МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №6

по дисциплине «Операционные системы»

Тема: Построение модуля динамической структуры

Студент гр.8382 _____ Фильцин И.В.

Преподаватель _____ Ефремов М.А..

Санкт-Петербург

2020

Цель работы

Исследование возможности построения загрузочного модуля динамической структуры. В отличии от предыдущих лабораторных работ в этой работе рассматривается приложение, состоящее из нескольких модулей, а не из одного модуля простой структуры. В этом случае разумно предположить, что все модули приложения находятся в одном каталоге и полный путь в этот каталог можно взять из среды, как это делалось в работе 2. Понятно, что такое приложение должно запускаться в соответствии со стандартами ОС.

В работе исследуется интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция 4B00h прерывания int 21h. Все загрузочные модули находятся в одном каталоге. Необходимо обеспечить возможность запуска модуля динамической структуры из любого каталога.

Ход работы

В ходе лабораторной программы был написан программный модуль типа .EXE, который подготавливает параметры для запуска загрузочного модуля и вызывает его.

Программа запускает модифицированную программу из 2 лабораторной работы, которая при выходе запрашивает у пользователя символ с клавиатуры и завершается с введенным кодом.

Результат запуска программы см. на рис. 1

Результат запуска программы и ввода Ctrl-C см. на рис. 2

Запуск программы из другой директории см. на рис. 3

