МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання курсового проекту з дисципліни
"Операційні системи"
для студентів спеціальності 121
"Інженерія програмного забезпечення"

2022

Методичні рекомендації до виконання курсового проекту з дисципліни "Операційні системи" для студентів спеціальності 121 "Інженерія програмного забезпечення"/ Уклад.: Сердюк С.М., Степаненко О.О., Качан О.І. – НУ «Запорізька політехніка», 2022. – 35 с.

Укладачі: С.М. Сердюк, к.т.н., доцент кафедри ПЗ,
О.О. Степаненко, к.т.н., доцент кафедри ПЗ,
О.І. Качан, асистент кафедри ПЗ.

Рецензент: В.І. Дубровін, к.т.н., проф. кафедри ПЗ.

Відповідальний за випуск: С.О. Субботін, зав. каф. ПЗ, д.т.н., професор

Затверджено
на засіданні кафедри
"Програмні засоби"

Протокол № 9 від 15.05.2022 р.

ЗМІСТ

1 ВСТУП

Метою курсового проекту є поглиблення та закріплення знань з навчальної дисципліни "Операційні системи", здобутих в ході лекційних і лабораторних робіт, а також практичне застосування знань, що були засвоєнні при виконанні завдань на самопідготовку і при вивченні забезпечуючих та суміжних навчальних дисциплін.

Тематика курсових проектів представлена у додатку. Допускається ініціативна тематика за умовою її утвердження керівником.

Теми курсових проектів видаються студентам керівником проекту з відміткою про дату видання і дату захисту у спеціальному графіку, затвердженому деканатом. При цьому студент ставить підпис за вказаний термін роботи.

2 ТЕОРЕТИЧНІ РОЗДІЛИ ТЕМАТИКИ КУРСОВИХ ПРОЕКТІВ

2.1 Драйвери пристроїв

Драйвер пристрою – це спеціальна програма, яка керує обміном з периферійними пристроями (принтером, дисковим накопичувачем і т.п.). Драйвер пристрою створюється його розробником, тим самим особливості апаратури залишаються схованими для операційної системи і програміста. Також драйвер пристрою може створюватись для організації роботи з ним в необхідному для конкретного використання режимі. Користуючись стандартним для кожної операційної системи інтерфейсом, програма користувача чи сама операційна система звертається до драйверу для виконання операцій керування, вводу чи виводу. Усі драйвери можна поділити на дві групи:

а) драйвери символьних пристроїв – обслуговують пристрої символьного вводу/виводу, такі, як принтери, клавіатура, послідовні адаптери і т.д.;

б) драйвери блочних пристроїв – вони орієнтовані на обмін даних блоками – це драйвери дискових пристроїв .

Установлені драйвери пристроїв можуть бути реалізовані тільки на мові ассемблера у вигляді COM-програми, однак не є такими, тому що мають іншу структуру.

Структура драйверу

Програма драйвера пристрою поділяється на три частини:

а) заголовок драйвера, який озаголовлює пристрій і містить інформацію про останні частини драйверу;

б) стратегія драйверу, яка зберігає інформацію про області даних, які створює MS DOS та називається заголовком запиту;

в) обробник переривання пристрою, який і містить код, що керує пристроєм.

Заголовок драйверу

Драйвер пристрою повинен починатися з заголовку драйвера. Він має довжину 18 байт, поділених на 5 полів, зміст яких приведено у табл. 2.1.

Таблиця 2.1 – Заголовок драйверу

№ п/п	Зміщення поля (байт)	Розмір поля (байт)	Опис поля
1	0	4	Показчик на заголовок наступного драйверу. Якщо зміщення адресу наступного драйвера дорівнює FFFFFFFFH, то це останній драйвер в ланцюзі
2	4	2	Атрибути драйверу
3	6	2	Зміщення програми стратегій драйверу
4	8	2	Зміщення програми обробки переривань
5	10	8	Ім’я пристрою (для символьних пристроїв) чи кількість пристроїв (для блочних пристроїв)

Приклад 1. Заголовок драйверу.

Початковий код для драйверу пристрою з ім’ям DEVICE12. Він замінює стандартний пристрій AUX, якій використовується MS DOS як додатковий пристрій (частіше послідовний порт), перехоплюючи функцію 4 переривання 21H, яка посилає символ в стандартний комунікаційний канал. Увесь драйвер пристрою складається з коду цього прикладу разом з кодом, який є у прикладі 2 і прикладі 3. Щоб отримати повну програму необхідно розташувати їх підряд один за одним.

```
CSEG SEGMENT PUBLIC 'CODE'   ;установлюємо кодовий сегмент
  ORG 0                      ;цей рядок необов'язковий
  ASSUME CS:CSEG,DS:CSEG,ES:CSEG
```

```
DEVICE12 PROC FAR          ;драйвер - це віддалена процедура
DD 0FFFFFFFFH              ;адреса наступного драйвера
DW 8000H                   ;байт атрибутів
DW DEV_STRATEGY            ;адреса процедури стратегії
DW DEV_INTERRUPT           ;адреса процедури переривання
DB 'AUX'                   ; ім'я пристрою (доповнене пробілами)
```

Стратегія драйверу

Для звертання до драйверу MS DOS формує у своїй області даних запит який складається із заголовка стандартного формату розміром 13 байт (табл. 2.2) і додаткової структури даних (змінної частини запита), довжина і формат якої залежать від типу запиту.

Після цього MS DOS читає з заголовка драйверу зміщення програми стратегії і передає їй управління, записуючи у регістри ES:BX адресу заголовка запиту.

Задача програми стратегії – запам'ятати цю адресу усередині тіла драйверу для подальшого використання або організувати чергу запитів обслуговування.

