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

Студентка гр. 7383 _____ Ханова Ю.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Постановка задачи.

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.

Ход работы.

В ходе выполнения данной работы был создан набор функций и структур данных, описанных в табл. 1-2.

Таблица 1 – Описание функций.

Название функции	**Назначение**
Write_info	Печатает: - сегментный адрес недоступной памяти, взятой из PSP в 16-ой с/с - сегментный адрес среды, передаваемой программе в 16-ой с/с - хвост командной строки в символьном виде
Write_sreda_path	Печатает: - содержимое области среды в символьном виде - путь загружаемого модуля
Write	Вызывает функцию печати строки
TETR_TO_HEX	Вспомогательная функция для работы функции BYTE_TO_HEX
BYTE_TO_HEX	Переводит число AL в коды символов 16-ой с/с, записывая получившееся в BL и BH

WRD_TO_HEX	Переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры

Таблица 2 - Описание структур данных.

Название	**Тип**	**Назначение**
off_mem_	db	сегментный адрес недоступной памяти, взятой из PSP в 16-ой с/с
Seg_adr	db	сегментный адрес среды, передаваемой программе в 16-ой с/с
TAIL	db	хвост командной строки в символьном виде
SREDA_	db	содержимое области среды в символьном виде
PATH_	db	путь загружаемого модуля
ENDL	db	Перенос строки
SPC	db	Отступ в начале строки

Была написана программа, которая выполняет следующие действия:

1) Печатает сегментный адрес первого байта недоступной памяти

2) Печатает сегментный адрес среды, передаваемой программе

3) Печатает хвост командной строки

4) Печатает содержимое области среды в символьном виде

5) Печатает путь загружаемого модуля

6) Выходит в DOS

Результаты работы программы представлены на рис. 1-2.

Рисунок 1 – Результат выполнения программы lr2.com (без хвоста командной строки)

Рисунок 2 – Результат выполнения программы lr2.com (с наличием хвоста командной строки)

Выводы.

В процессе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей, а также префикс сегмента программы (PSP) и среда, передаваемая программе. Код программы lr2.asm представлен в приложении А

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти:

1) На какую область памяти указывает адрес недоступной памяти?

   На границу оперативной памяти и на границу области, доступной для загрузки программ.

2) Где расположен этот адрес по отношению области памяти, отведенной программе?

   Сразу за областью памяти, отведенной программе.

3) Можно ли в эту область памяти писать?

   Можно, т.к. в DOS нет защиты памяти.

Среда, передаваемая программе:

1) Что такое среда?

   Среда представляет собой область памяти, в которой в виде символьных строк записаны значения переменных, называемых переменными среды.

2) Когда создается среда? Перед запуском приложения или в другое время?

   Среда создается при загрузке DOS, а при запуске приложения копируется в новую область памяти.

3) Откуда берется информация, записываемая в среду?

   В MS-DOS она берется из системного файла autoexec.bat.

ПРИЛОЖЕНИЕ А

lr2.asm

```
TESTPC SEGMENT
            ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
            ORG   100H
START: JMP BEGIN

;данные

off_mem_    db 'Segment address of the first byte of inaccessible memory:
',0DH,0AH,'$'
Seg_adr_    db 'Segmental address of the environment passed to the program:
',0DH,0AH,'$'
TAIL_       db 'Command-line tail:    ','$'
SREDA_            db 'The contents of the environment area in the symbolic form:
',0DH,0AH,'$'
PATH_       db 'Load module path: ',0DH,0AH,'$'
ENDL        db 0DH,0AH,'$'
SPC         db '        ','$'



Write PROC near
      mov AH,09h
      int 21h
      ret
Write ENDP

Write_info PROC near

      mov ax,es:[2]
      mov di,offset off_mem_
      add di, 62
      call WRD_TO_HEX
      mov dx,offset off_mem_
      call Write

      mov ax,es:[2Ch]
      mov di,offset Seg_adr_
      add di, 65
      call WRD_TO_HEX
      mov dx,offset Seg_adr_
      call Write

      mov dx,offset TAIL_
      call Write
```

```
	mov cx,0
	mov cl,es:[80h]
	cmp cl,0
	je TAIL_END
	mov dx,81h
	mov bx,0
	mov ah,02h
	TAIL_loop:
		mov dl,es:[bx+81h]
		int 21h
		inc   bx
	loop TAIL_loop

	TAIL_END:
		mov dx,offset ENDL
		call Write
		ret
Write_info ENDP

Write_sreda_path PROC near
	mov DX, OFFSET SREDA_
	call Write
	mov dx,offset SPC
	call Write
	xor SI, SI
	mov BX, ES:[2Ch]
	mov ES, BX
	mov AH, 02h

	env_loop:
		mov DL, ES:[SI]
		int 21h
		inc SI
		cmp WORD PTR ES:[SI], 0000h
		je gotopath
		cmp BYTE PTR ES:[SI], 00h
		je linenew
		jmp env_loop

	linenew:
		mov dx,offset ENDL
		call Write
		mov dx,offset SPC
		call Write
		mov AH, 02h
		inc SI
		jmp env_loop
```

```
	gotopath:
		mov dx,offset ENDL
		call Write
		add SI, 4
		mov DX, OFFSET PATH_
		call Write
		mov dx,offset SPC
		call Write
		mov AH, 02h

		path_loop:
			mov DL, ES:[SI]
			int 21h
			inc SI
			cmp BYTE PTR ES:[SI], 00h
			jne path_loop
	ret
Write_sreda_path 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
;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
	pop CX
	ret
BYTE_TO_HEX ENDP

; перевод в 16с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
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

BEGIN:
        call Write_info
        call Write_sreda_path
        xor AL,AL
        mov AH,4Ch
        int 21H
TESTPC ENDS
 END START
```