

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

Студентка гр. 6383

Иолшина В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. Также нужно исследовать интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция 4B00h прерывания int 21h.

Таблица 1 - Описание процедур.

Название процедуры	Назначение
PRINT	вызывает функцию печати строки
BYTE_TO_HEX	переводит байтовое число из регистра AL в шестнадцатеричную систему счисления, записывая получившееся в al и ah
TETR_TO_HEX	вспомогательная функция для работы функции BYTE_TO_HEX, которая переводит из двоичной в шестнадцатеричную систему счисления
DEAL	запускает вызываемый модуль
PAR_BL	создаёт блок параметров
MEMORY_	освобождает лишнюю память

Действия, выполняемые программой.

1. Освобождение память и обработка ошибок, которые могли произойти.
2. Создание блоков параметров.
3. Запуск вызываемого модуля.
4. Вывод кода завершения при отсутствии ошибок, при их наличии – их обработка.

Результаты выполнения программы представлены на рис. 1-6.

1. Запуск программы lr6.asm, вызывающей программу lr2.com, которая останавливается, ожидая ввод символа. Действие изображено на рис. 1.

```
K:\>lr6
Address of unavailable memory:9FFF
Address of environment: 02C9
Tail:
Content of the environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
PATH of the loadable module:
```

Рисунок 1 – Результат выполнения программы lr6.exe

2. На рис. 2 представлен ввод символа.

```
K:\LAB6>lr6
Address of unavailable memory:9FFF
Address of environment: 02CA
Tail:
Content of the environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
PATH of the loadable module:
K:\LAB6\LR2.COM
Everything is fine!
Code of finish: 46
```

Рисунок 2 – Результат выполнения программы lr6.exe

3. Запустим программу снова, теперь введём сочетание клавиш Ctrl+C. Действие программы показано на рис. 3.

```
K:\LAB6>lr6
Address of unavailable memory:9FFF
Address of environment: 02CA
Tail:
Content of the environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
PATH of the loadable module:
K:\LAB6\LR2.COM
Everything is fine!
Code of finish: 03
```

Рисунок 3 – Результат выполнения программы lr6.exe

4. Запустим программу lr6.exe, когда оба файла не находятся в текущем каталоге. Выполнение программы lr6.exe показано на рис. 4.

```

K:\LAB6>cd ..
K:\>\LAB6\lr6
Address of unavailable memory:9FFF
Address of environment: 02CA
Tail:
Content of the environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
PATH of the loadable module:
K:\LAB6\LR2.COM
Everything is fine!
Code of finish: 46

```

Рисунок 4 – Результат выполнения программы lr6.exe

5. Запустим программу снова, теперь введём сочетание клавиш Ctrl+C. Действие программы показано на рис. 5.

```

K:\>\LAB6\lr6
Address of unavailable memory:9FFF
Address of environment: 02CA
Tail:
Content of the environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
PATH of the loadable module:
K:\LAB6\LR2.COM
Everything is fine!
Code of finish: 03

```

Рисунок 5 – Результат выполнения программы lr6.exe

6. Запуск программы, когда файлы находятся в разных каталогах. Результат выполнения на рис. 6.

```

K:\>cd LAB6
K:\LAB6>lr6
Error of the file

```

Рисунок 6 – Результат повторного выполнения программы lr6.exe

Выводы.

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры, а также реализовано и исследовано взаимодействие между вызывающим и вызываемым модулями.

Ответы на контрольные вопросы.

1. Как реализовано прерывание Ctrl+C?

При нажатии сочетания клавиш Ctrl+C, вызывается прерывание int 23h, при этом управление передаётся по адресу 0000:008Ch. Данный адрес копируется в PSP функциями DOS 26h (создать PSP) и 4Ch (EXEC), а затем восстанавливается при выходе из программы.

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

В таком случае программа заканчивается в точке вызова функции 4Ch прерывания int 21h.

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

Программа заканчивается в месте вызова функции 01h прерывания 21h, то есть там, где ожидается ввод символа.

ПРИЛОЖЕНИЕ А

6.asm

CODE SEGMENT

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

START: JMP BEGIN

PRINT PROC near

mov AH,09h

int 21h

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

FUNC_ PROC

mov ax,STACK

sub ax,CODE

add ax,100h

mov bx,ax

mov ah,4ah

int 21h

jnc stepF

```

        call DEAL

stepF:
call PAR_BL
push es
push bx
push si
push ax
mov es,es:[2ch]
mov bx,-1
stepS:
        add bx,1
        cmp word ptr es:[bx],0000h
        jne stepS
add bx,4
mov si,-1
stepT:
        add si,1
        mov al,es:[bx+si]
        mov _PATH[si],al
        cmp byte ptr es:[bx+si],00h
        jne stepT
add si,1
stepT2:
        mov _PATH[si],0
        sub si,1
        cmp byte ptr es:[bx+si],'\\'
        jne stepT2
add si,1
mov _PATH[si],'I'
add si,1
mov _PATH[si],'r'
add si,1
mov _PATH[si],'2'
add si,1
mov _PATH[si],','
add si,1
mov _PATH[si],'C'
add si,1
mov _PATH[si],'O'
add si,1
mov _PATH[si],'M'
pop ax
pop si
pop bx
pop es
ret

