

ВСМИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Операционные системы»
Тема: Исследование интерфейсов программных модулей

Студентка гр. 9382

Круглова В.Д.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Необходимые сведения для составления программы.

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. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять.

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.

Рисунок 1 — Формат PSP

Область среды содержит последовательность символьных строк вида: имя=параметр. Каждая строка завершается байтом нулей. В первой строке указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM. Следующие строки содержат

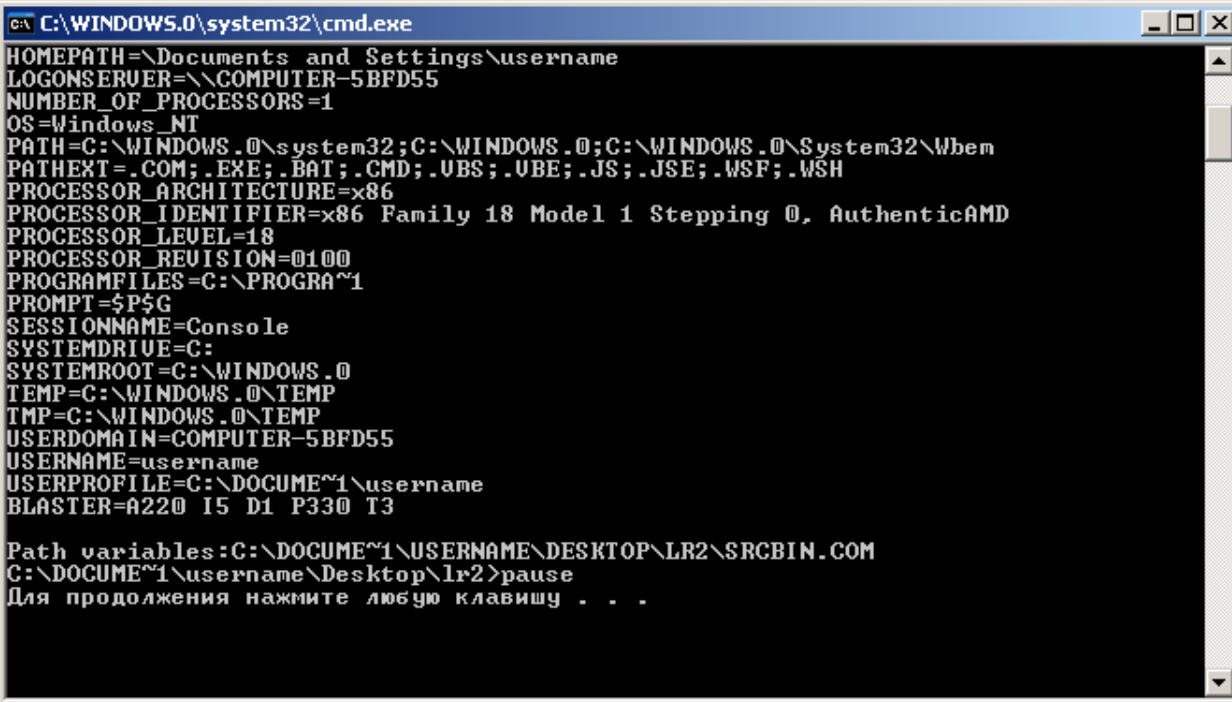
информацию, задаваемую командами PATH, PROMPT, SET. Среда заканчивается также байтом нулей. Таким образом, два нулевых байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

Ход выполнения работы

Была написана COM программа (см. Приложение А). Программа обращается к определенным участкам PSP и выводит:

1. Адрес начала недоступной памяти в шестнадцатеричном виде
2. Сегментный адрес среды программы в шестнадцатеричном виде
3. Хвост командной строки в символах
4. Содержимое переменных среды
5. Путь загружаемого модуля

Результаты работы программы



```
C:\WINDOWS.0\system32\cmd.exe
HOMEPATH=\Documents and Settings\username
LOGONSERVER=\\COMPUTER-5BFD55
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
PATH=C:\WINDOWS.0\system32;C:\WINDOWS.0;C:\WINDOWS.0\System32\Wbem
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 18 Model 1 Stepping 0, AuthenticAMD
PROCESSOR_LEVEL=18
PROCESSOR_REVISION=0100
PROGRAMFILES=C:\PROGRA~1
PROMPT=$P$G
SESSIONNAME=Console
SYSTEMDRIVE=C:
SYSTEMROOT=C:\WINDOWS.0
TEMP=C:\WINDOWS.0\TEMP
TMP=C:\WINDOWS.0\TEMP
USERDOMAIN=COMPUTER-5BFD55
USERNAME=username
USERPROFILE=C:\DOCUME~1\username
BLASTER=A220 I5 D1 P330 T3

Path variables:C:\DOCUME~1\USERNAME\Desktop\LR2\SRCBIN.COM
C:\DOCUME~1\username\Desktop\lr2>pause
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2 – Результат работы программы.

Ответы на контрольные вопросы:

Сегментный адрес недоступной памяти.

1. На какую область памяти указывает адрес недоступной памяти?

На сегментный адрес последнего параграфа памяти, используемого DOS для запуска программ.

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

За областью памяти, которую DOS отводит пользовательским программам.

3. Можно ли в эту область памяти писать?

Можно, т. к. DOS это не контролирует.

Среда, передаваемая программе.

1. Что такое среда?

Область памяти, содержащая переменные среды (символьные строки вида «ИМЯ = ПАРАМЕТР»).

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

Среда создается после старта командного интерпретатора при запуске DOS. При запуске программы информация из переменных среды передается в среду программы.

3. Откуда берется информация, записываемая в среду?

Из системного файла autoexec.bat.

Вывод

В ходе данной лабораторной работы был исследован интерфейс загрузочного модуля , PSP и среда, передаваемая программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

LR2 SEGMENT

ASSUME CS:LR2, DS:LR2, ES:NOTHING, SS:NOTHING
ORG 100H ; PSP

START: JMP BEGIN

;DATA SEGMENT

UN_MEM db 'Forbidden memory:', '\$'
ENV_ADDR db 'Environment address:', '\$'
COM_TAIL db 'Command tail:', '\$'
NO db 'no', '\$'
ENV_D db 'Environment data:' , '\$'
PATH db 'Path variables:' , '\$'
ENDL db 0dh, 0ah, '\$'

;DATA ENDS

;CODE SEGMENT

PRINT_STR PROC near

push AX
mov ah, 09h
int 21h
pop AX
ret

PRINT_STR ENDP

PRINT_LN PROC near

call PRINT_STR
mov DX, offset ENDL
call PRINT_STR
ret

PRINT_LN ENDP

PRINT_HEX_2B PROC

