

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. 0382

Охотникова Г.С.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет 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 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.

В качестве вызываемой программы необходимо взять программу ЛР 2, которая распечатывает среду и командную строку. Эту программу следует немного модифицировать, вставив перед выходом из нее обращение к функции ввода символа с клавиатуры. Введенное значение записывается в регистр AL и затем происходит обращение к функции выхода 4Ch прерывания int 21h.

Шаг 2. Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры.

Введите произвольный символ из числа A-Z. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

Шаг 3. Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры.

Введите комбинацию символов Ctrl-C. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

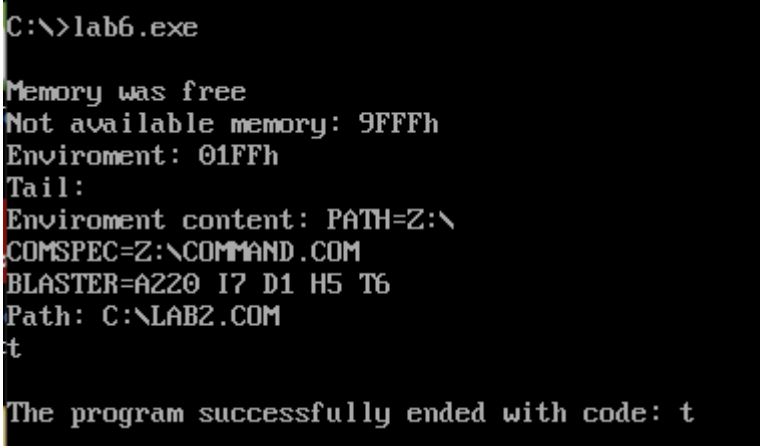
Шаг 4. Запустите отлаженную программу, когда текущим каталогом является какой-либо другой каталог, отличный от того, в котором содержатся разработанные программные модули.

Повторите ввод комбинаций клавиш. Занесите полученные данные в отчет.

Шаг 5. Запустите отлаженную программу, когда модули находятся в разных каталогах. Занесите полученные данные в отчет.

Выполнение работы.

1. При выполнении данной лабораторной работы был написан модуль EXE, который выполняет указанные в задании функции.
2. Был запущен модуль lab6.exe из директории с разработанными модулями с введенным символом t.



```
C:\>lab6.exe

Memory was free
Not available memory: 9FFFh
Enviroment: 01FFh
Tail:
Enviroment content: PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path: C:\LAB2.COM
t

The program successfully ended with code: t
```

Рисунок 1 — Результат работы модуля lab6.exe

3. Был запущен модуль lab6.exe из директории с разработанными модулями с введенной комбинацией CTRL+C. Так как в DOSBOX нет реализации обработки данной комбинации клавиш, то это — символ сердечка и программа работает так же, как в предыдущем пункте.

```

C:\>lab6.exe

Memory was free
Not available memory: 9FFFh
Enviroment: 01FFh
Tail:
Enviroment content: PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path: C:\LAB2.COM
♥

The program successfully ended with code: ♥

```

Рисунок 2 — Результат запуска модуля lab6.com

4. Был запущен модуль lab6.exe из другой директории с нажатием клавиши w.

```

C:\>cd \testing

C:\TESTING>lab6.exe

Memory was free
Not available memory: 9FFFh
Enviroment: 01FFh
Tail:
Enviroment content: PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path: C:\TESTING\LAB2.COM
w

The program successfully ended with code: w

```

Рисунок 3 — Результат работы модуля из другой директории

5. Модуль был запущен из другой директории, при этом загрузочный модуль находился в другом каталоге.

```

C:\TESTING>lab6.exe

Memory was free
Error: File not found.

```

Рисунок 4 — Результат работы из разных каталогов

Исходный программный код см. в приложении А.

Контрольные вопросы.

1. Как реализовано прерывание CTRL+C?

