**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**по лабораторной работе №6**

**по дисциплине «Операционные системы»**

# **Тема: Построение модуля динамической структуры**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9383 |  | Орлов Д.С. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2021

**Цель работы.**

Исследование возможности построения загрузочного модуля динамической структуры.

**Задание.**

**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) Вызываемой модуль запускается с использованием загрузчика.

3) После запуска проверяется выполнение загрузчика, а затем ре-

зультат выполнения вызываемой программы. Необходимо проверить причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

В качестве вызываемой программы необходимо взять программа 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 полученный результат 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.

**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. Был написан и отлажен программный модуль типа **.ЕХЕ,** который выполняет необходимые функции.
2. Запущена отлаженная программа, когда текущим каталогом является каталог с разработанными модулями. lb6.exe вызывает lab2.com, которая остановилась, ожидая символ с клавиатуры. Был введен символ 1.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 - Пример запуска программы lb6.exe

1. Была повторно запущена программа и введена комбинация ctrl + c. Так как в DOSBOX не реализована обработка данной комбинации, сtrl + с выводится как сердце.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 - Результат работы программы на шаге 3

1. Программы были запущены из другого каталога.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 - Пример работы программы на шаге 4

1. Была запущена программа lb6.exe, когда модули находились в разных каталогах.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 - Вывод программы на шаге 5

**Ответы на вопросы.**

**1. Как реализовано прерывания Ctrl-C?**

При нажатии сочетания клавиш Ctrl+C управление передается по адресу 0000:008C после срабатывания прерывания 23h. Адрес копируется в PSP функциями 26h и 4Ch. При выходе из программы адрес восстанавливается.

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

В точке вызова функции 4Ch прерывания int 21h.

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

Программа завершится в точке, где была считана комбинация клавиш Ctrl+C: в месте ожидания нажатия клавиши, на функции 01h вектора прерывания 21h

**Вывод.**

В результате выполнения работы были исследованы возможности построения загрузочного модуля динамической структуры.

**ПРИЛОЖЕНИЕ А**

**КОД ПРОГРАММЫ**

code segment

assume CS:code, DS:data

println proc

push AX

push DX

mov AH, 09h

int 21h

mov AH, 02h

mov DL, 0Ah

int 21h

mov DL, 0Dh

int 21h

pop DX

pop AX

ret

println endp

num2dec proc

push AX

push BX

push CX

push DX

push SI

mov BX, 10

xor CX, CX

xor DX, DX

div10: div BL

mov DL, AH

push DX

xor AH, AH

inc CX

xor DX, DX

cmp AL, 0

jne div10

loop10: pop DX

xor DH, DH

add DL, 30h

mov byte ptr [SI], DL

inc SI

loop loop10

pop SI

pop DX

pop CX

pop BX

pop AX

ret

num2dec endp

free proc

push AX

push BX

push DX

xor DX, DX

mov AX, offset endprog

add AX, offset enddata

add AX, 300h

mov BX, 16

div BX

mov BX, AX

inc BX

mov AH, 4Ah

int 21h

jnc endf

cmp AX, 7

je efree7

cmp AX, 8

je efree8

cmp AX, 9

je efree9

jmp efreeU

efree7: mov DX, offset ferr7

jmp eprint

efree8: mov DX, offset ferr8

jmp eprint

efree9: mov DX, offset ferr9

jmp eprint

efreeU: mov DX, offset unknown

eprint: call println

endf: pop DX

pop BX

pop AX

ret

free endp

setp proc

push AX

mov AX, ES:[2Ch]

mov param, AX

mov param + 2, ES

mov param + 4, 80h

pop AX

ret

setp endp

getpath proc

push DX

push DI

push SI

push ES

xor DI, DI

mov ES, ES:[2Ch]

mov DL, ES:[DI]

jmp check

nextc: inc DI

mov DL, ES:[DI]

check: cmp DL, 00h

jne nextc

inc DI

mov DL, ES:[DI]

cmp DL, 00h

jne nextc

xor SI, SI

add DI, 3

getc: mov DL, ES:[DI]

cmp DL, 00h

je getf

mov byte ptr fpath[SI], DL

inc DI

inc SI

jmp getc

getf: dec SI

mov DL, fpath[SI]

cmp DL, '\'

jne getf

inc SI

xor DI, DI

addp: mov DL, fname[DI]

cmp DL, '$'

je endg

mov fpath[SI], DL

inc DI

inc SI

jmp addp

endg: mov fpath[SI], 00h

pop ES

pop SI

pop DI

pop DX

ret

getpath endp

callp proc

push AX

push DX

push DS

push ES

mov keepSS, SS

mov keepSP, SP

mov AX, DS

mov ES, AX

mov BX, offset param

mov DX, offset fpath

mov AX, 4B00h

int 21h

mov DX, keepSP

mov SP, DX

mov SS, keepSS

pop ES

pop DS

jnc ok

cmp AX, 1

je ecall1

cmp AX, 2

je ecall2

cmp AX, 5

je ecall5

cmp AX, 8

je ecall8

cmp AX, 10

je ecallA

cmp AX, 11

je ecallB

jmp ecallU

ecall1: mov DX, offset cerr1

jmp print

ecall2: mov DX, offset cerr2

jmp print

ecall5: mov DX, offset cerr5

jmp print

ecall8: mov DX, offset cerr8

jmp print

ecallA: mov DX, offset cerrA

jmp print

ecallB: mov DX, offset cerrB

jmp print

ecallU: mov DX, offset unknown

jmp print

ok: mov AX, 4D00h

int 21h

cmp AH, 0

je good

cmp AH, 1

je eprog1

cmp AH, 2

je eprog2

cmp AH, 3

je eprog3

jmp ecallU

eprog1: mov DX, offset lerr1

jmp print

eprog2: mov DX, offset lerr2

jmp print

eprog3: mov DX, offset lerr3

jmp print

good: mov DX, offset lgood

mov SI, DX

add SI, 37

call num2dec

print: push AX

push DX

mov AH, 02h

mov DL, 0Dh

int 21h

mov AH, 02h

mov DL, 0Ah

int 21h

pop DX

pop AX

call println

pop DX

pop AX

ret

callp endp

main proc far

mov AX, data

mov DS, AX

call free

call setp

call getpath

call callp

; --- End ---

mov AX, 4C00h

int 21h

main endp

endprog:

code ends

data segment

param dw 7 dup (0)

fname db 'lab2.com$'

fpath db 64 dup (0), '$' ; or maybe more

keepSS dw 0

keepSP dw 0

unknown db 'Unknown error$'

ferr7 db 'Error: memory control block destroyed$'

ferr8 db 'Error: not enough memory to execute the function$'

ferr9 db 'Error: invalid memory block address$'

cerr1 db 'Error: function number is incorrect$'

cerr2 db 'Error: file could not be found$'

cerr5 db 'Disk error$'

cerr8 db 'Error: insufficient memory$'

cerrA db 'Error: wrong environment string$'

cerrB db 'Error: wrong format$'

lgood db 'The program has ended with the code: $'

lerr1 db 'The program terminated by Ctrl-Break$'

lerr2 db 'The program terminated by device error$'

lerr3 db 'The program terminated by function 31h$'

enddata db 0

data ends

stack segment stack

db 256 dup (?)

stack ends

end main