**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №7**

**по дисциплине «Операционные системы»**

Тема: Построение модуля оверлейной структуры

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Поляков Н.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы.**

Исследование возможности построения модуля оверлейной структуры и механизма вызова его из программы.

**Описание функций.**

|  |  |
| --- | --- |
| Имя | Описание |
| PRINT | Вызывает функцию 9 прерывания 21h |
| FIND\_OVL\_SIZE | Определяет размер оверлейного модуля и выделяет для него память |
| FIND\_PATH | Формирует путь до файла оверлейного модуля и сохраняет его в памяти |
| FREE\_MEM | Освобождает память, неиспользуемую программой |
| CALL\_OVL | Осуществляет выполнение кода оверлейной структуры |

**Описание структур данных.**

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| ERROR\_FREEING | db | 'Error when freeing memory: $' |
| ERROR\_MCM | db | 'MCB is destroyed$' |
| ERROR\_NO\_MEM | db | 'Not enough memory for function processing$' |
| ERROR\_WRONG\_ADDR | db | 'Wrong addres of memory block$' |
| ERROR\_UNKNOWN | db | 'Unknown error$' |
| STRENDL | db | Строка с символами перевода строки |
| OVL\_PATH | db | Строка для хранения пути до оверлейного модуля |
| DTA | db | Буфер DTA |
| KEEP\_PSP | dw | Адрес PSP |
| PATH\_TO | db | 'Path to the called file: ','$' |
| ERROR\_FILE\_NOT\_FOUND | db | 'The file was not found!',13,10,'$' |
| ERROR\_ROUTE\_NOT\_FOUND | db | 'The route was not found!',13,10,'$' |
| ERROR\_ALLOC | db | 'Failed to allocate memory to load overlay!',13,10,'$' |
| BLOCK\_ADDR | dw | Снгментный адрес блока освобожденной для оверлейного модуля памяти |
| CALL\_ADDR | dw | Адрес, по которому вызывается оверлейный модуль |
| ERROR\_OVL\_LOAD | db | 'The overlay was not been loaded: ' |
| ERROR\_NON\_EXIST\_FUNC | db | 'a non-existent function!',13,10,'$' |
| ERROR\_TOO\_MANY\_FILSE | db | 'too many open files!',13,10,'$' |
| ERROR\_NO\_ACCESS | db | 'no access!',13,10,'$' |
| ERROR\_LOW\_MEMORY | db | 'low memory!',13,10,'$' |
| ERROR\_INCOR\_ENV | db | 'incorrect environment!',13,10,'$' |
| OVL1 | db | Относительный путь до первого оверлейного модуля |
| OVL2 | db | Относительный путь до второго оверлейного модуля |

**Тестирование.**

1. Запуск программы с оверлейными модулями в той же директории.

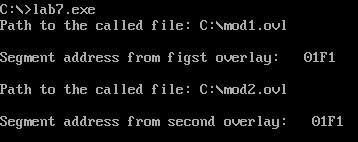
****

Рис.1 Результат работы программы lab7.exe.

1. Запуск программы из другой директории.

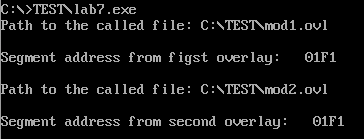
****

Рис.2 Результат работы программы lab7.exe при запуске из другой директории.

1. Запуск программы при отсутствии одного из оверлейных модулей.

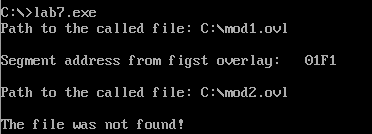


Рис.3 Результат работы программы lab7.exe при отсутствии одного оверлейного модуля.

**Ответы на контрольные вопросы.**

1. Как должна быть устроена программа, если в качестве оверлейного сегмента использовать .COM модули?

Ответ: В памяти, отведенной для оверлейного модуля необходимо перед загрузкой в нее модуля оставить 100h свободного места и в первые байты этой памяти поместить код безусловного перехода на начало .COM модуля, который находится через 100h байт далее в памяти.

**Вывод.**

В ходе выполнения лабораторной работы была изучена структура оверлейного модуля и метод вызова его из программы.

**Приложение 1. Код программы.**

**lab7.asm**

ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK

DATA SEGMENT

ERROR\_FREEING db 'Error when freeing memory: $'

ERROR\_MCM db 'MCB is destroyed$'

ERROR\_NO\_MEM db 'Not enough memory for function processing$'

ERROR\_WRONG\_ADDR db 'Wrong addres of memory block$'

ERROR\_UNKNOWN db 'Unknown error$'

STRENDL db 13,10,'$'

OVL\_PATH db 64 dup (0), '$'

DTA db 43 DUP (?)

