

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

Студент гр. 9383

Крейсманн К.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры.

Задание

Шаг1. 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) Вызываемой модуль запускается с использованием загрузчика.

3) После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Необходимо проверить причину завершения и , в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

В качестве вызываемой программы необходимо взять программа 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 полученный результат 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.

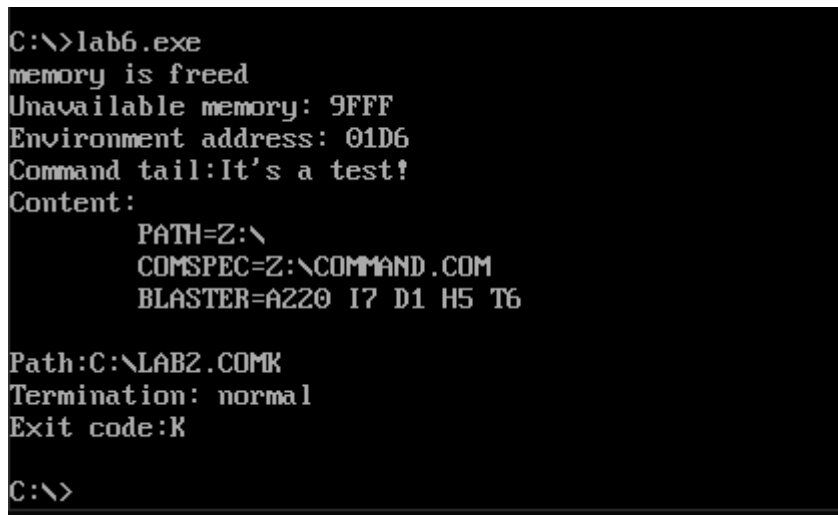
Шаг 4. Запустите отлаженную программу, когда текущим каталогом является какой-либо другой каталог, отличный от того, в котором содержатся разработанные программные модули.

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. Был разработан и отлажен программный модуль типа .exe.
2. Программа запущена из директории с разработанными модулями, был введен символ K. Результат работы программа представлен на рисунке 1.



```
C:\>lab6.exe
memory is freed
Unavailable memory: 9FFF
Environment address: 01D6
Command tail:It's a test!
Content:
    PATH=Z:\
    COMSPEC=Z:\COMMAND.COM
    BLASTER=A220 I7 D1 H5 T6

Path:C:\LAB2.COMK
Termination: normal
Exit code:K

C:\>
```

Рисунок 1

3. Программа запущена и завершена с помощью Ctrl + C. В виду того, что в DOSBOX не реализована обработка данного сочетания, Ctrl+C – это символ сердечка. Результат работы программы представлен на рисунке 2.

```

C:\>lab6.exe
memory is freed
Unavailable memory: 9FFF
Environment address: 01D6
Command tail:It's a test!
Content:
    PATH=Z:\
    COMSPEC=Z:\COMMAND.COM
    BLASTER=A220 I7 D1 H5 T6

Path:C:\LAB2.COM
Termination: normal
Exit code:
C:\>_

```

Рисунок 2

4. Программа запущена из другой директории. Результат представлен на рисунке 3.

```

C:\>lab6\lab6.exe
memory is freed
Unavailable memory: 9FFF
Environment address: 01D6
Command tail:It's a test!
Content:
    PATH=Z:\
    COMSPEC=Z:\COMMAND.COM
    BLASTER=A220 I7 D1 H5 T6

Path:C:\LAB6\LAB2.COM
Termination: normal
Exit code:0

```

Рисунок 3

5. Программа запущена, когда программный и загрузочный модули находятся в разных директориях, результат работы программы на рисунке 4.

```

C:\>lab6
memory is freed
error: file not found
C:\>_

```

Рисунок 4

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ ПО ЛАБОРАТОРНОЙ №6

1) Как реализовано прерывания Ctrl-C?

При нажатии данной комбинации клавиш управление передается по адресу 0000:008C. Этот адрес копируется в PSP функциями 26h и 4ch, при выходе из программы адрес восстанавливается.

2) В какой точке заканчивается вызываемая программа, если код 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 нажатие клавиш, т.е. при ожидании ввода символа.

Выводы.

Была исследована возможность построения загрузочного модуля динамической структуры.

Приложение А.

Исходный код программы lab6.asm:

```
dosseg
.model small
.stack 400h

.data

err_free db 0
_psp dw 0

string_free db 'memory is freed',0dh,0ah,$'
string_mcb_des db 'error: mcb is destroyed',0dh,0ah,$'
string_memory_n_enough db 'error: not enough memory', 0dh,0ah,$'
string_mcb_addr db 'error: invalid mcb address',0dh,0ah,$'

string_inv_num_func db 'error: invalid function number',0dh,0ah,$'
string_not_file db 'error: file not found',0dh,0ah,$'
string_disk db 'error: disk error',0dh,0ah,$'
string_wrong_env db 'error: wrong string of environment',0dh,0ah,$'
string_wrong_format db 'error: wrong format',0dh,0ah,$'

string_end_1 db 0dh,0ah,'Termination: normal',0dh,0ah,$'
string_end_2 db 0dh,0ah,'Termination: ctrl-break',0dh,0ah,$'
string_end_3 db 0dh,0ah,'Termination: device error',0dh,0ah,$'
string_end_4 db 0dh,0ah,'Termination: 31h function',0dh,0ah,$'
```

string_path db 128 DUP(0)

string_cmd_1 db 12

str db "It's a test!",0

block dw 0

dd 0

dd 0

dd 0

temp_sp dw 0

temp_ss dw 0

string_end_code db 'Exit code: ',0dh,0ah,'\$'

