

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

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Операционные системы»
Тема: Сопряжение стандартного и пользовательского обработчиков
прерываний

Студент гр. 9382

Кузьмин Д. И.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2021

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерывания получает управление по прерыванию (int 09h) при нажатии клавиши на клавиатуре. Он обрабатывает скан-код и осуществляет определенные действия, если скан-код совпадает с определенными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передается стандартному прерыванию.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет такие же функции, как в программе ЛР 4, а именно:

1) Проверяет, установлено ли пользовательское прерывание с вектором 09h.

2) Если прерывание не установлено то, устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний. Адрес точки входа в стандартный обработчик прерывания находится в теле пользовательского обработчика. Осуществляется выход по функции 4Ch прерывания int 21h.

3) Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h. Выгрузка прерывания по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h. Для того чтобы проверить установку прерывания, можно поступить следующим образом. Прочитать адрес, записанный в векторе прерывания. Предположим, что этот адрес указывает на точку входа в установленный резидент. На определенном, известном смещении в теле резидента располагается сигнатура, некоторый код,

который идентифицирует резидент. Сравнив известное значение сигнатуры с реальным кодом, находящимся в резиденте, можно определить, установлен ли 3 резидент. Если значения совпадают, то резидент установлен. Длину кода сигнатуры должна быть достаточной, чтобы сделать случайное совпадение маловероятным. Программа должна содержать код устанавливаемого прерывания в виде удаленной процедуры. Этот код будет работать после установки при возникновении прерывания. Он должен выполнять следующие функции:

- 1) Сохранить значения регистров в стеке при входе и восстановить их при выходе.
- 2) При выполнении тела процедуры анализируется скан-код.
- 3) Если этот код совпадает с одним из заданных, то требуемый код записывается в буфер клавиатуры.
- 4) Если этот код не совпадает ни с одним из заданных, то осуществляется передача управления стандартному обработчику прерывания.

Шаг 2. Запустите отлаженную программу и убедитесь, что резидентный обработчик прерывания 09h установлен. Работа прерывания проверяется введением различных символов, обрабатываемых установленным обработчиком и стандартным обработчиком.

Шаг 3. Также необходимо проверить размещение прерывания в памяти. Для этого запустите программу ЛР 3, которая отображает карту памяти в виде списка блоков МСВ. Полученные результаты поместите в отчет.

Шаг 4. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.

Шаг 5. Запустите отлаженную программу с ключом выгрузки и убедитесь, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3. Полученные результаты поместите в отчет.

Шаг 6. Ответьте на контрольные вопросы.

Выполнение работы.

1) Первым шагом было создание пользовательского обработчика прерываний, который срабатывает на каждую нажатую клавишу и в зависимости от нее либо обрабатывает ее (если нажата клавиша 1, то выводится символ D), либо переход к стандартному обработчику прерываний.

2) Далее была создана функция, проверяющая, установлено ли пользовательское прерывание и если нет, то устанавливает его, загружая при этом резидентную программу.

3) Проверка прерываний осуществляется с помощью функции 31h прерывания int 21h, а установка – с помощью функции 25h.

4) Далее была релизована проверка параметра /un, отвечающего за выгрузку прерывания. В случае, если параметр указан, при помощи функции 49h прерывания int 21h освобождается память, занимаемая резидентной программой и восстанавливается исходный вектор прерываний.

Исходный код см. в приложении Б

Результаты работы программы см. в приложении А

Контрольные вопросы.

1) Какого типа прерывания использовались в работе?

В работе использовались:

программные прерывания – int 21h

аппаратные прерывания – int 09h

2) Чем отличается скан код от кода ASCII??

