МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Дисципліна «Оперційні системи»

Лабораторна робота "Робота з файлами"

Запоріжжя

2025

Завдання:

- Ознайомитись з системними викликами для роботи з файлами, такими як: **open()**, **close()**, **read()**, **write()**, **lseek()**. А також з додатковими функціями **time()** та **sleep()**.
- Продемонструвати у роботі кожну з цих функцій (досить продемонструвати частину коду програми, де використовується функція).

Номер індивідуального завдання обчислюється за формулою:

(< номер завдання **>) = (<** свій номер у списку **> - 1) % 5 + 1**

Свій номер у списку дивись на останній сторінці поточного документу (Додаток 2).

- (**1**) Створити програму копіювання файлів (**fcopy**).
- (**2**) Створити програму пошуку у файлі заданої послідовності символів (**fsearch**).
- (**3**) Створити програму порівняння двух файлів (**fcompare**).
- (4) Створити програму, яка перезапише файл у зворотньому порядку (**freverse**).
- (5) Створити програму, яка дозволить розглядати файл, як масив (**farray**).

(1) **Технічні умови** (до програми **fcopy**)**:**

- Програма копіює будь які файли (текстові або бінарні) будь якого розміру.
- Ім’я файлів (орігінала та копії) вказуються у командному рядку.
- Максимальний обсяг даних, який може бути завантаженний у пам’ять не повинен перевищувати **64 кБ**
- Впродовж копіювання файлу на екран виводится прогрес виконаної роботи. Формат прогресу вибирається самостійно.
- По завершанню копіювання програма повідомляє скількі байтів було скопійовано та впродовж якого часу (у секундах).

(2) **Технічні умови** (до програми **fsearch**)**:**

- Програма обробляє тільки текстові файли.
- Ім’я файлу та послідовність символів до пошуку вказуються у командному рядку. Послідовність символів **ASCII** для пошуку записується у лапках.
- Максимальний обсяг даних, який може бути завантаженний у пам’ять не повинен перевищувати **64 кБ**
- Впродовж пошуку на екран виводится прогрес виконаної роботи. Формат прогресу вибирається самостійно.

- По завершанню пошуку програма повідомляє результат пошуку (знайдено чи ні), скількі байтів було переглянуто (до моменту знаходження вказанної послідовності символів) та впродовж якого часу (у секундах). Якщо вказана послідовність символів знайдена, то повідомляється индекс входження послідовності символів у файлі (номер байту з якого починається знайдена послідовність), у протилежному випадку повідомляється про її відсутність.

(3) **Технічні умови** (до програми **fcompare**)**:**

- Програма порівнює будь які файли (текстові або бінарні) будь якого розміру.
- Ім’я файлів вказуються у командному рядку.
- Максимальний обсяг даних, який може бути завантаженний у пам’ять не повинен перевищувати **64 кБ**
- Впродовж порівняння файлів на екран виводится прогрес виконаної роботи. Формат прогресу вибирається самостійно.
- По завершанню роботи програма повідомляє з якого по номеру байта у файлах знайдено не співпадіння, а також час роботи програми (у секундах).

(4) **Технічні умови** (до програми **freverse**)**:**

- Ім’я файлу вказуються у командному рядку.
- Максимальний обсяг даних, який може бути завантаженний у пам’ять не повинен перевищувати **64 кБ**.
- Реверс файлу по байтовий.
- Впродовж роботи на екран виводится прогрес виконаної роботи. Формат прогресу вибирається самостійно.
- По завершанню роботи програма повідомляє яку кількість перестановок було здіснено, а також час роботи програми (у секундах).

(5) **Технічні умови** (до програми **farray**)**:**

- Файл даних типизований, тип запису файлу **int**. Файл ствоюється самостійно, окремою програмою (**fmake**). Кількість записів не меньша ніж 1000000. Значення записів генеруються випадково.
- Ім’я файлу даних вказуються у командному рядку при старті основної програми **farray**.

- Максимальний обсяг даних, який може бути завантаженний у пам’ять не повинен перевищувати **64 кБ**.
- Розробити слідуючий набір функцій для роботи з файлом:
  - **`void add(const int net_val)`** Додає до файлу новий запис (кількість записів файлів збільшується, новий дописується в кінець файлу)
  - **`int get_at(const size_t index)`** Повертає значення запису фалу за індексом index
  - **`void set_at(const size_t index, const int new_val)`** Зберігає у файлі нове значення запису за індексом index
  - **`void remove(const size_t index)`** Видаляє запис з файлу за індексом index (новий файл не створювати, видаляти методом зсуву)
- Після відкриття файлу програма запитує у оператора індекс запису, значення якого необхідно прочитати, та вивести у вихідний потік. Якщо оператор на запит вводить значення **-1**, програма завершує роботу. Записи індексуються з нуля. Якщо оператор вводить індекс неіснуючого запису, програма повідомляє про відповідну помилку та продовжує роботу.