Запуск программы, когда модули находятся в разных директориях см. на рис. 4

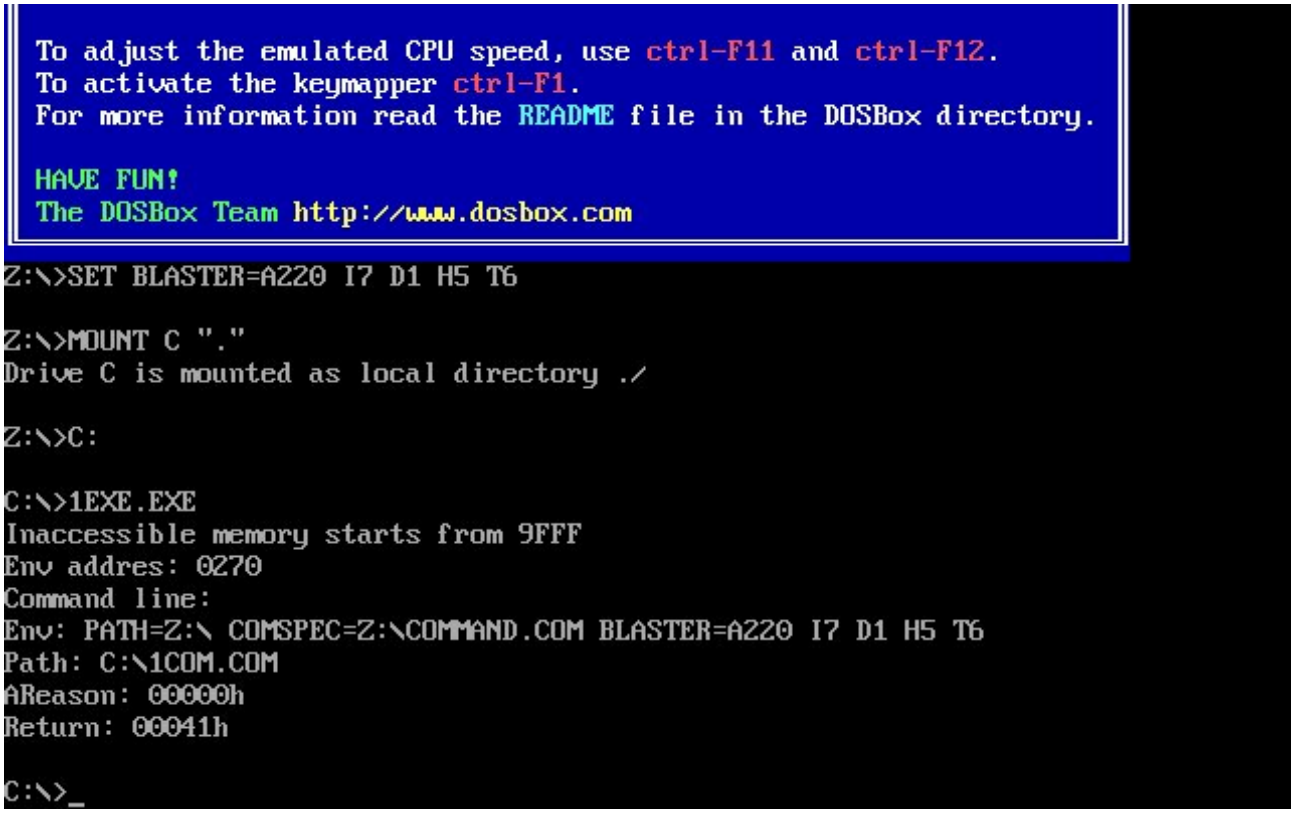

Рис. 1: Запуск программы

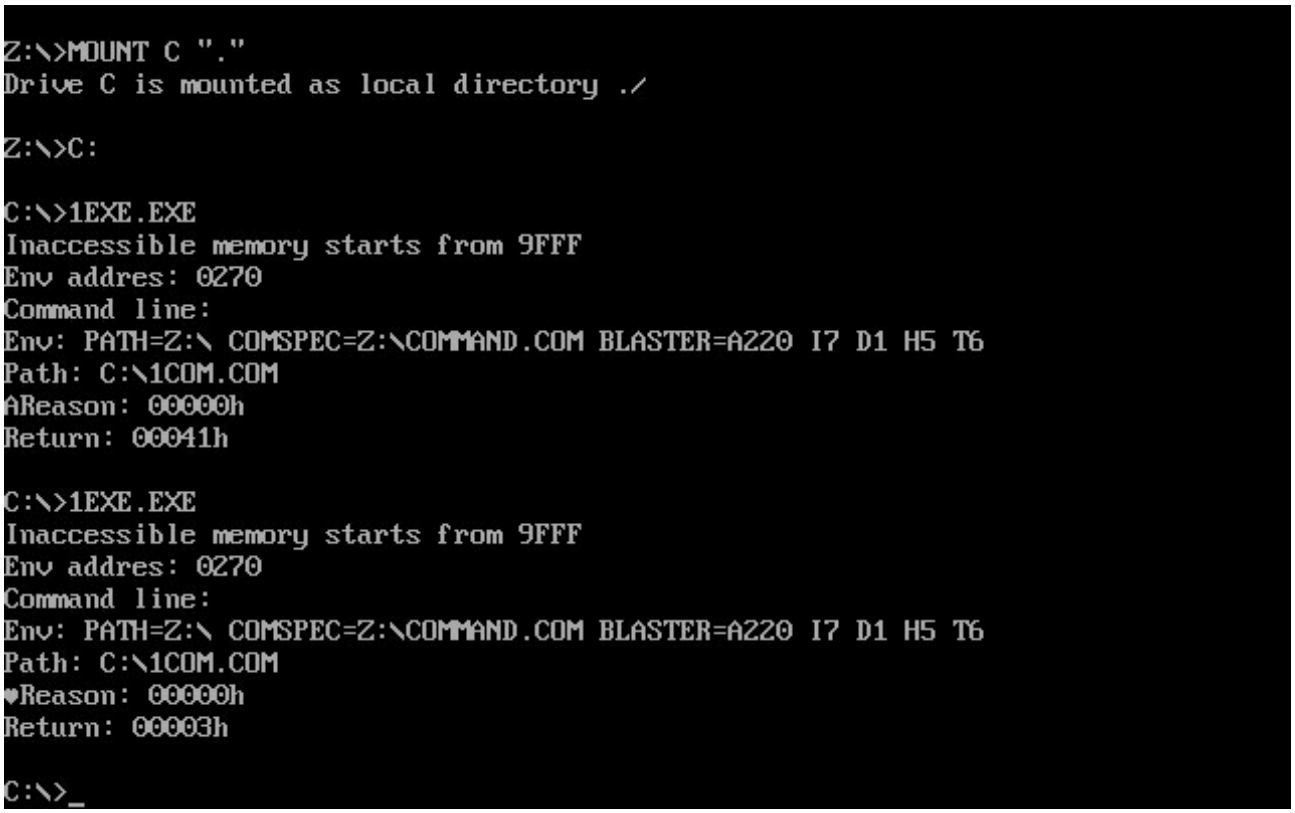

Рис. 2: Результат запуска программы и ввода Ctrl-C

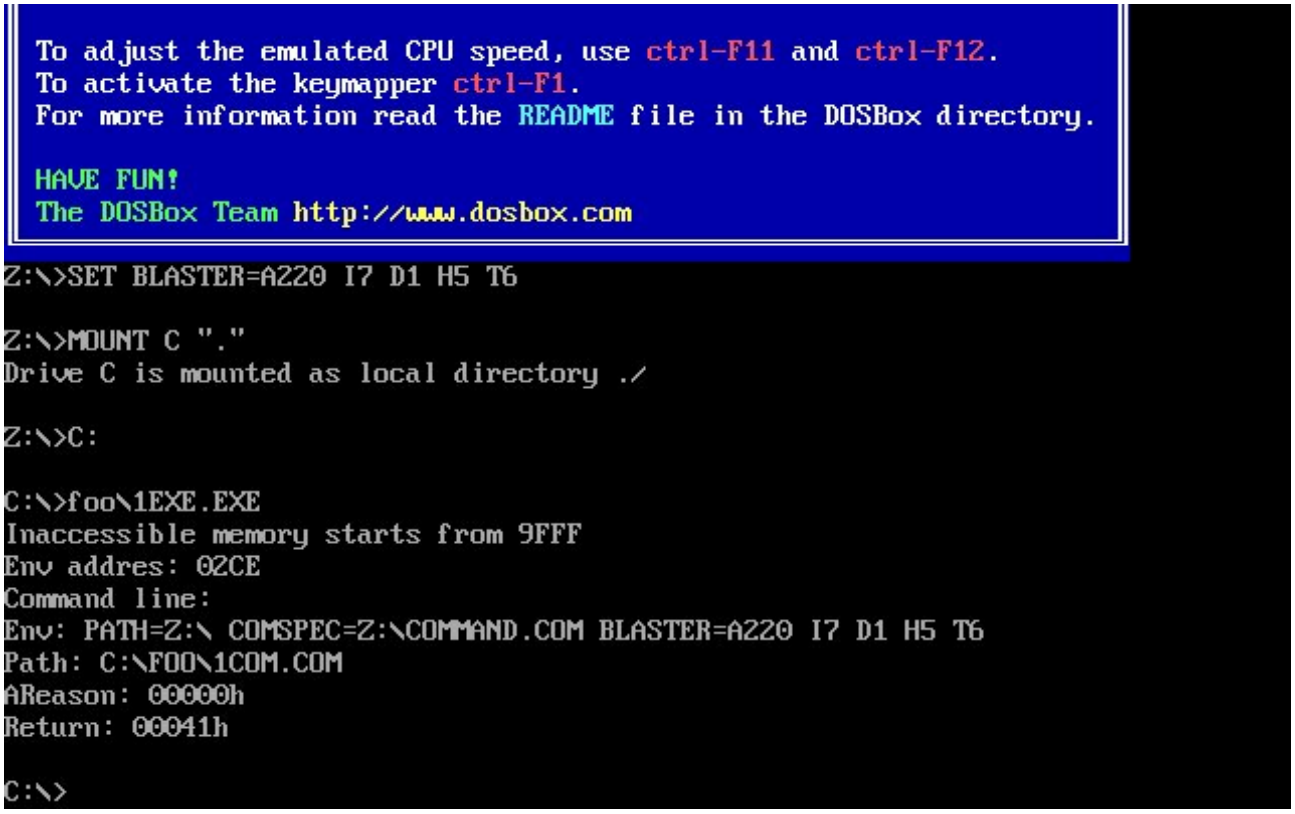

Рис. 3: Запуск программы из другой директории

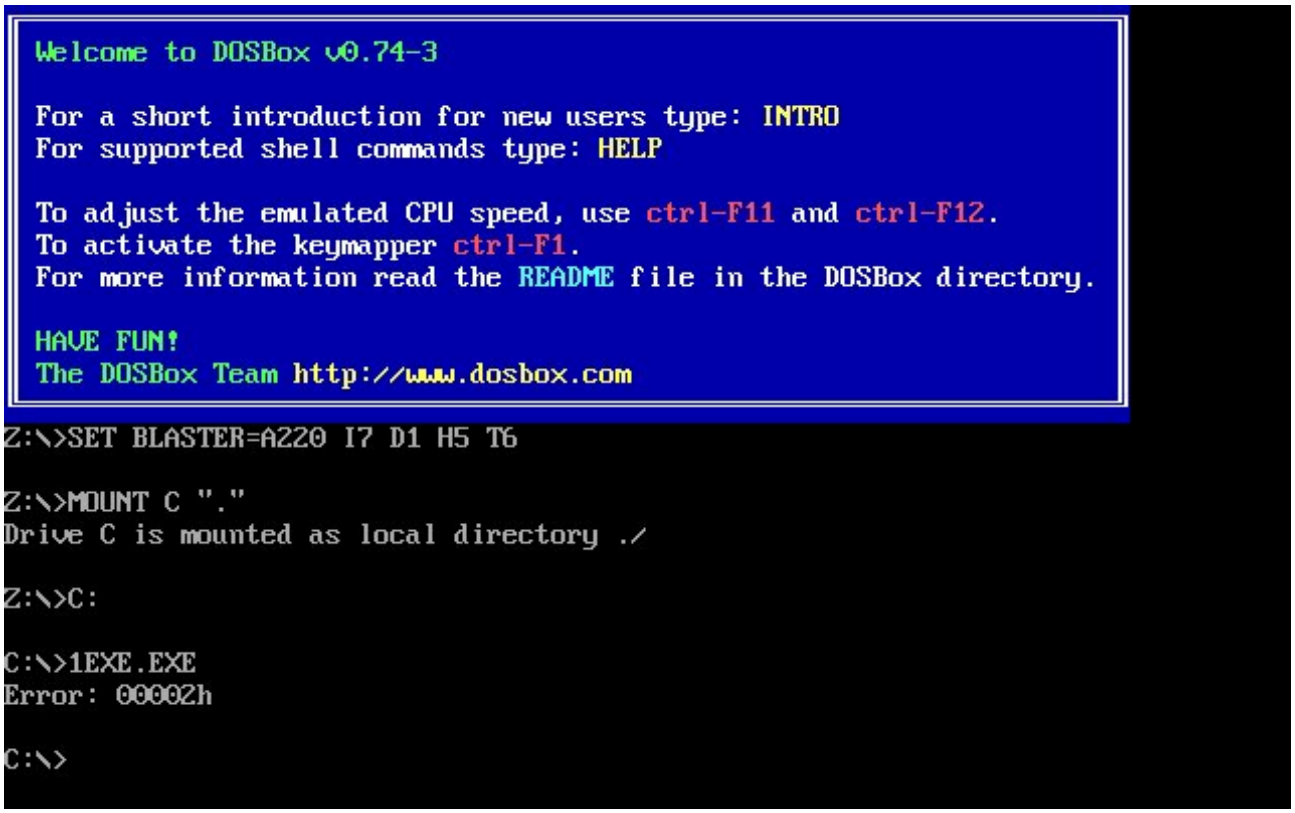

Рис. 4: Запуск программы, когда модули находятся в разных директориях

Контрольные вопросы

1) Как реализовано прерывание Ctrl-C.

