

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. 9382

Дерюгин Д.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

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:

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.

Вызываемый модуль запускается с использованием загрузчика.

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.

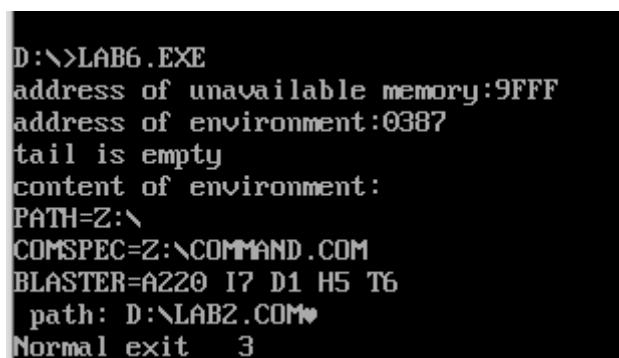
Запустили программу из текущего каталога



```
D:\>LAB6.EXE
address of unavailable memory:9FFF
address of environment:0387
tail is empty
content of environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
path: D:\LAB2.COM
Normal exit 113
```

Рис. 1 Результат работы программы

Теперь введем комбинацию CTRL-C. Как видно на скриншоте ниже, программа не выдала ошибку, так как в dos не реализована данная комбинация.



```
D:\>LAB6.EXE
address of unavailable memory:9FFF
address of environment:0387
tail is empty
content of environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
path: D:\LAB2.COM
Normal exit 3
```

Рис. 2 Ввод комбинации ctrl-c

Теперь изменим текущий каталог и снова запустим нашу программу.

```
D:\>cd ../a  
  
D:\A>LAB6.EXE  
address of unavailable memory:9FFF  
address of environment:0387  
tail is empty  
content of environment:  
PATH=Z:\  
COMSPEC=Z:\COMMAND.COM  
BLASTER=A220 I7 D1 H5 T6  
path: D:\A\LAB2.COM  
Normal exit 113
```

Рис. 3 Результат запуска программы с другой директории

Теперь запустим нашу программу, когда модули находятся в разных каталогах. На рисунке 4 видно, что программа выдала ошибку о том, что файл не найден

```
D:\>cd ../a  
  
D:\A>LAB6.EXE  
File not found
```

Рис. 4 Результат работы программы, когда модули в разных папках

Ответы на вопросы.

1. Как реализовано прерывание Ctrl-C?

При нажатии комбинации управление передается по адресу 0000:008ch. Этот адрес копируется в PSP функциями 26h и 4ch и восстанавливается из PSP, когда программа завершается.

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 было произведено нажатие клавиши, то есть в месте ожидания нажатия 01h прерывания int 21h.

Выводы.

В ходе данной лабораторной работы были получены навыки построения загрузочного модуля динамической структуры.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ

ASTACK SEGMENT STACK

DW 1000 DUP (?)

ASTACK ENDS

DATA SEGMENT

BLOCKED_DESTROYED db 'Block destroyed' , 13, 10, '\$'

LOW_MEMORY db 'Low memory', 13, 10, '\$'

INVALID_ADRESS_OF_BLOCK db 'Invalid adress of block', 13, 10, '\$'

NUMBER_OF_FUNCTION_INVALID db 'Number of function invalid', 13, 10, '\$'

FILE_NOT_FOUNT db 'File not found', 13, 10, '\$'

DISK_ERROR db 'Disk error', 13, 10, '\$'

INCORRECT_ENVIRONMENT_STRING db 'Incorret environment string', 13, 10, '\$'

INVALID_FORMAT db 'Invalid format', 13, 10, '\$'

NORMAL db 13, 10, 'Normal exit ', 13, 10, '\$'

CTRL_BREAK db 'Programm finished using ctrl-break ', 13, 10, '\$'

DEVICE_ERROR db 13, 10, 'Device error ', 13, 10, '\$'

FUNCTION_31H db 13, 10, 'Finished by 31h ', 13, 10, '\$'

