**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. 7381 | Минуллин М.А. |
| Преподаватель | Ефремов М.А. |

Санкт-Петербург

2019

**Цель работы.**

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

**Необходимые сведения для составления программы.**

При начальной загрузке программы формируется PSP, который размещается в начале первого сегмента программы. PSP занимает 256 байт и располагается с адреса, кратного границе сегмента. При загрузке модулей типа .COM все сегментные регистры указывают на адрес PSP. При загрузке типа .EXE сегментные регистры DS и ES указывают на PSP. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять. Формат PSP представлен в табл. 1.

Таблица 1 – формат PSP.

|  |  |  |
| --- | --- | --- |
| Смещение | Длина поля (байты) | Содержимое поля |
| 0 | 2 | int 20h |
| 2 | 2 | Сегментный адрес первого байта недоступной памяти. Программа не должна модифицировать содержимое памяти этим адресом. |
| 4 | 6 | Зарезервировано |
| 10 | 4 | Вектор прерывания 22h |
| 14 | 4 | Вектор прерывания 23h |
| 18 | 4 | Вектор прерывания 24h |
| 44 | 2 | Сегментный адрес среды, передаваемый программе. |
| 5Ch | 16 | Область форматируется как стандартный неоткрытый бок управления файлом (FCB). |
| 6Ch | 20 | Область форматируется как стандартный неоткрытый бок управления файлом(FCB). Перекрывается, если FCB с адреса 5Ch открыт. |
| 80h | 1 | Число символов в хвосте командной строки. |
| 81h |  | Хвост командной строки – последовательность символов после имени вызываемого модуля. |

Область среды содержит последовательность строк вида: имя=параметр. Каждая строка завершается байтом нулём.

В первой строке указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM. Следующие строки содержат информации, задаваемую командами PATH, PROMPT, SET.

Среда заканчивается также байтом нулём. Таким образом, два нулевых байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

**Ход работы.**

Был написан и отлажен программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1. Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.
2. Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
3. Хвост командной строки в символьном виде.
4. Содержимое области среды в символьном виде.
5. Путь загружаемого модуля.

Результаты работы программы представлены на рис. 1.

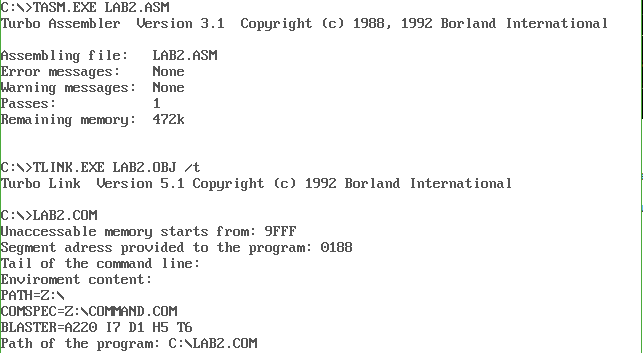


Рисунок 1 – результат работы программы.

**Сегментный адрес недоступной памяти.**

В: На какую область памяти указывает адрес недоступной памяти?

О: Адрес недоступной памяти указывается на область памяти, которую нельзя модифицировать.

В: Где расположен этот адрес по отношению области памяти, отведённой программе?

О: Сегмент находится непосредственно после памяти, выделенной программе.

В: Можно ли в эту область памяти писать?

О: Нет, нельзя.

**Среда, передаваемая программе.**

В: Что такое среда?

О: Среда – это совокупность переменных среды, которые могут использоваться приложениями для получения некоторой системной информации и для передачи данных между программами. Переменная среды – символьная строка в коде ACII вида имя=параметр.

В: Когда создаётся среда? Перед запуском приложения или в другое время?

О: При загрузке программы.

В: Откуда берётся информацию, записываемая в среду?

О: Блок окружения наследуется от программы-родителя.

