**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 |  | Калашников М.А. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи**

**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.

**Сведения о функциях и структурах данных управляющей программы.**

Сведения о функциях:

|  |  |
| --- | --- |
| Название | Назначение |
| TETR\_TO\_HEX | перевод полвины байта из 2-ой СС в 16-ую СС |
| BYTE\_TO\_HEX | перевод байта, расположенного в AL, в 2 символа 16-ой числа в AX |
| WRD\_TO\_HEX | перевод в 16-ую СС 16-ти разрядного числа |
| BYTE\_TO\_DEC | перевод в 10-ую СС |
| PRINT\_STR | Печать строки из DX, и печать перехода на новую строку, если CL!=0 |
| PRINT\_MEM | Печать адреса недоступной памяти |
| PRINT\_ENV | Печать начала сегментного адреса среды |
| PRINT\_ARGS | Печать хвоста командной строки |
| PRINT\_ENV\_PARAMS | Печать содержимого области среды и пути загружаемого модуля |

Сведения о структурах данных:

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| S\_MEM\_ADDR | db | Адрес недоступной памяти |
| S\_ENV\_ADDR | db | Адрес среды |
| S\_TAIL | db | Хвост командной строки |
| S\_NEV\_PARM | db | Содержимое области среды |
| S\_PATH | db | Путь загружаемого модуля |
| S\_ENDSTR | db | Переход на новую строку |

**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 без аргументов



рис.2 Результат работы программы с аргументами

**Описание результатов исследования проблем, поставленных в лабораторной работе**

**Сегментный адрес недоступной памяти.**

1. На какую область памяти указывает адрес недоступной памяти?

Ответ: Адрес недоступной памяти указывает на конец оперативной памяти и на границу области, доступной для загрузки программ, за ним располагается ROM BIOS;

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

Ответ: Адрес расположен сразу за областью памяти, отведенной программе (начиная с адреса 9FFF);

1. Можно ли в эту область памяти писать?

Ответ: При условии отсутствия в управляющей программе операционной системы механизма защиты памяти (как в DOS), запись в эту область возможна.

**Среда, передаваемая программе.**

1. Что такое среда?

Ответ: среда – это область памяти, содержащая переменные, в которых хранятся данные о некоторых директориях операционной системы и конфигурации компьютера, записанные в виде строк формата *имя=параметр*;

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

Ответ: среда создается при загрузке DOS, перед запуском приложения эта среда только копируется в новую область памяти;

1. Откуда берется информация, записываемая в среду?

Ответ: информация, записываемая в среду, берется из системного файла AUTOEXEC.BAT.

**Заключение**

В ходе работы было проведено 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 (РSР) и помещает его адрес в сегментный регистр.

Приложения

**Код программы** **code\_com**.**asm**

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

S\_ENDSTR db 0DH,0AH,'$'

S\_MEM\_ADDR db 'Unavailable memory address: h$'

S\_ENV\_ADDR db 'Segment environment address: h$'

S\_TAIL db 'Tail: $'

S\_ENV\_PARM db 'Environment params: $'

S\_PATH db 'Path: $'

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

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 ax

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

pop ax

ret

BYTE\_TO\_DEC ENDP

PRINT\_STR PROC near

push ax

push ds

mov AH,09h

int 21h

test cl, cl

je new\_line

dec cl

jmp quit

new\_line:

push es

pop ds

mov dx, OFFSET S\_ENDSTR

int 21h

quit:

pop ds

pop ax

ret

PRINT\_STR ENDP

PRINT\_MEM PROC NEAR

mov ax, ds:[02h]

mov di, offset S\_MEM\_ADDR

add di, 31

call WRD\_TO\_HEX

mov dx, offset S\_MEM\_ADDR

call PRINT\_STR

ret

PRINT\_MEM ENDP

PRINT\_ENV PROC NEAR

mov ax, ds:[2Ch]

mov di, offset S\_ENV\_ADDR

add di, 32

call WRD\_TO\_HEX

mov dx, offset S\_ENV\_ADDR

call PRINT\_STR

ret

PRINT\_ENV ENDP

PRINT\_ARGS PROC NEAR

mov dx, OFFSET S\_TAIL

inc cl

call PRINT\_STR

mov bx, 80h

mov al, [bx]

cmp al, 0

je empty

mov ah, 0

mov di, ax

mov bl, [di+81h]

mov byte ptr [di+81h], '$'

mov dx, 82h

call PRINT\_STR

mov [di+81h], bl

empty:

ret

PRINT\_ARGS ENDP

PRINT\_ENV\_PARAMS PROC NEAR

test ch, ch

jne only\_path

mov dx, offset S\_ENV\_PARM

call PRINT\_STR

jmp pep\_begin

only\_path:

mov dx, offset S\_PATH

inc cl

call PRINT\_STR

pep\_begin:

mov bx, 2Ch

mov ax, [bx]

mov ds, ax

mov di,0

mov dx, di

check\_for\_env\_end:

cmp byte ptr [di], 0

je exit

check\_for\_param\_endl:

inc di

cmp byte ptr [di], 0

jne check\_for\_param\_endl

mov byte ptr [di], '$'

call PRINT\_STR

mov byte ptr [di], 0

inc di

mov dx, di

test ch, ch

jne exit

jmp check\_for\_env\_end

exit:

push es

pop ds

ret

PRINT\_ENV\_PARAMS ENDP

BEGIN:

xor cl, cl

call PRINT\_MEM

call PRINT\_ENV

call PRINT\_ARGS

mov ch, 0

call PRINT\_ENV\_PARAMS

mov ch, 1

call PRINT\_ENV\_PARAMS

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START