

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

Кафедра МОЭВМ

ОТЧЕТ
лабораторная работа №2
по дисциплине «Операционные системы»
Тема: Исследование интерфейсов командных модулей

Студент гр. 9381

Николаев А.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Основные теоретические положения.

При начальной загрузке программы формируется PSP, который размещается в начале первого сегмента программы. PSP занимает 256 байт и располагается с адреса, кратного границе сегмента. При загрузке модулей типа COM все сегментные регистры указывают на адрес PSP. При загрузке модуля типа EXE сегментные регистры DS и ES указывают на PSP.

Формат PSP.

Смещение	Длина поля (байт)	Содержимое поля
0	2	int 20h
2	2	Сегментный адрес первого байта недоступной памяти. Программа не должна модифицировать содержимое памяти за этим адресом.
4	6	Зарезервировано
0Ah (10)	4	Вектор прерывания 22h (IP, CS)
0Eh (14)	4	Вектор прерывания 23h (IP, CS)
12h (18)	4	Вектор прерывания 24h (IP, CS)
2Ch (44)	2	Сегментный адрес среды, передаваемой программе.

5Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB)
6Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается, если FCB с адреса 5Ch открыт.
80h	1	Число символов в хвосте командной строки.
81h		Хвост командной строки - последовательность символов после имени вызываемого модуля.

Ход работы.

Был написан код .COM, который выводит следующую информацию в консоль.

- Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде
- Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде
- Хвост командной строки, в символьном виде
- Содержимое области среды в символьном виде
- Путь загружаемого модуля

```
C:\>lab2.COM
Memory Address: 9FFF
Env. Address: 0188
Tail:
Env. Data:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Path:
C:\LAB2.COM

C:\>_
```

Функции и структуры данных программы:

Имя	Описание
PRINT	Выводит строку в консоль.
TETR_TO_HEX	Переводит половину байта в символ.
BYTE_TO_HEX	Переводит байт в два символа в 16СС.
WREAD_TO_HEX	Переводит число в символьную строку в 16СС.

Имя	Содержание
STR_1	Memory Address:
STR_2	Env. Address:
STR_3	Tail:
STR_4	Env. Data:
STR_5	\$ (Переход на новую строку)
STR_6	Path:

Контрольные вопросы

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

Область недоступной памяти, начинается с адреса сегментной памяти, следующей за памятью, выделяемой программе.

2. Где расположен этот адрес по отношению области памяти, отведенной программе?

Адрес указывает на байт, следующий за последним байтом памяти, выделенной для программы.

3. Можно ли в эту область памяти писать

Можно, контроля доступа к памяти нет.

Среда, передаваемая программе

1. Что такое среда?

Среда – область памяти, в которой хранятся значения переменных в формате «переменная = значение», нулевой байт, обозначающий конец строки.

2. Когда создается среда? Перед запуском приложения или в другое время?

Когда одна программа запускает другую программу, то запущенная программа получает свой экземпляр блока среды, который является копией родительского. Запуск осуществляется интерпретатором COMMAND.COM, который имеет свою среду, которая называется корневой, запуск программного интерпретатора осуществляется при запуске ОС. Среда создается перед запуском программирования, копирование переменных производится для каждой программы.

3. Откуда берется информация, записываемая в среду?

AUTOEXEC.BAT - системный пакетный файл, который содержит информацию о ключевых переменных среды.

Выводы.

Был исследован интерфейс управляющей программы и загрузочных модулей. Был изучен PSP и среда, передаваемая программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR2.ASM

```
CODE        SEGMENT
ASSUME      CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
ORG         100H
START:      JMP     BEGIN
STR1        DB "Memory Address:      ", 13,10, "$"
STR2        DB "Env. Address:        ", 13,10, "$"
STR3        DB "Tail:                                ", 13,10, "$"
STR4        DB "Env. Data: ", 13, 10, "$"
STR5        DB 13, 10, "$"
STR6        DB "Path: ", 13, 10, "$"

PRINT       PROC    NEAR
            PUSH    AX
            MOV     AH, 09H
            INT     21H
            POP     AX
            RET
PRINT       ENDP

TETR_TO_HEX PROC    NEAR

            AND     AL, 0FH
            CMP     AL, 09H
            JBE     NEXT
            ADD     AL, 07H
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, 4H
            SHR     AL, CL
            CALL    TETR_TO_HEX
            POP     CX
            RET
BYTE_TO_HEX ENDP

WREAD_TO_HEX PROC    NEAR
```

```

        PUSH    BX
        MOV     BH, AH
        CALL    BYTE_TO_HEX
        MOV     [DI], AH
        DEC     DI
        MOV     [DI], AL
        DEC     DI
        MOV     AL, BH
        CALL    BYTE_TO_HEX
        MOV     [DI], AH
        DEC     DI
        MOV     [DI], AL
        POP     BX
        RET
WREAD_TO_HEX    ENDP

```

BEGIN:

```

        MOV     AX, DS:[02H]
        MOV     DI, OFFSET STR1
        ADD     DI, 19
        CALL    WREAD_TO_HEX
        MOV     DX, OFFSET STR1
        CALL    PRINT

```

```

        MOV     AX, DS:[2CH]
        MOV     DI, OFFSET STR2
        ADD     DI, 17
        CALL    WREAD_TO_HEX
        MOV     DX, OFFSET STR2
        CALL    PRINT

```

```

        XOR     CX, CX
        XOR     SI, SI
        MOV     CL, DS:[80H]
        CMP     CL, 0
        JE      NO_TAIL
        MOV     DI, OFFSET STR3
        ADD     DI, 6H

```

READ:

```

        MOV     AL, DS:[81H + SI]
        MOV     [DI], AL
        INC     DI
        INC     SI
        LOOP    READ

```

NO_TAIL:


```

        MOV     DX, OFFSET STR3
        CALL    PRINT
        MOV     DX, OFFSET STR4
        CALL    PRINT
        XOR     DI, DI
        MOV     BX, 2CH
        MOV     DS, [BX]
BEGIN_STRING:
        CMP     BYTE PTR [DI], 00H
        JE      ENTR
        MOV     DL, [DI]
        MOV     AH, 02H
        INT     21H
        JMP     END_DATA
ENTR:
        PUSH    DS
        MOV     CX, CS
        MOV     DS, CX
        MOV     DX, OFFSET STR5
        CALL    PRINT
        POP     DS
END_DATA:
        INC     DI
        CMP     WORD PTR [DI], 0001H
        JE      PATH
        JMP     BEGIN_STRING
PATH:
        PUSH    DS
        MOV     AX, CS
        MOV     DS, AX
        MOV     DX, OFFSET STR6
        CALL    PRINT
        POP     DS
        ADD     DI, 2
CIRCLE:
        CMP     BYTE PTR [DI], 00H
        JE      END_PATH
        MOV     DL, [DI]
        MOV     AH, 02H
        INT     21H
        INC     DI
        JMP     CIRCLE
END_PATH:
        PUSH    DS
        MOV     CX, CS
        MOV     DS, CX
        MOV     DX, OFFSET STR5

```

```
CALL PRINT
POP DS
```

```
XOR     AL, AL
MOV     AH, 4CH
INT     21H
```

```
CODE    ENDS
END      START
```