**Выводы.**

В ходе выполнения лабораторной работы был написан .COM модуль, который извлекает информацию о загрузочном модуле и среде из PSP и выводит её на экран. Исследован интерфейс управляющей программы, а также изучены особенности такого понятия как среда.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ ТЕКСТ .COM МОДУЛЯ**

CSEG    SEGMENT

assume  CS:CSEG, DS:CSEG, ES:NOTHING, SS:NOTHING

ORG 100H

START:  jmp BEGIN

str1 db 'Unaccessable memory starts from: ', '$'

mem\_addr db 5 dup(?)

str2 db 'Segment adress provided to the program: ', '$'

seg\_addr db 5 dup(?)

tail\_msg db 'Tail of the command line: ', '$'

tail db 50 dup(?)

content\_msg db 'Enviroment content: ', 0Dh, 0Ah, '$'

content db 256 dup(?)

path\_msg db 'Path of the program: ', '$'

path db 50 dup(?)

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

WRITE proc near

    mov cx, 2

cycle:

    xchg al, ah

    push ax

    call BYTE\_TO\_HEX

    mov [si], al

    inc si

    mov [si], ah

    inc si

    pop ax

    loop cycle

    mov byte ptr [si], '$'

    ret

WRITE endp

WRT\_CONTENT proc near

    mov bx, 2Ch

    mov ax, es:bx

    mov es, ax

    xor si, si

search:

    cmp byte ptr [es:si], 0

    jne wrt

    mov byte ptr [di], 0Dh

    mov byte ptr [di + 1], 0Ah

    add di, 2

    inc si

    cmp byte ptr [es:si], 0

    je end\_table

wrt:

    mov al, [es:si]

    mov [di], al

    inc di

    inc si

    jmp search

end\_table:

    add si, 3

    mov byte ptr [di], '$'

    mov di, offset path

wrt\_path:

    cmp byte ptr [es:si], 0

    je done

    mov bl, [es:si]

    mov [di], bl

    inc di

    inc si

    jmp wrt\_path

done:

    mov byte ptr [di], 0Dh

    mov byte ptr [di + 1], 0Ah

    mov byte ptr [di + 2], '$'

    ret

WRT\_CONTENT endp

BEGIN:

    mov bx, 2

    mov ax, es:bx

    mov si, offset mem\_addr

    call WRITE

    mov ah, 09h

    mov dx, offset str1

    int 21h

    mov dx, offset mem\_addr

    int 21h

    mov dl, 0Dh

    mov ah, 02h

    int 21h

    mov dl, 0Ah

    int 21h

    mov bx, 2Ch

    mov ax, es:bx

    mov si, offset seg\_addr

    call WRITE

    mov ah, 09h

    mov dx, offset str2

    int 21h

    mov dx, offset seg\_addr

    int 21h

    mov dl, 0Dh

    mov ah, 02h

    int 21h

    mov dl, 0Ah

    int 21h

    mov bx, 80h

    xor ch, ch

    mov cl, es:bx

    mov bx, 81h

    add bx, cx

    dec bx

    mov si, offset tail

    add si, cx

    mov byte ptr [si], '$'

    dec si

wrt\_tail:

    cmp cx, 0

    je skip

    mov ah, es:bx

    mov [si], ah

    dec si

    dec bx

    dec cx

    jmp wrt\_tail

skip:

    mov dx, offset tail\_msg

    mov ah, 09h

    int 21h

    mov dx, offset tail

    int 21h

    mov dl, 0Dh

    mov ah, 02h

    int 21h

    mov dl, 0Ah

    int 21h

    mov di, offset content

    call WRT\_CONTENT

    mov dx, offset content\_msg

    mov ah, 09h

    int 21h

    mov dx, offset content

    int 21h

    mov dx, offset path\_msg

    int 21h

    mov dx, offset path

    int 21h

    xor  al, al

    mov  ah, 4Ch

    int  21h

CSEG    ends

end START