При нажатии данной комбинации клавиш срабатывает прерывание int 23h. Управление передается по адресу 0000:008C. Этот адрес копируется в PSP при помощи функции 26h и 4ch. После выхода из программы адрес восстанавливается

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

При выполнении функции 4ch прерывания int 21h.

3. В какой точке заканчивается вызываемая программа по прерыванию CTRL+C?

В том месте, где произошло прерывание — там, где ожидается нажатие клавиш.

Выводы.

Были исследованы возможности построения загрузочного модуля динамической структуры.

ПРИЛОЖЕНИЕ А.

Название файла: lab6.asm

```
AStack SEGMENT STACK
```

```
        DW 128 DUP(?)
```

```
AStack ENDS
```

```
DATA SEGMENT
```

```
;блок параметров
```

```
P_BLOCK DW 0
```

```
        dd 0
```

```
        dd 0
```

```
        dd 0
```

```
file_name DB 'lab2.com', 0
```

```
file_path db 128 DUP(0)
```

```
flag DB 0
```

```
CMD DB 1h, 0dh
```

```
KEEP_SS DW 0
```

```
KEEP_SP DW 0
```

```
KEEP_PSP DW 0
```

```
STR_MEMORY_FREE DB 'Memory was free' , 0dh, 0ah, '$'
```

```
STR_ERROR_CRASH DB 'Error: MCB crashed.', 0dh, 0ah, '$'
```

```
STR_ERROR_NO_MEMORY DB 'Error: Not enough memory.', 0dh, 0ah,  
'$'
```

```
STR_ERROR_ADDRESS DB 'Error: Invalid memory addressess.', 0dh,  
0ah, '$'
```

```
STR_ERROR_NUMBER DB 'Error: Invalid function number.', 0dh, 0ah,  
'$'
```

```
STR_ERROR_NO_FILE DB 'Error: File not found.', 0dh, 0ah, '$'
```

```
STR_ERROR_DISK DB 'Error with disk.', 0dh, 0ah, '$'
```

```
STR_ERROR_MEMORY DB 'Error: Insufficient memory.', 0dh, 0ah,  
'$'
```

```
STR_ERROR_ENVIROMENT DB 'Error: Wrong string of environment.',  
0dh, 0ah, '$'
```

```
STR_ERROR_FORMAT DB 'Error: Wrong format.', 0dh, 0ah, '$'
```

```

        STR_ERROR_DEVICE DB 0dh, 0ah, 'Error: Device error.' , 0dh, 0ah,
'$'

        STR_END_CODE DB 0dh, 0ah, 'The program successfully ended with
code:  ' , 0dh, 0ah, '$'

        STR_END_CTRL DB 0dh, 0ah, 'The program was interrupted by ctrl-
break' , 0dh, 0ah, '$'

        STR_END_INTER DB 0dh, 0ah, 'The program was ended by
interruption int 31h' , 0dh, 0ah, '$'

        NEW_STR DB 0DH,0AH,'$'

        DATA_END DB 0

DATA ENDS

CODE SEGMENT

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

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

FREE_MEMORY PROC
        push ax
        push bx
        push cx
        push dx
        mov ax, offset DATA_END
        mov bx, offset PR_END
        add bx, ax
        mov cl, 4
        shr bx, cl
        add bx, 2bh
        mov ah, 4ah
        int 21h
        jnc end_proc
        mov flag, 1

```

```

error_crash:
    cmp ax, 7
    jne error_no_memory
    mov dx, offset STR_ERROR_CRASH
    call PRINT
    jmp ret_p

error_no_memory:
    cmp ax, 8
    jne error_address
    mov dx, offset STR_ERROR_NO_MEMORY
    call PRINT
    jmp ret_p

error_address:
    cmp ax, 9
    mov dx, offset STR_ERROR_ADDRESS
    call PRINT
    jmp ret_p

end_proc:
    mov flag, 1
    mov dx, offset NEW_STR
    call PRINT
    mov dx, offset STR_MEMORY_FREE
    call PRINT

ret_p:
    pop dx
    pop cx
    pop bx
    pop ax
    ret

FREE_MEMORY ENDP

LOAD PROC
    push ax
    push bx

