МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №6

по дисциплине «Операционные системы»

Тема: Построение модуля динамической структуры

Студент гр. 8381 Муковский Д.В.

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.

Написан текст исходного EXE модуля, который выполняет 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. Необходимо проверять причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения. В качестве вызываемой программы была взята программа ЛР 2, которая распечатывает среду и командную строку.

Полученный модуль был отлажен. Далее отлаженная программа была запущена, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается,

ожидая символ с клавиатуры. Введен символ «q». Результаты выполнения программы представлены на рис. 1.

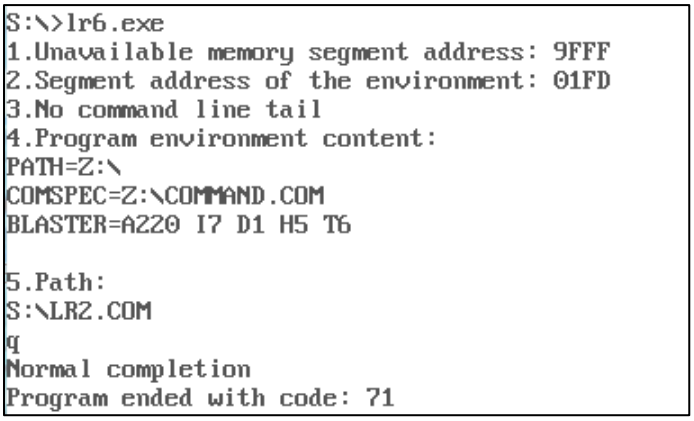

Рисунок 1 – Результат выполнения LR6.EXE, когда текущим каталогом является каталог с разработанными модулями.

Далее программа была запущена, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры. Введена комбинация Ctrl-C. Результаты выполнения представлены на рис. 2.

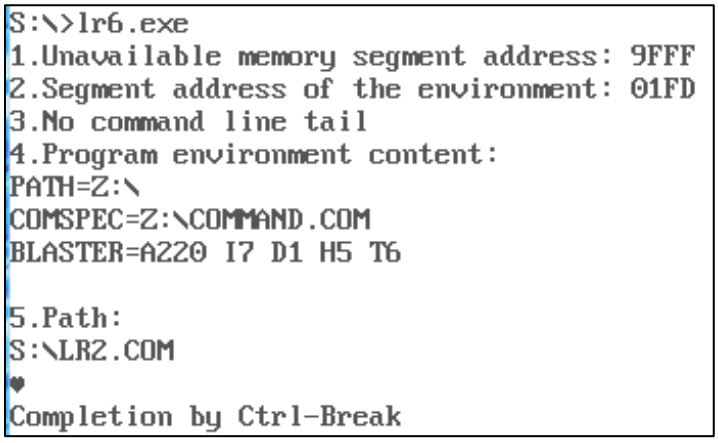

Рисунок 2 – Результат выполнения LR6.EXE, когда текущим каталогом является каталог с разработанными модулями и введена комбинация Ctrl-C.

Далее отлаженная программа была запущена, когда оба модуля находятся не в текущем каталоге. Введен символ «q». Результаты выполнения программы представлены на рис. 3.

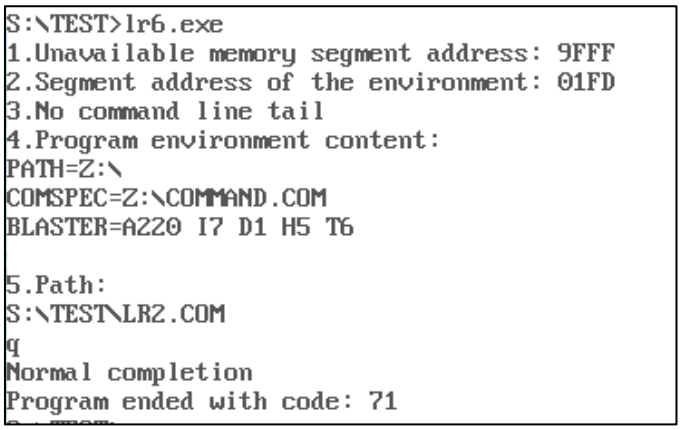

Рисунок 3 – Повторный запуск программы, когда оба модуля не в текущем каталоге и введен символ «q».

Далее отлаженная программа была запущена, когда оба модуля находятся не в текущем каталоге. Введена комбинация Ctrl-C. Результаты выполнения представлены на рис. 4.

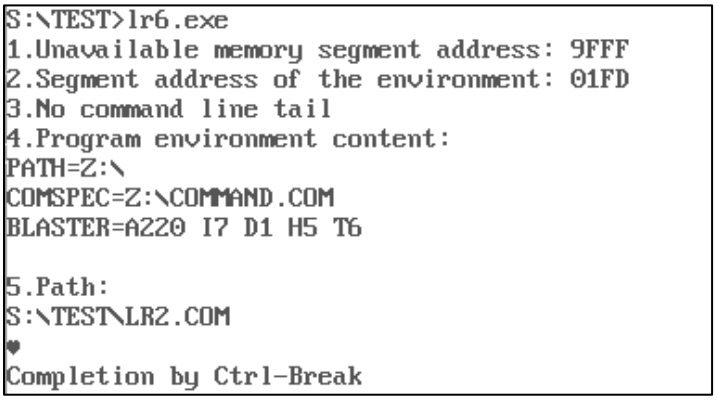

Рисунок 4 – Повторный запуск программы, когда оба модуля не в текущем каталоге и введена комбинация Ctrl-C.

Далее отлаженная программа была запущена, когда модули находятся в разных каталогах. Результаты выполнения представлены на рис. 5.

