

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. 7383

Корякин М.П.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

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.

Описание функций.

- ROUT - пользовательский обработчик прерываний, печатающий цифру «9» при нажатии на кнопку «w».
- PRINT - вызывает функцию печати строки.
- PROV_ROUT - проверяет, установлен ли пользовательский обработчик прерывания, и если нет – устанавливает его. В ином случае, если хвост равен '/un', восстанавливает стандартное
- SET_ROUT - устанавливает пользовательское прерывание.
- DEL_ROUT - удаляет пользовательское прерывание.
- SAVE_STAND - сохраняет адрес стандартного прерывания в KEEP_IP, KEEP_CS.

Результат работ написанной программы.

```
C:\>13-1
available memory: 648912 byte
extended memory: 15360 Kb
Address  Owner      Size Name
016F     0008         16
0171     0000         64
0176     0040        256
0187     0192        144
0191     0192     648912  L3-1
C:\>
```

Рисунок 1 – Состояние памяти до запуска программы.

```

C:\>I5
Res is load

C:\>I3-1
available memory: 647360 byte
extended memory: 15360 Kb
Address      Owner      Size Name
016F        0008         16
0171        0000         64
0176        0040        256
0187        0192        144
0191        0192       1376 L5
01E8        01F3        144
01F2        01F3     647360 L3-1

C:\>_

```

Рисунок 2 – Состояние памяти после запуска программы.

```

C:\>I5
Res is load

C:\>q9erty_

```

Рисунок 3 – Проверка работы прерывания, при нажатии на «w»
выводится «9».

```

C:\>I5
Res is load

C:\>I5
Res already load

```

Рисунок 4 – Тестирование повторного запуска программы.

```

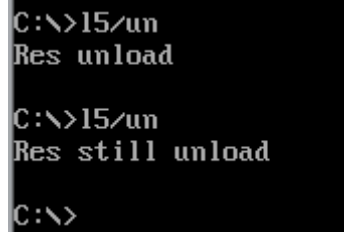
C:\>I5/un
Res unload

C:\>I3-1
available memory: 648912 byte
extended memory: 15360 Kb
Address      Owner      Size Name
016F        0008         16
0171        0000         64
0176        0040        256
0187        0192        144
0191        0192     648912 L3-1

C:\>

```

Рисунок 5 – Состояние памяти после выгрузки резидента.



```
C:\>15/un  
Res unload  
  
C:\>15/un  
Res still unload  
  
C:\>
```

Рисунок 6 – Тестирование повторной выгрузки резидента.

Выводы.

В данной лабораторной работе была исследована возможность встраивания пользовательского прерывания в стандартный обработчик от клавиатуры.

Ответы на контрольные вопросы.

1. Какого типа прерывания использовались в работе?

В работе использовались программные и аппаратные прерывания.

2. Чем отличается скан код от кода ASCII?

Скан код – это код, который присвоен каждой клавише на клавиатуре, он используется драйвером клавиатуры для распознавания нажатой клавиши. А код ASCII – это такая таблица, в которой каждому символу соответствует какое-либо число.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД.

```
ASTACK SEGMENT STACK
    dw 100h dup (?)
ASTACK ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK
ROUT PROC FAR
    jmp go_
    SIGNATURA dw 0ABCDh
    KEEP_PSP dw 0
    KEEP_IP dw 0
    KEEP_CS dw 0
    INT_STACK DW 100 dup (?)
    KEEP_SS DW 0
    KEEP_AX DW ?
    KEEP_SP DW 0
go_:
    mov KEEP_SS, SS
    mov KEEP_SP, SP
    mov KEEP_AX, AX
    mov AX, seg INT_STACK
    mov SS, AX
    mov SP, 0
    mov AX, KEEP_AX
    push ax
    push es
    push ds
    push dx
    push di
    push cx
    mov al, 0
    in al, 60h
    cmp al, 11h
    je do_req
    pushf
    call dword ptr cs:KEEP_IP
    jmp skip
do_req:
    in al, 61h
    mov ah, al
    or al, 80h
    out 61h, al
    xchg ah, al
    out 61h, al
    mov al, 20h
    out 20h, al
buf_push:
    mov al, 0
```

```

        mov ah,05h
        mov cl,'9' ;inst symbol
        mov ch,00h
        int 16h
        or al,al
        jz skip
        mov ax,0040h
        mov es,ax
        mov ax,es:[1Ah]
        mov es:[09h],ax
        jmp buf_push
skip:
        pop cx
        pop di
        pop dx
        pop ds
        pop es
        mov al,20h
        out 20h,al
        pop ax
        mov AX,KEEP_SS
        mov SS,AX
        mov AX,KEEP_AX
        mov SP,KEEP_SP

        iret
ROUT ENDP
LAST_BYTE:
;-----
PRINT PROC
        push ax
        mov ah,09h
        int 21h
        pop ax
        ret
PRINT ENDP
;-----
PROV_ROUT PROC
        mov ah,35h
        mov al,09h
        int 21h
        mov si,offset SIGNATURA
        sub si,offset ROUT
        mov ax,0ABCDh
        cmp ax,ES:[BX+SI]
        je ROUT_EST
            call SET_ROUT
            jmp PROV_KONEC
ROUT_EST:
        call DEL_ROUT
PROV_KONEC:

