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. 8382 _____ Чирков С.А.

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.

Выполнение работы.

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

- 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.
- Вызываемый модуль запускается с использованием загрузчика.
- После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Проверяется причина завершения и, в зависимости от значения, выводится соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

В качестве вызываемой программы используется программа из лабораторной работы 2, которая распечатывает среду и командную строку.

Результат работы программы в одном каталоге при введении буквы английского алфавита показан на рисунке 1. Результат работы программы в одном каталоге при введении Ctrl-C показан на рисунке 2. На рисунках 3-4 показан вывод программы при работе в разных каталогах.

```
C:\>lr6.exe
Address of inaccessible memory is 9FFF
Address of program environment is 1177
Tail of command line is
Environment data:  PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path of file: C:\LR2.COM
a
program run normally
exit code - 61
```

Рисунок 1. Ввод буквы a.

```
C:\>lr6.exe
Address of inaccessible memory is 9FFF
Address of program environment is 1177
Tail of command line is
Environment data:  PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path of file: C:\LR2.COM
♥
program run with int 31h
```

Рисунок 2. Ввод Ctrl-Break

```
C:\>lab6\lr6.exe
Address of inaccessible memory is 9FFF
Address of program environment is 1177
Tail of command line is
Environment data:  PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path of file: C:\LAB6\LR2.COM
1
program run normally
exit code - 31
```

Рисунок 3. Запуск из другого каталога

```
C:\>lab6\lr6.exe
file not found
```

Рисунок 4. Запуск при модулях в разных каталогах

Контрольные вопросы.

1. Как реализовано прерывание Ctrl-C?

Вызывается обработчик Ctrl-C int 23h, передающий управление вызывающей программе.

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?

В точке вызова функции 1h прерывания int 21h.

Выводы.

В ходе работы была исследована возможность построения загрузочного модуля динамической 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,'$'
argblock dw 0
      dd 0
      dd 0
      dd 0
path db '                                        ',10,13,'$',0
keep_ss dw 0
keep_sp dw 0
keep_ds dw 0
er1 db 'wrong function number',10,13,'$'
er2 db 'file not found',10,13,'$'
er5 db 'disc error',10,13,'$'
er8 db 'insufficient amount of memory',10,13,'$'
er10 db 'wrong environment string',10,13,'$'
er11 db 'wrong formatting',10,13,'$'
norm0 db 10,13,'program run normally',10,13,'$'
norm1 db 10,13,'program run with ctrl-break',10,13,'$'
norm2 db 10,13,'program run with device error',10,13,'$'
norm3 db 10,13,'program run with int 31h',10,13,'$'
endprog db '    ',10,13,'$'
exit db 'exit code - $'
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 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:
sub si, 7
mov [si], byte ptr 'L'
mov  [si+1], byte ptr 'R'
mov  [si+2], byte ptr '2'
mov  [si+3], byte ptr '.'
mov  [si+4], byte ptr 'C'
mov  [si+5], byte ptr 'O'
mov  [si+6], byte ptr 'M'
mov [si+7], byte ptr 0
pop di
pop si
pop es
pop es
PATHSTR 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

BEGIN PROC  far
mov bx, es
mov ax, DATA
mov ds, ax
mov keep_ds, ds
mov keep_sp, sp
mov keep_ss, ss

call FREEMEM
call PATHSTR

push ds
mov keep_sp, sp
mov keep_ss, ss
pop es
mov bx, offset argblock
mov dx, offset path
mov ax, 4B00h
int 21h
mov bx, ax
mov ax, DATA
mov ds, ax
mov ax, bx
mov ds, keep_ds
mov ss, keep_ss
mov sp, keep_sp
jc errorint
mov ah, 4dh
int 21h
cmp ax, 0
je ok0
```

```
cmp ax, 1
je ok1
cmp ax, 2
je ok2
cmp ax, 3
je ok3
ok0:
mov dx, offset norm0
mov ah, 9
int 21h
mov di, offset endprog
call BYTE_TO_HEX
mov [di], al
inc di
mov [di], ah
mov dx, offset exit
mov ah, 9
int 21h
mov dx, offset endprog
jmp endbegin
ok1:
mov dx, offset norm1
jmp endbegin
ok2:
mov dx, offset norm2
jmp endbegin
ok3:
mov dx, offset norm3
jmp endbegin
errorint:
cmp ax, 1
je except1
cmp ax, 2
je except2
cmp ax, 5
je except5
cmp ax, 8
je except8
cmp ax, 10
je except10
cmp ax, 11
je except11
```

```
except1:
mov dx, offset er1
jmp endbegin
except2:
mov dx, offset er2
jmp endbegin
except5:
mov dx, offset er5
jmp endbegin
except8:
mov dx, offset er8
jmp endbegin
except10:
mov dx, offset er10
jmp endbegin
except11:
mov dx, offset er11
endbegin:
mov ah, 9
int 21h
xor AL,AL
mov AH,4Ch
int 21H
BEGIN      ENDP

CODE    ENDS
FLAG SEGMENT
FLAG ENDS
END     BEGIN
```