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. 8382 _____ Чирков С.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование возможности построения загрузочного модуля оверлейной структуры. Исследуется структура оверлейного сегмента и способ загрузки и выполнения оверлейных сегментов. Для запуска вызываемого модуля используется функция 4B03h прерывания int 21h. Все загрузочные и оверлейные модули находятся в одном каталоге.

В этой работе также рассматривается приложение, состоящее из нескольких модулей, поэтому все модули помещаются в один каталог и вызываются с использованием полного пути.

Выполнение работы.

В процессе выполнения лабораторной работы был написан и отлажен программный модуль типа .EXE, выполняющий следующие функции:

- Освобождает память для загрузки оверлеев.
- Читает размер файла оверлея и запрашивает объем памяти, достаточный для его загрузки.
- Файл оверлейного сегмента загружается и выполняется.
- Освобождается память, отведенная для оверлейного сегмента.
- Затем предыдущие действия выполняются для следующего оверлейного сегмента.

Также были написаны и отлажены оверлейные сегменты, которые выводят адрес сегмента их загрузки.

Результат работы программы в одном каталоге показан на рисунке 1. Результат работы программы при запуске из другого каталога показан на рисунке 2. На рисунках 3-4 показан вывод программы при отсутствии одного из оверлейных модулей.

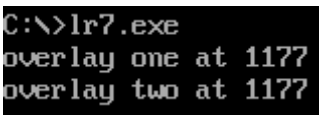

Рисунок 1. Запуск программы.

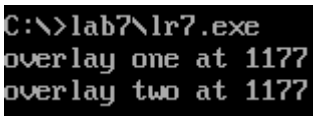

Рисунок 2. Запуск из другого каталога

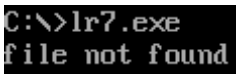

Рисунок 3. Запуск при отсутствии модуля 1

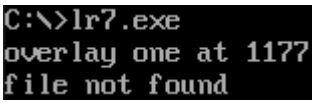

Рисунок 4. Запуск при отсутствии модуля 2

Контрольные вопросы.

1. Как должна быть устроена программа, если в качестве оверлейного сегмента использовать .com модуль?

Обязательно при использовании .com модуля необходимо учитывать наличие с нулевого адреса PSP, а с адреса 100h – самой функции с возвратом retf. Таким образом .com модули также можно использовать для оверлейных сегментов.

Выводы.