```

```

        ret
PROV_ROUT ENDP
;-----

SET_ROUT PROC
    mov ax,KEEP_PSP
    mov es,ax
    cmp byte ptr es:[80h],0
        je UST
    cmp byte ptr es:[82h], '/'
        jne UST
    cmp byte ptr es:[83h], 'u'
        jne UST
    cmp byte ptr es:[84h], 'n'
        jne UST

    mov dx,offset PRER_NE_SET_VIVOD
    call PRINT
    ret
UST:
    call SAVE_STAND
    mov dx,offset PRER_SET_VIVOD
    call PRINT
    push ds
    mov dx,offset ROUT
    mov ax,seg ROUT
    mov ds,ax
    mov ah,25h
    mov al,09h
    int 21h
    pop ds
    mov dx,offset LAST_BYTE
    mov cl,4
    shr dx,cl
    add dx,1
    add dx,40h
    mov al,0
    mov ah,31h
    int 21h
    ret
SET_ROUT ENDP
;-----

DEL_ROUT PROC
    push dx
    push ax
    push ds
    push es
    mov ax,KEEP_PSP
    mov es,ax
    cmp byte ptr es:[82h], '/'
        jne UDAL_KONEC

```

```

    cmp byte ptr es:[83h], 'u'
        jne UDAL_KONEC
    cmp byte ptr es:[84h], 'n'
        jne UDAL_KONEC
    mov dx, offset PRER_DEL_VIVOD
    call PRINT
    CLI
    mov ah, 35h
    mov al, 09h
    int 21h
    mov si, offset KEEP_IP
    sub si, offset ROUT
    mov dx, es:[bx+si]
    mov ax, es:[bx+si+2]
    mov ds, ax
    mov ah, 25h
    mov al, 09h
    int 21h
    mov ax, es:[bx+si-2]
    mov es, ax
    mov ax, es:[2ch]
    push es
    mov es, ax
    mov ah, 49h
    int 21h
    pop es
    mov ah, 49h
    int 21h
    STI
    jmp UDAL_KONEC2
UDAL_KONEC:
    mov dx, offset PRER_UZHE_SET_VIVOD
    call PRINT
UDAL_KONEC2:
    pop es
    pop ds
    pop ax
    pop dx
    ret
DEL_ROUT ENDP
;-----
SAVE_STAND PROC
    push ax
    push bx
    push es
    mov ah, 35h
    mov al, 09h
    int 21h
    mov KEEP_CS, ES
    mov KEEP_IP, BX
    pop es

```



```

        pop bx
        pop ax
        ret
SAVE_STAND ENDP
;-----
BEGIN:
        mov ax,DATA
        mov ds,ax
        mov KEEP_PSP, es
        call PROV_ROUT
        xor AL,AL
        mov AH,4Ch
        int 21H
CODE ENDS
DATA SEGMENT
        PRER_SET_VIVOD db 'Res is load',0DH,0AH,'$'
        PRER_DEL_VIVOD db 'Res unload',0DH,0AH,'$'
        PRER_UZHE_SET_VIVOD db 'Res already load',0DH,0AH,'$'
        PRER_NE_SET_VIVOD db 'Res still unload',0DH,0AH,'$'
        STRENDL db 0DH,0AH,'$'
DATA ENDS
END BEGIN

```