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. 8382 _____ Щеглов А.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

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 и среды, передаваемой программе.

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.

Результат работы программы представлен на рисунке 1.

Рис 1.

Контрольные вопросы.

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

От 9FFFh до FFFFh. Адрес 9FFFh, находящийся в PSP, подразумевает начало памяти, в которую нельзя загрузить программу.

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

Сразу же после памяти, выделенной программе

3. Можно ли в эту область памяти писать?

Да, так как DOS не контролирует обращение программ к памяти, но это может отразиться на работе программы

Среда, передаваемая программе

1. Что такое среда?

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,PROMPT,SET.

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

Среда создается при загрузке операционной системы.

3. Откуда берется информация, записываемая в среду?

Среда копируется из содержимого среды родительского процесса. Также можно изменить ее переменные или просмотреть ее содержимое с помощью команды set.

Вывод.

В ходе работы был исследован интерфейс управляющей программы, который состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные.

ПРИЛОЖЕНИЕ А

Исходный код программы lr2.asm

```
AS TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
    start: jmp BEGIN

ENDL db 13, 10, '$'
INACCES_MEM db "Inaccesible memory adress      ", 13, 10, '$'
SEG_ADRESS db "Environment address      ", 13, 10, '$'
TAIL db "Command line tail    ", '$'
CONTENT_S db "Environment content ", 13, 10, '$'
MODULE db "Module path ", '$'

PRINT PROC near
    push dx
    push ax
    mov ah, 09h
    mov dx, di
    int 21h
    pop ax
    pop dx
    ret
PRINT 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

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

BEGIN:

    mov ax, cs:[2h]
    mov di, offset INACCES_MEM
    push di
    add di, 29
    call WRD_TO_HEX
    pop di
    call PRINT

    mov ax, cs:[2ch]
    mov di, offset SEG_ADRESS
    push di
    add di, 23
    call WRD_TO_HEX
    pop di
    call PRINT

    mov di, offset TAIL
    call PRINT
    xor cx, cx
    mov cl, cs:[80h]
    cmp cx, 0
    je TAIL_end
    mov si, 81h
    mov ah, 02h

TAIL_LOOP:
    mov dl, cs:[si]
    int 21h
    inc si
    LOOP TAIL_LOOP

TAIL_end:
    mov di, offset ENDL
    call PRINT
```

```
    mov di, offset CONTENT_S
    call PRINT
    mov si, 2Ch
    mov es, [si]
    mov si, 0
    mov ah, 02h

CONTENT_S_OUT_LOOP:
    mov dl, 0
    cmp dl, es:[si]
    je CONTENT_S_END
CONTENT_S_IN_LOOP:
    mov dl, es:[si]
    int 21h
    inc si
    cmp dl, 0
    jne CONTENT_S_IN_LOOP
    jmp CONTENT_S_OUT_LOOP

CONTENT_S_END:
    mov di, offset ENDL
    call PRINT

    mov di, offset MODULE
    call PRINT
    add si, 3

MODULE_LOOP:
    mov dl, es:[si]
    int 21h
    inc si
    cmp dl, 0
    jne MODULE_LOOP


    mov di, offset ENDL
    call PRINT

    mov ah,4Ch
    int 21h


TESTPC ENDS
END START
```