Генерируется прерывание 23h, которое завершает запущенную программу.

2) В какой точке заканчивается вызываемая программа, если код причины завершения 0?

Внутри прерывания 21h ф-ии 4ch.

3) В какой точке заканчивается вызываемая программа по прерыванию Ctrl-C?

Внутри прерывания 23h.

Вывод

В ходе лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

Приложение А. Исходный код программы

```
.model small

.stack 100h

.data

error_string db 'Error: 00000h', 13, 10, '$'

segm dw 0
seg_offset_cmd dd 0
seg_offset_fcbf dd 0
seg_offset_fcbs dd 0
program_path db 100 dup(0)

keep_ss dw 0
keep_sp dw 0
keep_ds dw 0

reason_code db 'Reason: 00000h', 13, 10, '$'
return_code db 'Return: 00000h', 13, 10, '$'

lb db 'Offset: 0000', 13, 10, '$'

.code

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

print_error_code proc near
  push di
  push dx

  mov di, offset error_string
  add di, 11
  call wrd_to_hex

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

  pop dx
  pop di
  ret
print_error_code endp
```

```
main:
  mov ax, @data
  mov ds, ax

  mov bx, offset last_byte
  mov ah, 04ah
  int 21h
  jnc run_prog

  call print_error_code
  jmp finish

  run_prog:
  mov bp, offset seg_offset_cmd
  mov [bp], es
  mov ah, 080h
  mov [bp + 2], ah

  mov bp, offset seg_offset_fcbf
  mov [bp], es
  mov ah, 05ch
  mov [bp + 2], ah

  mov bp, offset seg_offset_fcbs
  mov [bp], es
  mov ah, 06ch
```

```
mov [bp + 2], ah

mov si, 02ch
mov es, es:[si]

mov si, 0

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

add si, 3
mov bp, offset program_path

print_for:
  mov dl, es:[si]
  mov ds:[bp], dl
  cmp dl, 0
```

```
  je finish_print
  inc si
  inc bp
  jmp print_for
finish_print:

sub bp, 8
mov ds:[bp], byte ptr '1'
mov ds:[bp + 1], byte ptr 'c'
mov ds:[bp + 2], byte ptr 'o'
mov ds:[bp + 3], byte ptr 'm'
mov ds:[bp + 4], byte ptr '.'
mov ds:[bp + 5], byte ptr 'c'
mov ds:[bp + 6], byte ptr 'o'
mov ds:[bp + 7], byte ptr 'm'

mov dx, offset program_path

mov keep_ss, ss
mov keep_sp, sp
mov keep_ds, ds

mov ax, ds
mov es, ax
mov bx, offset segm

mov ax, 04b00h
```

```
int 21h

mov ss, keep_ss
mov sp, keep_sp
mov ds, keep_ds

jnc good_run

call print_error_code
jmp finish

good_run:

mov ah, 04dh
int 21h

push ax
xor al, al
xchg ah, al
mov di, offset reason_code
add di, 12
call wrd_to_hex

pop ax
xor ah, ah
mov di, offset return_code
add di, 12
```

```
  call wrd_to_hex

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

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

  finish:
  mov ah, 04ch
  int 21h

last_byte:

end main
```

Приложение Б. Исходный код программы из лр.2

```
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_l
    or al, 30h
    mov [si], al
  end_l:
    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

  mov ah, 01h
  int 21h

  mov ah, 4ch
  int 21h

testpc ends
       end start
```