**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:

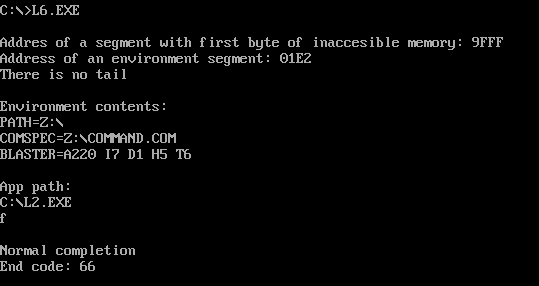


Рис.1 Запуск программы L6.exe

Программа вызывает другую программу (L2.EXE), которая останавливается, ожидая символ с клавиатуры.

1. Запустим L6.exe и введем комбинацию Ctrl-C. Программа и модуль в одной папке.

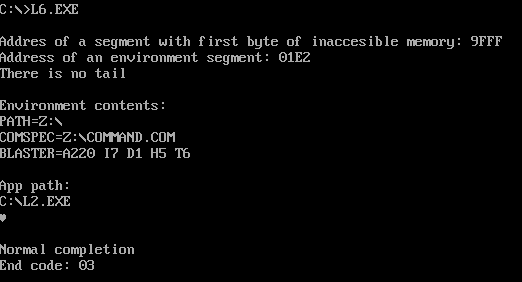


Рис.2 Запуск программы L6.exe c Ctrl-C

1. Запуск программы, когда текущим каталогом является какой-либо другой каталог, отличный от того в котором содержатся разработанные программные модули.

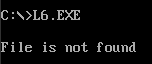


Рис.3 Запуск программы из другого каталога.

1. Укажем путь в хвосте:



Рис.4 Запуск программы из другого каталога.

**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 LR6.asm**

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

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

AStack SEGMENT STACK

DW 64 DUP(0)

AStack ENDS

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

DATA SEGMENT

ENVIRONMENT\_ADRESS dw 0

CMD\_ADDRESS dw 0

CMD\_OFFSET dw 0

FSB1\_ADDRESS dw 0

FSB1\_OFFSET dw 0

FSB2\_ADDRESS dw 0

FSB2\_OFFSET dw 0

DYNAMIC\_STRUCT db 'This is module with dynamic struct!', 0dh, 0ah, '$'

SUB\_MODULE\_PATH db ' ', 0dh, 0ah, '$', 00h

SUB\_MODULE\_START db'Sub module was executed!', 0dh, 0ah, '$'

EOL db ' ',0dh, 0ah,'$'

KEEP\_SS dw 0

KEEP\_SP dw 0

DATA ENDS

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

CODE SEGMENT

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

MAIN\_PROC PROC FAR

mov ax, DATA

mov ds, ax

mov dx, offset DYNAMIC\_STRUCT

call PRINT\_STRING

mov dx, offset EOL

call PRINT\_STRING

;clear mem

mov cl, 04h

mov ah, 4ah

mov bx, offset NEED\_MEM\_AREA

shr bx, cl

add bx, 30h

int 21h

;fill parameters

mov ax, es

mov ENVIRONMENT\_ADRESS, 00h

mov word ptr CMD\_ADDRESS, ax

mov word ptr CMD\_OFFSET, 0080h

mov word ptr FSB1\_ADDRESS, ax

mov word ptr FSB1\_OFFSET, 005Ch

mov word ptr FSB2\_ADDRESS, ax

mov word ptr FSB2\_OFFSET, 006Ch

;get sub module path

push es

push dx

push bx

mov es, es:[2Ch]

mov bx, 00h

START\_C1:

mov dl, es:[bx]

cmp dl, 00h

je END\_C1

inc bx

jmp START\_C1

END\_C1:

inc bx

mov dl, es:[bx]

cmp dl, 00h

jne START\_C1

push di

mov di, offset SUB\_MODULE\_PATH

add bx, 03h

PATH\_START:

mov dl, es:[bx]

cmp dl,00h

je PATH\_END

mov [di], dl

inc di

inc bx

jmp PATH\_START

PATH\_END:

sub di, 5h

mov [di + 0], byte ptr '2'

