

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №6
по дисциплине «Операционные системы»
Тема: «Построение модуля динамической структуры»

Студент гр. 7381

Вологдин М.Д.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

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.

Описание функций:

PRINT	Процедура вызова прерывания, печатающего строку
BYTE_TO_HEX	Процедура перевода числа AL в коды символов 16-ой с/с, записывая получившееся в al и ah
FREE_MEM	Процедура освобождения лишней памяти
CREATE_PARAM_BLOCK	Процедура создания блока параметров
RUN_CHILD	Процедура запуска вызываемого модуля

Описание структур данных:

STR_ERR_*	Строки, оповещающие о различных ошибках
STR_NRML_END STR_CTRL_BREAK STR_DEVICE_ERROR	Строки, содержащие причины завершения дочерней программы

STR_RSDNT_END STR_UNKNWN	
STR_END_CODE	Строка, оповещающая, что далее следует код завершения
PARMBLOCK	Указатель на блок параметров
STD_CHILD_PATH	Стандартное имя вызываемого модуля – “LAB2.EXE”. Используется, если нет хвоста командной строки
CHILD_PATH	Строка, используемая для хранения имени вызываемого модуля, если есть хвост командной строки
KEEP_SS KEEP_SP	Переменные для сохранения регистров SS и SP перед вызовом модуля

Выполнение работы.

Был написан программный модуль .EXE, который выполняет следующие функции:

1. 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.
2. Вызываемый модуль запускается с использованием загрузчика.
3. После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Необходимо проверять причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

Тестирование

- 1) Запуск отлаженной программы, когда текущим каталогом является каталог с разработанными модулями и ввод произвольного символа из числа A-Z.

```
C:\>LAB6.EXE

Address of a segment with first byte of inaccessible memory: 9FFF
Address of an environment segment: 01E7
There is no tail

Environment contents:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

App path:
C:\LAB2.EXE
f
Normal end
End code: 66
```

- 2) Запуск отлаженной программы, когда текущим каталогом является каталог с разработанными модулями и ввод комбинации символов Ctrl-C. В DOSBox не поддерживается данная комбинация. Должно было быть выведено «End by Ctrl-Break».

```
C:\>LAB6.EXE

Address of a segment with first byte of inaccessible memory: 9FFF
Address of an environment segment: 01E7
There is no tail

Environment contents:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

App path:
C:\LAB2.EXE
```

- 3) Запуск отлаженной программы, когда текущим каталогом является другой каталог, отличный от того, где содержатся программные модули и ввод комбинация клавиш.

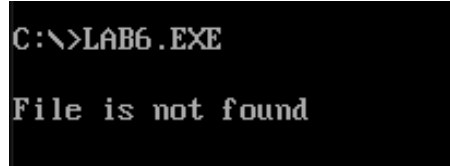
```
C:\>lab6\lab6.exe

Address of a segment with first byte of inaccessible memory: 9FFF
Address of an environment segment: 0246
There is no tail

Environment contents:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

App path:
C:\LAB2.EXE
w
Normal end
End code: 77
```

- 4) Запуск отлаженной программы, когда модули находятся в разных каталогах.



```
C:\>LAB6.EXE  
File is not found
```

Выводы

В результате выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

Ответы на контрольные вопросы

1. Как реализовано прерывание Ctrl-C?

Ответ: При нажатии сочетания клавиш Ctrl-C или Ctrl-Break вызывается прерывание int 23h, которое завершает текущий процесс, при этом управление передается по адресу 0000:008c.

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

Ответ: Если код причины завершения 0, то вызываемая программа заканчивается в месте вызова функции 4Ch прерываний int 21h.

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

Ответ: При нажатии сочетания клавиш Ctrl+C программа завершает работу в том месте, где программа ожидала ввода символа, т.е. в точке вызова функции 01h прерывания int 21h.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

CODE SEGMENT

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

START: JMP BEGIN

;-----

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

;-----

FREE_MEM PROC

MOV AX,STACKSEG

MOV BX,ES

SUB AX,BX

ADD AX,10H

MOV BX,AX

MOV AH,4AH

INT 21H

JNC FREE_MEM_SUCCESS

MOV DX,OFFSET STR_ERR_FREE_MEM

CALL PRINT

CMP AX,7

MOV DX,OFFSET STR_ERR_MCB_DESTROYED

JE FREE_MEM_PRINT_ERROR

CMP AX,8

MOV DX,OFFSET STR_ERR_NOT_ENOUGH_MEM

JE FREE_MEM_PRINT_ERROR

CMP AX,9

MOV DX,OFFSET STR_ERR_WRNG_MEM_BL_ADDR

FREE_MEM_PRINT_ERROR:

CALL PRINT

MOV DX,OFFSET STRENDL

CALL PRINT

XOR AL,AL

MOV AH,4CH

INT 21H

```

        FREE_MEM_SUCCESS:
        RET
FREE_MEM ENDP
;-----

```

```

CREATE_PARAM_BLOCK PROC
        MOV AX, ES:[2CH]
        MOV PARMBLOCK,AX
        MOV PARMBLOCK+2,ES
        MOV PARMBLOCK+4,80H
        RET
CREATE_PARAM_BLOCK ENDP
;-----

