error.', 0DH,0AH, '$'
    MESSAGE_NOT_ENOUGH_MEMORY db 'Not enough memory.', 0DH,0AH, '$'
    MESSAGE_INVALID_ENV_STR db 'Invalid environment string.', 0DH,0AH, '$'
    MESSAGE_INVALID_FORMAT db 'Invalid format.', 0DH,0AH, '$'
    MESSAGE_TERM_NORMAL db 0DH,0AH, 'Normal termination $'
    MESSAGE_TERM_CTRLBREAK db 0DH,0AH, 'Ctrl-Break termination $'
    MESSAGE_TERM_DEVICE_ERROR db 0DH,0AH, 'Device error termination $'
    MESSAGE_TERM_FUNCTION db 0DH,0AH, '31h termination $'
    MESSAGE_TERM_CODE db 'with code    .$'
    PATH db 128 dup(0)
DATA ENDS

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

FREE_MEMORY PROC near
    mov bx, offset LAST_SEGMENT
    mov ax, es
    sub bx, ax
    mov cl, 4h
    shr bx, cl
    mov ah, 4Ah
    int 21h
    jc Err
    jmp EndProcNormal

Err:
    cmp ax, 7
    je DamageControl
    cmp ax, 8
    je NoMemory
    cmp ax, 9
    je InvalidAddress
    jmp EndProcError
DamageControl:
    mov dx, offset MESSAGE_DAMAGE_CONTROL
    jmp EndProcError
NoMemory:
    mov dx, offset MESSAGE_NO_MEMORY
    jmp EndProcError
InvalidAddress:
    mov dx, offset MESSAGE_INVALID_ADDRESS
    jmp EndProcError
EndProcError:
    call WRITE
    xor al, al
    mov ah, 4Ch
    int 21h

```

```

EndProcNormal:
    ret
FREE_MEMORY ENDP

CREATE_BLOCK PROC near
    mov bx, offset PARAMETER_BLOCK
    mov ax, es

    mov cx, 0
    mov [bx], cx
    mov [bx+2], ax
    mov cx, 80h
    mov [bx+4], cx
    mov [bx+6], ax
    mov cx, 5Ch
    mov [bx+8], cx
    mov [bx+10], ax
    mov cx, 6Ch
    mov [bx+12], cx

    ret
CREATE_BLOCK ENDP

CREATE_STRING PROC near
    mov es, es:[2Ch]
    mov si, 0

EnvLoop:
    mov dl, es:[si]
    cmp dl, 00h
    je EnvEnd
    inc si
    jmp EnvLoop

EnvEnd:
    inc si
    mov dl, es:[si]
    cmp dl, 00h
    jne EnvLoop
    add si, 03h

    lea di, PATH
PathLoop:
    mov dl, es:[si]
    cmp dl, 00h
    je PathEnd
    mov [di], dl
    inc di
    inc si
    jmp PathLoop

PathEnd:
    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 [di+8], byte ptr 0

    ret
CREATE_STRING ENDP

RUN_PROC PROC near
    push ds
    mov KEEP_SP, sp

```

```

    mov KEEP_SS, ss

    mov ax, ds
    mov es, ax
    mov bx, offset PARAMETER_BLOCK
    mov dx, offset PATH
    mov ax, 4B00h
    int 21h
    pop ds
    mov sp, KEEP_SP
    mov ss, KEEP_SS
    jnc ProgramLoaded

    cmp ax, 1
    je InvalidNumFunction
    cmp ax, 2
    je NoFile
    cmp ax, 5
    je DiskError
    cmp ax, 8
    je NotEnoughMemory
    cmp ax, 10
    je InvalidEnvString
    cmp ax, 11
    je InvalidFormat

InvalidNumFunction:
    mov dx, offset MESSAGE_INVALID_NUM_FUNC
    jmp EndProc
NoFile:
    mov dx, offset MESSAGE_FILE_NOT_FOUND
    jmp EndProc
DiskError:
    mov dx, offset MESSAGE_DISK_ERROR
    jmp EndProc
NotEnoughMemory:
    mov dx, offset MESSAGE_NOT_ENOUGH_MEMORY
    jmp EndProc
InvalidEnvString:
    mov dx, offset MESSAGE_INVALID_ENV_STR
    jmp EndProc
InvalidFormat:
    mov dx, offset MESSAGE_INVALID_FORMAT
    jmp EndProc
EndProc:
    call WRITE
    xor al, al
    mov ah, 4Ch
    int 21h

ProgramLoaded:
    mov ax, 4D00h
    int 21h
    cmp ah, 0
    je Normal
    cmp ah, 1
    je CtrlBreak
    cmp ah, 2
    je DeviceError
    cmp ah, 3
    je Func31h
Normal:
    mov dx, offset MESSAGE_TERM_NORMAL
    jmp PrintCode
CtrlBreak:
    mov dx, offset MESSAGE_TERM_CTRLBREAK
    jmp PrintCode
DeviceError:
    mov dx, offset MESSAGE_TERM_DEVICE_ERROR
    jmp PrintCode
Func31h:
    mov dx, offset MESSAGE_TERM_FUNCTION

```

```

        jmp PrintCode

PrintCode:
    call WRITE
    mov di, offset MESSAGE_TERM_CODE
    call BYTE_TO_HEX
    add di, 10
    mov [di], al
    inc di
    mov [di], ah
    mov dx, offset MESSAGE_TERM_CODE
    call WRITE
    ret
RUN_PROC ENDP

WRITE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP

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

MAIN PROC FAR
    mov ax, DATA
    mov ds, ax

    call FREE_MEMORY
    call CREATE_BLOCK
    call CREATE_STRING
    call RUN_PROC

    xor ax, ax
    mov ah, 4Ch
    int 21h
    ret
MAIN ENDP
CODE ENDS
LAST_SEGMENT SEGMENT
LAST_SEGMENT ENDS
END MAIN

```