```



```

push cx
push dx
push ds
push es
mov KEEP_SP, sp
mov KEEP_SS, ss
mov ax, DATA
mov es, ax
mov bx, offset P_BLOCK
mov dx, offset CMD
mov [bx+2], dx
mov [bx+4], ds
mov dx, offset file_path
mov ax, 4b00h ;вызов загрузчика ОС
int 21h
mov ss, KEEP_SS
mov sp, KEEP_SP
pop es
pop ds
jnc load_okey

cmp ax, 1
jne error_no_file

mov dx, offset STR_ERROR_NUMBER
call PRINT

jmp end_load

error_no_file:
cmp ax, 2
jne error_disk
mov dx, offset STR_ERROR_NO_FILE
call PRINT
jmp end_load

error_disk:
cmp ax, 5
jne error_memory

```

```

        mov dx, offset STR_ERROR_DISK
        call PRINT
        jmp end_load

error_memory:
        cmp ax, 8
        jne error_enviroment
        mov dx, offset STR_ERROR_MEMORY
        call PRINT
        jmp end_load

error_enviroment:
        cmp ax, 10
        jne error_format
        mov dx, offset STR_ERROR_ENVIROMENT
        call PRINT
        jmp end_load

error_format:
        cmp ax, 11
        mov dx, offset STR_ERROR_FORMAT
        call PRINT
        jmp end_load

load_okey:
        mov ah, 4dh
        mov al, 00h
        int 21h
        cmp ah, 0
        jne contrl
        push di
        mov di, offset STR_END_CODE
        mov [di+44], al
        pop si
        mov dx, offset NEW_STR
        call PRINT
        mov dx, offset STR_END_CODE
        call PRINT
        jmp end_load

```

```

contrl:
    cmp ah, 1
    jne error_device
    mov dx, offset STR_END_CTRL
    call PRINT
    jmp end_load

error_device:
    cmp ah, 2
    jne end_interrupt
    mov dx, offset STR_ERROR_DEVICE
    call PRINT
    jmp end_load

end_interrupt:
    cmp ah, 3
    mov dx, offset STR_END_INTER
    call PRINT

end_load:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
LOAD ENDP

PATH PROC
    push ax
    push bx
    push cx
    push dx
    push di
    push si
    push es
    mov ax, KEEP_PSP
    mov es, ax
    mov es, es:[2ch]

```

```

    mov bx, 0

find_path:
    inc bx
    cmp byte ptr es:[bx-1], 0
    jne find_path
    cmp byte ptr es:[bx+1], 0
    jne find_path
    add bx, 2
    mov di, 0

find_loop:
    mov dl, es:[bx]
    mov byte ptr [file_path + di], dl
    inc di
    inc bx
    cmp dl, 0
    je end_find_loop
    cmp dl, '\'
    jne find_loop
    mov cx, di
    jmp find_loop

end_find_loop:
    mov di, cx
    mov si, 0

end_p:
    mov dl, byte ptr [file_name + si]
    mov byte ptr [file_path + di], dl
    inc di
    inc si
    cmp dl, 0
    jne end_p
    pop es
    pop si
    pop di
    pop dx
    pop cx

```

```

        pop bx
        pop ax
        ret
PATH ENDP

MAIN PROC far
    push ds
    xor ax, ax
    push ax
    mov ax, DATA
    mov ds, ax
    mov KEEP_PSP, es
    call FREE_MEMORY
    cmp flag, 0
    je end_main
    call PATH
    call LOAD

end_main:
    xor al, al
    mov ah, 4ch
    int 21h
MAIN ENDP
PR_END:
CODE ENDS
END MAIN

```