**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. 6383 |  | Никитин К.В. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы.**

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерывания получает управление по прерыванию (int 09h) при нажатии клавиши на клавиатуре. Он обрабатывает скан-код и осуществляет определенные действия, если скан-код совпадает с определенными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передается стандартному прерыванию.

**Описание функций и структур данных.**

|  |  |
| --- | --- |
| Название | Назначение |
| TETR\_TO\_HEX | перевод половины байта в AL в символ в 16-ричной с/c |
| BYTE\_TO\_HEX | перевод байта в AL в два символа в 16-ричной c/c |
| BYTE2DEC | перевод байта в AL, в два символа в 10-тичной c/c |
| WORD2HEX | перевод числа в регистре AX, в символьную строку в 16-ричной c/c |
| FORMATRES | вывод символа и его кода |
| MAKEPATH | подготовка полного пути |
| MAIN | головная функция |

**Последовательность действий, выполняемых утилитой.**

Модуль типа .EXE выполняет следующие функции:

1. 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.
2. Вызываемый модуль запускается с использованием загрузчика.
3. 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 взята программа lr2, которая распечатывает среду и командную строку.

**Ход работы.**

1. Запустим отлаженную программу, когда текущим каталогом является каталог с разработанными модулями и введём произвольный символ из числа a-z (см. Рис.1):

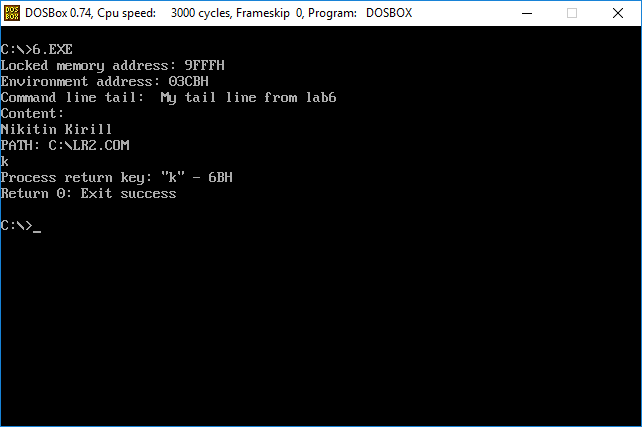


Рисунок 1 – Запуск программы lab6.exe, введенный символ q

1. Запустим программу lab6.exe и введем комбинацию символов Ctrl-C (см. Рис.2):

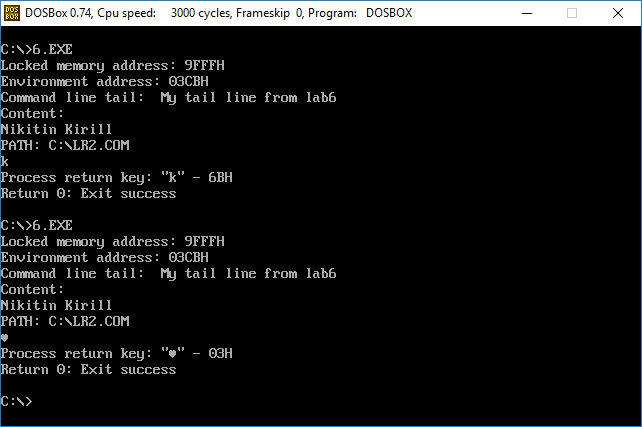


Рисунок 2 – Тестирование ввода комбинации клавиш Ctrl-C (DOSBOX)

Причина нормального завершения заключается в том, что DOSBOX игнорирует Ctrl-C.

1. Запуск программы, когда текущий каталог отличается от каталога, в котором содержатся разработанные программные модули (см. Рис.4):

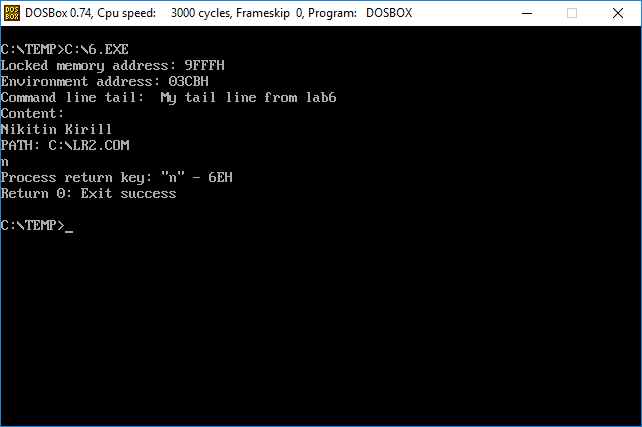


Рисунок 4 – Запуск программы lab6.exe из другого каталога

1. Запуск программы, когда модули находятся в разных каталогах (см. Рис.5):