Таблиця 2.2 – Запит до заголовку драйверу

№ п/п	Зміщення (байт)	Розмір (байт)	Опис поля
1	0	1	Загальний розмір блока запиту у байтах (довжина заголовку запиту плюс довжина змінної частини запиту)
2	1	1	Номер пристрою. Використовується для блочних пристроїв; вказуючи, з яким пристроєм (з числа, що були обслужені цим драйвером) буде працювати операційна система
3	2	1	Код команди, яку потрібно виконати. Може мати значення від 00 h до 18h
4	3	2	Слово стану пристрою. Заповнюється драйвером перед поверненням керування операційній системі
5	5	8	Зарезервовано

Приклад 2. Процедура стратегії пристрою.

```
DEV_STRATEGY: MOV CS:KEEP_ES,ES
      MOV CS:KEEP_BX,BX
      RET
  KEEP_CS DW ?
  KEEP_BX DW ?
```

Обробник переривань пристрою

Одразу після виклику програми стратегії MS DOS викликає програму обробки переривань, визначивши її адресу із заголовку драйвера.

Програма обробки переривань вибирає щойно записану програмою стратегії адресу заголовка запиту і виконує ту функцію, номер якої записаний в запиті. Номер функції знаходиться у заголовку запита. Результати виконання функції програма переривань записує в спеціально відведене поле заголовку запита, після чого процедура звернення MS DOS до драйверу завершується.

Існує 13 типів функцій, які може виконувати встановлений драйвер пристрою. Процедура яка виконує необхідну функцію, визначається за допомогою 13-словної таблиці, в якій знаходяться зміщення для 13 типів функцій.

Функції завжди перечислюються у такому порядку.

1. INITIALIZE (ініціалізація)
2. CHECK_MEDIA (перевірка носія)
3. MAKE _BPB
4. IOCTL_IN
5. IMPUT_DATA (ввід даних)
6. NONDESTRUCT_IN
7. IMPUT_STATUS (статус вводу)
8. CLEAR_INPUT (очищення вводу)
9. OUTPUT_DATA (вивід даних)
10. OUTPUT_VERIFY (перевірка виводу)
11. JUTHUT_STATUS (статус виводу)
12. CLEAR_OUTPUT (очищення виводу)
13. IOCTL_OUT

Приклад 3. Процедура обробки переривань.

В цьому прикладі приведена загальна форма процедури обробки переривання, за виключенням реального коду, що керує пристроєм. Тут використовується функція виводу. Процедура, що виконує вивід отримує з заголовка запит адреси буфера, в якому знаходяться дані, що виводяться (зміщення 14). Вона також зчитує число байтів, що потрібно вивести (зміщення 18). Коли процедура завершить вивід даних, то вона встановлює слово статусу в заголовку запита (зміщення 3) і поверне управління. Коли операція успішна, то потрібно встановити біт 8 слова статусу.

```
;---ініціалізація обробника переривань пристрою
 DEV_INTERRUPT: PUSH ES      ;зберігаємо регістри
      PUSH DS
      PUSH AX
      PUSH BX
      PUSH CX
      PUSH DX
      PUSH SI
      PUSH DI
      PUSH BP
  MOV AX,CS:KEEP_ES          ;ES:BX вказує на заголовок запиту
  MOV ES,AX
  MOV BX,CS:KEEP_BX
  MOV AL,ES:[BX+2]           ;отримуємо код команди із заголовку
  SHL AL,1                   ;помножуємо на 2)
  SUB AH,AH                  ;обнуляємо AH
  LEA DI,FUNCTIONS           ;DI вказує на зміщення до(перед)
                             таблиці(ею)
  ADD DI,AX                  ;додаємо зміщення в таблиці
  JMP WORD PTR [DI]          ;переходимо на адресу з таблиці
 FUNCTIONS LABEL WORD        ;це таблиця функцій
  DW INITIALIZE
  DW CHECK_MEDIA
  DW MAKE_BPB
  DW IOCTL_IN
  DW INPUT_DATA
  DW NONDESTRUCT_IN
  DW INPUT_STATUS
  DW CLEAR_INPUT
  DW OUTPUT_DATA
```

```
  DW OUTPUT_VERIFY
  DW OUTPUT_STATUS
  DW CLEAR_OUTPUT
  DW IOCTL_OUT
;-вихід із драйверу, якщо функція не підтримується
 CHECK_MEDIA:
 MAKE_BPB:
 IOCTL_IN:
 INPUT_DATA:
 NONDESTRUCT_IN:
 INPUT_STATUS:
 CLEAR_INPUT:
 OUTPUT_VERIFY:
 OUTPUT_STATUS:
 CLEAR_OUTPUT:
 IOCTL_OUT:
  OR ES:WORD PTR [BX]+3,8103H       ;модифікуємо статус
  JMP QUIT
;-процедури для двох кодів, що підтримуються
 INITIALIZE: LEA AX,E_O_P           ;зміщення кінця програми в AX
  MOV ES:WORD PTR [BX]+14,AX        ;помістимо його у заголовок
  MOV ES:WORD PTR [BX]+16,CS
  ...
  (тут іде ініціалізація пристрою)
  ...
  JMP QUIT
OUTPUT_DATA: MOV CL,ES:[BX]+18      ;отримуємо число, символ
  CBW CX                    ;CX використовуємо як лічильник
  MOV AX,ES:[BX]+16         ;отримуємо адресу буфера даних
  MOV DS,AX
  MOV DX,ES:[BX]+14
                                ...
                        (операції по виводу)
                                ...
  JMP QUIT
;-вихід з модифікацією байту статусу в заголовку запита
QUIT: OR ES:WORD PTR [BX]+3,100H    ;установлюємо біт 8
  POP BP                            ;повертаємо регістри
  POP DI
  POP SI
  POP DX
  POP CX
```

```
  POP BX
  POP AX
  POP DS
  POP ES
  RET
E_O_P:                              ;мітка кінця програми
DEVICE12 ENDP
CSEG ENDS
  END DEVICE12
```

Підключення драйверу

Для підключення драйверу до операційной системи файл CONFIG.SYS повинен містити рядок виду

DEVISE=<шлях_файлу_драйверу >_< параметри>

Завантажені драйвери знаходяться у списку драйверів перед резидентними драйверами

2.2 Оперативна пам'ять

Основна пам'ять

Основна пам'ять ПЕОМ знаходиться в межах 1 Мбайту адресного простору. Розподіл основної пам'яті зображено у табл. 2.3.

