

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по практической работе № 2
по дисциплине «Операционные системы»
Тема: Исследование интерфейсов программных модулей

Студентка гр. 9383

Сергиенкова А.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

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 состоит префикс сегмента программы (PSP) и помещает его адрес в сегментные регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Постановка задачи.

Для выполнения лабораторной работы необходимо написать и отладить программный модуль **.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.

Выполнение работы.

Были использованы функции:

TETR_TO_HEX – перевод десятичной цифры в код символа;

BYTE_TO_HEX – перевод байта в 16 с/с в символьный код;

WRD_TO_HEX – перевод слова в 16 с/с в символьный код;

BYTE_TO_DEC – перевод байта в 16 с/с в символьный код в 10 с/с.

Были составлены функции:

WRITESTRING – вывод строки на экран;

PSP_MEM - получение адреса недоступной памяти;

PSP_ENV – получение адреса среды;

PSP_TAIL - получение хвоста командной строки;

PSP_CONTENT - Получения содержимого области среды и пути загружаемого файла.

Также были объявлены строки для вывода информации:

- M_ADRESS db 'Locked memory address: h',13,10,'\$'
- E_ADRESS db 'Environment address: h',13,10,'\$'
- TAIL_STR db 'Command line tail: ',13,10,'\$'
- NULL_TAIL db 'In Command tail no sybmols',13,10,'\$'
- CONTENT db 'Content:',13,10, '\$'
- END_STRING db 13, 10, '\$'
- PATH db 'Path: ',13,10,'\$'

В результате выполнения были получены следующие значения(рис.1):



```
F:\>lab2.com hello
Locked memory address: 9FFF
Environment address: 0188
Command line tail: hello
Content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path:
F:\LAB2.COM
F:\>
```

Рисунок 1 – результат работы программы

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?

На сегментный адрес основной оперативной памяти, расположенной после программы.

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

Адрес недоступной памяти расположен с адреса 9FFF, после области памяти, отведённой программе.(В сторону бОльших адресов).

3. Можно ли в эту область памяти писать?

Можно, так как в DOS общее адресное пространство, также там нет механизмов защиты.

Среда, передаваемая программе:

1. Что такое среда?

Среда - область памяти, в которой в виде символьных строк записаны значения переменных, называемых переменными среды. В данной области памяти любые текстовые величины, байт 0 завершает каждую строку.

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

Изначально среда создаётся при запуске ОС. Но она также может быть изменена перед запуском приложения, в соответствии с требованиями. Также, когда одна программа запускает другую программу, то запущенная программа получает свой собственный экземпляр блока среды, который является точной копией среды родителя.

3. Откуда берётся информация, записываемая в среду?

Из системного файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства.

Выводы.

Был исследован интерфейс управляющей программы и загрузочных модулей, а также был изучен префикс сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Lab2.asm

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

; Данные

M_ADRESS db 'Locked memory address: ',13,10,'\$'

E_ADRESS db 'Environment address: ',13,10,'\$'

TAIL_STRING db 'Command line tail: ',13,10,'\$'

NULL_TAIL db 'In Command tail no sybmols',13,10,'\$'

CONTENT db 'Content:',13,10, '\$'

END_STRING db 13, 10, '\$'

PATH db 'Path: ',13,10,'\$'

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
```

WSTRING PROC near

mov AH,09h

int 21h

ret

WSTRING ENDP

PSP_MEM PROC near

;MEM

mov ax,ds:[02h]

mov di, offset M_ADRESS

add di, 26

call WRD_TO_HEX

mov dx, offset M_ADRESS

call WSTRING

ret

PSP_MEM ENDP

PSP_ENV PROC near

;ENV

mov ax,ds:[2Ch]

mov di, offset E_ADRESS

add di, 24

call WRD_TO_HEX

mov dx, offset E_ADRESS

call WSTRING

ret

PSP_ENV ENDP


```

PSP_TAIL PROC near
;TAIL
    mov cl, ds:[80h]
    mov si, offset TAIL_STRING
    add si, 19
    cmp cl, 0h
    je empty_tail
    xor di, di
    xor ax, ax
readtail:
    mov al, ds:[81h+di]
    inc di
    mov [si], al
    inc si
    loop readtail
    mov dx, offset TAIL_STRING
    jmp end_tail
empty_tail:
    mov dx, offset NULL_TAIL
end_tail:
    call WSTRING
    ret
PSP_TAIL ENDP

```

```

PSP_CONTENT PROC near
;ENV CONTENT
    mov dx, offset CONTENT
    call WSTRING
    xor di, di

```

```

    mov ds, ds:[2Ch]
read_str:
    cmp byte ptr [di], 00h
    jz end_str
    mov dl, [di]
    mov ah, 02h
    int 21h
    jmp find_end
end_str:
    cmp byte ptr [di+1], 00h
    jz find_end
    push ds
    mov cx, cs
    mov ds, cx
    mov dx, offset END_STRING
    call WSTRING
    pop ds
find_end:
    inc di
    cmp word ptr [di], 0001h
    jz read_path
    jmp read_str
read_path:
    push ds
    mov ax, cs
    mov ds, ax
    mov dx, offset PATH
    call WSTRING
    pop ds

```

```

        add di, 2
loop_path:
        cmp byte ptr [di], 00h
        jz complete
        mov dl, [di]
        mov ah, 02h
        int 21h
        inc di
        jmp loop_path
complete:
        ret
PSP_CONTENT ENDP

BEGIN:
        call PSP_MEM
        call PSP_ENV
        call PSP_TAIL
        call PSP_CONTENT

        xor AL,AL
        mov AH,4Ch
        int 21H
TESTPC ENDS
END START

```