

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по лабораторной работе №6
по дисциплине «Операционные системы»
Тема: Построение модуля динамической структуры

Студент гр. 8382

Нечепуренко Н.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. Исполняемый модуль должен вызывать другую программу и анализировать код возврата.

Выполнение работы.

Для выполнения работы был реализован .exe модуль, который сначала освобождает неиспользуемую память, затем загружает модифицированный модуль лабораторной работы №2, который ждёт пользовательского ввода одного символа и возвращает его в код завершения. После завершения вызываемого модуля вызывающая программа анализирует код и причину завершения вызываемой программы и выводит информацию на экран. Полный исходный код программы представлен в Приложении А.

Пусть оба модуля лежат в корневой директории. Выполним запуск модуля динамической структуры и введём «с». Результат представлен на рисунке 1.



```
C:\>lab6
Normal shrink!
C:\lab2.com Inaccessible memory starts at 9FFF.
Environment address is 1179.
Command line tail is
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Module path:C:\LAB2.COM
cLoad lab2!
Normal exit!
Exit code is 63
```

Рисунок 1 – Корневая директория, введено: с

Повторим запуск, но теперь введём сочетание клавиш «ctrl-c». Результат приведён на рисунке 2.

```

C:\>lab6
Normal shrink!
C:\lab2.com Inaccessible memory starts at 9FFF.
Environment address is 1179.
Command line tail is
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Module path:C:\LAB2.COM
♥Load lab2!
Normal exit!
Exit code is 03

```

Рисунок 2 – Корневая директория, введено: ctrl-c

Теперь создадим директорию «test» и переместим в неё оба модуля. Повторим оба вызова (см. рис. 3 и 4).

```

C:\>TEST\LAB6.EXE
Normal shrink!
C:\TEST\lab2.com Inaccessible memory starts at 9FFF.
Environment address is 1179.
Command line tail is
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Module path:C:\TEST\LAB2.COM
cLoad lab2!
Normal exit!
Exit code is 63

```

Рисунок 3 – Директория test, введено: c

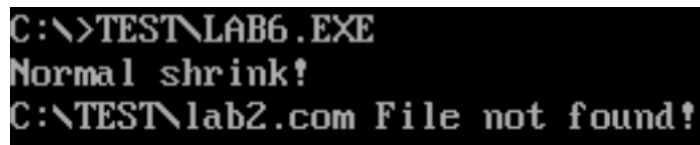
```

C:\>TEST\LAB6.EXE
Normal shrink!
C:\TEST\lab2.com Inaccessible memory starts at 9FFF.
Environment address is 1179.
Command line tail is
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Module path:C:\TEST\LAB2.COM
♥Load lab2!
Normal exit!
Exit code is 03

```

Рисунок 4 – Директория test, введено: ctrl-c

Удалим вызываемый модуль из директории test, запустим вызывающий модуль. Результат представлен на рисунке 5.



```
C:\>TEST\LAB6.EXE
Normal shrink!
C:\TEST\lab2.com File not found!
```

Рисунок 5 – Вызов модуля, которого нет в текущей директории
Получаем ожидаемое сообщение о том, что файл не найден.

Контрольные вопросы.

1. Как реализовано прерывание Ctrl-C?

При нажатии сочетания клавиш ctrl-c генерируется прерывание DOS 23h и управление передаётся вызывающей программе.

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

Программа завершается при вызове функции 4ch прерывания DOS 21h (в регистре al находится код возврата), что означает штатное завершение программы.

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

Программа завершается на вводе символа, т.е. на вызове функции 01h прерывания DOS 21h.

Выводы.

В результате выполнения работы был реализован модуль динамической структуры, были получены знания о работе механизмов завершения программы, например, об обработке сочетания «ctrl-c».

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
astack segment stack
    dw 100 dup(?)
astack ends
```

```
dataseg segment
    msg_shrink_normal db "Normal shrink!", 13, 10, "$"
    msg_shrink_destroyed db "Control block is destroyed!", 13, 10, "$"
    msg_shrink_notenough db "Not enough memory!", 13, 10, "$"
    msg_shrink_invalidadr db "Invalid address!", 13, 10, "$"
    msg_shrink_error_offset dw offset msg_shrink_destroyed
                                dw offset msg_shrink_notenough
                                dw offset msg_shrink_invalidadr
    msg_load_invalid_num_func db "Invalid function number!", 13, 10, "$"
    msg_load_fnf db "File not found!", 13, 10, "$"
    msg_load_disk db "Disk error!", 13, 10, "$"
    msg_load_notenough db "Not enough memory to load!", 13, 10, "$"
    msg_load_envir db "Invalid envir str!", 13, 10, "$"
    msg_load_format db "Invalid format!", 13, 10, "$"
    msg_load_normal db "Load lab2!", 13, 10, "$"
    msg_load_err_offset dw 0
                                dw offset msg_load_invalid_num_func
                                dw offset msg_load_fnf
                                dw 0
                                dw 0
                                dw offset msg_load_disk
                                dw 0
                                dw 0
                                dw offset msg_load_notenough
                                dw 0
                                dw offset msg_load_envir
                                dw offset msg_load_format
    msg_exit_normal db "Normal exit!", 13, 10, "$"
    msg_exit_cbreak db "Ctrl+Break exit!", 13, 10, "$"
    msg_exit_crit db "Critical error exit!", 13, 10, "$"
    msg_exit_resident db "31h exit!", 13, 10, "$"
    msg_exit_code_offset dw offset msg_exit_normal
                                dw offset msg_exit_cbreak
                                dw offset msg_exit_crit
                                dw offset msg_exit_resident
```