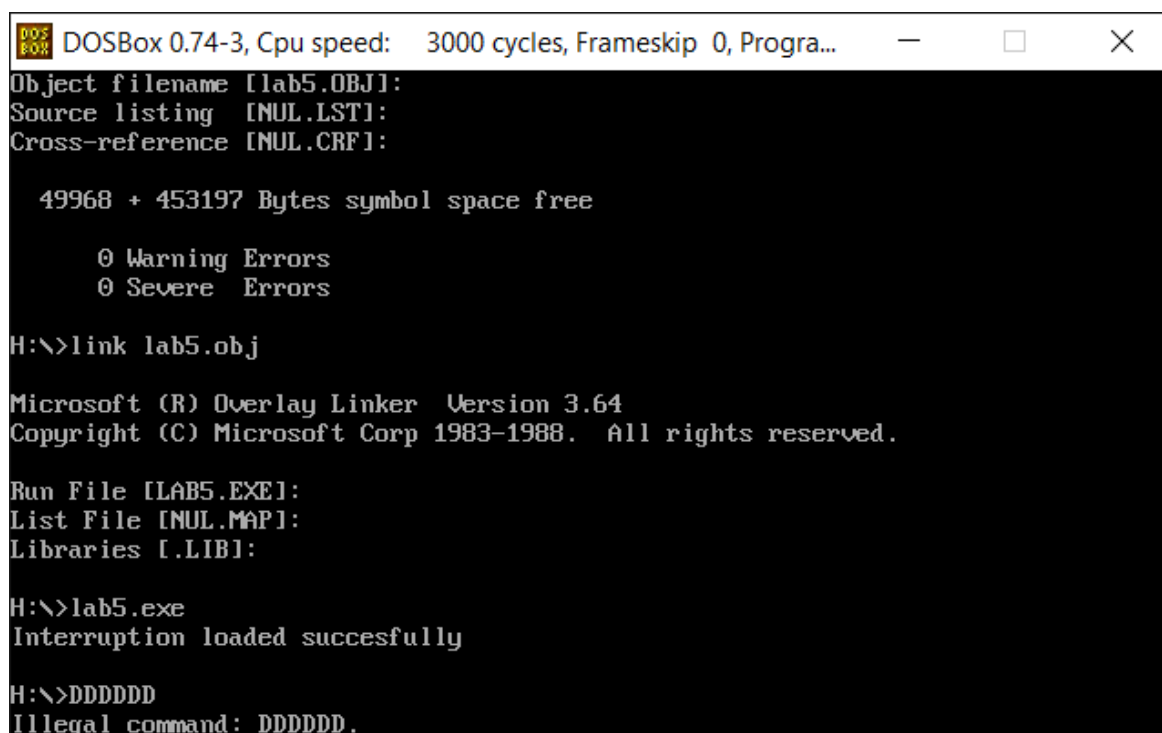
Скан код – это код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознает, какая клавиша была нажата(или отпущена) ASCII код – это уникальный код для каждого символа из таблицы ASCII. Например, для клавиши А скан-код нажатия – 1E, скан-код отпущения – 9E

ASCII код - 65.

Выводы.

Были получены навыки разработки программ, встраивающих пользовательский обработчик прерываний в стандартный обработчик от клавиатуры

