**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 |  | Быков И. В. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

# Постановка задачи.

* 1. **Цель работы.**

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системный данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

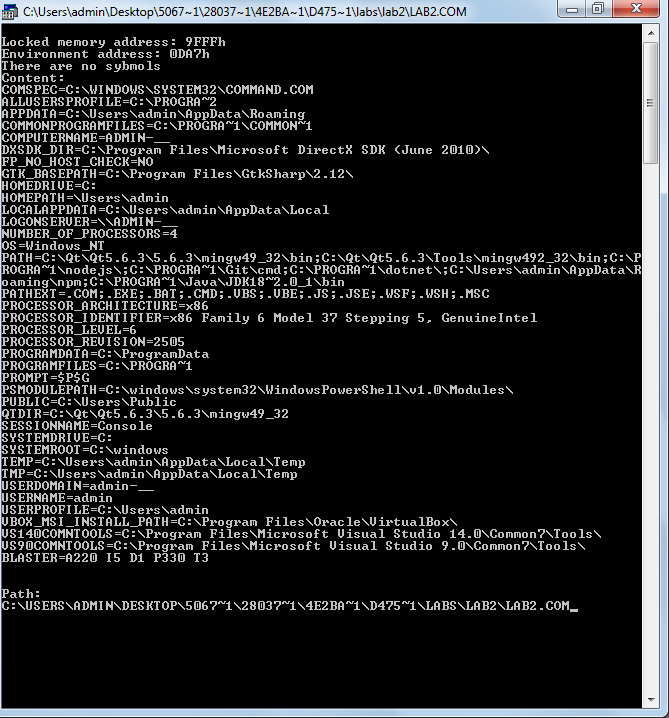
* 1. **Функции и структуры данных.**

|  |  |
| --- | --- |
| Название процедуры (функции) | Описание процедуры (функции) |
| WRITE\_MSG | 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 |
| INFO | Вывод на экран:   * Сегментный адрес недоступной памяти * Сегментный адрес среды * Хвост командной строки * Содержимое области среды * Путь загружаемого модуля |

* 1. **Последовательность действий, выполняемых утилитой.**

1. Определение и вывод сегментного адреса недоступной памяти
2. Определение и вывод сегментного адреса среды
3. Определение и вывод хвоста командной строки
4. Определение и вывод содержимого области среды
5. Определение и вывод пути загружаемого модуля

Результат работы программы представлен на рис. 1.



*Рисунок 1. Результат работы программы lab2.com*

В первой строке указан сегментный адрес недоступной памяти: 9FFFh, во второй строке указан сегментный адрес среды: 0DA7h, третья строка говорит о том, что программа запускалась без хвоста. Последняя строка – это путь загружаемого модуля.

# Вывод.

В результате выполнения данной работы были исследованы интерфейс управляющей программы и загрузочных модулей, исследованы префикс сегмента программы (PSP) и среды, передаваемой программе. Исходный код полученной программы находится в Приложении А.

# Ответы на вопросы.

* 1. **Сегментный адрес недоступной памяти.**
     1. *На какую область памяти указывает адрес недоступной памяти?*
  + На область оперативной памяти.
    1. *Где расположен этот адрес по отношению области памяти, отведённой программе?*
  + Этот адрес расположен сразу за концом памяти, которая отводится программе.
    1. *Можно ли в эту область памяти писать?*
  + Можно, если отсутствует защита памяти (как в DOS)
  1. **Среда, передаваемая программе.**
     1. *Что такое среда?*
  + Среда – набор переменных, в которых хранятся настройки ОС.
    1. *Когда создаётся среда? Перед запуском приложения или в другое время?*
  + Среда создается при загрузке DOS. А при запуске программы эта среда копируется в новую область памяти.
    1. *Откуда берётся информация, записываемая в среду?*
  + Из реестра ОС.

# Приложение А. lab2.asm

TESTPC SEGMENT

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

org 100h

START: JMP BEGIN

; DATA

MEMORY\_ db 13, 10, "Locked memory address: h$" ; 17 symbols

ENVIRONMENT\_ db 13, 10, "Environment address: h$" ; 23 symbols

TAIL\_ db 13, 10, "Command line tail: $" ; 21 symbols

EMPTY\_ db 13, 10, "There are no sybmols$"

CONTENT\_ db 13, 10, "Content:", 13, 10, "$"

ENT\_ db 13, 10, "$"

PATH\_ db 13, 10, "Path:", 13, 10, "$" ; 8 symbols

;PROCEDURES

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

WRITE\_MSG PROC near

push ax

mov ah, 09h

int 21h

pop ax

ret

WRITE\_MSG ENDP

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

INFO PROC near

; MEMORY

mov ax, ds:[02h]

mov di, offset MEMORY\_

add di, 28

call WRD\_TO\_HEX

mov dx, offset MEMORY\_

call WRITE\_MSG

; ENVIRONMENT

mov ax, ds:[2Ch]

mov di, offset ENVIRONMENT\_

add di, 26

call WRD\_TO\_HEX

mov dx, offset ENVIRONMENT\_

call WRITE\_MSG

; TAIL

xor cx, cx

mov cl, ds:[80h]

mov si, offset TAIL\_

add si, 20

test cl, cl

jz empty

xor di, di

xor ax, ax

readtail:

mov al, ds:[81h+di]

mov [si], al

inc di

inc si

loop readtail

mov dx, offset TAIL\_

call WRITE\_MSG

jmp nextaction

empty:

mov dx, offset EMPTY\_

call WRITE\_MSG

nextaction: nop

; ENVIRONMENT CONTENT

mov dx, offset CONTENT\_

call WRITE\_MSG

xor di, di

mov bx, 2Ch

mov ds, [bx]

readstring:

cmp byte ptr [di], 00h

jz pressenter

mov dl, [di]

mov ah, 02h

int 21h

jmp findend

pressenter:

push ds

mov cx, cs

mov ds, cx

mov dx, offset ENT\_

call WRITE\_MSG

pop ds

findend:

inc di

cmp word ptr [di], 0001h

jz readpath

jmp readstring

readpath:

push ds

mov ax, cs

mov ds, ax

mov dx, offset PATH\_

call WRITE\_MSG

pop ds

add di, 2

pathloop:

cmp byte ptr [di], 00h

jz final

mov dl, [di]

mov ah, 02h

int 21h

inc di

jmp pathloop

final:

ret

INFO 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

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

; CODE

BEGIN:

call INFO

mov ah, 10h

int 16h

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START