```

FUNC_ ENDP

MEMORY_ PROC

```
    mov ax,STACK
    mov bx,es
    sub ax,bx
    add ax,10h
    mov bx,ax
    mov ah,4Ah
    int 21h
    jnc FIN
```

```
    mov dx,offset ERRMEM
    call PRINT
    cmp ax,7
    mov dx,offset MCB_
    je MEM_PRINT
    cmp ax,8
    mov dx,offset NOTMEM
    je MEM_PRINT
    cmp ax,9
    mov dx,offset ERRADR
```

```
MEM_PRINT:
    call PRINT
    mov dx,offset STRING
    call PRINT
```

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

FIN:

ret

MEMORY_ ENDP

PAR_BL PROC

```
    mov ax, es:[2Ch]
    mov PARAM_S,ax
    mov PARAM_S+2,es
    mov PARAM_S+4,80h
    ret
```

PAR_BL ENDP

DEAL PROC

```
    lea dx, _PATH
        xor ch,ch
        mov cl,es:[80h]
        cmp cx,0
        je UNTAIL
        mov si,cx
        push si
            METOCHKA:
                mov al,es:[81h+si]
                mov [offset _PATH+si-1],al
                dec si
        loop METOCHKA
        pop si
        mov [_PATH+si-1],0
        mov dx,offset _PATH
    UNTAIL:
        push ds
        pop es
        mov bx,offset PARAM_S
        mov KEEP_SP, SP
        mov KEEP_SS, SS
        mov ax,4b00h
        int 21h
        jnc FIN_
        push ax
        mov ax,DATA
        mov ds,ax
        pop ax
        mov SS,KEEP_SS
        mov SP,KEEP_SP
        cmp ax,1
        mov dx,offset ERRFUNCT
        je DEAL_PRINT
        cmp ax,2
        mov dx,offset ERRFILE
        je DEAL_PRINT
        cmp ax,5
        mov dx,offset ERRDISK
        je DEAL_PRINT
        cmp ax,8
        mov dx,offset NOTMEM_
        je DEAL_PRINT
        cmp ax,10
        mov dx,offset ERRENV
        je DEAL_PRINT
```

```

        cmp ax,11
        mov dx,offset ERRFORM
        DEAL_PRINT:
        call PRINT
        mov dx,offset STRING
        call PRINT
        xor AL,AL
        mov AH,4Ch
        int 21H
FIN_:
        mov ax,4d00h
        int 21h
        cmp ah,0
        mov dx,offset GOOD
        je REASONS
        cmp ah,1
        mov dx,offset END_CTRL
        je REASONS
        cmp ah,2
        mov dx,offset ERRDEVICE
        je REASONS
        cmp ah,3
        mov dx,offset ERRRES
        REASONS:
                call PRINT
                mov dx,offset STRING
                call PRINT
        mov dx,offset CODEELEM
        call PRINT
        call BYTE_TO_HEX
        push ax
        mov ah,02h
        mov dl,al
        int 21h
        pop ax
        xchg ah,al
        mov ah,02h
        mov dl,al
        int 21h
        mov dx,offset STRING
        call PRINT
        ret
DEAL ENDP

```

```

BEGIN:

```

```

        mov ax,DATA
        mov ds,ax
        call MEMORY_
        call FUNC_
        call DEAL
        xor AL,AL
        mov AH,4Ch
        int 21H
CODE ENDS

```

DATA SEGMENT

```

ERRMEM                db 'Error of clear memory: $'
MCB_                  db 'MCB is destroyed$'
NOTMEM                db 'Not enough memory$'
ERRADR                db 'Error of the address$'
ERRFUNCT              db 'Error of function number$'
ERRFILE               db 'Error of the file$'
ERRDISK               db 'Error of the disk$'
NOTMEM_               db 'Not enough memory$'
ERRENV                db 'Error of env$'
ERRFORM               db 'Error of format$'
GOOD                  db 0DH, 0AH, 'Everything is fine!$'
END_CTRL              db 'End ctrl$'
ERRDEVICE             db 'Error of device$'
ERRRES                db 'End 31h$'
CODEELEM              db 'Code of finish: $'
STRING                db 0DH,0AH,$'
PARAM_S               dw 0
                        dd 0
                        dd 0
                        dd 0
_PATH                 db 50h dup ('$')
KEEP_SS               dw 0
KEEP_SP               dw 0

```

DATA ENDS

STACK SEGMENT STACK

```

        dw 64h dup (?)

```

STACK ENDS

END START