Таблиця 2.3 – Розподіл основної пам'яті

Діапазон адресів	Зміст
1	2
0000h-9FFFFh-Conventionel(Base) Memory,640 Кбайтів-стандартна базова пам′ять досяжна DOS і програм реального режиму	
0000:0000	Вектори переривань
0000:0400	Область даних BIOS
0000:0500	Область даних MS DOS
xxxx:0000	Область програм MS DOS. В ній знаходяться розширення BIOS, обробник переривань MS DOS, буфери, внутрішні структури даних, драйвери пристроїв, що завантажуються
xxxx:0000	Резидента порція командного процесору command.com

Продовження таблиці 2.3

1	2
xxxx:0000	Резиденті програми
xxxx:0000	Завантажені прикладні програми типу *.com чи *.exe
xxxx:0000	Транзитна порція command.com
A0000h-FFFFFh – Upper Memory Area (UMA), 384 Кбайтів – верхня пам′ять, зарезервована для системних потреб	
A000:0000	Пам'ять EGA, якою можна користуватися в деяких режимах
B000:0000	Пам'ять монохромного відео контролеру
B800:0000	Пам'ять відео контролеру CGA
C800:0000	Зовнішнє ПЗУ
F600:0000	ПЗУ інтерпретатору BASIC
FE00:0000	ПЗУ BIOS

***Векторна таблиця зв′язку* MS DOS**

В області даних MS DOS основні структури даних організовані у вигляді дерева. Коренем є векторна таблиця зв’язку (табл. 2.4), яка має дреси усіх останніх структур: список блоків керування пам′яттю MCB, список блоків керування пристроями MS DOS, таблицю файлів, дискові буфери і т.п.

У векторній таблиці зв’язку є і інша корисна інформація, що відкриває доступ до всіх внутрішніх структур даних операційної системи.

Таблиця 2.4 – Поля векторної таблиці зв’язку

№ п/п	Зміщення (байт)	Розмір (байт)	Опис поля
1	**2**	**3**	**4**
1	-2	2	Сегмент першого блоку пам′яті MCB
2	0	4	Указник на перший блок керування пристроями MS DOS(MS DOS Devise Control Block)
3	4	4	Указник на таблицю файлів MS DOS

Продовження таблиці 2.4

1	2	3	4
4	8	4	Указник на драйвер CLOCKS, що встановлений у файлі config.sys чи резидентний
5	2	4	Указник на драйвер CON, що встановлений у файлі config.sys чи резидентний
6	16	2	Максимальний розмір блоку (у байтах) для пристрою, що виконує передачу даних окремими блоками
7	18	4	Указник на структуру, яка описує дискові буфери
8	22	4	Указник на масиви інформації про пристрої
9	26	4	Указник на таблицю FCB
10	30	2	Розмір таблиці FCB
11	32	1	Число пристроїв, що виконують передачу даних окремими блоками
12	33	1	Значення LASTDRIVE у config.sys файлі (не вказуючи дорівнює п'яти)
13	34	–	Початок драйверу NULL.Цей драйвер завжди перший у списку драйверів MS DOS

Зміщення для кожного поля приведено відносно адреси, одержаної за допомогою не документованої функції 52h переривання INT 21h (отримана адреса знаходиться в ES:BX).

Приклад 4. Початковий текст функції, на мові C++, що повертає адресу другого поля векторної таблиці зв’язку.

```
void far *get_cvt(void)
{     union REGS inregs, outregs;
      struct SREGS segregs;
      inregs.h.ah = 0x52;
      intdosx(&inregs, &outregs, &segregs);
      return(MK_FP(segregs.es,outregs.x.bx));
}
```

Примітка

Інформація про векторну таблицю зв'язку приведена для версії MS DOS 6.22. Дана інформація відсутня у документації на операційну систему MS DOS, і відповідно може змінюватись від версії до версії.

Блоки керування пам'ятю MCB (Memory Control Block)

Зона оперативної пам′яті, починаючи з області програм MS DOS і до відеопам′яті, розбита на фрагменти. Перед кожним фрагментом знаходиться блок керування пам′ятю.

Всередині блоку MCB зберігається довжина, що описується даним MCB (табл. 2.5), фрагменту пам′яті. Наступний фрагмент пам′яті, починається одразу за попереднім. Усі блоки керування пам′ятю зв′язані у список.

Блоки MCB бувають двох типів – M і Z. M-блоки – це проміжні блоки. Блок типу Z є останнім і єдиним блоком у списку.

Таблиця 2.5 – Формат блоку MCB

Зміщення (байт)	**Розмір (байт)**	**Опис**
0	1	Тип блоку MCB (M чи Z)
1	2	Зміщення компоненти адреси власника блоку, цей блок завжди вирівняний на межі параграфу (коли 0, то блок описує сам себе)
3	2	Число параграфів у цьому блоці
5	11	Зарезервовано

