**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

**отчет**

по лабораторной работе №**6**

по дисциплине «Операционные системы»

на тему: Построение модуля динамической структуры

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Кухарев М.А. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы**

Исследование возможности построения загрузочного модуля динамической структуры. В отличии от предыдущих лабораторных работ в этой работе рассматривается приложение, состоящее из нескольких модулей, а не из одного модуля простой структуры. В этом случае разумно предположить, что все модули приложения находятся в одном каталоге и полный путь в этот каталог можно взять из среды, как это делалось в работе 2. Понятно, что такое приложение должно запускаться в соответствии со стандартами ОС.

В работе исследуется интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция 4В00h прерывания int 21h. Все загрузочные модули находятся в одном каталоге. Необходимо обеспечить возможность запуска модуля динамической структуры из любого каталога.

**Ход работы**

1. Запустим отлаженную программу, когда текущим каталогом является каталог с разработанными модулями:



Рис.1 Запуск программы LAB6.exe

Программа вызывает другую программу (LAB2.EXE), которая останавливается, ожидая символ с клавиатуры.

1. Запустим LAB6.exe и введем комбинацию Ctrl-C. Программа и модуль в одной папке.

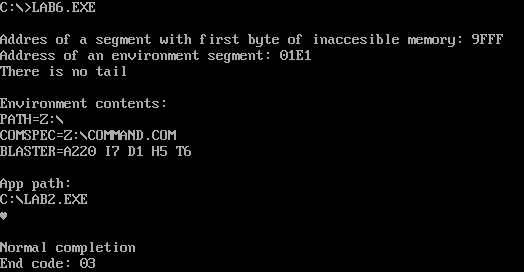


Рис.2 Запуск программы LAB6.exe c Ctrl-C

1. Переместим модуль L2.EXE в другой каталог.



Рис.3

1. Находясь в каталоге «С:», запустим программу, которая находится в каталоге «С:\test»



*Рисунок 4.*

1. Находясь в каталоге «С:\test», запустим программу, которая находится в каталоге «С:»



*Рисунок 5.*

**Ответы на контрольные вопросы.**

1. **Как реализовано прерывание Ctrl-C?**

Прерывание 23h вызывается, если была нажаты комбинации клавиш Ctrl-C или Ctrl-Break. Адрес, по которому передается управление (0000:008c). Управление передаётся тогда, когда DOS распознает, что пользователь нажал Ctrl-Break или Ctrl-C. Адрес по вектору INT 23h копируется в поле PSP Ctrl-Break Address функциями DOS 26h (создать PSP) и 4Ch (EXEC).

Исходное значение адреса обработчика Ctrl-Break восстанавливается из PSP при завершении программы. Таким образом, по завершении порожденного процесса будет восстановлен адрес обработчика Ctrl-Break из родительского процесса.

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

Если код причины завершения 0, то вызываемая программа заканчивается в месте вызова функции 4Ch прерываний int 21h.

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

В месте, где программа ожидала ввода символа: в точке вызова функции 01h прерывания int 21h.

**Приложение A: Код программы LAB6.asm**

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACKSEG

START: JMP BEGIN

;---------------------------------------

; Вызывает прерывание, печатающее строку.

PRINT PROC near

push ax

mov AH,09h

int 21h

pop ax

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

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR\_TO\_HEX

pop CX

ret

BYTE\_TO\_HEX ENDP

; Функция освобождения лишней памяти

CLEARMEMORY PROC

; Вычисляем в BX необходимое количество памяти для этой программы в параграфах

mov ax,STACKSEG ; В ax сегментный адрес стека

mov bx,es

sub ax,bx ; Вычитаем сегментный адрес PSP

add ax,10h ; Прибавляем размер стека в параграфах

mov bx,ax

; Пробуем освободить лишнюю память

mov ah,4Ah

int 21h

jnc CLEARMEMORY\_SUCCESS

; Обработка ошибок

mov dx,offset ERR\_CLEARMEMORY

call PRINT

cmp ax,7

mov dx,offset ERR\_MCB\_DESTROYED

je CLEARMEMORY\_PRINT

cmp ax,8

mov dx,offset ERR\_FEW

je CLEARMEMORY\_PRINT

cmp ax,9

mov dx,offset ERR\_WRONG\_ADDR

CLEARMEMORY\_PRINT:

call PRINT

mov dx,offset STRENDL

call PRINT

; Выход в DOS

xor AL,AL

mov AH,4Ch

int 21H

CLEARMEMORY\_SUCCESS:

ret

CLEARMEMORY ENDP

; Функция создания блока параметров

PARAMETERS\_BLOCK PROC

mov ax, es:[2Ch]

mov PARAMETERSBLOCK,ax ; Кладём сегментный адрес среды

mov PARAMETERSBLOCK+2,es ; Сегментный адрес параметров командной строки(PSP)

mov PARAMETERSBLOCK+4,80h ; Смещение параметров командной строки

ret

PARAMETERS\_BLOCK ENDP

; Функция запуска дочернего процесса

CHILD\_PROCESS PROC

mov dx,offset STRENDL

call PRINT

; Устанавливаем DS:DX на имя вызываемой программы

mov es,es:[2ch]

mov si,0

next1:

mov dl,es:[si]

cmp dl,0

je end\_path

inc si

jmp next1

end\_path:

inc si

mov dl,es:[si]

cmp dl,0

jne next1

add si,3

lea di,CHILD\_PATH

next2:

mov dl, es:[si]

cmp dl,0

je end\_copy

mov [di],dl

inc di

inc si

jmp next2

end\_copy:

sub di,8

mov [di], byte ptr 'l'

mov [di+1], byte ptr 'a'

mov [di+2], byte ptr 'b'

mov [di+3], byte ptr '2'

mov [di+4], byte ptr '.'