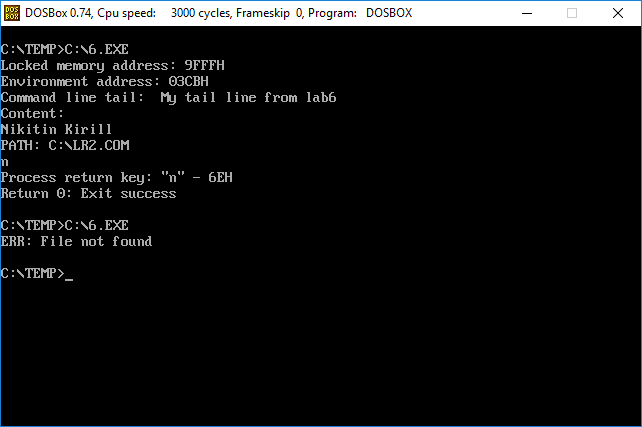


Рисунок 5 – Запуск программы lab6.exe

**Заключение.**

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

**Ответы на контрольные вопросы.**

1. Как реализовано прерывание Ctrl-C?

Вектор прерывания 23h, находящийся по адресу 0000:008Ch, содержит адрес, по которому DOS передает управление после обнаружения нажатия пользователем клавиш Ctrl-C или Ctrl-Break. Обычная системная обработка Ctrl-Break сводится к немедленному снятию программы.

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

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

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

По прерыванию Ctrl-C вызываемая программа заканчивается в точке вызова функции 01h прерывания int 21h.

**ПРИЛОЖЕНИЕ А**

**6.asm**

.286

SSTACK SEGMENT STACK

DW 64H DUP (0)

SSTACK ENDS

ENVIROMENT\_ SEGMENT 'ENVIRONMENT'

DB 'Nikitin Kirill',0

ENVEND DB 0

ENVIROMENT\_ ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:SSTACK, ES:NOTHING

KEEP\_SS DW 0

KEEP\_SP DW 0

KEEP\_ERR DW 0

TETR2HEX PROC NEAR

and AL,0FH

cmp AL,09

jbe NEXT

add AL,07

NEXT: add AL,30H

ret

TETR2HEX ENDP

BYTE2HEX PROC NEAR

push CX

mov AH,AL

call TETR2HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR2HEX

pop CX

ret

BYTE2HEX ENDP

WORD2HEX PROC NEAR

push BX

mov BH,AH

call BYTE2HEX

mov [DI],AH

dec DI

mov [DI],AL

dec DI

mov AL,BH

call BYTE2HEX

mov [DI],AH

dec DI

mov [DI],AL

pop BX

ret

WORD2HEX ENDP

BYTE2DEC PROC NEAR

push CX

push DX

push AX

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\_L

or AL,30H

mov [SI],AL

END\_L: pop AX

pop DX

pop CX

ret

BYTE2DEC ENDP

MAKEPATH MACRO FILENAME,RESULTPATH

push ES

push BX

push DX

push DI

push SI

mov ES,ES:[002CH]

xor BX,BX

ROAD2ZERO:mov DX,ES:[BX]

cmp DX,0000H

jz READPATH

inc BX

jmp ROAD2ZERO

READPATH: add BX,4

mov DI,OFFSET RESULTPATH

COPYPATH: mov DL,ES:[BX]

mov [DI],DL

inc DI

inc BX

cmp DL,00H

jnz COPYPATH

BACKPATH: dec DI

mov DL,[DI]

cmp DL,92

jne BACKPATH

mov SI,OFFSET FILENAME

COPYNAME: inc DI

mov DL,[SI]

mov [DI],DL

inc SI

cmp DL,00H

jnz COPYNAME

pop SI

pop DI

pop DX

pop BX

pop ES

ENDM

FORMATRES PROC NEAR

push SI

push AX

mov SI,OFFSET STR\_CODE\_RETURN+21

mov [SI],AL

call BYTE2HEX

mov SI,OFFSET STR\_CODE\_RETURN+26

mov [SI],AL

mov [SI+1],AH

pop AX

pop SI

ret

FORMATRES ENDP

puts MACRO STRING

push AX

push DX

mov DX,OFFSET STRING

mov AH,09H

int 21H

pop DX

pop AX

ENDM

; Точкавхода

MAIN PROC NEAR

mov AX,DATA

mov DS,AX

mov BX,OFFSET PROGEND

mov AH,4AH

int 21H

jnc PREPARE

cmp AX,07H

jne TESTERR8

puts STR\_ERROR\_7

jmp EXIT

TESTERR8: cmp AX,08H

jne TESTERR9

puts STR\_ERROR\_8

jmp EXIT

TESTERR9: cmp AX,09H

jne UNERRFREE

puts STR\_ERROR\_9

jmp EXIT

UNERRFREE:puts STR\_UNKNOWN\_ERROR

jmp EXIT

PREPARE:

