

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

ОТЧЕТ
по лабораторной работе №6
по дисциплине «Операционные системы»
Тема: Построение модуля динамической структуры

Студент гр. 9381

Прибылов Н.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. В отличие 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.

Постановка задачи.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет функции:

1) Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится он сам. Вызываемому модулю передается новая среда, созданная вызывающим модулем и новая командная строка.

2) Вызываемый модуль запускается с использованием загрузчика.

3) После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Необходимо проверять причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения. В качестве вызываемой программы необходимо взять программу ЛР 2, которая распечатывает среду и командную строку. Эту программу следует немного модифицировать, вставив перед выходом из нее обращение к функции ввода символа с клавиатуры. Введенное значение записывается в регистр AL и затем происходит обращение к функции выхода 4Ch прерывания int 21h.

Шаг 2. Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры. Введите произвольный символ из числа A-Z. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

Шаг 3. Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры. Введите комбинацию символов Ctrl-C. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

Шаг 4. Запустите отлаженную программу, когда текущим каталогом является какой-либо другой каталог, отличный от того, в котором содержатся разработанные программные модули. Повторите ввод комбинаций клавиш. Занесите полученные данные в отчет.

Шаг 5. Запустите отлаженную программу, когда модули находятся в разных каталогах. Занесите полученные данные в отчет.

Выполнение работы.

1. Была запущена программа из директории с разработанными модулями и введен символ z.

```
X:\>lab6.exe
>Inaccessible memory address: 9FFF
>Environment address: 01F9
>Command line tail:
>Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
>Module path:
X:\LAB62.COMz

In lab6:
Termination reason: Normal termination
Exit code: z

X:\>
```

2. Программа вновь была запущена и завершена при помощи Ctrl-C.

```
X:\>lab6.exe
>Inaccessible memory address: 9FFF
>Environment address: 01F9
>Command line tail:
>Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
>Module path:
X:\LAB62.COM♥

In lab6:
Termination reason: Normal termination
Exit code: ♥

X:\>_
```

3. Программа была запущена из другой директории.

```

X:\LAB6>lab6.exe
>Inaccessible memory address: 9FFF
>Environment address: 01F9
>Command line tail:
>Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
>Module path:
X:\LAB6\LAB62.COMx

In lab6:
Termination reason: Normal termination
Exit code: x

X:\LAB6>

```

4. Программа была запущена, причём модули находятся в разных каталогах.

```

X:\>lab6.exe

In lab6:
Error: file not found!

X:\>_

```

Контрольные вопросы.

1. Как реализовано прерывание Ctrl-C?

При нажатии клавиш Ctrl-C управление передаётся по адресу 0000:008Ch. Этот адрес копируется в PSP функциями 26h и 4Ch и восстанавливается из PSP при выходе из программы.

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

При выполнении функции 4Ch прерывания int 21h.

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

Программа завершится непосредственно в том месте, в котором произошло нажатие сочетания клавиш (то есть в месте ожидания нажатия клавиши: 01h вектора прерывания 21h).

Выводы.