Приклад 5. Програма знаходить у пам′яті програмні блоки MCB і виводить для кожного ім′я власника.

```
/*Компілятор Turbo C, Turbo C++ або Borland C++*/
#include <stdio.h>
#include <dos.h>
int main (void)
{
union REGS regs;
```

```
    struct SREGS segregs;
    typedef struct
    {
    unsigned char marker;
    unsigned int  owner;
    unsigned int  sizePara;
    unsigned char dummy [3];
    unsigned char name [8];
    }mcb;
    mcb far *ptr;
    unsigned int segm, i;
    regs.h.ah=0x52;
    intdosx(&regs, &regs, &segregs);
    segm=peek (segregs.es,regs.x.bx-2);
    printf(" Адреса MCB Розмір блока ім'я власника\n"
    "          пам'яті(байт)    \n\n ");
    ptr=MK_FP(segm,0);
    print ("04X:0000  8lu*,segm, (long) ptr->size
Para*16);
    if (!  Ptr->owner) printf("Блок   вільний");
    else if (_osmajor>=4&&(segm +1)=ptr->owner)
    {
    for (i=0;i<=7;i++) (printf("c",ptr->name[i]);
    }
    printf("\n\");
    while (ptr ->>marker='M')
    {
    segm=segm+ptr->sizePara+1;
    ptr=MK_FP(segm,0);
    printf( " 04X:0000 8lu*, segm,
    (long)ptr->sizePara*16);
    if (!ptr-> owner ) printf ("Блок   вільний");
    else if (_osmajor>=4&&(segm+1)==ptr->owner)
    {
    for(i=0;i<=7;i++)(printf ("C",  ptr->name[i]);
    }
    printf ("\n");
    }
    return 0;
    }
```

Розширена пам′ять

Пам′ять, вища ніж 1 Мб, називається розширеною пам′яттю - Extended Memory. Є дві специфікації використання цієї пам′яті:

а) EMS (Expanded Memory Specificatson) – програмна специфікація використання розширеної пам′яті DOS-програмами через 4 сторінки по 16 Кбайт, які відображають розширену пам′ять в область UMA (здебільшого з адреси D0000h);

б) XMS (Extended Memory Specification) - програмна специфікація використання розширеної пам′яті DOS-програми через переключення у захищений режим і навпаки.

2.3 Визначення ресурсів і діагностика ПЕОМ

Діагностика ПЕОМ включає в себе**:**

– визначення ресурсів ПЕОМ;

– одержання технічних, швидкісних та ін. характеристик як ПЕОМ в цілому, так і окремих її вузлів;

– знаходження місць з можливими помилками і збоїв у роботі обладнання.

Визначення ресурсів ПЕОМ зазвичай проходить у наступних випадках:

– коли необхідно з′ясувати, чи можлива коректна робота тієї чи іншої програми на ПЕОМ даної конфігурації (чи вдосталь оперативної чи дискової пам′яті, чи є принтер, чи відповідає відеоадаптер прийнятому стандарту і т.п.);

– при інсталяції (установці) програмного продукту на певну ПЕОМ, щоб правильно конфігурувати (відлагодити) програму для подальшої роботи;

– при інсталяції (установці) програмного продукту на з′ясовану ПЕОМ, з ціллю відвернення масового тиражування неліцензійної копії. В цьому випадку проводиться ще й діагностика ресурсів для "прив′язки", яку установлюють програми до особливості даної ПЕОМ;

– просто для одержання користувачем інформації, що його цікавить, про ресурси, які має ПЕОМ.

Діагностика комп'ютерних вірусів є окремим випадком загальної діагностики ПЕОМ.

Деяку інформацію про конфігурацію комп'ютера можна одержати з енергонезалежної пам′яті CMOS, яку мають комп′ютери класу IBM PC AT і PS/2. Місткість даної пам'яті 64 байти.

В ній зберігається різноманітна інформація (табл. 2.6), яка включає в себе поточну дату та поточний час, конфігурацію обладнання і т.д.

Таблиця 2.6 – Інформація про структуру енергозалежної пам’яті CMOS

Адреса	Опис
00H–0DH	Використовується годинником реального часу
0EH	Байт стану автоматичного тестування POST
0FH	Байт стану при поверненні у реальний режим
10H	Тип накопичувача на гнучких дисках
11H	Зарезервовано
12H	Тип накопичувача на твердих дисках (коли <15)
13H	Зарезервовано
14H	Байт конфігурації обладнання
15H–16H	Розмір базової пам'яті
17H–18H	Об'єм розширеної пам'яті вище 1Мбайт
19H	Тип твердого диску C (коли >15)
1AH	Тип твердого диску D (коли >15)
1BH–20H	Зарезервовано
21H–2DH	Зарезервовано
2EH–2FH	Пам'ять для контрольної суми змісту адресів від 10h до 20h
30H–31H	Об'єм розширеної пам'яті вище 1Мбайт
32H	Поточне століття у двійково-десятковому вигляді (наприклад 20)
33H	Інформація для допомоги
34H–3FH	Зарезервовано

Для того щоб прочитати байт з енергонезалежної пам'яті, необхідно спочатку надіслати потрібну адресу байту у порт 70h, а потім виконати команду читання байту із порту 71h.

Приклад 6. Програма для визначення розміру розширеної (extended) пам'яті.

```
/*Компілятор  Turbo C, Turbo C++ або Borland C++*/
     #include <stdio.h>
     #include <dos.h>
     #define AT 0xFC
     #define  PS_2_30 0 x FA
     #define  PS_2_80 0 x F8
     { unsigned char type_ibm, high_byte, low_byte;
     type_ibm=peekb (0 x F000,0 x FFFE;)
/*Байт F000:FFFE у ПЗП – код моделі IBM PC */
     if (type_ibm= =AT | |
     tipe_ibm= = PS_2_30| | tipe_ibm= = PS_2_80) }
     outportb (0 x 70 ,0 x 17);
     low_bute = inportb (0 x 71)
     outportb (0 x 70 ,0 x 18);
     high_byte = inportb (0 x 71);
     printf ("Об’єм розширеної пам’яті і Кбайт\ n",
high_bute
                  *256+low_bute);
     else
     printf("Розширена пам'ять відсутня\n ");
     return0;}
```

При необхідності здобути повнішу інформацію про ресурси ПЕОМ, чи провести їх детальний аналіз, використовують різні методи, орієнтовані на визначений пристрій з використанням, коли необхідно, опиту відповідних портів, використанням функцій BIOS і DOS, які працюють з даними пристроями (Приклад 6).