MAKEPATH FILE,FULLPATH

push DS

push ES

mov KEEP\_SS,SS

mov KEEP\_SP,SP

push DS

pop ES

mov DX,OFFSET FULLPATH

mov BX,OFFSET ENV

mov AH,4BH

mov AL,00H

int 21H

mov SS,KEEP\_SS

mov SP,KEEP\_SP

pop ES

pop DS

jc ERROR1

mov AH,4DH

int 21H

call FORMATRES

puts STR\_CODE\_RETURN

cmp AH,00H

jne RET1

puts STR\_RETURN\_0

jmp EXIT

RET1: cmp AH,01H

jne RET2

puts STR\_RETURN\_1

jmp EXIT

RET2: cmp AH,02H

jne RET3

puts STR\_RETURN\_2

jmp EXIT

RET3: cmp AH,03H

jne UNRET

puts STR\_RETURN\_3

jmp EXIT

UNRET: puts STR\_UNKNOWN\_RETURN

jmp EXIT

ERROR1: cmp AX,01H

jne ERROR2

puts STR\_ERROR\_1

jmp EXIT

ERROR2: cmp AX,02H

jne ERROR3

puts STR\_ERROR\_2

jmp EXIT

ERROR3: cmp AX,03H

jne ERROR5

puts STR\_ERROR\_3

jmp EXIT

ERROR5: cmp AX,05H

jne ERROR8

puts STR\_ERROR\_5

jmp EXIT

ERROR8: cmp AX,08H

jne ERRORA

puts STR\_ERROR\_8

jmp EXIT

ERRORA: cmp AX,0AH

jne ERRORB

puts STR\_ERROR\_A

jmp EXIT

ERRORB: cmp AX,0BH

jne UNERR

puts STR\_ERROR\_B

jmp EXIT

UNERR: puts STR\_UNKNOWN\_ERROR

EXIT: xor AL,AL

mov AH,4CH

int 21H

ret

MAIN ENDP

PROGEND:

CODE ENDS

DATA SEGMENT

STR\_SUCCESS DB 'Free: success',0DH,0AH,'$'

STR\_ERROR\_1 DB 'ERR: Invalid function number',0DH,0AH,'$'

STR\_ERROR\_2 DB 'ERR: File not found',0DH,0AH,'$'

STR\_ERROR\_3 DB 'ERR: Path not found',0DH,0AH,'$'

STR\_ERROR\_5 DB 'ERR: Access denied',0DH,0AH,'$'

STR\_ERROR\_7 DB 'ERR: Memory control blocks destroyed',0DH,0AH,'$'

STR\_ERROR\_8 DB 'ERR: Insufficient memory',0DH,0AH,'$'

STR\_ERROR\_9 DB 'ERR: Invalid memory block address',0DH,0AH,'$'

STR\_ERROR\_A DB 'ERR: Invalid environment',0DH,0AH,'$'

STR\_ERROR\_B DB 'ERR: Invalid format',0DH,0AH,'$'

STR\_UNKNOWN\_ERROR DB 'ERR: Unknown',0DH,0AH,'$'

STR\_RETURN\_0 DB 'Return 0: Exit success',0DH,0AH,'$'

STR\_RETURN\_1 DB 'Return 1: Ctrl-Break',0DH,0AH,'$'

STR\_RETURN\_2 DB 'Return 2: Device error',0DH,0AH,'$'

STR\_RETURN\_3 DB 'Return 3: 31H exit',0DH,0AH,'$'

STR\_UNKNOWN\_RETURN DB 'Return ?: unknown',0DH,0AH,'$'

STR\_CODE\_RETURN DB 'Process return key: " " - H',0DH,0AH,'$'

FILE DB 'LR2.COM',0

FULLPATH DB 128 DUP (0)

CMDLINE DB 24," My tail line from lab6",0DH

ENV DW ENVIROMENT\_

DW OFFSET CMDLINE

DW SEG CMDLINE

DD 0

DD 0

DATA ENDS

END MAIN