ПРИЛОЖЕНИЕ А. ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ



```
DOS FOR DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Object filename [lab5.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

49968 + 453197 Bytes symbol space free

0 Warning Errors
0 Severe Errors

H:\>link lab5.obj

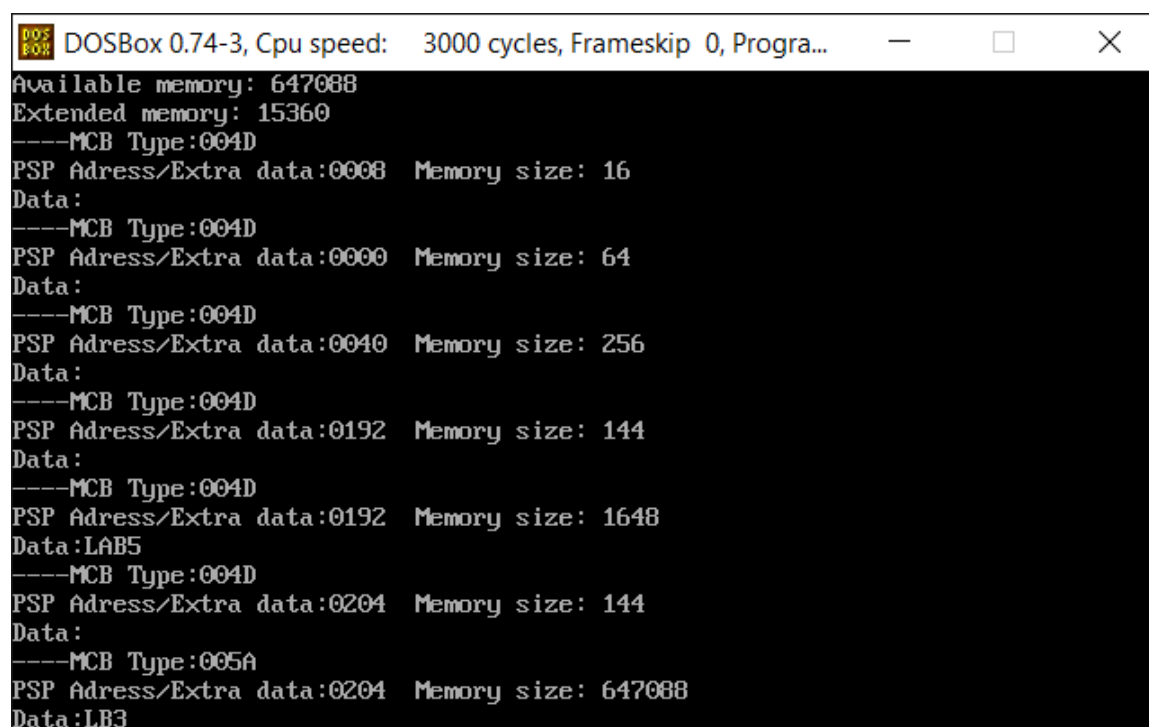
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run File [LAB5.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

H:\>lab5.exe
Interruption loaded succesfully

H:\>DDDDDD
Illegal command: DDDDDD.
```

Рис. 1 - Установка резидентного обработчика прерывания и демонстрация того, что он обрабатывает нажатые клавиши и заменяет '1' на 'D'



```
DOS FOR DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Available memory: 647088
Extended memory: 15360
----MCB Type:004D
PSP Address/Extra data:0008 Memory size: 16
Data:
----MCB Type:004D
PSP Address/Extra data:0000 Memory size: 64
Data:
----MCB Type:004D
PSP Address/Extra data:0040 Memory size: 256
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 144
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 1648
Data:LAB5
----MCB Type:004D
PSP Address/Extra data:0204 Memory size: 144
Data:
----MCB Type:005A
PSP Address/Extra data:0204 Memory size: 647088
Data:LB3
```

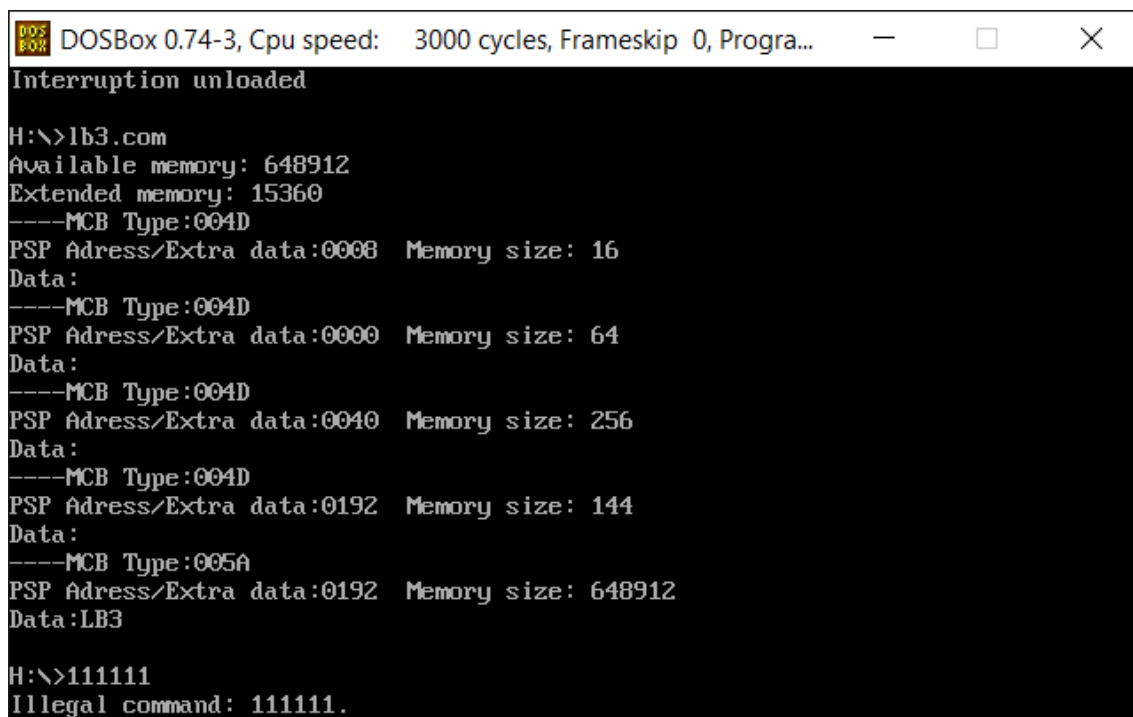
Рис. 2 - Состояние памяти после загрузки резидентной части

