**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Операционные системы»**

Тема: **Исследование интерфейсов программных модулей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Никитин К.В. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

**Постановка задачи.**

Исследование интерфейса управляющей программы и загрузочных модулей.

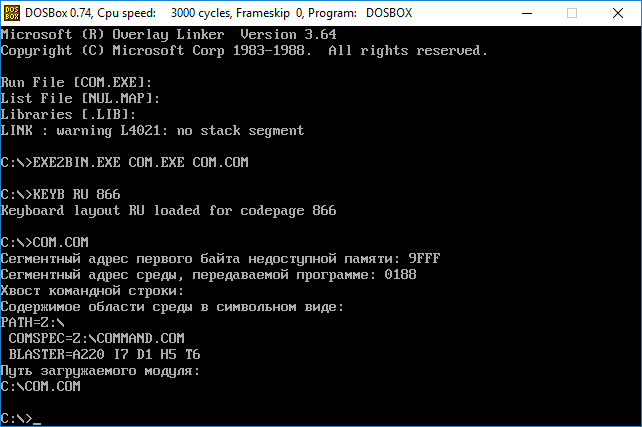
Описание функций и структур данных:

|  |  |
| --- | --- |
| **Название** | **Назначение** |
| WRITEMSG | Вывод сообщения на экран |
| TETR\_TO\_HEX | Десятичная цифра переводится в код символа |
| BYTE\_TO\_HEX | Байт в AL переводится в два символа шестн. числа в AX |
| WRD\_TO\_HEX | Перевод в 16 с/с 16-ти разрядного числа |
| BYTE\_TO\_DEC | Перевод байта в 16-ной с/с в символьный код в 10-ной с/с |
| GET\_ADRESS\_DEFIECY\_MEM | Получает адрес недопустимой памяти |
| GET\_SEG\_ADRESS\_ENV | Получает сегментный адрес среды |
| TAIL | Получает окончание командной строки |
| ENV | Получает содержимое среды в символьном виде |

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Назначение** |
| DEFIECY\_MEM\_MSG | db | Выводит адрес первого байта недопустимой памяти |
| DEFIECY\_MEM | db | Хранение адреса недопустимой памяти |
| SEG\_ADRESS\_ENV\_MSG | db | Выводит сегментный адрес среды |
| SEG\_ADRESS\_ENV | db | Хранение сегментного адреса среды |
| TAIL\_ | db | Выводит окончание командной строки |
| ENV\_ | db | Выводит содержимое области среды в символьном виде |
| PATH\_ | db | Выводит загружаемого модуля |

Последовательность действий, выполняемых утилитой:

1. Печатает сегментный адрес первого байта недоступной памяти
2. Печатает сегментный адрес среды, передаваемой программе
3. Печатает хвост командной строки
4. Печатает содержимое области среды в символьном виде
5. Печатает путь загружаемого модуля
6. Выходит в DOS



*Рисунок 1. Результат выполнения программы com.com*

**Заключение.**

В процессе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей.

**Ответы на контрольные вопросы.**

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на область памяти, в которой нельзя выделять память.

1. Где расположен этот адрес по отношению области памяти, отведенной программе?

Этот адрес расположен после области памяти, отведенной программе.

1. Можно ли в эту область памяти писать?

Можно, в MS-DOS нет защиты памяти.

Среда, передаваемая программе:

1. Что такое среда?

Среда представляет собой область памяти, в которой в виде символьных строк записаны значения переменных среды.

1. Когда создается среда? Перед запуском приложения или в другое время?

Среда загружается в память при загрузке ОС, при запуске приложения копируется из родительского процесса.

1. Откуда берется информация, записываемая в среду?

В MS-DOS она берется из файла autoexec.bat, ее копии могут дополняться новыми данными.

# Приложение А.

# сom.asm

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

DEFIECY\_MEM\_MSG db 'Сегментный адрес первого байта недоступной памяти: '

DEFIECY\_MEM db ' ',0DH,0AH,'$'

SEG\_ADRESS\_ENV\_MSG db 'Сегментный адрес среды, передаваемой программе: '

SEG\_ADRESS\_ENV db ' ',0DH,0AH,'$'

TAIL\_ db 'Хвост командной строки: ',0DH,0AH,'$'

ENV\_ db 'Содержимое области среды в символьном виде: ',0DH,0AH,'$'

PATH\_ db 'Путь загружаемого модуля: ',0DH,0AH,'$'

ENDL db 0DH,0AH,'$'

WRITEMSG PROC

push ax

mov ah,09h

int 21h

pop ax

ret

WRITEMSG ENDP

GET\_ADRESS\_DEFIECY\_MEM PROC

mov ax,es:[2]

mov di,offset DEFIECY\_MEM+3

call WRD\_TO\_HEX

lea dx,DEFIECY\_MEM\_MSG

call WRITEMSG

ret

GET\_ADRESS\_DEFIECY\_MEM ENDP

GET\_SEG\_ADRESS\_ENV PROC

mov ax,es:[2Ch]

mov di,offset SEG\_ADRESS\_ENV+3

call WRD\_TO\_HEX

lea dx,SEG\_ADRESS\_ENV\_MSG

call WRITEMSG

ret

GET\_SEG\_ADRESS\_ENV ENDP

TAIL PROC

mov dx,offset TAIL\_

call WRITEMSG

mov cx,0

mov cl,es:[80h]

cmp cl,0

je TAIL\_END

mov dx,81h

mov bx,0

mov ah,02h

TAIL\_loop:

mov dl,es:[bx+81h]

int 21h

inc bx

loop TAIL\_loop

mov dx,offset ENDL

call WRITEMSG

TAIL\_END:

ret

TAIL ENDP

ENV PROC

mov dx,offset ENV\_

call WRITEMSG

push es

mov ax,es:[2Ch]

mov es,ax

mov ah,02h

mov bx,0

ENV\_loop:

mov dl,es:[bx]

int 21h

inc bx

cmp byte ptr es:[bx],00h

jne ENV\_loop

mov dx,offset ENDL

call WRITEMSG

cmp word ptr es:[bx],0000h

jne ENV\_loop

add bx,4

mov dx,offset PATH\_

call WRITEMSG

ENV\_loop2:

mov dl,es:[bx]

int 21h

inc bx

cmp byte ptr es:[bx],00h

jne ENV\_loop2

mov dx,offset ENDL

call WRITEMSG

pop es

ret

ENV 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

WRD\_TO\_HEX PROC near

push BX

mov BH,AH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

dec DI

mov AL,BH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

pop BX

ret

WRD\_TO\_HEX ENDP

BYTE\_TO\_DEC PROC near

push CX

push DX

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 DX

pop CX

ret

BYTE\_TO\_DEC ENDP

BEGIN:

call GET\_ADRESS\_DEFIECY\_MEM

call GET\_SEG\_ADRESS\_ENV

call TAIL

call ENV

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START