Приклад 7. Визначення наявності VGA-карти з використанням функції відео-BIOS з кодом 1AH. (Призначення цієї функції - повертати інформацію про те, у якому режимі працює відеокарта і який неактивний режим може підтримуватися. Ця функція є у всіх BIOS VGA і MCGA/):

```
     MOV AX,1A00h ;виклик інформаційної функції BIOS
     INT 10h
     CMP AL,1Ah   ;Функція BIOS підтримується, якщо
                  AL=1Ah
     JNE M_NOVGA
;Визначення, в якому режимі працює VGA
      MOV CL,BH   ;BL-активний дісплей, BH-неактивний
      XOR BH,BH   ;BX-відеорежим активного
      XOR CH,CH   ;CX-відеорежим неактивного
      CMP BX,07h  ;Активний VGA монохромний?
      JE M_VGA    ;VGA існує
      CMP CX,O7h  ;Неактивний VGA монохромний?
      JE M_VGA    ;VGA існує
      CMP BX,08h  ;Активний VGA кольоровий?
      JE M_VGA    ;VGA існує
      CMP CX,08h  ;Неактивний VGA кольоровий?
      JNE M_M
M_VGA:
...Виконання подальших дій з урахуванням наявності VGA
M_M:
      CMP BX,0Ch        ;Активний MCGA кольровий?
      JE M_MCGA
      CMP CX,0Ch
      JNE ,M_NOVGA      ;Не активний MCGA кольоровий?
M_MCGA:
...Виконання подальших дій з урахуванням наявності MCGA
M_NOVGA:
..Виконання подальших дій з урахуванням відсутності VGA
```

2.4 Захист інформації

У світі комп′ютерних технологій, що швидко розвиваються, актуальним є питання захисту інформації. Методи захисту різноманітні як по кінцевій цілі, так і по технічному виконанню; їх можна поділити на механічні, апаратні і програмні.

До механічних методів захисту належать різні кришки, чохли з замками (закриваючи, наприклад, дисковод гнучких дисків чи сітьових вимикачів), клейкі пластини для приклеювання терміналу до комп′ютеру, а комп′ютеру до столу, приміщення з сигналізацією і багато іншого.

Апаратні засоби реалізуються у вигляді спеціальних модулей, що підключаються до системного каналу комп′ютеру чи портів вводу-виводу і здійснюють обмін кодовими послідовностями з захищеними програмами.

Найбільш різноманітні програмні засоби. До них належать програми цифрації даних по заданому користувачем ключу, адміністратори дисків, які дозволяють обмежити доступ користувачів до окремих логічних дисків, методи установки програмного продукту з дистрибутивних дискет, які дозволяють виконати установку не більше заданого числа разів, запуск захищених програм за допомогою ключових дискет, які не копіюються, засоби захисту програм від копіювання та багато іншого.

Окремо можна виділити системи, що призначені для захисту інформації від дій комп′ютерних "вірусів" і відновлення пошкодженої інформації в результаті цих дій.

В системах обмеження (розмежування, контролю) доступу до інформації, а також до ресурсів обчислювальної системи використовуються диспетчеризація і протоколювання.

Диспетчеризація - це ієрархічне розмеження доступу до ресурсів ЕОМ як для груп користувачів, так і для окремих користувачів з присвоєнням кожній групі чи користувачу імені і паролю, які мають визначений пріоритет і необхідних для відкриття сеансу роботи чи доступу до окремих ресурсів ЕОМ.

Разом з диспетчеризацією звичайно використовується і протоколювання. В протокол заноситься інформація про користувачів, які проводять сеанси роботи, час початку і час закінчення кожного сеансу, інформація яка реєструє дії користувача під час сеансу роботи (список задач, виконаних під час сеансу, намагання доступу до системної, захищеної інформації і т.ін.).

Як протоколи так і списки імен і паролей повинні бути підвласні тільки особі, яка обслуговує систему (володіє системним паролем). Для цього використовують різні методи захисту цієї інформації, такі як:

– шифрування захищених даних (Приклад 8);

– організація збереження захищених даних і доступу до них нестандартними для DOS методами (Приклад 9).

Приклад 8. Найпростіший алгоритм побайтового шифрування (Кожний наступний байт шифрується шляхом сумування з попереднім).

```
/ * Запис*/
SPPOW (char a[], int dl)
{
int j;
for (j=0;j<dl;j++)
a [j+1]=a[j]+a[j+1];
a[0]=a[0]+a[dl-1];
return;
}
/*Читання*/
SPPOW (char a[], int dl)
{
int j;
a[0]=a[0]+a[dl-1];
for (j=0; j<dl;j++)
a[j+1]=a[j+1]-a[j];
return;
}
```

Приклад 9. Використання вільної області останнього кластеру файла для зберігання позначки.

1. Процедура запису позначки:

```
/* Запис п'яти байтів "metka" за межі файлу, ім'я якого задається в командному рядку запуску задачі*/
main (ig,v)
int ig;
char *v[];
{
unsigned char a[]="metka";
int n,j,k;
long l;
long filelength ();
n=_open (v[l],4);      /*відкрити файл*/
```

```
    if (n<0) abort ();
    i= filelength (n);/*визначити його довжину у
байтах*/
    lseek (n,01,2);   /*піти у кінець файлу*/
    write (n,a,5);    /*записати у кінець файлу
позначку*/
    close (n);        /*закрити файл*/
    n=_open (v[1],4); /*знов відкрити файл*/
    if (n<0) abort ();
    chsize (n , i);   /*зменшити його розмір на 5
байтів*/
    close (n);        /*закрити файл*/
```

2. Процедура читання позначки:

```
    /*Читання позначки за межами файлу. Ім'я файлу
задається в командному рядку*/
    main (ig,v)
    int ig;
    char *v[];
    {
    unsigned char a[6];
    int n ;
    long l ;
    n=_open (v[l],4); /*відкрити файл*/
    if (n<0) abort();
    i=filelength (n);/*визначити його довжину у
байтах*/
    lseek (n,51,2);   /*вийти за межі файлу на 5
байтів*/
    write (n,a,0);    /*записати 0 байтів за
межами файлу*/
    close (n);        /*закрити файл*/
    n=_open (v[1],4); /*знову відкрити файл*/
    if(n<0)abort();   /*встановити вказник на 5-й
від кінця файлу байт. На цей час у файлі вже
включена позначка - це останні 5 байтів файлу*/
    lseek (n,-51,2);
    _read (n,a,5);
```

```
chsize (n,i);
close (n);        /*закрити файл*/
if (a [0]="m"  && a [1]="e"  &&  a [2]="t");
printf ("\n Помітка знайдена");
else printf ("\n Помітка відсутня");
exit (0);
}
```

Серед багатьох систем програмного захисту інформації окремо можна виділити системи захисту від копіювання чи системи захисту авторських прав, що забезпечують заборону нелегального розповсюдження, використання чи зміни програмних продуктів.