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
PSP Address/Extra data:0008 Memory size: 16
Data:
----MCB Type:004D
PSP Address/Extra data:0000 Memory size: 64
Data:
----MCB Type:004D
PSP Address/Extra data:0040 Memory size: 256
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 144
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 1648
Data:LAB5
----MCB Type:004D
PSP Address/Extra data:0204 Memory size: 144
Data:
----MCB Type:005A
PSP Address/Extra data:0204 Memory size: 647688
Data:LB3
H:\>lab5.exe
Interruption is already loaded
```

Рис. 3 - Определение уже установленного прерывания

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Interruption is already loaded
H:\>lab5.exe /un
Interruption unloaded
H:\>lb3.com
Available memory: 648912
Extended memory: 15360
----MCB Type:004D
PSP Address/Extra data:0008 Memory size: 16
Data:
----MCB Type:004D
PSP Address/Extra data:0000 Memory size: 64
Data:
----MCB Type:004D
PSP Address/Extra data:0040 Memory size: 256
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 144
Data:
----MCB Type:005A
PSP Address/Extra data:0192 Memory size: 648912
Data:LB3
```

Рис. 4- Выгрузка прерывания и состояние памяти после выгрузки.



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

Interruption unloaded

H:\>lb3.com

Available memory: 648912

Extended memory: 15360

-----MCB Type:004D

PSP Address/Extra data:0008 Memory size: 16

Data:

-----MCB Type:004D

PSP Address/Extra data:0000 Memory size: 64

Data:

-----MCB Type:004D

PSP Address/Extra data:0040 Memory size: 256

Data:

-----MCB Type:004D

PSP Address/Extra data:0192 Memory size: 144

Data:

-----MCB Type:005A

PSP Address/Extra data:0192 Memory size: 648912

Data:LB3

H:\>111111

Illegal command: 111111.

Рис. 4 - Отсутствие обработки клавиши 1

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД

Файл lab5.asm

```
ASTACK      SEGMENT      STACK
            DW 300 DUP (?)
ASTACK      ENDS

DATA SEGMENT
resident_set db 'Interruption is already loaded' , 13, 10, '$'
resident_not_set db 'Interruption loaded succesfully', 13, 10, '$'
unload db 'Interruption unloaded', 13, 10, '$'
param db ' /un'
psp dw 0
DATA ENDS
CODE SEGMENT
    .386
    ASSUME CS:CODE, DS:DATA, SS:ASTACK
; Процедуры
ROUT PROC FAR

    jmp start
    keep_ip dw 0
    keep_cs dw 0
    sign dw 0A35Fh
    keep_sp dw 0
    start_adress dw 0
    keep_ss dw 0
    keep_ax dw 0
    int9_vect dd 0
    REQ_KEY db 2h
    new_stack db 64 dup (?)

start:
    mov keep_sp, sp
    mov keep_ss, ss
    mov keep_ax, ax
    mov sp, offset new_stack
    add sp, 64
    mov ax, seg new_stack
    mov ss, ax

    push ax
    push cx
    push ds
    push es

    in al, 60h
    cmp al, REQ_KEY
    je do_req

    pop es
    pop ds
    pop cx
    pop ax
    mov ss, keep_ss
    mov sp, keep_sp
```

```

        mov ax, keep_ax

        jmp cs:[int9_vect]

do_req:
push ax
in al, 61h
mov ah, al
or al, 80h
out 61h, al
xchg ah, al
out 61h, al
mov al, 20h
out 20h, al
pop ax

;запись символа 'D' вместо '1'
write:
mov ah, 05h
mov cl, 'D'
int 16h
xor ch, ch
or al, al
jnz skip
jmp exrout

skip:
mov es, ax
mov al, es:[41AH]
mov es:[41CH], al
jmp write

exrout:
pop es
pop ds
pop cx
pop ax
mov ss, keep_ss
mov sp, keep_sp
mov ax, keep_ax
iret

ROUT ENDP
;-----
RESTORE_VECTOR PROC NEAR

        CLI
        PUSH DS
        push dx
        push ax
        MOV DX, es:keep_ip
        MOV AX, es:keep_cs
        MOV DS, AX
        MOV AH, 25H
        MOV AL, 09h
        INT 21H
        pop ax
        pop dx