В ходе работы была исследована возможность построения загрузочного модуля оверлейной структуры.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ LR1COM.ASM

```
AStack SEGMENT STACK 'STACK'
DW 80h DUP(?)
AStack ENDS

DATA SEGMENT
err1 db 'memory control block ruined',10,13,'$'
err2 db 'not enough memory',10,13,'$'
err3 db 'wrong memory block address',10,13,'$'
DTA db 43 dup (?)
keep_psp dw 0
keep_ds dw 0
keep_ss dw 0
keep_sp dw 0
path db 80h dup (0), '$'
one db 'one.ovl',0
two db 'two.ovl',0
notfile db 'file not found',10,13,'$'
notpath db 'path not found',10,13,'$'
notmem db 'memory error',10,13,'$'
block dw 0
address dd 0
manyfiles db 'many files loaded',10,13,'$'
noaccess db 'no access',10,13,'$'
notexist db 'not exist',10,13,'$'
notenoughmemory db 'not enough memory',10,13,'$'
wrongenv db 'wrong environment',10,13,'$'
DATA ENDS

CODE SEGMENT
ASSUME SS:AStack,DS:DATA,CS:CODE

FREEMEM PROC near
mov bx, offset FLAG
mov ax, ds
sub bx, ax
mov cl, 4
shr bx, cl
mov ah, 4ah
```

```
int 21h
jc errfree
jmp endfree
errfree:
cmp ax, 7
je exc1
cmp ax, 8
je exc2
cmp ax, 9
je exc3
exc1:
mov dx, offset err1
jmp endexc
exc2:
mov dx, offset err2
jmp endexc
exc3:
mov dx, offset err3
endexc:
mov ah, 9
int 21h
xor ax, ax
mov ah, 4ch
int 21h
endfree:
ret
FREEMEM ENDP

PATHSTR PROC near
push dx
push ds
mov dx, seg DTA
mov ds, dx
mov dx, offset DTA
mov ah, 1ah
int 21h
pop ds
pop dx

mov es, keep_psp

push es
```

```
push dx
push si
push di
mov es, es:[2ch]
mov di, 0
loop1:
mov dl, es:[di]
cmp dl, 0
je loop2
inc di
jmp loop1
loop2:
inc di
mov dl, es:[di]
cmp dl, 0
jne loop1
add di, 3
mov si, offset path
pathloop:
mov dl, es:[di]
cmp dl, 0
je endpath
mov [si], dl
inc si
inc di
jmp pathloop
endpath:
mov di, cx
sub si, 7
notrdy:
mov dl, byte ptr [di]
mov byte ptr [si], dl
inc di
inc si
cmp dl, 0
jne notrdy

mov dl, '$'
mov byte ptr [si], dl
pop di
pop si
pop dx
```

```
pop es
ret
PATHSTR ENDP

OVERLAYSIZE PROC near
push ax
push cx
push dx
push es
push ds
push di
mov ax, seg path
mov ds, ax
mov dx, offset path
mov ah, 4eh
int 21h
jc err4e
mov di, offset DTA
mov ax, [di+1ah]
mov bx, [di+1ch]
mov cl, 4
shr ax, cl
mov cl, 12
shl bx, cl
add bx, ax
inc bx
mov ah, 48h
int 21h
jc err48h
mov block, ax
pop di
pop ds
pop es
pop dx
pop cx
pop ax
ret
err4e:
cmp ax, 2
je nofile
cmp ax, 3
je nopath
```

```
nofile:
mov dx, offset notfile
jmp quit
nopath:
mov dx, offset notpath
jmp quit
err48h:
mov dx, offset notmem
quit:
mov ah, 9
int 21h
pop di
pop ds
pop es
pop dx
pop cx
pop ax
mov al, 0
mov ah, 4Ch
int 21h
OVERLAYSIZE ENDP

OVERLAYRUN PROC near
push ax
push bx
push dx
push es

push es
mov keep_ds, ds
mov keep_ss, ss
mov keep_sp, sp
mov bx, seg block
mov es, bx
mov bx, offset block
mov dx, offset path
mov ah, 4bh
mov al, 3
int 21h
mov dx, keep_ds
mov ss, keep_ss
mov sp, keep_sp
```

```
pop es

jc errrun
mov ax, block
mov word ptr address+2, ax
call address
mov ax, block
mov es, ax
mov ah, 49h
int 21h
pop es
pop dx
pop bx
pop ax
mov es, keep_psp
ret
errrun:
cmp  ax, 1
je errrun1
cmp  ax, 2
je errrun2
cmp  ax, 3
je errrun3
cmp  ax, 4
je errrun4
cmp  ax, 5
je errrun5
cmp  ax, 8
je errrun8
cmp  ax, 10
je errrun10
errrun1:
mov dx, offset notexist
jmp quitovl
errrun2:
mov dx, offset notfile
jmp quitovl
errrun3:
mov dx, offset notpath
jmp quitovl
errrun4:
mov dx, offset manyfiles
```

```
jmp quitovl
errrun5:
mov dx, offset noaccess
jmp quitovl
errrun8:
mov dx, offset notenoughmemory
jmp quitovl
errrun10:
mov dx, offset wrongenv
quitovl:
mov ah, 9
int 21h
pop es
pop dx
pop bx
pop ax
mov es, keep_psp
ret
OVERLAYRUN ENDP

BEGIN PROC  far
mov bx, es
mov ax, DATA
mov ds, ax

mov keep_psp, es

call FREEMEM
mov cx, offset one
call PATHSTR
call OVERLAYSIZE
call OVERLAYRUN

xor bx, bx
mov cx, offset two
call PATHSTR
call OVERLAYSIZE
call OVERLAYRUN

xor AL,AL
mov AH,4Ch
int 21H
```

```
BEGIN      ENDP

CODE    ENDS
FLAG SEGMENT
FLAG ENDS
END     BEGIN
```