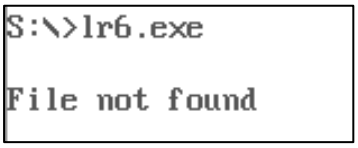

Рисунок 5 – Запуск программы, когда модули находятся в разных каталогах.

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. Но есть исключения, а именно: функции 06h и 07h, нечувствительные к Ctrl+C, и функции 02H и 09H, которые анализируют кольцевой буфер на предмет наличия там Ctrl+C один раз на каждые 64 вызова.

Далее адрес по вектору INT 23H копируется в поле PSP Ctrl-Break Address функциями DOS 26H (создает PSP) и 4CH (EXEC). Проверка на Ctrl+C осуществляется независимо от перенаправления ввода-вывода, а также независимо от состояния системного флага BREAK.

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

В месте вызова функции 4CH прерывания INT 21H.

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

В точке вызова функции 01h прерывание INT 21h, которое ожидает ввод символа с клавиатуры.

Вывод.

В ходе выполнения лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры, а также были получены знания по работе с интерфейсом между вызывающим и вызываемым модулями по управлению и по данным.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR6.ASM

```
DATA SEGMENT

      FILENAME db 'LR2.COM', 0
      PARAM_BLOCK dw 7 dup(0)
      SAVE_SS dw 0
      SAVE_SP dw 0
      SAVE_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$'
      ERR3_ENDING db 13,10,'Completion by function 31h$'

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

DATA  ENDS


ASTACK     SEGMENT  STACK
DW 256 DUP(?)
ASTACK  ENDS

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


;-------------------------------------------------------------------
-----------
;ВЫВОДИТ БАЙТ(AL) В 16 С/С
HEX_BYTE_PRINT PROC NEAR
           PUSH AX
           PUSH BX
           PUSH DX
           MOV AH, 0
           MOV BL, 10H
```

```
            DIV BL
            MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH - ВТОРАЯ
            MOV AH, 02H
            CMP DL, 0AH
            JL PRINT_1 ;ЕСЛИ В DL - ЦИФРА
            ADD DL, 7   ;СДВИГ В ASCII С ЦИФР ДО БУКВ
      PRINT_1:
            ADD DL, 48
            INT 21H
            MOV DL, DH
            CMP DL, 0AH
            JL PRINT_2
            ADD DL, 7
      PRINT_2:
            ADD DL, 48
            INT 21H;
            POP DX
            POP BX
            POP AX
            RET
HEX_BYTE_PRINT ENDP
;-------------------------------------------------------------------------
-----------
PRINT PROC NEAR
            PUSH AX
            MOV AH, 09H
            INT 21H
            POP AX
            RET
PRINT ENDP
;-------------------------------------------------------------------------
-----------


FREE_MEMORY     PROC NEAR
            push ax
            push bx
            push cx
            push dx
            push es
            push ds

            mov BX, offset PROGRAM_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_FUNC_FM

            irpc case, 789
                 cmp ax, &case&
                 je ERRM_&case&
            endm

            irpc met, 789
                 ERRM_&met&:
                 mov dx, offset ERR&met&_MEM
                 call PRINT
                 mov ax, 4C00h
                 int 21h
            endm

            END_FUNC_FM:

            pop ds
            pop es
            pop dx
            pop cx
            pop bx
            pop ax
      RET
FREE_MEMORY      ENDP
;-------------------------------------------------------------------------
-----------

LOAD proc near
            push ax
            push bx
            push cx
            push dx
            push es
            push ds
            push si
            push di
            push ss
            push sp

            mov es, es:[2Ch]
            mov si,0
            lea di, FILENAME_F

            SKIP_ENVIR:
                 mov dl, es:[si]
                 cmp dl, 00
                 je ENVIR_ENDING
                 inc si
                 jmp SKIP_ENVIR
```

```
ENVIR_ENDING:
     inc si
     mov dl, es:[si]
     cmp dl, 00
     jne SKIP_ENVIR
     add si, 3
PATH_S:
     mov dl, es:[si]
     cmp dl, 00
     je WRITE_NAME
     mov [di], dl
     inc si
     inc di
     jmp PATH_S
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 SAVE_SS, ss
mov SAVE_SP, sp

push ds
pop es

mov bx, offset PARAM_BLOCK
mov dx, offset FILENAME_F

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

jnc NO_LOAD_ERRORS
mov bx, DATA
mov ds, bx
mov SS, SAVE_SS
mov SP, SAVE_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 PRINT
            mov ax, 4C00h
            int 21h
      endm

      NO_LOAD_ERRORS:

      mov ax, 4D00h
      int 21h

      cmp al,3 ;код завершения при CTRL+C(сердечко), т.к. в DOSBOX
не работает настоящее прерывание
      je CTRLC_END


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

      irpc met, 0123
            ERRE_&met&:
                  mov dx, offset ERR&met&_ENDING
                  call PRINT
                  jmp LOAD_END
      endm
      LOAD_END:
      mov dx,0
      mov dx, offset COMPLETION
      call PRINT
      call HEX_BYTE_PRINT
      jmp POP_ALL


      CTRLC_END:
            mov dx, offset ERR1_ENDING
            call PRINT

      POP_ALL:
      pop sp
      pop ss
      pop di
      pop si
      pop ds
      pop es
      pop dx
      pop cx
      pop bx
      pop ax
```

```
          RET
LOAD ENDP

MAIN proc far
    push ds
    xor ax, ax
    push ax
    mov ax, DATA
    mov ds, ax
     mov SAVE_PSP, ES
     call FREE_MEMORY
     mov AX,SAVE_PSP
     mov ES,AX
     call LOAD
     mov ax, 4C00h
     int 21h
     ret
MAIN endp
PROGRAM_END:
CODE ENDS


end MAIN
```