**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. 6381 |  | Поляков Н.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы.**

Исследование возможности построения загрузочного модуля динамической структуры и интерфейса между вызывающей и вызываемой программами.

**Описание функций.**

|  |  |
| --- | --- |
| Имя | Описание |
| PRINT | Вызывает функцию 9 прерывания 21h |
| TETR\_TO\_HEX | Осуществляет перевод половины байта в AL в символ в 16-ричной с/c и помещает результат в AL |
| BYTE\_TO\_HEX | Осуществляет перевод байта в AL, в 16-тичную c/c и помещает результат по адресу DI |
| FREE\_MEM | Освобождает память, неиспользуемую программой |
| CREATE\_BLOCK | Заполняет блок параметров нужными значениями |
| RUN\_PROG | Запускает программу и обрабатывает ошибки |

**Описание структур данных.**

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| KEEP\_SS | dw | Сохраненное значение регистра ss |
| KEEP\_SP | dw | Сохраненное значение регистра sp |
| ERROR\_FREEING | db | Строка 'Error when freeing memory: $' |
| ERROR\_MCM | db | Строка 'MCB is destroyed$' |
| ERROR\_NO\_MEM | db | Строка 'Not enough memory for function processing$' |
| ERROR\_WRONG\_ADDR | db | Строка 'Wrong addres of memory block$' |
| ERROR\_NUKNOWN | db | Строка 'Unknown error$' |
| ERROR\_WRONG\_NUM | db | Строка 'Function number is wrong$' |
| ERROR\_FILE\_NOT\_FOUND | db | Строка 'File is not found$' |
| ERROR\_DISK | db | Строка 'Disk error$' |
| ERROR\_NO\_MEM2 | db | Строка 'Not enough memory$' |
| ERROR\_WRONG\_ENV | db | Строка 'Wrong environment string$' |
| ERROR\_WRONG\_FORMAT | db | Строка 'Wrong format$' |
| END\_NORM | db | Строка 'Normal end$' |
| END\_CTRL\_С | db | Строка 'End by Ctrl-C$' |
| END\_ERROR | db | Строка 'End by device error$' |
| END\_31h | db | Строка 'End by 31h function$' |
| END\_NUKNOWN | db | Строка 'End by unknown reason$' |
| END\_CODE | db | Строка 'End code: $' |
| STRENDL | db | Строка с символами перевода строки |
| PROG\_PATH | db | 'LAB2.COM',0 |
| PAR\_BLOCK | dw dd dd dd | Блок параметров |

**Тестирование.**

1. Запуск в случае, когда обе программы находятся в одной директории.

****

Рис.1 Результат работы программы lab6.exe.

1. Выход из вызываемой программы при помощи Ctrl+C.

****

Рис.2 Результат работы программы lab6.exe при нажатии Ctrl+C.

1. Запуск программы из другой директории.



Рис.3 Запуск программы lab6.exe из другой директории.



Рис.4 Запуск программы lab6.exe из другой директории при нажатии Ctrl+C.

1. Запуск программы в случае, когда lab6.exe и lab2.com находятся в разных директориях.

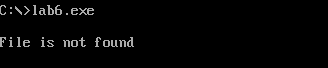
****

Рис.5 Запуск программы lab6.exe c вызываемой программой в другой директории.

**Ответы на контрольные вопросы.**

1. Как реализовано прерывание Ctrl-C?

Ответ: Функции ввода проверяют наличие в буфере клавиатуры кода 03, который помещается туда при нажатии этой комбинации клавиш. В случае его там присутствия, вызывается прерывание int 23h и программа завершает работу с кодом 03.

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

Ответ: В месте вызова функции 4Ch прерывания int 21h.

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

Ответ: в месте вызова функции 01h прерывания int 21h.

**Вывод.**

В ходе выполнения лабораторной работы был изучен интерфейс взаимодействия вызываемой и вызывающей программ и разработана программа, вызывающая другую программу.

