

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.8382

Синельников М.Р

Преподаватель

Ефремов М.А

Санкт-Петербург

2020

Цель работы.

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

Ход работы.

1)

```
C:\>os6
Address not available memory: 9FFF
Environment address: 0353
Tail:
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
PATH to file:
C:\OS_2.COM
a
termination is ok
CODE: a
```

рисунок 1 — оба модуля находятся в одной директории

Введён символ a

2)

```
C:\>os6
Address not available memory: 9FFF
Environment address: 0353
Tail:
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
PATH to file:
C:\OS_2.COM
♥
termination is ok
CODE: ♥
```

рисунок 2 — оба модуля в одной директории

Введена комбинация Ctrl - C

3)

```

C:\>test\os6.exe
Address not available memory: 9FFF
Environment address: 0353
Tail:
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
PATH to file:
C:\OS_2.COM
c
termination is ok
CODE: c

```

рисунок 3 — оба модуля в другой директории

Введён символ c

4)

```

C:\>test\os6.exe
Address not available memory: 9FFF
Environment address: 0353
Tail:
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
PATH to file:
C:\OS_2.COM
♥
termination is ok
CODE: ♥

```

рисунок 4 — оба модуля в другой директории

Введена комбинация Ctrl - C

5)

```

Z:\>C:
C:\>test1\OS6.EXE
file is not found

```

рисунок 5 — модули в разных директориях

Контрольные вопросы.

1) Как реализовано прерывание Ctrl-C?

Если обнаруживается Ctrl-C, вызывается прерывание int 23h, что приводит к завершению процесса.

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

При вызове функции 4ch прерыванию int 21h.

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

При вызове функции 01h прерывания int 21h — функции считывания символа.

Вывод.

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

Приложение А

Исходный код файла OS6.asm

```
AStack SEGMENT      STACK
                dw 50 DUP(0)
```

```
AStack ENDS
```

```
DATA  SEGMENT
```

```
Message1 db 'Main memory block is destroyed', 0dh, 0ah, '$'
```

```
Message2 db 'Memory is not enough', 0dh, 0ah, '$'
```

```
Message3 db 'Address is not recognised', 0dh, 0ah, '$'
```

```
Message4 db 'function's number is wrong', 0dh, 0ah, '$'
```

```
Message5 db 'file is not found', 0dh, 0ah, '$'
```

```
Message6 db 'disk error', 0dh, 0ah, '$'
```

```
Message7 db 'memory size is not enough', 0dh, 0ah, '$'
```

```
Message8 db 'error with string environments', 0dh, 0ah, '$'
```

```
Message9 db 'format is wrong', 0dh, 0ah, '$'
```

```
Message10 db 'termination is ok', 0dh, 0ah, '$'
```

```
Message11 db 'termination after ctrl_break', 0dh, 0ah, '$'
```

```
Message12 db 'termination after device mistake', 0dh, 0ah, '$'
```

```
Message13 db 'termination after 31h function', 0dh, 0ah, '$'
```

```
code_string db 'CODE:      ', 0dh, 0ah, '$'
```

```
PARAMETER_BLOCK db 0
```

```
DD      0
```

```
DD      0
```

```
DD      0
```

```
PATH DB      128 DUP(0)
```

```
keep_ss dw 0
```

```
keep_sp dw 0
```

```
keep_ds dw 0
```

```
DATA ENDS
```

```
CODE  SEGMENT
```