Структура захисту від копіювання

Структура захисту від копіювання складається, як правило, із наступних компонент:

а) модуль перевірки недублюємої чи оригінальної інформації – перевіряє існування не копійованих ознак на носії інформації чи технічні характеристики визначеної ПЕОМ.

По розміщенню цього модуля можна виділити 3 основні типи захисту:

– системи з резидентним модулем перевірки, які створені за технологією файлового вірусу;

– системи з зовнішнім модулем перевірки, який винесено у окрему програму;

– системи з внутрішніми функціями перевірки (захищені пакети програм);

б) модуль захисту від перегляду і аналізу логіки системи;

в) модуль відповідності з захищеними структурами - забезпечує правильну роботу захищених програм і адекватне сприйняття захищених даних у випадку легальних копій.

Деякі автори вважають за необхідне додавати до системи захисту блок відповідної реакції, який повинен робити різні караючі дії у випадку нелегальних копій.

Приклад 10. Форматування доріжки ключової дискети не у полі копіювання.

```
#include  <bios.h>
...
// установити номер доріжки 41
ntrk=81;
// записати у буфер
for  (i=0 ; i<512 ; i++)
buffer [i]=0;
// заповнення ідентифікаторів формату
for (k=0; k<9; k++)
{
// номер доріжки
buf[4*k]=ntrk;
// номер поверхні
buf[4*k+1]=0;
// номер сектору від 1 до 9
buf[4*k+2]=k+1;
// довжина сектору 512 байт
buf[4*k+3]=2;
}
// форматування 41 доріжки
flag=biosdisk (5,0,0,ntrk,1,9,buf);
if (flag!=0)
{
printf ("Помилка форматування");
return(0);
}
// запись 1-го сектору на знов у
відформатовану доріжку
flag=biosdisk (3,0,0,ntrk,1,1,buffer);
if (flag!=0)
{
printf ("Помилка запису");
return(0);
}
```

Приклад 11. Поставлення позначки на жорсткому диску, використовуючи операцію довгого запису (робота з розширеною до 516 байт довжиною сектора).

```
;компіляція в COM.файл.
.MODEL TYNY
.CODE
ORG  100h
START:
PUSH CS
POP DS
MOV AH,2          ;функція звичайного читання
MOV AL,1          ;кількість секторів
MOV BX,OFFSET BF  ;зміщення      буферу      з
інформацією
MOV CX, 0003h
MOV DX,DS
MOV ES,DX
MOV DX, 0080h
INT 13h
MOV AH ,0Bh       ;функція додаткового запису
MOV AL,1
MOV BX, OFFSET BF ;зміщення      буферу      з
інформацією
MOV CX, 0003h     ;сегмент буферу
MOV DX,DS
MOV ES,DX
MOV DX,0080h
INT 13h           ;обнулимо додаток у 4 байти
MOV WORD PTR DOP,0
MOV WORD PTR DOP+2,0
MOV AH, 0AH;
MOV AL,1
MOV BX,OFFSET BF  ;зміщення      буферу      з
інформацією
MOV CX, 0003h     ;сегмент буферу
MOV DX,DS
MOV ES,DX
```

```
    MOV DX,0080h
    INT 13h          ;виводить      на      екран
додатковий байт
```

2.5 Інтерфейс користувача

Інтерфейс людина-комп′ютер забезпечує зв′язок між користувачем та процесом, який виконує деяке завдання.

Це дає користувачу можливість визначити, які завдання робити активними у даний момент часу, як передати їм дані для обробки і отримувати результати обробки. З точки зору програмного забезпечення склад інтерфейсу складається з двох компонент: набір процесів вводу-виводу і процес діалогу. Користувач ОС взаємодіє з інтерфейсом: через інтерфейс він посилає вхідні дані та приймає вихідні. Процеси по виконанню завдань викликаються інтерфейсом у ті моменти часу, які потрібно.

Основою для класифікації інтерфейсу людина-комп′ютер є структура діалогу. Вона визначає ступінь взаємодії між системою та користувачем. Також вона задає основні форми керування цими взаємодіями.

Традиційно відокремлюють чотири основні структури діалогів типу:

– питання-відповідь;

– меню;

– екранних форм;

– на базі команд.

Для оцінки придатності любого діалогу користуються наступними основними критеріями – природністю, послідовністю, стислістю, підтримкою користувача і гнучкістю.

Природний діалог – це такий діалог, який не примушує користувача, який взаємодіє з системою, змінювати свої традиційні способи рішення задач.

Послідовність можна розуміти так, що користувач, який засвоїв роботу однієї частини системи, не заплутається, коли розбиратиметься з особливостями іншої її частини.

Стислий діалог потребує від користувача вводу тільки мінімуму інформації, необхідної для роботи системи.

Підтримка користувача у процесі діалогу – це та міра допомоги, яку діалог дає користувачу у процесі його роботи з цією системою. Основними аспектами цієї підтримки є:

– кількість і якість інструкцій;

– характер звісток про помилки.

Гнучкість діалогу – це міра того, наскільки добре він відповідає різним рівням підготовки і виробництво роботи користувача. Така гнучкість передбачує, що діалог може порядковувати свою структуру чи вхідні дані.