**Приложение 1. Код программы.**

**lab6.asm**

DATA SEGMENT

ERROR\_FREEING db 'Error when freeing memory: $'

ERROR\_MCM db 'MCB is destroyed$'

ERROR\_NO\_MEM db 'Not enough memory for function processing$'

ERROR\_WRONG\_ADDR db 'Wrong addres of memory block$'

ERROR\_NUKNOWN db 'Unknown error$'

ERROR\_WRONG\_NUM db 'Function number is wrong$'

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

ERROR\_DISK db 'Disk error$'

ERROR\_NO\_MEM2 db 'Not enough memory$'

ERROR\_WRONG\_ENV db 'Wrong environment string$'

ERROR\_WRONG\_FORMAT db 'Wrong format$'

END\_NORM db 'Normal end$'

END\_CTRL\_С db 'End by Ctrl-C$'

END\_ERROR db 'End by device error$'

END\_31h db 'End by 31h function$'

END\_NUKNOWN db 'End by unknown reason$'

END\_CODE db 'End code: $'

STRENDL db 13,10,'$'

PAR\_BLOCK dw 0

dd ?

dd 0

dd 0

PROG\_PATH db 20h dup(0)

KEEP\_SS dw 0

KEEP\_SP dw 0

DATA ENDS

CODE SEGMENT

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

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

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

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

FREE\_MEM PROC

mov ax,STACKSEG

mov bx,es

sub ax,bx

add ax,10h

mov bx,ax

mov ah,4Ah

int 21h

jnc MEM\_FREED

mov dx,offset ERROR\_FREEING

call PRINT

cmp ax,7

mov dx,offset ERROR\_MCM

je FREE\_MEM\_PRINT\_ERROR

cmp ax,8

mov dx,offset ERROR\_NO\_MEM

je FREE\_MEM\_PRINT\_ERROR

cmp ax,9

mov dx,offset ERROR\_WRONG\_ADDR

FREE\_MEM\_PRINT\_ERROR:

call PRINT

mov dx,offset STRENDL

call PRINT

xor AL,AL

mov AH,4Ch

int 21H

MEM\_FREED:

ret

FREE\_MEM ENDP

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

CREATE\_BLOCK PROC

mov ax, es:[2Ch]

mov PAR\_BLOCK,ax

mov PAR\_BLOCK+2,es

mov PAR\_BLOCK+4,80h

ret

CREATE\_BLOCK ENDP

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

RUN\_PROG PROC

mov dx,offset STRENDL

call PRINT

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,PROG\_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 'c'

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

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

mov dx,offset PROG\_PATH

mov KEEP\_SP, sp

mov KEEP\_SS, ss

mov ax,ds

mov es,ax

mov bx,offset PAR\_BLOCK

mov ax,4b00h

int 21h

jnc NO\_ERRORS

push ax

mov ax,DATA

mov ds,ax

pop ax

mov SS,KEEP\_SS

mov SP,KEEP\_SP

cmp ax,1

mov dx,offset ERROR\_WRONG\_NUM

je PRINT\_ERROR

cmp ax,2

mov dx,offset ERROR\_FILE\_NOT\_FOUND

je PRINT\_ERROR

cmp ax,5

mov dx,offset ERROR\_DISK

je PRINT\_ERROR

cmp ax,8

mov dx,offset ERROR\_NO\_MEM2

je PRINT\_ERROR

cmp ax,10

mov dx,offset ERROR\_WRONG\_ENV

je PRINT\_ERROR

cmp ax,11

mov dx,offset ERROR\_WRONG\_FORMAT

je PRINT\_ERROR

mov dx,offset ERROR\_NUKNOWN

PRINT\_ERROR:

call PRINT

mov dx,offset STRENDL

call PRINT

xor al,al

mov ah,4Ch

int 21H

NO\_ERRORS:

mov ax,4d00h

int 21h

cmp ah,0

mov dx,offset END\_NORM

je PRINT\_NORMAL\_END

cmp ah,1

mov dx,offset END\_CTRL\_С

je PRINT\_NORMAL\_END

cmp ah,2

mov dx,offset END\_ERROR

je PRINT\_NORMAL\_END

cmp ah,3

mov dx,offset END\_31h

je PRINT\_NORMAL\_END

mov dx,offset END\_NUKNOWN

PRINT\_NORMAL\_END:

call PRINT

mov dx,offset STRENDL

call PRINT

mov dx,offset END\_CODE

call PRINT

call BYTE\_TO\_HEX

push ax

mov ah,2

mov dl,al

int 21h

pop ax

xchg ah,al

mov ah,2

mov dl,al

int 21h

mov dx,offset STRENDL

call PRINT

ret

RUN\_PROG ENDP

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

MAIN PROC FAR

mov ax,data

mov ds,ax

call FREE\_MEM

call CREATE\_BLOCK

call RUN\_PROG

xor al,al

mov ah,4Ch

int 21H

ret

MAIN ENDP

CODE ENDS

STACKSEG SEGMENT STACK

dw 80h dup (?)

STACKSEG ENDS

END MAIN