Был построен загрузочный модуль динамической структуры. Исследован интерфейс между вызывающим и вызываемым модулями по управлению и по данным.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab6.asm

```
CODE segment
ASSUME CS:CODE, DS:CODE, SS:STACK

    block      db 14 dup(0)
    path       db 50 dup(0)
    _ss        dw 0
    _sp        dw 0
    mem_error  db 'Error: Memory cannot be allocated!$'
    lab2_name  db 'lab62.com', 0
    message    db 'In lab6:$'
    error1     db 'Error: invalid function number!$'
    error2     db 'Error: file not found!$'
    error3     db 'Error: disk error!$'
    error4     db 'Error: not enough memory!$'
    error5     db 'Error: wrong environment string!$'
    error6     db 'Error: invalid format!$'
    error7     db 'Error: unknown error!$'
    reason     db 'Termination reason: $'
    reason1    db 'Normal termination$'
    reason2    db 'Ctrl-Break termination$'
    reason3    db 'Device error termination$'
    reason4    db '31h function termination$'
    reason5    db 'Unknown termination reason$'
    exit_code  db 'Exit code: $'
    endl       db 13,10,'$'
    teststr    db 4,' cmd tail to 2 $',0

main proc near
    mov ax, seg CODE
    mov ds, ax

    mov bx, seg CODE
    add bx, offset CodeSegEnd
    add bx, 256
    mov cl, 4h
    shr bx, cl
    mov ah, 4Ah
    int 21h
    jnc mem_ok
    mov dx, offset mem_error
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp main_exit

mem_ok:
    mov es, es:[002Ch]
```

```

        xor bx, bx

next:
    mov dl, byte PTR es:[bx]
    cmp dl, 0h
    je first_0
    inc bx
    jmp next
first_0:
    inc bx
    mov dl, byte PTR es:[bx]
    cmp dl, 0h
    je second_0
    jmp next
second_0:
    add bx,3

    push si
    mov si, offset path
next1:
    mov dl, byte PTR es:[bx]
    mov [si], dl
    inc si
    inc bx
    cmp dl, 0
    jne next1

next2:
    mov al, [si]
    cmp al, '\'
    je next3
    dec si
    jmp next2

next3:
    inc si

    push di
    mov di, offset lab2_name
next4:
    mov ah, [di]
    mov [si], ah
    inc si
    inc di
    cmp ah, 0
    jne next4

    pop di
    pop si

    mov _sp, sp
    mov _ss, ss
    mov ax, ds
    mov es, ax
    push ax
    mov ax, seg teststr
    mov [block+3], ah
    mov [block+2], al

```



```

pop ax
mov ax, offset teststr
mov [block+4], al
mov [block+5], al
mov bx, offset block
mov dx, offset path
mov ax, 4B00h
int 21h

```

```

push ax
mov dx, offset endl
mov ah, 09h
int 21h
mov ah, 09h
int 21h
mov dx, offset message
mov ah, 09h
int 21h
mov dx, offset endl
mov ah, 09h
int 21h
pop ax

```

```

jc errors
jmp run_ok

```

errors:

```

cmp ax, 1
jne err2
mov dx, offset error1
mov ah, 09h
int 21h
mov dx, offset endl
mov ah, 09h
int 21h
jmp main_exit

```

err2:

```

cmp ax, 2
jne err3
mov dx, offset error2
mov ah, 09h
int 21h
mov dx, offset endl
mov ah, 09h
int 21h
jmp main_exit

```

err3:

```

cmp ax, 5
jne err4
mov dx, offset error3
mov ah, 09h
int 21h
mov dx, offset endl
mov ah, 09h
int 21h
jmp main_exit

```

```

err4:
    cmp ax, 8
    jne err5
    mov dx, offset error4
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp main_exit

err5:
    cmp ax, 10
    jne err6
    mov dx, offset error5
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp main_exit

err6:
    cmp ax, 11
    jne err7
    mov dx, offset error6
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp main_exit

err7:
    mov dx, offset error7
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp main_exit

run_ok:
    mov ax, seg CODE
    mov ds, ax
    mov ss, _ss
    mov sp, _sp

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

    mov ah, 4Dh
    int 21h

    push ax
    cmp ah, 0

```

```

    jne reason_tag2
    mov dx, offset reason1
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp print_exit_code

reason_tag2:
    cmp ah, 1
    jne reason_tag3
    mov dx, offset reason2
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp print_exit_code

reason_tag3:
    cmp ah, 2
    jne reason_tag4
    mov dx, offset reason3
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp print_exit_code

reason_tag4:
    cmp ah, 3
    jne reason_tag5
    mov dx, offset reason4
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h
    jmp print_exit_code

reason_tag5:
    mov dx, offset reason5
    mov ah, 09h
    int 21h
    mov dx, offset endl
    mov ah, 09h
    int 21h

print_exit_code:
    mov dx, offset exit_code
    mov ah, 09h
    int 21h
    pop ax
    mov dl, al
    mov ah, 02h
    int 21h

```

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

main_exit:
        xor al, al
        mov ah, 4Ch
        int 21h
        ret
main endp

CodeSegEnd:
CODE ends

STACK segment stack
        dw 128 dup (?)
STACK ends

end main
```