Чотири традиційні структури діалогу у різних ступенях відповідають перечисленим основним критеріям. З точки зору взаємодії користувача та прикладної програми, один з них здається більш природним ніж інші. Вони передбачають різні ступені підтримки користувача.

Структура діалогу типу запитання-відповідь основана на аналогії з звичайним інтерв′ю. Система бере на себе роль інтерв′юера і отримує інформацію від користувача у вигляді відповідей на питання. Відповіді звичайно вводяться користувачем у вигляді текстового рядка, який набирають з клавіатури. Цей рядок може являти собою чи об′єкт із списків можливих відповідей чи будь-які дані. Природність діалогу у значній мірі залежить від характеру питань, що задає користувач або система.

Структура діалогу типу запитання-відповідь у значній мірі забезпечує підтримку користувача, тому що стисле навідне запитання при розумній побудові може бути зроблено самопояснюючим. У навідне запитання можна включити додаткову інформацію, наприклад формат виводу. Коли користувачу потрібна довідка, система може надати йому інформацію, яка потрібна йому для відповіді на поставлене питання. Дана структура не гарантує мінімального об′єму вводу, який оцінюється у кількості нажатих клавіш.

Меню є одним із поширених механізмів організації запитів на ввід даних під час діалогу, що керується системою. Суть структури меню полягає у тому, що у користувача є список можливих варіантів даних для вводу, з яких потрібно вибрати те, що потрібно. Меню у вигляді рядка даних може з′явитися зверху чи знизу екрану і часто залишається в цій позиції впродовж усього діалогу. Додаткове меню

подається у вигляді блоків даних, які випадають на екран" у поточному положенні курсору. Це меню зникає після вибору варіанту.

Один із способів отримання інформації від інших людей є таким, щоб задавати їм питання і вислуховувати їх відповіді. Такий спосіб знайшов відображення у структурі типу запитання-відповідь. Другий підхід - отримати від них усю інформацію загалом шляхом заповнювання форми. Цей принцип збору інформації лежить в основі структури діалогу типу екранної форми. У данному випадку перед користувачем ставиться запитання. Ця множина питань постійна у тому розумінні, що відповідь на попереднє запитання не впливає на те, яке запитання буде задане наступним.

Структура діалогу на основі мови команд досить розповсюджена, як і структура діалогу типу меню, в першу чергу тому, що вона дуже часто використовується в операційних системах. Звичайно діалог на базі мови команд працює у телетайпному режимі. Система нічого не виводить окрім постійної підказки, яка означає готовність системи до роботи.

Інтерфейси командних мов були поширені багато років. Однак у останні роки пішли зміни у інтерфейсі між комп′ютерними системами та їх користувачами. Головним чином вони стосуються взаємин з недосвідченими користувачами і направлені на зміну традиційного меню – структурованого діалогу. Типовим серед цих розробок є багаторазовий WIMP-інтерфейс, який можна розшифрувати наступним чином:

W – інформація подається користувачу на екрані дісплея у вигляді декількох вікон (windows);

I – об'єкти, з якими має справу система, подаються піктографічно у вигляді ікон (icons);

M – вибірка проводиться за допомогою маніпулятору типу "миша";

P – означає меню, які автоматично з′являються (pop-up) на екрані чи які користувач може "витягнути " (pull down) із рядку меню, яка знаходиться у верхній частині екрану.

Більшість систем мають в основі структури діалогу типу запитання-відповідь, меню чи структуру на базі команд. Рідко удається побудувати діалог для усієї системи, використовуючи лише

одну із цих структур. Більшість діалогів базуються на змішаних структурах, що об'єднують в собі декілька основних структур.

Приклад 12. Макроозначення для роботи з маніпулятором "миша".

```
;ініціалізація миші
m reset macro
xor ax,ax
int 33h
endm
;вивід та погасання покажчика миші на екран
m_show macro
mov ax,0001h
int 33h
endm
m_hide macro
mow ax,0002h
int 33h
endm
;отримати положення миші і стану клавіш
m_getpos macro
mow ax,0003h
int 33h
endm
;установити горизонт. межу
setx macro minx, maxx
mov ax,0007h
mov cx, minx
mov dx, maxx
int 33h
endm
;установити вертик. межу
sety macro miny, maxy
mov ax,0008h
mov cx, miny
mov dx, maxy
int 33h
endm
```

3 ОСНОВНІ ЕТАПИ КУРСОВОГО ПРОЕКТУ, ЇХ ЗМІСТ ТА ТРУДОЄМКІСТЬ

Усі завдання на проект передбачають наявність дослідницької (пошукової) частини, для виконання якої студент повинен всебічно вивчити, маючи по обраній (чи указаній викладачем) темі, публікації та практичні програмні реалізації. Відібраний матеріал необхідно систематизувати, проаналізувати і узагальнити.

Огляд і аналіз проведеного пошуку є основою для складання технічного завдання (ТЗ) на тему, яку розроблюють у курсовому проекті, а також повинен бути наданий у вигляді пояснювальної записки з обов'язковими посиланнями на джерела інформації.

Для виконання практичної частини проекту, тобто програмної реалізації теми курсового проекту, студент складає ТЗ, яке узгоджується керівником. В ТЗ повинно обов’язково бути присутнє наступне положення:

– повна назва курсового проекту;

– мета розробки і її призначення;

– стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків;

– основні вимоги до програми, що розробляється, початковим даним і результату;

– алгоритм функціонування програми;

– обґрунтований вибір апаратно-технічних засобів, операційної системи і мови програмування;

– точно визначенні системні функції, які використовує програма;

– основні обмеження на установку і використання програми.

Узгоджене керівником курсового проекту ТЗ є невід′ємною частиною пояснювальної записки. Робота по створенню ТЗ є першим етапом курсового проекту і складає 25% його трудоємкості.