KEEP\_PSP dw 0

PATH\_TO db 'Path to the called file: ','$'

ERROR\_FILE\_NOT\_FOUND db 'The file was not found!',13,10,'$'

ERROR\_ROUTE\_NOT\_FOUND db 'The route was not found!',13,10,'$'

ERROR\_ALLOC db 'Failed to allocate memory to load overlay!',13,10,'$'

BLOCK\_ADDR dw 0

CALL\_ADDR dd 0

ERROR\_OVL\_LOAD db 'The overlay was not been loaded: '

ERROR\_NON\_EXIST\_FUNC db 'a non-existent function!',13,10,'$'

ERROR\_TOO\_MANY\_FILSE db 'too many open files!',13,10,'$'

ERROR\_NO\_ACCESS db 'no access!',13,10,'$'

ERROR\_LOW\_MEMORY db 'low memory!',13,10,'$'

ERROR\_INCOR\_ENV db 'incorrect environment!',13,10,'$'

OVL1 db 'mod1.ovl',0

OVL2 db 'mod2.ovl',0

DATA ENDS

;---------------------------------------------------------------

CODE SEGMENT

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

;--------------------------------------------------------------------------------

FREE\_MEM PROC

mov bx,ss

add bx,10h

mov ax,es

sub bx, ax

mov ah,4ah

int 21h

jnc MEM\_FREED

mov dx,offset ERROR\_FREEING

call PRINT

cmp ax,7

mov dx,offset ERROR\_MCM

je FREE\_MEM\_PRINT\_ERROR

cmp ax,8

mov dx,offset ERROR\_NO\_MEM

je FREE\_MEM\_PRINT\_ERROR

cmp ax,9

mov dx,offset ERROR\_WRONG\_ADDR

je FREE\_MEM\_PRINT\_ERROR

mov dx,offset ERROR\_UNKNOWN

FREE\_MEM\_PRINT\_ERROR:

call PRINT

mov dx,offset STRENDL

call PRINT

xor AL,AL

mov AH,4Ch

int 21H

MEM\_FREED:

ret

FREE\_MEM ENDP

;---------------------------------------------------------------

FIND\_PATH PROC

push ds

push dx

mov dx, seg DTA

mov ds, dx

mov dx,offset DTA

mov ah,1Ah

int 21h

pop dx

pop ds

push es

push dx

push ax

push bx

push cx

push di

push si

mov es, KEEP\_PSP

mov ax, es:[2Ch]

mov es, ax

xor bx, bx

COPY\_CONT:

mov al, es:[bx]

cmp al, 0h

je STOP\_COPY\_CONT

inc bx

jmp COPY\_CONT

STOP\_COPY\_CONT:

inc bx

cmp byte ptr es:[bx], 0h

jne COPY\_CONT

add bx, 3h

mov si, offset OVL\_PATH

COPY\_PATH:

mov al, es:[bx]

mov [si], al

inc si

cmp al, 0h

je STOP\_COPY\_PATH

inc bx

jmp COPY\_PATH

STOP\_COPY\_PATH:

sub si, 9h

mov di, bp

ENTRY\_WAY:

mov ah, [di]

mov [si], ah

cmp ah, 0h

je STOP\_ENTRY\_WAY

inc di

inc si

jmp ENTRY\_WAY

STOP\_ENTRY\_WAY:

mov dx, offset PATH\_TO

call PRINT

mov dx, offset OVL\_PATH

call PRINT

mov dx, offset STRENDL

call PRINT

pop si

pop di

pop cx

pop bx

pop ax

pop dx

pop es

ret

FIND\_PATH ENDP

;---------------------------------------------------------------

FIND\_OVL\_SIZE PROC

push ds

push dx

push cx

xor cx, cx

mov dx, seg OVL\_PATH

mov ds, dx

mov dx, offset OVL\_PATH

mov ah,4Eh

int 21h

jnc FILE\_FOUND

cmp ax,3

je Error3

mov dx, offset ERROR\_FILE\_NOT\_FOUND

jmp EXIT\_FILE\_ERROR

Error3:

mov dx, offset ERROR\_ROUTE\_NOT\_FOUND

EXIT\_FILE\_ERROR:

call PRINT

pop cx

pop dx

pop ds

xor al,al

mov ah,4Ch

int 21H

FILE\_FOUND:

push es

push bx

mov bx, offset DTA

mov dx,[bx+1Ch]

mov ax,[bx+1Ah]

mov cl,4h

shr ax,cl

mov cl,12

sal dx, cl

add ax, dx

inc ax

mov bx,ax

mov ah,48h

int 21h

jnc SUCSESS\_ALLOC

mov dx, offset ERROR\_ALLOC

call PRINT

xor al,al

mov ah,4Ch

int 21h

SUCSESS\_ALLOC:

