

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

ОТЧЁТ
по лабораторной работе №2
по дисциплине «Операционные системы»
Тема: Исследование интерфейсов программных модулей

Студент гр.8382

Фильцин И.В.

Преподаватель

Ефремов М.А..

Санкт-Петербург

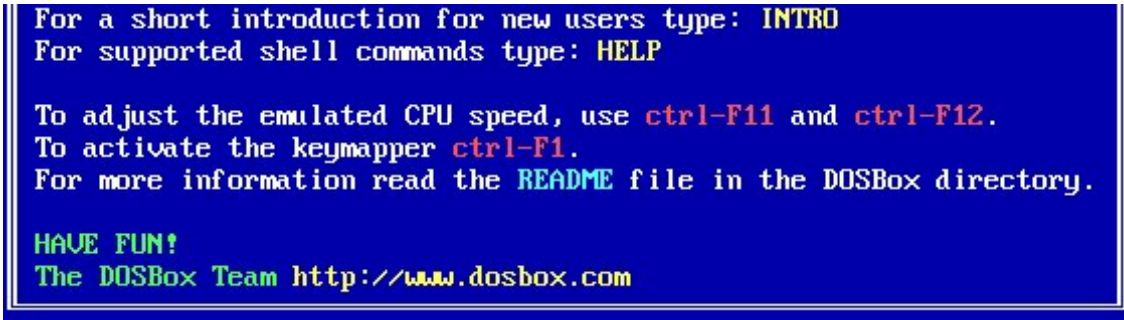
2020

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 (PSP) и среды, передаваемой программе.

Ход работы

В ходе лабораторной работы был написан код программы (см. исх. код в приложении А) и получен .COM модуль, который выводит некоторую информацию (Сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, область среды, путь загружаемого модуля) (см. рис. 1)



```
For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C "."
Drive C is mounted as local directory ./

Z:\>C:

C:\>1COM.COM
Inaccessible memory starts from 9FFF
Env address: 0188
Command line:
Env: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path: C:\1COM.COM

C:\>_
```

Рис. 1

Контрольные вопросы

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

На верхнюю границу доступной памяти в системе.

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

До выделенной программе памяти. (Старшие адреса сверху)

3) Можно ли в эту область памяти писать?

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

Среда передаваемая программе

1) Что такое среда?

Набор строк вида имя=параметр, 0

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

В момент загрузки ОС.

3) Откуда берётся информация, записываемая в среду?

Информация записывается в среду при помощи команды set имя=параметр

Вывод

В ходе лабораторной работы были изучены интерфейс управляющей программы и загрузочных модулей, PSP и среда, передаваемая программе.

Приложение А. Исходный код программы

```
testpc segment

    assume CS:testpc, ds:testpc, es:nothing, ss:
        nothing
    org 100h
start: jmp begin

inaccessible_label db 'Inaccessible memory starts from
    ', '$'
inaccessible_value db '0000', 13, 10, '$'

env_label db 'Env addres: ', '$'
env_value db '0000', 13, 10, '$'

line_label db 'Command line:', '$'

os_env_label db 'Env: ', '$'

path_label db 'Path: ', '$'

rn_label db 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

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_1
    or al, 30h
    mov [si], al
end_1:
    pop dx

```

```

    pop cx
    ret
byte_to_dec endp

print_command_line proc near
    push bx
    push si
    push ax

    cmp cx, 0
    je finish

    mov ah, 02h
    mov si, 081h

while:
    mov dl, [si]
    int 21h
    inc si
    loop while

finish:
    pop ax
    pop si
    pop bx
    ret
print_command_line endp

```



```

print_env proc near

    push ax
    push dx

    mov ah, 02h
    mov si, 0

out_print:
    mov dl, es:[si]
    cmp dl, 0
    je finish_1
while_print:
    mov dl, es:[si]
    int 21h
    inc si
    cmp dl, 0
    je out_print
    jmp while_print

finish_1:
    pop dx
    pop ax
    ret
print_env endp

print_path proc near
    push ax

```

push dx

mov ah, 02h

print_for:

mov dl, es:[si]

int 21h

cmp dl, 0

je finish_print

inc si

jmp print_for

finish_print:

pop dx

pop ax

ret

print_path endp

printrn proc near

push ax

push dx

mov dx, offset rn_label

mov ah, 09h

int 21h

```
    pop dx
    pop ax
    ret
printrn endp
```

begin:

```
    mov dx, offset inaccessible_label
    mov ah, 09h
    int 21h
```

```
    mov si, 02h
    mov ax, [si]
    mov di, offset inaccessible_value
    add di, 3
    call wrd_to_hex
```

```
    mov dx, offset inaccessible_value
    mov ah, 09h
    int 21h
```

```
    mov dx, offset env_label
    int 21h
```

```
    mov si, 02ch
    mov ax, [si]
    mov di, offset env_value
```

```

add di, 3
call wrd_to_hex

mov dx, offset env_value
mov ah, 09h
int 21h

mov dx, offset line_label
int 21h

xor ch, ch
mov si, 080h
mov cl, [si]

call print_command_line
call printrn

mov dx, offset os_env_label
mov ah, 09h
int 21h

mov si, 02ch
mov es, [si]

call print_env
call printrn

```

```
mov dx, offset path_label
mov ah, 09h
int 21h
```

```
add si, 3
```

```
call print_path
call printrn
```

```
xor al, al
mov ah, 4ch
int 21h
```

```
testpc ends
      end start
```