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

Санкт-Петербург

2018

**Цель работы.**

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче 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. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять.

Формат PSP:

|  |  |  |
| --- | --- | --- |
| Смещение | Длина поля (байт) | Содержимое поля |
| 0 | 2 | int 20h |
| 2 | 2 | Сегментный адрес первого байта недоступной памяти. Программа не должна модифицировать содержимое памяти за этим адресом. |
| 4 | 6 | Зарезервировано |
| 0Ah(10) | 4 | Вектор прерывания 22h (IP, CS) |
| 0Eh(14) | 4 | Вектор прерывания 23h (IP, CS) |
| 12h(18) | 4 | Вектор прерывания 24h (IP, CS) |
| 2Ch(44) | 2 | Сегментный адрес среды, передаваемой программе. |
| 5Ch |  | Область форматируется как стандартный неоткрытый блок управления файлом (FCB) |
| 6Ch |  | Область форматируется как стандартный неоткрытый блок 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 символьных строк вида:

*имя=параметр*

Каждая строка завершается байтом нулей.

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

Среда заканчивается также байтом нулей. 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.

**Описание функций.**

|  |  |
| --- | --- |
| Имя | Описание |
| TETR\_TO\_HEX | Осуществляет перевод половины байта в AL в два символа в 16-ричной с/c и помещает результат в AL |
| BYTE\_TO\_HEX | Осуществляет перевод байта в AL в 16-ричную c/c и помещает результат по адресу DI |
| WRD\_TO\_HEX | Осуществляет перевод слова в AX в 16-ричную c/c и помещает результат по адресу DI |
| BYTE\_TO\_DEC | **О**существляет перевод байта в AL, в 10-тичную c/c и помещает результат в SI |
| FIND\_ADDR\_MEM | **О**пределяет сегментный адрес недоступной памяти и выводит его на экран |
| FIND\_ADDR\_ENV | Определяет сегментный адрес среды, передаваемой программе и выводит их на экран |
| FIND\_TAIL | Определяет хвост командной строки и выводит их на экран |
| FIND\_ENV\_AND\_PATH | **О**пределяет содержимое области среды и путь загружаемого модуля и выводит их на экран |

**Описание структур данных.**

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| ADDR\_MEM | db | Строка “Address of inaccessible memory: ” |
| ADDR\_ENV | db | Строка “Address of the environment: ” |
| TAIL | db | Строка “Tail command line: ” |
| CONTENT | db | Строка “Content of the environment area:” |
| PATH | db | Строка “Path of the loadable module:” |
| BUF | db | Буфер размера 100 байт |

**Последовательность действий, выполняемых программой.**

Модуль .СОМ выбирает и выводит на экран:

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.

**Результат работы программы.**

****

Рис.1 Результат работы программы.

**Ответы на контрольные вопросы.**

1. **Сегментный адрес недоступной памяти.**
2. На какую область памяти указывает адрес недоступной памяти?

Ответ: На область памяти, в которой нельзя выделять память. Там находится, например, память, используемая видеоадаптером и ROM BIOS.

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

Ответ: Сразу за областью, выделенной для программы (9FFFh).

1. Можно ли в эту область памяти писать?

Ответ: Можно, так как в DOS нет защиты памяти.

1. **Среда, передаваемая программе.**
2. Что такое среда?

Ответ: Среда – это набор переменных, необходимых для работы приложения.

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

Ответ: Среда создаётся при загрузке DOS и копируется в новую область памяти перед запуском приложения.

1. Откуда берется информация, записываемая в среду?

Ответ: Информация, записываемая в среду, берется из системного файла autoexec.bat.

**Вывод.**

В результате выполнения лабораторной работы было исследовано содержимое PSP, интерфейс управляющей программы и и среда, передаваемая программе.

**Приложение 1. Код программы.**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

ADDR\_MEM db 'Address of inaccessible memory: ',13,10,'$'

ADDR\_ENV db 'Address of the environment: ',0dh,0ah,'$'

TAIL db 'Tail command line: ',13,10,'$'

CONTENT db 'Content of the environment area:',13,10,'$'

PATH db 'Path of the loadable module:',13,10,'$'

BUF db 100 DUP(0)

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

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

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

FIND\_ADDR\_MEM PROC NEAR

push ax

push di

push dx

mov ax,ds:[02h]

lea di,ADDR\_MEM

add di,36

call WRD\_TO\_HEX

mov ah,09h

lea dx,ADDR\_MEM

int 21h

pop dx

pop di

pop ax

ret

FIND\_ADDR\_MEM ENDP

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

FIND\_ADDR\_ENV PROC NEAR

push ax

push di

push dx

mov ax, ds:[2Ch]

lea di, ADDR\_ENV

add di, 31

call WRD\_TO\_HEX

mov ah,09h

lea dx,ADDR\_ENV

int 21h

pop dx

pop di

pop ax

ret

FIND\_ADDR\_ENV ENDP

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

FIND\_TAIL PROC NEAR

push ax

push cx

push bx

push si

push di

xor cx,cx

mov ch,ds:[80h]

mov si,81h

mov di,offset TAIL

add di,20

start1:

cmp ch,0

je end1

mov al, ds:[si]

mov [di], al

inc di

inc si

dec ch

jmp start1

end1:

mov ah,09h

lea dx,TAIL

int 21h

pop di

pop si

pop dx

pop cx

pop ax

ret

FIND\_TAIL ENDP

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

FIND\_ENV\_AND\_PATH PROC near

push ax

push bx

push cx

push es

lea dx,CONTENT

mov ah,09h

int 21h

mov di,[2ch]

mov es,[di]

mov cx,0

mov si,0

mov bx,0

mov dx,0

mov cx,0

mov di,offset BUF

start2:

mov cx,0

cmp es:[si+bx],cx

je last\_line

cmp es:[si+bx],ch

je new\_line

mov cx,es:[si+bx]

mov [di+bx],cx

inc bx

jmp start2

new\_line:

mov dh,13

mov [di+bx],dh

inc bx

mov dh,10

mov [di+bx],dh

inc bx

mov dh,24h

mov [di+bx],dh

lea dx,BUF

int 21h

sub bx,1

add si,bx

mov bx,0

jmp start2

last\_line:

mov dh,13

mov [di+bx],dh

inc bx

mov dh,10

mov [di+bx],dh

inc bx

mov dh,24h

mov [di+bx],dh

lea dx,BUF

int 21h

sub bx,1

add si,bx

mov bx,0

jmp end2

end2:

lea dx,PATH

int 21h

add si,3

start3:

mov cx,0

cmp es:[si+bx],ch

je end3

mov cx,es:[si+bx]

mov [di+bx],cx

inc bx

jmp start3

end3:

mov dh,13

mov [di+bx],dh

inc bx

mov dh,10

mov [di+bx],dh

inc bx

mov dh,24h

mov [di+bx],dh

lea dx,BUF

int 21h

push es

pop cx

pop bx

pop ax

ret

FIND\_ENV\_AND\_PATH ENDP

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

BEGIN:

call FIND\_ADDR\_MEM

call FIND\_ADDR\_ENV

call FIND\_TAIL

call FIND\_ENV\_AND\_PATH

xor al, al

mov ah, 4ch

int 21h

TESTPC ENDS

END START