Зауваження:

“Ознайомитись” – це означа:

Чітко розуміти призначення той чи іншої функції

До якої **C** бібліотеки належить функція

Знати синтаксис її опису:

ім’я функції

параметри функції (призначення, типи)

що повертає функція по завершенню роботи:

у випадку успішного виконання

у випадку неуспішного виконання

Звіт:

Звіт оформлюється відповідно стандарту. Обов’язкові розділи звіту:

- тема роботи
- цілі роботи
- завдання

- інструменти (мова програмування, параметри компьютера на якому тестувалися програми)
- теоритична частина
- практична частина
  - загальний опис програми
  - розбиття програми на блоки, блок-схема програми
  - текст програми з коментарем
  - протокол роботи програми для кожного з тестів
- висновки

Приклад оформлення тексту програми:

- Дивись у **Додатку 1**

Теоритичні відомості

Файл – це іменована область пам’яті на носії інформації. Файли складаються з ***записів***. Один запис – це така структура даних, яка обробляється програмою, як єдиний блок даних. Тип файлу визначається по тому, як з ним працює програма. Один і той же файл можно обробляти як типизований, з чітко прописаною структурою записів так і не типизований, коли така структура запису не визначена. У випадку нетипизованого файлу запис виглядае як окремий байт даних. Тому нетипизований файл можно назвати як байтовий файл.

Щоб працювати з файлом, спочатку, необхідно його відкрити. Для цього операційна система надає програмі *системний виклик* ***open()***. Відкривши файл, над ним дозволяється виконувати наступні дії: додавати нові або змінювати вже існуючи дані (записи) (*системний виклик* ***write()***), читати вже існуючи у файлі записи (*системний виклик* ***read()***), керувати положенням курсору читання та запису (*системній виклик* ***lseek()***), закрити файл (*системний виклик* ***close()***). Слід зауважити, що після відкриття файлу курсор читання та запису встановлюється на нуль – на початок файлу, та після кожної операції читання або зберігання даних курсор переміщується на початок наступного запису у файлі, а якщо наступного запису не існує, то на кінець файлу. Функція **lseek()** дозволяє перемістити курсор читання та запису у будь яке положення в межах файлу, що дає змогу читати або записувати дані у будь якому порядку (прямий доступ до записів файлу – як у масивах доступ до його елементів). Завдяки наявності такої функції файл можно уявляти як звичайний масив, елементи якого називаються записами.

Щоб заміряти час робити програми існує функція **time()**, яка повертає поточний час у форматі одного цілого числа – кількість секунд від початку *комп’ютерної єри* (**1 січня 1970**). Для вимірювання часу роботи програми цю функцію слід викликати двічі: на початку роботи програми та перед її завершенням. Різниці отриманих значень і буде часом роботи програми, у секундах.

При необхідності, заблокувати роботу програми на деякий час, операційна система надає функцію **sleep()**. Параметр функції вказує, на скільки секунд процес повинен бути заблокований. У цей час процес не використовує центральний процесор, тобто не заважає виконанню інших задач.

Далі розглянемо синтаксис опису усіх вище згаданих функцій.

Системний виклик open() - відкриття файлу

```
#include <fcntl.h>
int open(const char *pathname, int flags, [ mode_t mode ]);

RETURN VALUE:
```

On success, **open**() return the new file descriptor (a nonnegative integer). On error, **-1** is returned and **errno** is set to indicate the error.

Зауваження: параметр, вказаний у прямокутних дужках є необов'язковим. Вказується тільки у випадку створення файлу.

Значення параметру **flags**:

Дозвіл:

```
O_RDONLY    Тільки читання
O_WRONLY    Тільки запис
O_RDWR      Читання та запис
```

Створення:

```
O_CREAT     Якщо такого файлу не існує, то створити його
O_TMPFILE   Тимчасовий файл
O_TRUNC     Скоротити до пустого файлу
```

Значення параметру **mode** (у випадку створення файлу):

```
S_IRWXU 00700 user has read, write, and execute permission
S_IRUSR 00400 user has read permission
S_IWUSR 00200 user has write permission
S_IXUSR 00100 user has execute permission

S_IRWXG 00070 group has read, write, and execute permission
S_IRGRP 00040 group has read permission
S_IWGRP 00020 group has write permission
S_IXGRP 00010 group has execute permission

S_IRWXO 00007 others have read, write, and execute permission
S_IROTH 00004 others have read permission
S_IWOTH 00002 others have write permission
S_IXOTH 00001 others have execute permission
```

Параметр **mode** моє сенс тільки під час створення файлу. Він визначає дозвіл для різних груп користувачів виконувати ті чи інші дії.

