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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Операционные системы»

Тема: Построение модуля динамической структуры

Студент гр. 8381 _____ Почаев Н.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

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

Основные теоретические положения.

В отличие 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.

Выполнение работы производилось на базе операционной системы Windows XP (32 bit), запускаемой в системе виртуализации VMware Workstation, в редакторе Notepad++. Сборка и отладка модулей производились с помощью компилятора MASM и отладчика AFD. Также в работе был использован консольный файловый менеджер Far Manager и HEX-редактор HxD. Для дополнительного тестирования и проверки функциональности программы использовался DOSBox.

Ход работы.

Созданные процедуры приведены в табл. 1.

Таблица 1 – Реализованный процедуры

Название процедуры	**Описание процедуры**
WRITE	Вывод строки на экран
Main	Основная процедура, выполняющая указанные действия
exitProg	Проверяет корректность выхода из дочернего модуля
freeMem	Освобождение места в памяти

Последовательность работы программы:

1. При запуске программы освобождается место в памяти.
2. Проверяется флаг переполнения CF.
3. Загружается дочерний модуль.
4. В конце дочернего модуля вызывается функция 01h прерывания 21h. Код символа, который нажал пользователь, сохраняется в AL и служит аргументов для функции 4Ch.
5. Управление переходит вызывающей программе.
6. Вызывающая программа проверяет корректность выхода из дочернего модуля и выводит соответствующее сообщение.

Результат запуска программы в рабочем каталоге с модулем из 2-ой ЛР представлен на рис. 1.

```
E:\Labs\lr6>LR6.EXE

Child process started
Locked memory address: 9FFF
Environment address: 1510
Command line tail: ▶з Л☺р ▬ Ц♥Л☺р Л☺р ╎╎♠♀◙◙♀◙ ♥♀◙F♀◙
                                                    ♦♀◙: Ц♥T Ц♥n Ц♥И Ц♥в Ц♥ ♥♀◙и
◙■◙-      ♀◙Л♠♀◙Л♀♀◙¶♀♀◙▼      ♀◙
Content: COMSPEC=C:\WINDOWS\SYSTEM32\COMMAND.COM
ALLUSERSPROFILE=C:\DOCUME~1\ALLUSE~1
APPDATA=C:\Documents and Settings\Администратор\Application Data
CLIENTNAME=Console
COMMONPROGRAMFILES=C:\PROGRA~1\COMMON~1
COMPUTERNAME=NIKITA-68071BC4
FP_NO_HOST_CHECK=NO
HOMEDRIVE=C:
HOMEPATH=\Documents and Settings\Администратор
LOGONSERVER=\\NIKITA-68071BC4
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
PATH=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 23 Model 1 Stepping 1, AuthenticAMD
PROCESSOR_LEVEL=23
PROCESSOR_REVISION=0101
PROGRAMFILES=C:\PROGRA~1
PROMPT=$P$G
SESSIONNAME=Console
SYSTEMDRIVE=C:
SYSTEMROOT=C:\WINDOWS
TEMP=C:\WINDOWS\TEMP
TMP=C:\WINDOWS\TEMP
USERDOMAIN=NIKITA-68071BC4
USERNAME=Администратор
USERPROFILE=C:\Documents and Settings\Администратор
BLASTER=A220 I5 D1 P330 T3

Path: E:\LABS\LR6\LR2.com
Press any key...
```

Рисунок 1 – Результат запуска программы в одном каталоге с модулем

Сразу после запуска программы, она загружает дочерний модуль, который ждёт нажатия клавиши. Если нажать любую клавишу, то код клавиши запишется в регистр AL, а дочерний модуль завершится и управление перейдёт вызывающей программе. Вызывающая программа выведет символ, который был нажат для завершения работы дочернего модуля (допустим была нажата клавиша k). Результирующий вывод программы представлен на рис. 2.

```
Path: E:\LABS\LR6\LR2.com
Press any key...
kk Process was end successfully, code:  6B
E:\Labs\lr6>
```

Рисунок 2 – Результат перехвата нажатия клавиши модулем

Если программа была завершена с помощью комбинации клавиш ctrl+c, то выведется соответствующее сообщение, данный факт представлен на рис. 3.

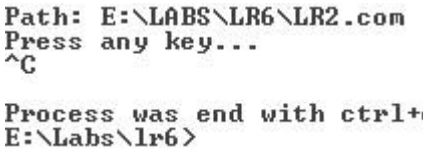

Рисунок 3 – завершение работы программы по нажатию Ctrl+C

Если дочернего модуля в каталоге с программой нет, то выведется соответствующее сообщение и программа завершится. Данный факт проиллюстрирован на рис. 4.

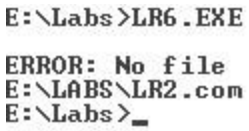

Рисунок 4 – Запуск программы в каталоге без модуля

Выводы.

