

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

ЛАБОРАТОРНАЯ РАБОТА № 5
по дисциплине «Операционные системы»
ТЕМА: Сопряжение стандартного и пользовательского обработчиков
прерываний.

Студентка гр. 9381

Москаленко Е.М.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерываний 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.

Ход работы.

1) Был написан и отлажен программный модуль типа .EXE, который выполняет такие же функции, как и в программе лабораторной работы №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 int21h.
3. Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.

Прерывание заменяет символы, вводимые с клавиатуры

(g → * k → !

c → ?).

Запустив модуль lab3, можно убедиться что пока прерывание не размещено в памяти:

```
0 Warning Errors
0 Severe Errors

F:\>link lab5

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

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

F:\>lab3_1
Available memory: 648912 b
Memory request failed
Extended memory: 15360 Kb
List of MCB:
MCB type: 4Dh PSP address: 0008h Size: 16 b
MCB type: 4Dh PSP address: 0000h Size: 64 b
MCB type: 4Dh PSP address: 0040h Size: 256 b
MCB type: 4Dh PSP address: 0192h Size: 144 b
MCB type: 4Dh PSP address: 0192h Size: 896 b LAB3_1
MCB type: 5Ah PSP address: 0000h Size: 648000 b -^b iF€iU

F:\>_
```

2) Запустила отлаженную программу и убедилась, что резидентный обработчик прерывания 09h установлен. Работа прерывания была проверена введением различных символов, обрабатываемых установленным обработчиком и стандартным обработчиком.

```
F:\>lab5
Interruption was loaded.

F:\>a!?!def*_
```

3) Проверила размещение прерывания в памяти. Для этого запустила программу лабораторной работы №3, которая отображает карту памяти в виде списка блоков MCB.

```

F:\>lab3_2
Available memory: 643696 b
Extended memory: 15360 Kb
List of MCB:
MCB type: 4Dh   PSP adress: 0008h   Size:      16 b
MCB type: 4Dh   PSP adress: 0000h   Size:       64 b
MCB type: 4Dh   PSP adress: 0040h   Size:     256 b
MCB type: 4Dh   PSP adress: 0192h   Size:     144 b
MCB type: 4Dh   PSP adress: 0192h   Size:    5040 b   LAB5
MCB type: 4Dh   PSP adress: 02D8h   Size:    5144 b
MCB type: 4Dh   PSP adress: 02D8h   Size:    5800 b   LAB3_2
MCB type: 5Ah   PSP adress: 0000h   Size:   642880 b   U LqA→

```

Как видно, резидент находится в памяти и успешно используется.

4) Запустила отлаженную программу еще раз и убедилась, что программа определяет установленный обработчик прерываний.

```

F:\>lab5
Interruption has been already loaded

F:\>*od ?an't !ill me_

```

5) Запустила отлаженную программу с ключом выгрузки и убедилась, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также запустила программу лабораторной работы №3.

Выгрузка:

```

F:\>lab5 /un
Interruption was unloaded.

F:\>god can't kill me

```

Карта памяти:

```

F:\>lab3_2
Available memory: 648912 b
Extended memory: 15360 Kb
List of MCB:
MCB type: 4Dh PSP adress: 0008h Size: 16 b
MCB type: 4Dh PSP adress: 0000h Size: 64 b
MCB type: 4Dh PSP adress: 0040h Size: 256 b
MCB type: 4Dh PSP adress: 0192h Size: 144 b
MCB type: 4Dh PSP adress: 0192h Size: 800 b LAB3_2
MCB type: 5Ah PSP adress: 0000h Size: 648096 b ❖ 0iαPSQ

```

Выводы.

В процессе выполнения данной работы производилось исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Было написано пользовательское прерывание от клавиатуры, которое анализирует скан-коды, выполняет вывод сообщения результата нажатия, а при несовпадении скан-кода передает управление стандартному обработчику.

Ответы на контрольные вопросы:

1. Какого типа прерывания использовались в работе?

Прерывания функции DOS (int 21h) и прерывания функций BIOS (16h, 09h) .

2. Чем отличается скан код от кода ASCII?

С помощью скан-кода драйвер клавиатуры распознает, какая именно клавиша была нажата

ASCII код – это код из таблицы всех имеющихся символов, которые могут быть выведены в консоль.

ПРИЛОЖЕНИЕ А

КОД ИСХОДНОЙ ПРОГРАММЫ

```
LB5 segment
assume cs:LB5, ds:DATA, ss:MY_STACK
```

```
MY_STACK segment stack
dw 256 dup(0)
MY_STACK ends
```

```
MY_INTERRUPT PROC FAR
    jmp Start
```

```
intData:
    key_value db 0
    new_stack dw 256 dup(0)
    signature dw 6666h
    keep_ip dw 0
    keep_cs dw 0
    keep_psp dw 0
    keep_ax dw 0
    keep_ss dw 0
    keep_sp dw 0
```

```
Start:
    mov keep_ax, ax
    mov keep_sp, sp
    mov keep_ss, ss
    mov ax, seg new_stack
    mov ss, ax
    mov ax, offset new_stack
    add ax, 256
    mov sp, ax

    push ax
    push bx
    push cx
    push dx
    push si
    push es
    push ds
    mov ax, seg key_value
    mov ds, ax
```

```

in al, 60h
cmp al, 22h ;g
je key_g
cmp al, 25h ;k
je key_k
cmp al, 2Eh ;c
je key_c

pushf
call dword ptr cs:keep_ip
jmp end_interruption

```

```

key_g:
    mov key_value, '*'
    jmp next_key

```

```

key_k:
    mov key_value, '!'
    jmp next_key

```

```

key_c:
    mov key_value, '?'