FILE_NAME db 'LAB2.COM', 0

fpb dw 0

dd 0

dd 0

dd 0

DATA ENDS

CODE SEGMENT

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

print PROC near

push ax

mov ah, 09h

int 21h

pop ax

ret

print ENDP

BYTE_TO_DEC PROC near

;Перевод в 10чную с/с, SI - адрес младшей цифры

```
    push CX
    push DX
    xor AH,AH
    xor DX,DX
    mov CX,10
loop_bd:    div CX
            or DL,30h
            mov [SI],DL
            dec SI
            xor DX,DX
            cmp AX,10
            jae loop_bd
            cmp AL,00h
            je end_1
            or AL,30h
            mov [SI],AL
end_1:      pop DX
            pop CX
            ret
```

BYTE_TO_DEC ENDP

SET_FPB PROC NEAR

```
    mov ax,es:[2ch]
    mov fpb,ax
    mov fpb+2,es
    mov fpb+4,80h
    ret
```

SET_FPB ENDP

PORSSESING_PROC PROC NEAR

```
    push ax
    mov ah, 4dh
    int 21h

    cmp ah, 0
    je normal_1

    cmp ah, 1
    je ctrl_break_1

    cmp ah, 2
```

```

        je device_error_1

        cmp ah, 3
        je func_31h

normal_1:
        mov si, offset NORMAL
        add si, 16
        call BYTE_TO_DEC
        mov dx, offset NORMAL
        call print
        jmp exit

ctrl_break_1:
        mov dx, offset CTRL_BREAK
        call print
        jmp exit

device_error_1:
        mov dx, offset DEVICE_ERROR
        call print
        jmp exit

func_31h:
        mov dx, offset FUNCTION_31H
        call print
        jmp exit

exit:
        pop ax
        ret
PORSSESING_PROC ENDP

CHANGE_MEMORY PROC NEAR

        jmp start
        KEEP_SS dw 0
        KEEP_SP dw 0
        KEEP_DS dw 0

start:

```



```

push bx
mov bx, 500
mov ah, 4ah
int 21h
pop bx
jnc success
; error

cmp ax, 7
je block_destroyed_1

cmp ax, 8
je low_memory_1

cmp ax, 9
je invalid_adress_of_block_1

jmp ending

block_destroyed_1:
mov dx, offset BLOCKED_DESTROYED
call print
jmp ending

low_memory_1:
mov dx, offset LOW_MEMORY
call print
jmp ending

invalid_adress_of_block_1:
mov dx, offset INVALID_ADRESS_OF_BLOCK
call print
jmp ending

;success
success:

call SET_FPB
mov bx, offset fpb

mov dx, offset FILE_NAME

```

```
mov KEEP_SS, ss
mov KEEP_SP, sp
mov KEEP_DS, ds
```

```
mov ax, 4B00h
int 21h
jnc prossesing
```

```
cmp ax, 1
je incorrect_number
```

```
cmp ax, 2
je not_found
```

```
cmp ax, 5
je disk_error_1
```

```
cmp ax, 8
je low_mem
```

```
cmp ax, 10
je incorrect_string
```

```
cmp ax, 11
je incorrect_format
jmp restore
```

```
incorrect_number:
mov dx, offset NUMBER_OF_FUNCTION_INVALID
call print
jmp restore
```

```
not_found:
mov dx, offset FILE_NOT_FOUNT
call print
jmp restore
```

```
disk_error_1:
mov dx, offset DISK_ERROR
```

```

    call print
    jmp restore

low_mem:
    mov dx, offset LOW_MEMORY
    call print
    jmp restore

incorrect_string:
    mov dx, offset INCORRECT_ENVIRONMENT_STRING
    call print
    jmp restore

incorrect_format:
    mov dx, offset INVALID_FORMAT
    call print
    jmp restore

prossesing:
    call PORSSSESING_PROC

restore:
    mov ss, keep_ss
    mov sp, keep_sp
    mov ds, keep_ds

    jmp ending

ending:
    ret
CHANGE_MEMORY ENDP

MAIN PROC FAR
    mov ax, DATA
    mov ds, ax
    call CHANGE_MEMORY
    mov ah, 4ch
    int 21h

```

```
MAIN ENDP  
CODE ENDS  
END MAIN
```