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

print_endl proc

push ax

push dx

mov dl, 13

mov ah, 02h

int 21h

mov dl, 10

int 21h

pop dx

pop ax

ret

print_endl endp

print_message proc

push ax

mov ah, 09h

int 21h

pop ax

ret

print_message endp

Prepare_memory proc

push ax

push dx

lea ax, end_of_prog

inc ax

mov bx, ax

mov al, 0

mov ah, 4ah

int 21h

jnc finish

cmp ax, 7

jne next1

lea dx, message1

call print_message

```
jmp error_exit
```

```
next1:
```

```
cmp ax,8
```

```
jne next2
```

```
lea dx,message2
```

```
call print_message
```

```
jmp error_exit
```

```
next2:
```

```
cmp ax,9
```

```
jne finish
```

```
lea dx,message3
```

```
call print_message
```

```
error_exit:
```

```
mov ax, 4C00h
```

```
int 21h
```

```
finish:
```

```
pop dx
```

```
pop ax
```

```
ret
```

```
Prepare_memory endp
```

```
create_block proc
```

```
push ax
```

```
push bx
```

```
push dx
```

```
mov dx, es
```

```
mov bx,offset PARAMETER_BLOCK
```

```
mov ax,0
```

```
mov [bx],ax
```

```
mov ax,80h
```

```
mov [bx + 2],dx
```

```
mov [bx + 4],ax
```

```
mov ax,5ch
```

```
mov [bx + 6],dx
```

```
mov [bx + 8],ax
```

```
mov ax,6ch
```

```

    mov [bx + 10],dx
    mov [bx + 12],ax

    pop dx
    pop bx
    pop ax
    ret
create_block endp

prepare_string proc
    mov es,es:[2ch]
    mov si,0
    cmp byte ptr es:[si],00h
    je go
    mov ah,02h
do2:
    mov dl,es:[si]
    inc si
    cmp byte ptr es:[si],0
    jne do2
    cmp byte ptr es:[si + 1],0
    jne do2

go:
    lea di, PATH
    add si,4
    cmp byte ptr es:[si],00h
    je path_end
do3:
    mov dl,es:[si]
    mov [di],dl
    inc si
    inc di
    cmp byte ptr es:[si],00h
    jne do3

path_end:
    sub di,7
    mov [di], byte ptr 'O'
    mov [di+1], byte ptr 'S'
    mov [di+2], byte ptr '_'
    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 00h
```

```
    ret
```

```
prepare_string endp
```

```
left_preparation proc
```

```
    mov keep_sp,sp  
    mov keep_ss,ss  
    mov keep_ds,ds  
    mov ax, ds  
    mov es, ax  
    mov bx, offset PARAMETER_BLOCK  
    mov dx, offset PATH  
    ret
```

```
left_preparation endp
```

```
run proc
```

```
    mov ax,4B00h  
    int 21h  
    call print_endl  
    mov ds,keep_ds  
    mov ss,keep_ss  
    mov sp,keep_sp
```

```
    jnc ok
```

```
error:
```

```
    cmp ax,1  
    je error1  
    cmp ax,2  
    je error2  
    cmp ax,5  
    je error5  
    cmp ax,8  
    je error8  
    cmp ax,10  
    je error10
```

```
cmp ax,11  
je error11
```

```
error1:  
lea dx,message4  
jmp print_error
```

```
error2:  
lea dx,message5  
jmp print_error
```

```
error5:  
lea dx,message6  
jmp print_error
```

```
error8:  
lea dx,message7  
jmp print_error
```

```
error10:  
lea dx,message8  
jmp print_error
```

```
error11:  
lea dx,message9  
jmp print_error
```

```
print_error:  
call print_message  
mov ax, 4C00h  
int 21h
```

```
ok:  
mov ax, 4D00h  
int 21h
```

```
cmp ah,0  
je result0  
cmp ah,1  
je result1  
cmp ah,2  
je result2
```

```
cmp ah,3
je result3
```

```
result0:
mov dx, offset message10
call print_message
push di
mov di,offset code_string
mov [di + 6], al
pop di
mov dx,offset code_string
call print_message
ret
```

```
result1:
lea dx, message11
jmp print_result
```

```
result2:
lea dx, message12
jmp print_result
```

```
result3:
lea dx, message13
jmp print_result
```

```
print_result:
call print_message
```

```
finish_run:
ret
```

```
run endp
```

```
Main PROC FAR
```

```
mov ax,data
mov ds,ax
call Prepare_memory
call create_block
call prepare_string
```

```
call left_preparation
call run
mov ax,4c00h
int 21h
ret
end_of_prog:
```

```
Main ENDP
```

```
CODE ENDS
```

```
END Main
```