

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. 9382
Преподаватель

Круглова В.Д.
Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. В отличие 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
Inaccessible memory: 9FFF
Enviroment address: 020C
Command line tail:
Enviroment: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path: C:\LR2.COM
e
Normal completion
Program ended with code: 65
```

Рисунок 1. Ввод буквы а.

```
C:\>LR6.EXE
Inaccessible memory: 9FFF
Enviroment address: 020C
Command line tail:
Enviroment: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path: C:\LR2.COM
*
Completion by Ctrl-Break
```

Рисунок 2. Ввод Ctrl-Break

```
C:\>LR6\LR6.EXE
Inaccessible memory: 9FFF
Enviroment address: 020C
Command line tail:
Enviroment: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path: C:\LR6\LR2.COM
q
Normal completion
Program ended with code: 71
```

Рисунок 3. Запуск из другого каталога

```
C:\>LR6\LR6.EXE
File not found
```

Рисунок 4. Запуск при модулях в разных каталогах

Контрольные вопросы.

1. Как реализовано прерывание Ctrl-C?

DOS выполняет команду INT 23h, если распознает, что была нажата клавиша Ctrl+C. В векторе 23h находится адрес программы, который завершает текущий процесс. Исключения: функции 06h и 07h, нечувствительные к Ctrl+C.

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, которое ожидает символ.

Выводы.

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ LR1COM.ASM

```
ASTACK SEGMENT STACK
DW 100h DUP(?)
ASTACK ENDS
```

```
DATA SEGMENT
```

```
FILENAME db 'LR2.COM', 0
PARAM_BLOCK dw 0
            dd 0
            dd 0
            dd 0

KEEP_SS    dw 0
KEEP_SP    dw 0
KEEP_PSP   dw 0
ERR7_MEM   db 13,10,'Memory control block is destroyed',13, 10,'$'
ERR8_MEM   db 13,10,'Not enough memory for function', 13, 10,'$'
ERR9_MEM   db 13,10,'Invalid adress',                  13, 10,'$'

ERR1_LOAD  db 13,10,'Incorrect function number',      13,
            10,'$'
ERR2_LOAD  db 13,10,'File not found',                  13, 10,'$'
ERR5_LOAD  db 13,10,'Disk error',                      13, 10,'$'
ERR8_LOAD  db 13,10,'Not enough memory',              13, 10,'$'
ERRA_LOAD  db 13,10,'Invalid environment',            13, 10,'$'
ERRB_LOAD  db 13,10,'Incorrect format',                13, 10,'$'

ERR0_ENDING db 13,10,'Normal completion$'
ERR1_ENDING db 13,10,'Completion by Ctrl-Break$'
ERR2_ENDING db 13,10,'Device error termination$'
ERR3_ENDING db 13,10,'Completion by function 31h$'

PATH       db 50 dup (0),'$'
COMPLETION db 13,10,'Program ended with code: $'
DATA_END   db 0
```

```
DATA ENDS
```

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA, SS:ASTACK
```

```
;-----
;print al in 16s/s
```

```

HEX_BYTE_PRINT PROC near
    push ax
    push bx
    push dx
    mov ah, 0
    mov bl,
    10h
    div bl
    mov dx, ax ; dl - первая цифра,
    dh - вторая mov ah, 2h cmp dl,
    0ah
    jl PRINT_1 ;если в dl - цифра
    add dl, 7 ;сдвиг в ASCII с цифр до
    букв PRINT_1:
    add dl,
    '0' int
    21h mov
    dl, dh
    cmp dl,
    0ah jl
    PRINT_2
    add dl, 7
    PRINT_2:
    add dl,
    '0' int
    21h pop
    dx pop
    bx pop
    ax
    ret
HEX_BYTE_PRINT ENDP
;-----
WriteMsg PROC
near    push ax
mov     ah,09h
int 21h pop ax
    ret
WriteMsg ENDP
;-----
FREE_MEM PROC near
    push ax
    push bx
    push cx
    push dx

    mov bx, offset _END
    mov ax, offset
    DATA_END add bx,
    ax add bx, 40Fh
    mov cl, 4 shr bx, cl
    mov ah,
    4Ah int 21h
    jnc   end_fm

```

```

irpc var1, 789          ;аналог
range-based c++      cmp ax, &var1&
                      je ERRM_&var1&
endm

```

```

irpc var2, 789
    ERRM_&var2&:
        mov dx, offset
ERR_&var2&_MEM    call
WriteMsg      mov ax, 4C00h
              int 21h
endm

```

```

end_fm:
pop dx
pop cx
pop bx
pop ax
ret
FREE_MEM ENDP

```

```

;-----

```

```

PREPAIR_DATA proc
near mov
ax,KEEP_PSP mov
es,ax

```

```

push ax
push es
push si
push di
push dx

```

```

mov es, es:
[2Ch] mov
si,0 lea di,
PATH
env_skip:
mov dl, es:
[si]
cmp dl, 00
je env_end
inc si
jmp env_skip
env_end:
inc si mov
dl, es:[si]
cmp dl, 00
jne
env_skip
add si, 3

```

```

write_path:
mov dl, es:
[si]
    cmp dl, 00
je
write_name
mov [di], dl
inc si
inc di
jmp
write_path
write_name:
mov si,0
file_name:
    mov dl, byte ptr
[FILENAME+si] mov
byte ptr [di-7], dl inc di
inc si test dl, dl jne
file_name

    mov KEEP_SS, ss
mov KEEP_SP, sp

    pop dx
pop di
pop si
pop es
pop ax
ret
PREPAIR_DATA ENDP
;-----
LOAD proc
near push
ax push bx
push dx
push ds
push ss
push sp
push ds
pop es

    mov bx, offset PARAM_BLOCK
    mov dx, offset PATH

    mov ah,
4bh mov
al, 0 int
21h

    jnc no_err
mov bx,
DATA mov
ds, bx mov

```



```

ss, KEEP_SS
mov sp,
KEEP_SP

irpc case, 1258AB
  cmp ax, 0&case&h je
  ERRL_&case&
endm

irpc met, 1258AB
  ERRL_&met&:
    mov dx, offset
  ERR&met&_LOAD call
  WriteMsg mov ax, 4C00h
    int 21h
endm

no_err:
mov ax,
4D00h int
21h

cmp al,3 ;код завершения при CTRL+C
je ctrl_c

irpc case, 0123
  cmp ah,
  &case& je
  ERRE_&case&
endm

irpc met, 0123
  ERRE_&met&:
    mov dx, offset ERR&met&_ENDING call WriteMsg jmp last_step
endm last_step: mov
dx,0 mov dx, offset
COMPLETION call
WriteMsg call
HEX_BYTE_PRINT
jmp __end
ctrl_c:
  mov dx, offset ERR1_ENDING
  call WriteMsg
__end:
pop sp
pop ss
pop ds
pop dx
pop bx
pop ax
ret
LOAD ENDP

```

```
MAIN proc far
mov ax, DATA
mov ds, ax
mov KEEP_PSP,
ES call
FREE_MEM call
PREPAIR_DATA
call LOAD mov
ax, 4C00h int
21h ret
MAIN endp
_END:
CODE ENDS

end MAIN
```