mov BLOCK\_ADDR, ax

pop bx

pop es

pop cx

pop dx

pop ds

ret

FIND\_OVL\_SIZE ENDP

;---------------------------------------------------------------

CALL\_OVL PROC

push dx

push bx

push ax

mov bx, seg BLOCK\_ADDR

mov es, bx

mov bx, offset BLOCK\_ADDR

mov dx, seg OVL\_PATH

mov ds, dx

mov dx, offset OVL\_PATH

mov ax, 4B03h

int 21h

push dx

jnc OVL\_NO\_ERROR

mov dx, offset ERROR\_OVL\_LOAD

call PRINT

cmp ax, 1

mov dx, offset ERROR\_NON\_EXIST\_FUNC

je OVL\_ERROR\_PRINT

cmp ax, 2

mov dx, offset ERROR\_FILE\_NOT\_FOUND

je OVL\_ERROR\_PRINT

cmp ax, 3

mov dx, offset ERROR\_ROUTE\_NOT\_FOUND

je OVL\_ERROR\_PRINT

cmp ax, 4

mov dx, offset ERROR\_TOO\_MANY\_FILSE

je OVL\_ERROR\_PRINT

cmp ax, 5

mov dx, offset ERROR\_NO\_ACCESS

je OVL\_ERROR\_PRINT

cmp ax, 8

mov dx, offset ERROR\_LOW\_MEMORY

je OVL\_ERROR\_PRINT

cmp ax, 10

mov dx, offset ERROR\_INCOR\_ENV

je OVL\_ERROR\_PRINT

mov dx, offset ERROR\_UNKNOWN

OVL\_ERROR\_PRINT:

call PRINT

jmp OVL\_RET

OVL\_NO\_ERROR:

mov AX,DATA

mov DS,AX

mov ax, BLOCK\_ADDR

;add ax,10

mov word ptr CALL\_ADDR+2, ax

call CALL\_ADDR

mov ax, BLOCK\_ADDR

mov es, ax

mov ax, 4900h

int 21h

mov AX,DATA

mov DS,AX

OVL\_RET:

pop dx

mov es, KEEP\_PSP

pop ax

pop bx

pop dx

ret

CALL\_OVL ENDP

;---------------------------------------------------------------

MAIN PROC FAR

mov ax,DATA

mov ds,ax

mov KEEP\_PSP, ES

call FREE\_MEM

mov bp, offset OVL1

call FIND\_PATH

call FIND\_OVL\_SIZE

call CALL\_OVL

mov bp, offset OVL2

call FIND\_PATH

call FIND\_OVL\_SIZE

call CALL\_OVL

xor al,al

mov ah,4Ch

int 21h

MAIN ENDP

CODE ENDS

STACK SEGMENT STACK

DW 80h DUP (?)

STACK ENDS

END MAIN

**mod1.asm**

ASSUME CS:OVL2,DS:OVL2,SS:NOTHING,ES:NOTHING

OVL2 SEGMENT

MAIN PROC FAR

push ds

push dx

push di

push ax

mov ax,cs

mov ds,ax

mov bx, offset STRING

add bx, 41

mov di, bx

mov ax, cs

call WRD\_TO\_HEX

mov dx, offset STRING

call PRINT

pop ax

pop di

pop dx

pop ds

retf

MAIN ENDP

;---------------------------------------------------------------

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

;--------------------------------------------------------------------------------

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

xor ah, ah

call BYTE\_TO\_HEX

mov [di], ah

dec di

mov [di], al

pop bx

ret

WRD\_TO\_HEX ENDP

;--------------------------------------------------------------------------------

STRING db 'Segment address from figst overlay: ',13,10,13,10,'$'

;--------------------------------------------------------------------------------

OVL2 ENDS

END MAIN

**mod2.asm**

ASSUME CS:OVL2,DS:OVL2,SS:NOTHING,ES:NOTHING

OVL2 SEGMENT

MAIN PROC FAR

push ds

push dx

push di

push ax

mov ax,cs

mov ds,ax

mov bx, offset STRING

add bx, 42

mov di, bx

mov ax, cs

call WRD\_TO\_HEX

mov dx, offset STRING

call PRINT

pop ax

pop di

pop dx

pop ds

retf

MAIN ENDP

;---------------------------------------------------------------

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

;--------------------------------------------------------------------------------

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

xor ah, ah

call BYTE\_TO\_HEX

mov [di], ah

dec di

mov [di], al

pop bx

ret

WRD\_TO\_HEX ENDP

;--------------------------------------------------------------------------------

STRING db 'Segment address from second overlay: ',13,10,13,10,'$'

;--------------------------------------------------------------------------------

OVL2 ENDS

END MAIN