**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Операционные системы»**

Тема: «Исследование интерфейсов программных модулей»

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Попов Н.В. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи.**

Цель работы: Исследование интерфейса управляющей программы и загрузочных модулей, префикса сегмента программы (PSP) и среды, передаваемой программе.

Таблица переменных

|  |  |  |
| --- | --- | --- |
| Переменные | Тип хранимых данных | Назначение |
| SegAdrUntouchMem | DB | Вывод строки 'Segment address of untouchable memory - ',10,13,'$' |
| SegAdrEnvir | DB | Вывод строки 'Segment address of environment - ',10,13,'$' |
| TailComStr | DB | Вывод строки 'Tail of command string - $' |
| EndStr | DB | Вывод строки 10,13,'$' |
| ContEnvAr | DB | Вывод строки 'Contents of the environment area - $' |
| WayMod | DB | Вывод строки 'Way of module - $' |

Описание функций и структур данных:

1. CHECK\_ASUM: выводит сегментный адрес недопустимой памяти, взятый из PSP, в 16 с/с.
2. CHECK\_SAE: выводит сегментный адрес среды, передаваемой программе, в 16 с/с.
3. CHECK\_TCS: выводит хвост командной строки в символьном виде.
4. CHECK\_CEA\_WM: выводит содержимое области среды в символьном виде и путь загружаемого модуля.
5. TETR\_TO\_HEX: вспомогательная функция для работы функции BYTE\_TO\_HEX.
6. BYTE\_TO\_HEX: переводит число AL в коды символов 16-ой с/с, записывая получившееся в bl и bh.
7. WRD\_TO\_HEX: переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры.
8. PRINT\_STR: выводит строку, помещенную в DX.

Последовательность действий, выполняемых утилитой:

1. Определение и печать сегментного адреса недостаточной памяти, взятого из PSP, в 16 с/с с помощью функции CHECK\_ASUM.
2. Определение и печать сегментного адреса среды, передаваемого программе, в 16 с/c с помощью функции CHECK\_SAE.
3. Определение и печать хвоста командной строки в символьном виде с помощью функции CHECK\_TCS.
4. Определение и печать содержимого области среды в символьном виде с помощью функции CHECK\_CEA\_WM.

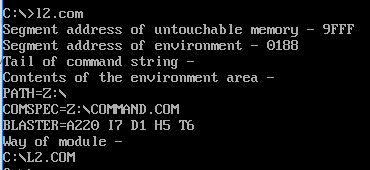


Рисунок 1. Работа программы L2.COM.

**Заключение.**

В процессе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей, а также префикс сегмента программы (PSP) и среды, передаваемой программе.

**Ответы на контрольные вопросы.**

**Сегментный адрес недоступной памяти**

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на адрес окончания основной оперативной памяти.

Оперативная память делится на основную и дополнительную память.

Основная, т. е. обычная, память, содержащаяся в первых 640 Кб оперативной памяти, содержит векторы прерываний, область данных BIOS и DOS, операционная система MS-DOS, загружаемые драйверы DOS, резидентная часть COMMAND.COM и свободная память для загружаемых прикладных и системых программ.

Дополнительная память в свою очередь включает верхнюю память (384 Кб) и расширенную память. В верхней памяти содержится такая информация, как графическая видеопамять, свободные адреса, текстовая видеозапись, ПЗУ расширений BIOS. Доступ к расширенной памяти осуществляется в защищенном режиме, поэтому для MS-DOC, работающей в реальном режиме, расширенная память недоступна. Несмотря на это, к первым 64 Кб расширенной памяти (область старшей памяти) можно обратиться с помощью драйвера HIMEM.SYS.

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

Адрес расположен за областью памяти, отведенной программе.

1. Можно ли эту в эту область памяти писать?

В этой области памяти хранятся такие данные, как, например, память, используемая видеоадаптером и ROM BIOS. Сюда можно писать, т. к. в MS-DOS отсутствует защита памяти.

**Среда, передаваемая программе**

1. Что такое среда?

Среда - это область памяти, содержащая переменные среды, записанные в виде строк "имя=параметр".

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

Среда создается при загрузке в DOS.

1. Откуда берется информация, записываемая в среду?

Информация, записываемая в среду, копируется из родительской программы.

**Приложение А**

**Исходный код L2.asm**

TESTPC SEGMENT

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

ORG 100H

START:

JMP BEGIN

SegAdrUntouchMem DB 'Segment address of untouchable memory - ',10,13,'$'

SegAdrEnvir DB 'Segment address of environment - ',10,13,'$'

TailComStr DB 'Tail of command string - $'

EndStr DB 10,13,'$'

ContEnvAr DB 'Contents of the environment area - $'

WayMod DB 'Way of module - $'

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

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

;байт в AL переводится в два символа шестн. числа в AX

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR\_TO\_HEX ; в AL старшая цифра

pop CX ; в AH младшая

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

;перевод в 16 с/с 16-ти разрядного числа

;AX - число, DI - адрес последего символа

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

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

PRINT\_STR PROC near ;Печать строки, помещенной в DX

mov ah,09h

int 21h

ret

PRINT\_STR ENDP

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

CHECK\_ASUM PROC near

mov ax,ds:[02h]

mov di,offset SegAdrUntouchMem+43

call WRD\_TO\_HEX

mov dx,offset SegAdrUntouchMem

call PRINT\_STR

ret

CHECK\_ASUM ENDP

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

CHECK\_SAE PROC near

mov ax,ds:[02ch]

mov di,offset SegAdrEnvir+36

call WRD\_TO\_HEX

mov dx,offset SegAdrEnvir

call PRINT\_STR

ret

CHECK\_SAE ENDP

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

CHECK\_TCS PROC near

mov dx,offset TailComStr

call PRINT\_STR

mov cl,ds:[080h]

cmp cl,0

je empty

mov ah,02h

xor di,di

newSymb:

mov dl,ds:[081h + di]

int 21h

inc di

loop newSymb

empty:

mov dx,offset EndStr

call PRINT\_STR

ret

CHECK\_TCS ENDP

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

CHECK\_CEA\_WM PROC near

lea dx,ContEnvAr

call PRINT\_STR

mov bx,1 ;checker

mov es,es:[2ch]

mov si,0

nextEl:

lea dx,EndStr

call PRINT\_STR

mov ax,si

endNotFound:

cmp byte ptr es:[si], 0

je endElemArea

inc si

jmp endNotFound

endElemArea:

push es:[si]

mov byte ptr es:[si], '$'

push ds

mov cx,es

mov ds,cx

mov dx,ax

call PRINT\_STR

pop ds

pop es:[si]

cmp bx,0

je final

inc si

cmp byte ptr es:[si], 01h

jne nextEl

lea dx,WayMod

call PRINT\_STR

mov bx,0

add si,2

jmp nextEl

final:

ret

CHECK\_CEA\_WM ENDP

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

BEGIN:

call CHECK\_ASUM

call CHECK\_SAE

call CHECK\_TCS

call CHECK\_CEA\_WM

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START