

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. 8383

Бессуднов Г. И.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2020

Цель работы.

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

Ход работы.

Был написан программный модуль LR5.EXE, который выполняет следующие функции:

- 1) Проверяет, установлено ли пользовательское прерывание с вектором 09h.
- 2) Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляет выход по функции 4Ch прерывания int 21h.
- 3) Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.
- 4) 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.

Обработчик прерываний обрабатывает клавиши гласных. Если была введена гласная, то она заменится на знак “*”. Ввиду этой особенности выгрузка прерывания происходит не по параметру /un, а по параметру /vn. Если же была нажата другая клавиша, то управление передается стандартному обработчику прерывания. Демонстрация работы программы представлена на рис. 1.



```
C:\>LR5.EXE  
C:\>*bcd*fgh*ijklm*pnrst*vwxyz
```

Рисунок 1 – Попытка ввода букв английского алфавита

С помощью программы LR3_1.COM было выведено состояние памяти. На рис. 2 показано состояние памяти после загрузки прерывания. На рис. 3 – после выгрузки прерывания.

```
Empty area
Size: 64 bytes

MCB 3
0004
Size: 256 bytes

MCB 4
1029
Size: 144 bytes

MCB 5
1029
Size: 704 bytes
LR5
MCB 6
109C
Size: 144 bytes

MCB 7
109C
Size: 648032 bytes
LR3_1
C:\>_
```

Рисунок 2 – После загрузки прерывания

```
C:\>LR3_1.COM
Aviabile memory: 648912 bytes
Extended memory: 15360 kbytes
MCB 1
MS DOS
Size: 16 bytes

MCB 2
Empty area
Size: 64 bytes

MCB 3
0004
Size: 256 bytes

MCB 4
1029
Size: 144 bytes

MCB 5
1029
Size: 648912 bytes
LR3_1
```

Рисунок 3 – После выгрузки прерывания

Как видно, после выгрузки прерывания, блоки, соответствующие программе LR5.EXE, удаляются.

Контрольные вопросы.

1. Какого типа прерывания использовались в работе?

Программные - 21h.

Аппаратные - 16h.

2. Чем отличается скан код от кода ASCII?

Скан-код - код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознает, какая клавиша была нажата. ASCII - код - это код символа в таблице ASCII - символов.

Выводы.

В ходе выполнения лабораторной работы была реализована программа, загружающая и выгружающая пользовательское прерывание от клавиатуры.

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ

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

ASTACK SEGMENT STACK
    DW 128 dup(0)
ASTACK ENDS

DATA SEGMENT
    IS_LOADED DB 0
    IS_VN DB 0
    MES_INT_LOADED DB 'Int loaded$'
    MES_INT_NOT_LOADED DB 'Int not loaded$'
DATA ENDS

MY_INT PROC FAR
    jmp MY_INT_START
    SYMB DB 0
    ID DW 6506h
    KEEP_AX DW 0
    KEEP_SS DW 0
    KEEP_SP DW 0
    KEEP_IP DW 0
    KEEP_CS DW 0
    KEEP_PSP DW 0
    INT_STACK DW 128 dup(0)

MY_INT_START:
    mov     KEEP_AX, AX
    mov     KEEP_SP, SP
    mov     KEEP_SS, SS
    mov     AX, SEG INT_STACK
    mov     SS, AX
    mov     AX, offset INT_STACK
    add     AX, 256
    mov     SP, AX
    push    BX
    push    CX
    push    DX
    push    SI
    push    DS
    push    BP
    push    ES

    mov     AX, SEG SYMB
    mov     DS, AX

    in      AL, 60h
    cmp     AL, 1Eh
    je      VOWEL
    cmp     AL, 12h
    je      VOWEL
    cmp     AL, 17h
    je      VOWEL
    cmp     AL, 18h
    je      VOWEL
```

```

        cmp     AL, 10h
        je      VOWEL
        cmp     AL, 16h
        je      VOWEL
        cmp     AL, 15h
        je      VOWEL
        pushf
        call    DWORD PTR CS:KEEP_IP
        jmp     MY_INT_END