```

msg_exit_code db "Exit code is  ", 13, 10, "$"
parameter_block dw ?
                dd ?
                dd ?
                dd ?
path db 100h dup("$")
filename db "lab2.com", 0, "$"
keep_ss dw 0
keep_sp dw 0
keep_ds dw 0

dataseg ends

codeseg segment
    assume ds:dataseg, cs:codeseg, ss:astack

shrink_memory proc near
    push ax
    push bx
    push cx
    push di
    push si
    mov ax, es
    sub bx, ax ; get the amount of memory this program use
    mov cl, 4
    shr bx, cl ; bx / 16 -> in paragraphs
    mov ax, 4a00h
    int 21h
    jc shrink_memory_error_occured
shrink_memory_normal:
    mov di, offset msg_shrink_normal
    call print
    jmp shrink_memory_final
shrink_memory_error_occured:
    sub ax, 7
    shl ax, 1
    mov bx, ax
    mov si, offset msg_shrink_error_offset
    mov di, ds:[si+bx]
    call print
    mov ax, 4c00h
    int 21h

```

```

shrink_memory_final:
    pop si
    pop di
    pop cx
    pop bx
    pop ax
    ret
shrink_memory endp

construct_param_block proc near
    push ax
    push bx
    push dx
    mov bx, offset parameter_block
    mov ax, 0 ; to inherit envir
    mov [bx], ax
    mov dx, es ; seg
    mov [bx+2], dx
    mov ax, 80h ; offs of num symbols in cmd
    mov [bx+4], ax
    mov [bx+6], dx ; seg
    mov ax, 5Ch ; 1st fcb
    mov [bx+8], ax
    mov [bx+10], dx
    mov ax, 6Ch ; 2nd fcb
    mov [bx+12], ax

construct_param_block_final:
    pop dx
    pop bx
    pop ax
    ret
construct_param_block endp

construct_path proc near
    push es
    push si
    push di
    push ax
    push cx
    mov es, es:[2Ch] ; envir addr
    mov si, 0

```

```

construct_path_skip_envir:
    mov al, es:[si]
    cmp al, 0
    je construct_path_if_all_skipped
    inc si
    jmp construct_path_skip_envir

construct_path_if_all_skipped:
    inc si
    mov al, es:[si]
    cmp al, 0
    jne construct_path_skip_envir

construct_path_find_module_path:
    add si, 3
    mov di, offset path
construct_path_copy:
    mov al, es:[si]
    cmp al, 0
    je construct_path_copy_name
    mov [di], al
    inc di
    inc si
    jmp construct_path_copy

construct_path_copy_name:
    sub di, 8
    mov cx, 9
    mov si, offset filename
construct_path_copy_name_loop:
    mov al, [si]
    mov [di], al
    inc si
    inc di
    loop construct_path_copy_name_loop

construct_path_final:
    mov di, offset path
    call print
    pop bx
    pop ax
    pop di

```



```

        pop si
        pop es
        ret
construct_path endp

load_lab2 proc near
    mov keep_ss, ss
    mov keep_sp, sp
    mov keep_ds, ds
    push bx
    push dx
    push ax
    push si
    push di

    mov bx, offset parameter_block
    mov dx, offset path
    mov ax, 4b00h
    int 21h ; call lab2
    jnc load_lab2_loaded
    mov bx, ax
    mov ax, keep_ss
    mov ss, ax
    mov sp, keep_sp
    mov ax, keep_ds
    mov ds, ax
    mov si, offset msg_load_err_offset
    shl bx, 1
    mov di, ds:[si+bx]
    call print
    mov ax, 4c00h
    int 21h
load_lab2_loaded:
    mov di, offset msg_load_normal
    call print
    mov ax, 4d00h ; get exit code of last proc
    int 21h ; ax = [reason][code]
    xor bx, bx
    mov bl, ah
    shl bx, 1
    mov si, offset msg_exit_code_offset
    mov di, ds:[si+bx]

```

```

        call print

load_lab2_exit_code:
    mov di, offset msg_exit_code
    push di
    add di, 13
    call BYTE_TO_HEX
    mov [di], al
    inc di
    mov [di], ah
    pop di
    call print

load_lab2_final:
    pop di
    pop si
    pop ax
    pop dx
    pop bx
    mov ax, 4c00h
    int 21
    ret
load_lab2 endp

main proc near
    mov ax, dataseg
    mov ds, ax
    call shrink_memory
    call construct_param_block
    call construct_path
    call load_lab2

    mov ax, 4c00h
    int 21h

main endp

print proc near
    ; prints di content
    push dx
    push ax
    mov ah, 9h

```

```

        mov dx, di
        int 21h
        pop ax
        pop dx
        ret
print 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
;байт в AL переводится в два символа шест. числа в AX
        push CX
        mov AH,AL
        call TETR_TO_HEX
        xchg AL,AH
        mov CL,4
        shr AL,CL
        call TETR_TO_HEX ;в AL старшая цифра
        pop CX ;в AH младшая
        ret
BYTE_TO_HEX ENDP

code_end:
codeseg ends
        end main

```