mov [di+5], byte ptr 'e'

mov [di+6], byte ptr 'x'

mov [di+7], byte ptr 'e'

mov dx,offset CHILD\_PATH ; Хвост есть, используем его

; Устанавливаем ES:BX на блок параметров

push ds

pop es

mov bx,offset PARAMETERSBLOCK

; Сохраняем SS, SP

mov KEEP\_SP, SP

mov KEEP\_SS, SS

; Вызываем загрузчик:

mov ax,4b00h

int 21h

jnc CHILD\_PROCESS\_SUCCESS

; Восстанавливаем DS, SS, SP

push ax

mov ax,DATA

mov ds,ax

pop ax

mov SS,KEEP\_SS

mov SP,KEEP\_SP

; Обрабатываем ошибки:

cmp ax,1

mov dx,offset ERR\_WRONG\_FUNCNUM

je CHILD\_PROCESS\_PRINT

cmp ax,2

mov dx,offset ERR\_FILE\_NOT\_FOUND

je CHILD\_PROCESS\_PRINT

cmp ax,5

mov dx,offset ERR\_DISK

je CHILD\_PROCESS\_PRINT

cmp ax,8

mov dx,offset ERR\_FEW2

je CHILD\_PROCESS\_PRINT

cmp ax,10

mov dx,offset ERR\_WRONG\_ENVIRON\_STR

je CHILD\_PROCESS\_PRINT

cmp ax,11

mov dx,offset ERR\_WRONG\_FORMAT

CHILD\_PROCESS\_PRINT:

call PRINT

mov dx,offset STRENDL

call PRINT

; Выходим в DOS

xor AL,AL

mov AH,4Ch

int 21H

CHILD\_PROCESS\_SUCCESS:

mov dx,offset STRENDL

call PRINT

mov ax,4d00h

int 21h

; Вывод причины завершения

cmp ah,0

mov dx,offset NORMAL\_COMPLETION

je CHILD\_PROCESS\_PRINT\_REASON

cmp ah,1

mov dx,offset CTRL\_BREAK

je CHILD\_PROCESS\_PRINT\_REASON

cmp ah,2

mov dx,offset DEVICE\_ERROR

je CHILD\_PROCESS\_PRINT\_REASON

cmp ah,3

mov dx,offset RESIDENT\_END

CHILD\_PROCESS\_PRINT\_REASON:

call PRINT

mov dx,offset STRENDL

call PRINT

; Вывод кода завершения:

mov dx,offset END\_CODE

call PRINT

call BYTE\_TO\_HEX

push ax

mov ah,02h

mov dl,al

int 21h

pop ax

xchg ah,al

mov ah,02h

mov dl,al

int 21h

mov dx,offset STRENDL

call PRINT

ret

CHILD\_PROCESS ENDP

BEGIN:

mov ax,data

mov ds,ax

call CLEARMEMORY

call PARAMETERS\_BLOCK

call CHILD\_PROCESS

xor AL,AL

mov AH,4Ch

int 21H

CODE ENDS

DATA SEGMENT

; Строки ошибок:

ERR\_CLEARMEMORY db 'Error with clear memory: $'

ERR\_MCB\_DESTROYED db 'MCB is destroyed$'

ERR\_FEW db 'Not enough memory$'

ERR\_WRONG\_ADDR db 'Wrong addres$'

; Ошибки от загрузчика OS

ERR\_WRONG\_FUNCNUM db 'Function number is wrong$'

ERR\_FILE\_NOT\_FOUND db 'File is not found$'

ERR\_DISK db 'Disk error$'

ERR\_FEW2 db 'Not enough memory$'

ERR\_WRONG\_ENVIRON\_STR db 'Wrong environment string$'

ERR\_WRONG\_FORMAT db 'Wrong format$'

; Строки, содержащие причины завершения дочерней программы

NORMAL\_COMPLETION db 'Normal completion$'

CTRL\_BREAK db 'End Ctrl-Break$'

DEVICE\_ERROR db 'End device error$'

RESIDENT\_END db 'End 31h function$'

END\_CODE db 'End code: $'

STRENDL db 0DH,0AH,'$'

; Блок параметров. Перед загрузкой дочерней программы на него должен указывать ES:BX

PARAMETERSBLOCK dw 0 ; Сегментный адрес среды

dd ? ; Сегментный адрес и смещение параметров командной строки

dd 0 ; Сегмент и смещение первого FCB

dd 0 ; Второго

CHILD\_PATH db 20h dup (0)

; Переменные для хранения SS, SP

KEEP\_SS dw 0

KEEP\_SP dw 0

DATA ENDS

STACKSEG SEGMENT STACK

dw 80h dup (?) ; 100h байт

STACKSEG ENDS

END START