Після узгодження ТЗ студент приступає до створення детального алгоритму функціонування програми і розробки інтерфейсу користувача. Інтерфейс може бути виконаний у текстовому чи у графічному вигляді з обов'язковим виконанням основних вимог по стандарту IBM-Microsoft, особливо у частині

використання спеціальних і функціональних клавіш, видів меню для користувача і вікон допомоги, існування статусних рядків підказки і вікон повідомлень програми, а також додаткових сервісних можливостей. Інтерфейс при цьому повинен відповідати темі, що реалізується в курсовому проекті, бути простим і зрозумілим для користувача ЕОМ без досвіду. Результатом цього етапу роботи є огляд і аналіз по пошуковій частині проекту, чітко документовані програмні модулі інтерфейсної частини програми (опис ідентифікаторів, призначення окремих програмних модулів, принципи роботи з інтерфейсом, посилання на використані матеріали при ведені розробки) і алгоритм функціонування усієї програми, що розробляється, які узгоджуються з керівником і також є невід′ємною частиною пояснювальної записки (при цьому тексти програмних модулів розміщуються в додатках, а алгоритм функціонування програми виноситься на плакат). Роботи по обробці алгоритму функціонування програми і розробки інтерфейсу користувача є другим етапом курсового проекту і складає 25% його трудоємкості.

На третьому етапі курсового проекту студенти виконують програмну реалізацію розробленого алгоритму, об′єднання інтерфейсної частини з основними програмними модулями і відлагоджування програми у цілому, а також кінцеве оформлення пояснювальної записки та графічної частини проекту, що по трудоємкості складає 40%. При цьому зберігаються вимоги до документування програмних модулів, наявність по тексту програм продуманих коментарів. Все це оформлюється у вигляді додатку "текст програми".

4 ВИМОГИ ДО ЗМІСТУ, ОФОРМЛЕННЯ І ОБ'ЄМУ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ ТА ГРАФІЧНОГО МАТЕРІАЛУ

Пояснювальна записка повинна бути оформлена відповідно з вимогами нормативно-технічної документації до текстових документів[4,5] і ЕСПД[1,2,3]. При цьому у пояснювальній записці обов′язковими є наступні розділи:

– титульний лист (згідно нормам МОНУ);

– завдання на розробку (2-х сторонній лист);

– реферат;

– зміст;

– перелік умовних означень, символів, одиниць і термінів;

– вступ (не більше однієї сторінки);

– огляд і аналіз стану тематики у теоретичному і практичному аспектах;

– аналіз технічного завдання;

– вибір та обґрунтування структури системи, що проектується (розробка теоретичної моделі);

– програмна реалізація (вибір засобів та компоновка проекту);

– керівництво оператора (інструкція для використання);

– керівництво програміста (інструкція для переробки або модернізації, та таблиці з назвами функцій та аргументами);

– висновки;

– перелік посилань.

Додаток 1. Технічне завдання (згідно з ЕСПД та СТП).

Додаток 2. Текст програми.

Додаток 3. Методика та результати іспитів (тестування) програмного продукту.

Графічна частина курсового проекту виконується на аркушах формату А3. Графічний ілюстраційний матеріал проходить нормоконтроль на відповідність ЕСПД.

Обсяг пояснювальної записки (без переліку додатків) 30...45 сторінок машинного тексту 14пт (1.5 інтервал), а графічна частина – 2 сторінки формату А3(перший – блок-схема або структурна схема програми).

5 КОНТРОЛЬ ЗА ХОДОМ ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ

Контроль за ходом курсового проекту організовує керівник проекту відповідно до трудоємкості етапів проектування, графіку навчального процесу і графіку поетапного контролю, які встановлює декан.

Для успішного виконання студентами курсового проекту керівник призначає консультації відповідно з планами учбового навантаження.

Порушення студентами етапності виконання курсового проекту без поважних причин неприпустимі. При зриві наступного етапу виконання проекту студент зобов'язаний надати письмові пояснення керівнику. Керівник зобов'язаний повідомити деканат щодо зриву студентом термінів виконання курсового проекту у наступному поетапному контролі, а у виключних випадках - в день отримання пояснення про неуспішність від студента.

6 ПОРЯДОК ЗАХИСТУ КУРСОВОГО ПРОЕКТУ

За три дні до встановлення терміну захисту курсового проекту студент зобов'язаний передати усі матеріали проекту, у тому числі тексти програм на машинному носії, своєму керівнику для оцінки стану готовності і рецензування проекту призначеним викладачем. Для оцінки готовності проекту до захисту керівнику дається два дні, після чого він або повертає курсовий проект студенту на доробку з указанням помилок і неточностей і призначає новий контрольний термін, або відправляє проект на рецензію зі своїм позитивним відгуком. Термін на рецензію – один день. За день до захисту студент отримує усі матеріали своєї роботи разом з відгуком керівника і рецензією для ознайомлення і підготовки до захисту.

Захист проводиться у спеціально призначені часи за графіком прийому курсових проектів, призначеною кафедрою комісією, в яку повинно входити не менше двох викладачів.

Під час захисту студент повинен проявити глибоке теоретичне знання і практичні навички у постановці задачі і її програмній реалізації для досягнення мети проекту. Разом з тим оцінці підлягає також вміння студента залишити прийняті у проекті рішення, довести правомірність загальної постановки задачі та вибраного підходу, проаналізувати можливі інші варіанти рішення поставленої задачі, знаходити їх слабкі і сильні сторони.

У випадку незадовільної оцінки усі рішення по повторним захистам приймає деканат.

ПЕРЕЛІК ПОСИЛАНЬ

1. Комп’ютерні науки, інформаційні технології та інженерія програмного забезпечення : навчальний посібник / під заг. ред. С.О. Субботіна. – Т. 1 Виконання, оформлення та захист випускних робіт бакалавра та атестаційних робіт магістра / [С. О. Субботін, С. К. Корнієнко, А. О. Олійник та ін.]. – Запоріжжя: ЗНТУ, 2018. – 133 с.

2. Методичний матеріал по документуванню програм. - Київ: АН України, СКТБ ПО ІК ім. В.М.Глушкова, ФАП АН України, 1992. – 70 с.