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

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

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

mov [di + 5], byte ptr 0h

pop di

pop bx

pop dx

pop es

;start sub module

mov KEEP\_SP, sp

mov KEEP\_SS, ss

push ds

mov ax, DATA

mov es, ax

mov bx, offset ENVIRONMENT\_ADRESS

mov ds, ax

mov dx, offset SUB\_MODULE\_PATH

mov ax, 4B00h

int 21h

pop ds

mov ss, KEEP\_SS

mov sp, KEEP\_SP

mov dx, offset EOL

call PRINT\_STRING

jnc COMPLETED

call ERROR\_PROCESSING

jmp M\_EXIT

COMPLETED:

call COMPL\_PROCESSING

M\_EXIT:

mov ah, 4Ch

int 21h

MAIN\_PROC ENDP

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

PRINT\_STRING PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT\_STRING ENDP

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

CLEAR\_MEMORY PROC NEAR

mov cl, 04h

mov ah, 4ah

mov bx, offset NEED\_MEM\_AREA

shr bx, cl

add bx, 33h

int 21h

ret

CLEAR\_MEMORY ENDP

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

FILL\_PARAMETERS PROC NEAR

mov ax, es

mov word ptr CMD\_ADDRESS, ax

mov word ptr CMD\_OFFSET, 0080h

mov word ptr FSB1\_ADDRESS, ax

mov word ptr FSB2\_OFFSET, 005Ch

mov word ptr FSB2\_ADDRESS, ax

mov word ptr FSB2\_OFFSET, 006Ch

ret

FILL\_PARAMETERS ENDP

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

COMPL\_PROCESSING PROC near

jmp completed\_begin

pfn db 'Programm finished normally with code 0!',13,10,'$'

pfwcb db 'Programm finished with Ctrl+Break!',13,10,'$'

pfweod db 'Programm finished with error of device!',13,10,'$'

pfw31h db 'Programm finished with 31h!',13,10,'$'

pfwc db 'Programm finished with code !',13,10,'$'

completed\_begin:

push ds

push ax

push dx

push bx

mov ah,4Dh

int 21h

push ax

mov ax,SEG pfn

mov ds,ax

pop ax

cmp ah,0h

jne not\_com\_0

mov dx,offset pfn

not\_com\_0:

cmp ah,1h

jne not\_com\_1

mov dx,offset pfwcb

not\_com\_1:

cmp ah,2h

jne not\_com\_2

mov dx,offset pfweod

not\_com\_2:

cmp ah,3h

jne not\_com\_3

mov dx,offset pfw31h

not\_com\_3:

mov ah,9h

push ax

int 21h

pop ax

mov dx,offset pfwc

mov bx,dx

add bx,1Ch

mov byte ptr [bx],al

int 21h

pop bx

pop dx

pop ax

pop ds

ret

COMPL\_PROCESSING ENDP

ERROR\_PROCESSING PROC near

jmp errors\_begin

wnof db 'Wrong number of function!',13,10,'$'

fnf db 'File not found!',13,10,'$'

de db 'Disc error!',13,10,'$'

nem db 'Not enough mmory!',13,10,'$'

wsoe db 'Wrong string of enviroment!',13,10,'$'

wf db 'Wrong format!',13,10,'$'

errors\_begin:

push ds

push ax

push dx

push ax

mov ax,SEG wnof

mov ds,ax

pop ax

cmp ax,1h

jne not\_err\_1

mov dx,offset wnof

not\_err\_1:

cmp ax,2h

jne not\_err\_2

mov dx,offset fnf

not\_err\_2:

cmp ax,5h

jne not\_err\_5

mov dx,offset de

not\_err\_5:

cmp ax,8h

jne not\_err\_8

mov dx,offset nem

not\_err\_8:

cmp ax,10h

jne not\_err\_10

mov dx,offset wsoe

not\_err\_10:

cmp ax,11h

jne not\_err\_11

mov dx,offset wf

not\_err\_11:

mov ah,9h

int 21h

pop dx

pop ax

pop ds

ret

ERROR\_PROCESSING ENDP

NEED\_MEM\_AREA:

CODE ENDS

END MAIN\_PROC