_end_data dd 0

.code

jmp m

WRITE_STR proc near

push ax

mov ah,9h

int 21h

pop ax

ret

WRITE_STR ENDP

FREE proc near

push ax

push bx

```
push cx
push dx

mov bx, offset _end_data
add bx, offset endofcode
add bx, 100
mov cl, 4h
shr bx, cl
mov ah, 4ah
int 21h
```

```
jnc ok
mov err_free, 1
```

```
cmp ax, 7
jne m1
mov dx, offset string_mcb_des
call write_str
jmp endFree
```

```
m1:
    cmp ax, 8
    jne m2
    mov dx, offset string_memory_n_enh
    call write_str
    jmp endFree
```

```
m2:
    cmp ax, 9
    mov dx, offset string_mcb_addr
    call write_str
    jmp endFree
```


ok:

```
mov dx,offset string_free  
call WRITE_STR
```

endFree:

```
pop dx  
pop cx  
pop bx  
pop ax
```

```
ret
```

FREE ENDP

path proc near

```
push ax  
push bx  
push cx  
push dx
```

```
mov es,_psp
```

```
mov es,es:[2ch] ;
```

```
xor di,di ; es:di - адрес переменных окружения  
;ищем 2 нуля, за ними путь
```

find_00:

```
mov al, es:[di]
```

```
inc di
```

```
cmp al,0
```

```
jne find_00
```

```

    mov al, es:[di]
    cmp al, 0
    jne find_00

    add di, 3
    mov si, offset string_path
n1:      ;смотрим путь
    mov cl, es:[di]
    mov byte ptr [si], cl
    cmp cl, 0h
    je n2
    inc si
    inc di
    jmp n1
n2:

    sub si, 8
    mov [si], byte ptr 'L'
    inc si
    mov [si], byte ptr 'A'
    inc si
    mov [si], byte ptr 'B'
    inc si
    mov [si], byte ptr '2'
    inc si
    mov [si], byte ptr '.'
    inc si
    mov [si], byte ptr 'C'
    inc si
    mov [si], byte ptr 'O'

```

```
inc si
mov [si],byte ptr 'M'
inc si
mov [si],byte ptr 0
inc si
mov [si],byte ptr '$'
```

```
pop dx
pop cx
pop bx
pop ax
ret
path endp
```

```
load proc near
```

```
push ax
push bx
push cx
push dx
```

```
mov temp_sp,sp
mov temp_ss,ss
push ds
push es
```

```
mov ax,@data
mov es,ax
mov bx,offset block; es:bx указывает на блок параметров
```

mov dx,offset string_path ; ds:dx указывает на строку,
содержащую путь и имя

mov word ptr [bx+2], offset string_cmd_1
mov word ptr [bx+4],ds

mov ax,4b00h
int 21h

pop es
pop ds
mov sp ,temp_sp
mov ss, temp_ss

jnc completion

err1:

cmp ax,1
jne err2
mov dx,offset string_inv_num_func
call write_str
jmp _end_load

err2:

cmp ax,2
jne err5
mov dx,offset string_not_file
call write_str
jmp _end_load

err5:

cmp ax,5
jne err8

```
mov dx,offset string_disk
```

```
call write_str
```

```
jmp _end_load
```

```
err8:
```

```
cmp ax,8
```

```
jne err10
```

```
mov dx,offset string_memory_n_enh
```

```
call write_str
```

```
jmp _end_load
```

```
err10:
```

```
cmp ax,10
```

```
jne err11
```

```
mov dx,offset string_wrong_env
```

```
call write_str
```

```
jmp _end_load
```

```
err11:
```

```
cmp ax,11
```

```
jne _end_load
```

```
mov dx,offset string_wrong_format
```

```
call write_str
```

```
jmp _end_load
```

```
completion:
```

```
mov ah,4dh
```

```
mov al,00h
```

```
int 21h
```

```
cmp ah,0
```

```
jne c2
```

```
mov si,offset string_end_code  
mov byte ptr [si+10],al
```

```
mov dx,offset string_end_1  
call write_str
```

```
mov dx,offset string_end_code  
call write_str
```

```
jmp _end_load
```

c2:

```
cmp ah,1  
jne c3  
mov dx,offset string_end_2  
call write_str  
jmp _end_load
```

c3:

```
cmp ah,2  
jne c4  
mov dx,offset string_end_3  
call write_str  
jmp _end_load
```

c4:

```
cmp ah,3  
jne _end_load  
mov dx,offset string_end_4  
call write_str
```

_end_load:

```
pop dx
```

```

        pop cx
        pop bx
        pop ax
    ret
load endp

MAIN PROC FAR
m:
    mov ax,@data
    mov ds,ax
    mov _psp,es

    call free
    cmp err_free,0
    je _next
    jmp end_prog
_next:
    call path
    call load

end_prog:

    xor al,al
    mov ah,4ch
    int 21h
endofcode:
MAIN ENDP

endath endp

```