```

```

        POP DS
        STI
        RET
RESTORE_VECTOR ENDP
end_rout:
WRITE_MSG PROC near
        push ax
        mov AH, 09h
        int 21h
        pop ax
        ret
WRITE_MSG ENDP

;-----
SET_INTERRUPT PROC NEAR

        mov ah, 35h
        mov al, 09h
        int 21h
        mov keep_cs, es
        mov keep_ip, bx
        mov word ptr int9_vect[02h], es
        mov word ptr int9_vect, bx

        PUSH DS
        PUSH AX
        PUSH DX
        MOV DX, OFFSET ROUT
        MOV AX, SEG ROUT

        MOV DS, AX
        MOV AH, 25H

        MOV AL, 09h
        INT 21H

        POP DX
        POP AX
        POP DS

        RET
SET_INTERRUPT ENDP

LOAD_TO_RESIDENT PROC NEAR

        push ax
        push bx
        push dx
        push cx

        mov dx, offset end_rout
        mov ax, cs
        add dx, ax
        shr dx, 4h

```

```

        add dx, 1Fh
        mov ah, 31h
        int 21h

        pop cx
        pop dx
        pop bx
        pop ax

        ret

LOAD_TO_RESIDENT ENDP
;-----
CHECK_VECTOR PROC NEAR

        PUSH AX
        PUSH BX
        PUSH ES
        PUSH SI

        MOV AH, 35H
        MOV AL, 09h
        INT 21H

        mov ax, 0A35Fh ;уникальное значение
        cmp ax, es:sign
        jne setres

        call CHECK_PARAM ;если установлено, переход к проверке
параметра /un
        jmp endthis

setres:      ;если прерывание не установлено
mov dx, offset resident_not_set
call WRITE_MSG
call SET_INTERRUPT
call LOAD_TO_RESIDENT

endthis:
POP SI
POP ES
POP BX
POP AX
RET
CHECK_VECTOR ENDP

UNLOAD_INTERRUPTION PROC NEAR

        call RESTORE_VECTOR

        mov ax, es:start_adress
        mov es, ax

        push es

```

```

        mov ax,es:[2ch]      ;среда
        mov es,ax
        mov ah,49h
        int 21h
        pop es

        mov ah,49h ;резидентная часть
        int 21h
        ret
UNLOAD_INTERRUPTION ENDP

CHECK_PARAM PROC NEAR

        push es
        mov es, start_adress
        mov cx, 4
        mov di, 81h
        mov si, offset param
        cld
        repe cmpsb
        jne notequal
        pop es
        call UNLOAD_INTERRUPTION
        mov dx, offset unload
        call WRITE_MSG
        jmp ex
notequal:
        pop es
        mov dx, offset resident_set
        call WRITE_MSG
ex:
        ret
CHECK_PARAM ENDP

MAIN      PROC FAR

        mov ax,DATA
        mov ds,ax
        mov psp, es
        mov start_adress, es
        call CHECK_VECTOR
        mov ah, 4ch
        int 21h

MAIN ENDP
CODE ENDS
        END MAIN

```