В ходе выполнения данной лабораторной работы была исследована работа и организация загрузочных модулей динамической структуры, также были приобретены навыки по загрузке и завершению дочерних модулей, аналогично была освоена обработка результатов завершения дочерних модулей.

Ответы на контрольные вопросы.

1. **Как реализовано прерывание Ctrl-C?**

Прерывание 23h вызывается, если была нажаты комбинации клавиш Ctrl-C или Ctrl-Break. Адрес, по которому передается управление (0000:008c). Управление передаётся тогда, когда DOS распознает, что пользователь нажал Ctrl-Break или Ctrl-C. Адрес по вектору INT 23h копируется в поле PSP Ctrl-Break Address функциями DOS 26h (создать PSP) и 4Ch (EXEC).

Исходное значение адреса обработчика Ctrl-Break восстанавливается из PSP при завершении программы. Таким образом, по завершении порожденного процесса будет восстановлен адрес обработчика Ctrl-Break из родительского процесса.

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

Если код причины завершения 0, то вызываемая программа заканчивается в месте вызова функции 4Ch прерываний int 21h.

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

В месте, где программа ожидала ввода символа: в точке вызова функции 01h прерывания int 21h.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
.model small
.data
st_docher_uspeh db 13, 10,    "Process was end successfully, code: $"
st_file_otst db 13, 10,       "ERROR: No file", 13, 10, "$"
st_docher_ctrlc db 13, 10,    "Process was end with ctrl+c$"
psp dw ?
filename db 50 dup(0)
eol db "$"
param dw 7 dup(?)
vrem_ss dw ?
vrem_sp dw ?
fmemerr db 0
.stack 100h
.code

TETR_TO_HEX   PROC  near
      and      AL,0Fh
      cmp      AL,09
      jbe      NEXT
      add      AL,07
      NEXT:      add      AL,30h
      ret
TETR_TO_HEX   ENDP

BYTE_TO_HEX   PROC  near
      push     CX
      mov      AH,AL
      call     TETR_TO_HEX
      xchg     AL,AH
      mov      CL,4
      shr      AL,CL
      call     TETR_TO_HEX
      pop      CX
      ret
BYTE_TO_HEX  ENDP

WRITE PROC
      push ax
      mov ah, 09h
      int 21h
      pop ax
      ret
WRITE ENDP
```

```
freeMem PROC
	lea bx, PROGEND
	mov ax, es
	sub bx, ax
	mov cl, 4
	shr bx, cl
	mov ah, 4Ah
	int 21h
	jc err
	jmp noterr
	err:
		mov fmemerr, 1
	noterr:
		ret
freeMem ENDP

exitProg PROC
	mov ah, 4Dh
	int 21h
	cmp ah, 1
	je errchild
	lea bx, st_docher_uspeh
	mov [bx], ax
	lea dx, st_docher_uspeh
	push ax
	call WRITE
	pop ax
	call byte_to_Hex
	push ax
	mov dl, ' '
	mov ah, 2h
	int 21h
	pop ax
	push ax
	mov dl, al
	mov ah, 2h
	int 21h
	pop ax
	mov dl, ah
	mov ah, 2h
	int 21h
	jmp exget
	errchild:
		lea dx, st_docher_ctrlc
		call WRITE
	exget:
		ret
```

```
exitProg ENDP

Main proc
      mov ax, @data
      mov ds, ax
      push si
      push di
      push es
      push dx
      mov es, es:[2Ch]
      xor si, si
      lea di, filename
      env_char:
            cmp byte ptr es:[si], 00h
            je env_crlf
            inc SI
            jmp env_next
      env_crlf:
            inc si
      env_next:
            cmp word ptr es:[si], 0000h
            jne env_char
            add si, 4
      abs_char:
            cmp byte ptr es:[si], 00h
            je vot
            mov dl, es:[si]
            mov [di], dl
            inc si
            inc di
            jmp abs_char
      vot:
            sub di, 5
            mov dl, '2'
            mov [di], dl
            add di, 2
            mov dl, 'c'
            mov [di], dl
            inc di
            mov dl, 'o'
            mov [di], dl
            inc di
            mov dl, 'm'
            mov [di], dl
            inc di
            mov dl, 0h
            mov [di], dl
```

```
            inc di
            mov dl, eol
            mov [di], dl
            pop dx
            pop es
            pop di
            pop si
            call freeMem
            cmp fmemerr, 0
            jne ex
            push ds
            pop es
            lea dx, filename
            lea bx, param
            mov vrem_ss, ss
            mov vrem_sp, sp
            mov ax, 4b00h
            int 21h
            mov ss, vrem_ss
            mov sp, vrem_sp
            jc erld
            jmp noterld
      erld:
            lea dx, st_file_otst
            call WRITE
            lea dx, filename
            call WRITE
            jmp ex
      noterld:
            call exitProg
      ex:
            mov ah, 4Ch
            int 21h
main ENDP

PROGEND PROC
PROGEND ENDP

end main
```