```

```

next_key:
    in al, 61h
    mov ah, al
    or     al, 80h
    out 61h, al
    xchg al, al
    out 61h, al
    mov al, 20h
    out 20h, al

```

```

print_key:
    mov ah, 05h
    mov cl, key_value
    mov ch, 00h
    int 16h
    or     al, al
    jz     end_interruption
    mov ax, 40h
    mov es, ax
    mov ax, es:[1ah]
    mov es:[1ch], ax
    jmp print_key

```

```

end_interruption:
    pop    ds
    pop    es
    pop     si
    pop    dx
    pop    cx
    pop    bx
    pop     ax

    mov sp, keep_sp
    mov ax, keep_ss
    mov ss, ax
    mov ax, keep_ax

    mov  al, 20h
    out 20h, al
    iret
MY_INTERRUPTION endp
_end:

```

```

is_int_loaded proc
    push ax
    push bx
    push si

    mov  ah, 35h
    mov  al, 09h
    int  21h
    mov  si, offset signature
    sub  si, offset MY_INTERRUPTION
    mov  ax, es:[bx + si]
    cmp  ax, signature
    jne  end_proc
    mov  is_load, 1

```

```

end_proc:
    pop  si
    pop  bx
    pop  ax
    ret
is_int_loaded endp

```



```

int_load proc
    push ax
    push bx
    push cx
    push dx
    push es
    push ds

    mov ah, 35h
    mov al, 09h
    int 21h
    mov keep_cs, es
    mov keep_ip, bx
    mov ax, seg MY_INTERRUPT
    mov dx, offset MY_INTERRUPT
    mov ds, ax
    mov ah, 25h
    mov al, 09h
    int 21h
    pop ds

    mov dx, offset _end
    mov cl, 4h
    shr dx, cl
    add dx, 10fh
    inc dx
    xor ax, ax
    mov ah, 31h
    int 21h

    pop es
    pop dx
    pop cx
    pop bx
    pop ax
ret
int_load endp

```

```

unload_interrupt proc
    cli
    push ax
    push bx
    push dx

```

```

push ds
push es
push si

mov ah, 35h
mov al, 09h
int 21h
mov si, offset keep_ip
sub si, offset MY_INTERRUPT
mov dx, es:[bx + si]
mov ax, es:[bx + si + 2]

push ds
mov ds, ax
mov ah, 25h
mov al, 09h
int 21h
pop ds

mov ax, es:[bx + si + 4]
mov es, ax
push es
mov ax, es:[2ch]
mov es, ax
mov ah, 49h
int 21h
pop es
mov ah, 49h
int 21h

sti

pop si
pop es
pop ds
pop dx
pop bx
pop ax

ret
unload_interrupt endp

is_unload_ proc

```

```

    push ax
    push es

    mov ax, keep_psp
    mov es, ax
    cmp byte ptr es:[82h], '/'
    jne end_unload
    cmp byte ptr es:[83h], 'u'
    jne end_unload
    cmp byte ptr es:[84h], 'n'
    jne end_unload
    mov is_un, 1

end_unload:
    pop es
    pop ax
    ret
is_unload_ endp

```

```

PRINT proc near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT endp

```

```

begin proc
    push ds
    xor ax, ax
    push ax
    mov ax, DATA
    mov ds, ax
    mov keep_psp, es

    call is_int_loaded
    call is_unload_
    cmp is_un, 1
    je unload
    mov al, is_load
    cmp al, 1
    jne load

```

```

        mov dx, offset str_loaded
        call PRINT
        jmp end_begin

load:
        mov dx, offset str_load
        call PRINT
        call int_load
        jmp end_begin

unload:
        cmp is_load, 1
        jne not_loaded
        mov dx, offset str_unload
        call PRINT
        call unload_interrupt
        jmp end_begin

not_loaded:
        mov dx, offset str_not_loaded
        call PRINT

end_begin:
        xor al, al
        mov ah, 4ch
        int 21h
begin endp
LB5 ends

DATA segment
        IS_LOAD          db                                0
        IS_UN            db                                0
        STR_LOAD         db    "Interruption was loaded.", 0dh,
0ah, "$"
        STR_LOADED       db    "Interruption has been already loaded",
0dh, 0ah, "$"
        STR_UNLOAD       db    "Interruption was unloaded.", 0dh,
0ah, "$"
        STR_NOT_LOADED   db    "Interruption is not loaded.", 0dh,
0ah, "$"
DATA ends
end begin

```