```

    push AX
    push BX
    mov BX, AX
    mov AL, AH
    call PRINT_HEX_1B
    mov AX, BX
    call PRINT_HEX_1B
    mov DX, offset ENDL
    call PRINT_STR
    pop BX
    pop AX
    ret
PRINT_HEX_2B ENDP

```

```

PRINT_HEX_1B PROC

```

```

    push AX
    push BX
    push DX
    mov AH, 0
    mov BL, 16
    div BL
    mov DX, AX
    mov AH, 02h
    cmp DL, 0Ah
        jl PRINT
    add DL, 7
PRINT:
    add DL, '0'
    int 21h;
    mov DL, DH
    cmp DL, 0Ah
        jl PRINT2
    add DL, 7
PRINT2:
    add DL, '0'

```

```

    int 21h;
    pop DX
    pop BX
    pop AX
    ret
PRINT_HEX_1B ENDP

```

```

PRINT_CHAR PROC
    push AX
    push DX
    xor DX, DX
    mov DL, AL
    mov AH, 02h
    int 21h
    pop DX
    pop AX
    ret
PRINT_CHAR ENDP

```

```

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 ; байт в AL переводится в два символа шестн. числа
    mov AH, AL
    call TETR_TO_HEX
    xchg AL, AH

```

В AX

```

mov CL,4
shr AL,CL
call TETR_TO_HEX ; в AL старшая цифра
pop CX ; в AH младшая
ret
BYTE_TO_HEX ENDP

```

WRD_TO_HEX PROC near ; 16 с/с 16 bit. В AX - число, DI - адрес последнего символа

```

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
WRD_TO_HEX ENDP

```

BYTE_TO_DEC PROC near ; 10 с/с, SI - адрес поля младшей цифры

```

push CX
push DX
xor AH,AH
xor DX,DX
mov CX,10
loop_bd:
div CX
or DL,30h
mov [SI],DL

```



```

    dec SI
    xor DX,DX
    cmp AX,10
    jae loop_bd
    cmp AL,00h
    je end_1
    or AL,30h
    mov [SI],AL
end_1:
    pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

```

BEGIN:

```

    push DS
    sub AX,AX
    push AX
    ; Начало
    ; Память
    mov DX, offset UN_MEM
    call PRINT_STR
    mov AX, DS:[2h]
    call PRINT_HEX_2B

    ; Адрес
    mov DX, offset ENV_ADDR
    call PRINT_STR
    mov AX, DS:[2Ch]
    call PRINT_HEX_2B

    ; Хвост
    mov dx, offset COM_TAIL
    call PRINT_STR

```

```

mov bx, 80h
mov ch, 0
mov cl, [bx]
cmp cx, 0
    je TAIL_END
mov bx, 81h
mov ah, 02h

```

TAIL_FOR:

```

mov dl, [bx]
int 21h
add bx, 1
loop TAIL_FOR

```

TAIL_END:

```

mov dx, offset ENDL
call PRINT_STR

```

; Окружение

```

mov DX, offset ENV_D
call PRINT_LN

```

```

mov SI, 0
mov BX, 2Ch
mov ES, [BX]

```

START_ENV:

```

cmp BYTE PTR ES:[SI], 0h
je NEW_LINE
mov AL, ES:[SI]
call PRINT_CHAR
    jmp CHECK_END

```

NEW_LINE:

```

mov DX, offset ENDL

```

```

        call PRINT_STR
CHECK_END:
        inc SI
        cmp WORD PTR ES:[SI], 0001h
        je WRITE_PATH
        jmp START_ENV

WRITE_PATH:
        mov DX, offset PATH
        call PRINT_STR
        add SI, 2
OUTPUT_PATH_FOR:
        cmp BYTE PTR ES:[SI], 00h
        je END_ENV_D
        mov AL, ES:[SI]
        call PRINT_CHAR
        inc SI
        jmp OUTPUT_PATH_FOR

END_ENV_D:

        ; Выход
        xor AL, AL
        mov AH, 4Ch
        int 21H
        ret

LR2 ENDS
        END START

```