```

```

RUN_CHILD PROC
        MOV DX,OFFSET STRENDL
        CALL PRINT

```

```

        MOV DX,OFFSET STD_CHILD_PATH

```

```

        XOR CH,CH
        MOV CL,ES:[80H]
        CMP CX,0
        JE RUN_CHILD_NO_TAIL
        MOV SI,CX
        PUSH SI
        RUN_CHILD_LOOP:
            MOV AL,ES:[81H+SI]
            MOV [OFFSET CHILD_PATH+SI-1],AL
            DEC SI
        LOOP RUN_CHILD_LOOP
        POP SI

```



```

MOV [CHILD_PATH+SI-1],0
MOV DX,OFFSET CHILD_PATH
RUN_CHILD_NO_TAIL:

PUSH DS
POP ES
MOV BX,OFFSET PARMBLOCK

MOV KEEP_SP, SP
MOV KEEP_SS, SS


MOV AX,4B00H
INT 21H
JNC RUN_CHILD_SUCCESS


PUSH AX
MOV AX,DATA
MOV DS,AX
POP AX
MOV SS,KEEP_SS
MOV SP,KEEP_SP


CMP AX,1
MOV DX,OFFSET STR_ERR_WRNG_FNCT_NUMB
JE RUN_CHILD_PRINT_ERROR
CMP AX,2
MOV DX,OFFSET STR_ERR_FL_NOT_FND
JE RUN_CHILD_PRINT_ERROR
CMP AX,5
MOV DX,OFFSET STR_ERR_DISK_ERR
JE RUN_CHILD_PRINT_ERROR

```

```

CMP AX,8
MOV DX,OFFSET STR_ERR_NOT_ENOUGH_MEM2
JE RUN_CHILD_PRINT_ERROR
CMP AX,10
MOV DX,OFFSET STR_ERR_WRONG_ENV_STR
JE RUN_CHILD_PRINT_ERROR
CMP AX,11
MOV DX,OFFSET STR_ERR_WRONG_FORMAT
JE RUN_CHILD_PRINT_ERROR
MOV DX,OFFSET STR_ERR_UNKNWN
RUN_CHILD_PRINT_ERROR:
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

```

```

XOR AL,AL
MOV AH,4CH
INT 21H

```

```

RUN_CHILD_SUCCESS:
MOV AX,4D00H
INT 21H

```

```

CMP AH,0
MOV DX,OFFSET STR_NRML_END
JE RUN_CHILD_PRINT_END_RSN
CMP AH,1
MOV DX,OFFSET STR_CTRL_BREAK
JE RUN_CHILD_PRINT_END_RSN
CMP AH,2
MOV DX,OFFSET STR_DEVICE_ERROR
JE RUN_CHILD_PRINT_END_RSN
CMP AH,3

```

```

MOV DX,OFFSET STR_RSDNT_END
JE RUN_CHILD_PRINT_END_RSN
MOV DX,OFFSET STR_UNKNWN
RUN_CHILD_PRINT_END_RSN:
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

```

```

MOV DX,OFFSET STR_END_CODE
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 STRENDL
CALL PRINT

```

```

RET

```

```

RUN_CHILD ENDP

```

```

;-----

```

```

BEGIN:

```

```

MOV AX,DATA
MOV DS,AX

```

```

CALL FREE_MEM
CALL CREATE_PARAM_BLOCK
CALL RUN_CHILD

```

```

        XOR AL,AL
        MOV AH,4CH
        INT 21H
CODE ENDS

```

DATA SEGMENT

```

        STR_ERR_FREE_MEM          DB 'ERROR WHEN FREEING
MEMORY: $'

        STR_ERR_MCB_DESTROYED     DB 'MCB IS DESTROYED$'
        STR_ERR_NOT_ENOUGH_MEM    DB 'NOT ENOUGH MEMORY
FOR FUNCTION PROCESSING$'

        STR_ERR_WRNG_MEM_BL_ADDR  DB 'WRONG ADDRES OF MEMORY
BLOCK$'

        STR_ERR_UNKNWN            DB 'UNKNOWN ERROR$'

        STR_ERR_WRNG_FNCT_NUMB    DB 'FUNCTION NUMBER IS
WRONG$'

        STR_ERR_FL_NOT_FND        DB 'FILE IS NOT FOUND$'
        STR_ERR_DISK_ERR          DB 'DISK ERROR$'
        STR_ERR_NOT_ENOUGH_MEM2   DB 'NOT ENOUGH MEMORY$'
        STR_ERR_WRONG_ENV_STR     DB 'WRONG ENVIRONMENT
STRING$'

        STR_ERR_WRONG_FORMAT      DB 'WRONG FORMAT$'

        STR_NRML_END              DB 'NORMAL END$'
        STR_CTRL_BREAK            DB 'END BY CTRL-BREAK$'
        STR_DEVICE_ERROR          DB 'END BY DEVICE ERROR$'
        STR_RSDNT_END             DB 'END BY 31H FUNCTION$'
        STR_UNKNWN                DB 'END BY UNKNOWN REASON$'
        STR_END_CODE              DB 'END CODE: $'

```

```

STRENDL DB 0DH,0AH,'$'

PARMBLOCK DW 0
            DD ?
            DD 0
            DD 0

CHILD_PATH      DB 50H DUP ('$')
STD_CHILD_PATH  DB 'LAB2.EXE',0
KEEP_SS DW 0
KEEP_SP DW 0
DATA ENDS
; CTEK
STACKSEG SEGMENT STACK
            DW 80H DUP (?) ; 100H БАЙТ
STACKSEG ENDS
END START

```