VOWEL:
        mov     SYMB, '*'

        in      AL, 61h
        mov     AH, AL
        or      AL, 80h
        out     61h, AL
        xchg    AL, AL
        out     61h, AL
        mov     AL, 20h
        out     20h, AL

WRITE_BUF:
        mov     AH, 05h
        mov     CL, SYMB
        mov     CH, 00h
        int     16h
        or      AL, AL
        jz      MY_INT_END

        mov     AX, 0040h
        mov     ES, AX
        mov     AX, ES:[1Ah]
        mov     ES:[1Ch], AX
        jmp     WRITE_BUF

MY_INT_END:
        pop     ES
        pop     BP
        pop     DS
        pop     SI
        pop     DX
        pop     CX
        pop     BX
        mov     SP, KEEP_SP
        mov     AX, KEEP_SS
        mov     SS, AX
        mov     AX, KEEP_AX
        mov     AL, 20h
        out     20h, AL
        IRET
        ret

MY_INT ENDP
END_OF_MY_INT:

MY_INT_CHECK PROC
        push    AX
        push    BX
        push    ES
        push    SI

```

```

        mov     AH, 35h
        mov     AL, 09h
        int     21h
        mov     SI, offset ID
        sub     SI, offset MY_INT
        mov     AX, ES:[BX + SI]
        cmp     AX, 6506h
        jne     CHECK_MY_INT_END
        mov     IS_LOADED, 1

CHECK_MY_INT_END:
        pop     SI
        pop     ES
        pop     BX
        pop     AX
        ret

MY_INT_CHECK ENDP

LOAD_MY_INT PROC
        push    AX
        push    BX
        push    CX
        push    DX
        push    DS
        push    ES

        mov     AH, 35h
        mov     AL, 09h
        int     21h
        mov     KEEP_CS, ES
        mov     KEEP_IP, BX
        push    DS
        mov     DX, offset MY_INT
        mov     AX, SEG MY_INT
        mov     DS, AX
        mov     AH, 25h
        mov     AL, 09h
        int     21h
        pop     DS
        mov     DX, offset END_OF_MY_INT
        add     DX, 10Fh
        mov     CL, 4h
        shr     DX, CL
        inc     DX
        xor     AX, AX
        mov     AH, 31h
        int     21h

        pop     ES
        pop     DS
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        ret

LOAD_MY_INT ENDP

VN_CHECK PROC
        push    AX
        push    ES

```

```

        mov     AX, KEEP_PSP
        mov     ES, AX
        cmp     byte ptr ES:[82h], '/'
        jne     CHECK_VN_END
        cmp     byte ptr ES:[83h], 'v'
        jne     CHECK_VN_END
        cmp     byte ptr ES:[84h], 'n'
        jne     CHECK_VN_END
        mov     IS_VN, 1

CHECK_VN_END:
        pop     ES
        pop     AX
        ret

VN_CHECK ENDP

VNLOAD_MY_INT 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_INT
    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

    pop     SI
    pop     ES
    pop     DS
    pop     DX
    pop     BX
    pop     AX
    STI
    ret
VNLOAD_MY_INT ENDP

WRITE_STR     PROC     NEAR
    push     AX

```



```

        mov     AH, 09h
        int     21h
        pop     AX
    ret
WRITE_STR    ENDP

MAIN PROC
    push    DS
    xor     AX, AX
    push    AX
    mov     AX, DATA
    mov     DS, AX
    mov     KEEP_PSP, ES

    call    MY_INT_CHECK
    call    VN_CHECK
    cmp     IS_VN, 1
    je      UNLOAD
    mov     AL, IS_LOADED
    cmp     AL, 1
    jne     LOAD
    mov     DX, offset MES_INT_LOADED
    call    WRITE_STR
    jmp     MAIN_END

LOAD:
    call    LOAD_MY_INT
    jmp     MAIN_END

UNLOAD:
    cmp     IS_LOADED, 1
    jne     NOT_EXIST
    call    VNLOAD_MY_INT
    jmp     MAIN_END

NOT_EXIST:
    mov     DX, offset MES_INT_NOT_LOADED
    call    WRITE_STR

MAIN_END:
    xor     AL, AL
    mov     AH, 4Ch
    int     21h
MAIN ENDP

CODE ENDS

END MAIN

```