Системний виклик close() - закриття файлу

```
#include <unistd.h>
int close(int fd);
```

RETURN VALUE:

close() returns zero on success.

On error, **-1** is returned, and **errno** is set to indicate the error

Системний виклик read() - читання записів з файлу

```
#include <unistd.h>
int read(int fd,              // Файловий дескриптор потоку (файлу)
         char* buf,           // Буфер введення
         size_t size);        // Довжина буфера, у байтах
```

RETURN VALUE:

On success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number. It is not an error if this number is smaller than the number of bytes requested; this may happen for example because fewer bytes are actually available right now (maybe because we were close to end-of-file, or because we are reading from a pipe, or from a terminal), or because **read()** was interrupted by a signal **SIGPIPE**.

On error, **-1** is returned, and **errno** is set to indicate the error. In this case, it is left unspecified whether the file position (if any) changes.

Системний виклик write() - зберігання записів у файлі

```
#include <unistd.h>
int write(int fd,             // Файловий дескриптор потоку (файлу)
          char* buf,          // Буфер виведення
          size_t size);       // Довжина буфера, у байтах
```

RETURN VALUE:

On success, the number of bytes written is returned. On error, **-1** is returned, and **errno** is set to indicate the error. Note that a successful **write**() may transfer fewer than count bytes. Such partial writes can occur for various reasons; for example, because there was insufficient space on the disk device to write all of the requested bytes, or because a blocked **write**() to a socket, pipe, or

similar was interrupted by a signal handler after it had transferred some, but before it had transferred all of the requested bytes. In the event of a partial write, the caller can make another **write**() call to transfer the remaining bytes. The subsequent call will either transfer further bytes or may result in an error (e.g., if the disk is now full). If count is zero and fd refers to a regular file, then **write**() may return a failure status if one of the errors below is detected. If no errors are detected, or error detection is not performed, 0 is returned without causing any other effect. If count is zero and fd refers to a file other than a regular file, the results are not specified.

Системний виклик lseek() - переміщення курсору читання та запису

```
#include <unistd.h>
off_t lseek(int fd            // Файловий дескриптор потоку (файлу)
            off_t offset      // Зсув (у байтах відносно точки відліку)
            int whence);      // Точка відліку
```

RETURN VALUE:

Upon successful completion, **lseek()** returns the resulting offset location as measured in bytes from the beginning of the file. On error, the value (off_t) **-1** is returned and **errno** is set to indicate the error.

Значення параметру **whence**:

SEEK_SET

The file offset is set to offset bytes.

SEEK_CUR

The file offset is set to its current location plus offset bytes.

SEEK_END

The file offset is set to the size of the file plus offset bytes.

Параметр **whence** встановлює точку відліку, відносно якої відбувається переміщення курсору на вказану кулькість байтів **offset**.

Додаток 1

Приклад оформлення тексту програми на **C**, яка демонструє використання системних викликів **`open()`**, **`read()`**, **`write()`** і **`close()`** в Linux з обробкою помилок.

```
#include <stdio.h>      // fprintf()
#include <stdlib.h>
#include <time.h>       // time()
#include <fcntl.h>      // open()
#include <unistd.h>     // read(), write(), close(), exit()
#include <errno.h>      // errno
#include <string.h>     // strerror()

#define BUF_SIZE 1024   // Розмір буферу обміну даних

int main() {
  time_t start_time = time(NULL);
  // Відкриваємо вхідний файл для читання
  int src_fd = open("input.txt", O_RDONLY);
  if (src_fd == -1) {
    fprintf(stderr, "Помилка відкриття input.txt: %s\n", strerror(errno));
    exit(EXIT_FAILURE);
  }
  // Відкриваємо вихідний файл для запису (створюємо, якщо не існує)
  int dest_fd = open("output.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
  if (dest_fd == -1) {
    fprintf(stderr, "Помилка створення output.txt: %s\n", strerror(errno));
    close(src_fd); // Закриваємо вже відкритий файл
    exit(EXIT_FAILURE);
  }
  // Створення буферу обміну
  char buffer[BUF_SIZE];
  ssize_t bytes_read, bytes_written;
  // Читання з одного файлу і запис у інший
  while ((bytes_read = read(src_fd, buffer, BUF_SIZE)) > 0) {
    bytes_written = write(dest_fd, buffer, bytes_read);
    if (bytes_written != bytes_read) {
      fprintf(stderr, "Помилка запису у файл: %s\n", strerror(errno));
      close(src_fd);
      close(dest_fd);
```

```
      exit(EXIT_FAILURE);
    }
  }
  if (bytes_read == -1) {
    fprintf(stderr, "Помилка читання з файлу: %s\n", strerror(errno));
  }
  // Закриття файлів
  if (close(src_fd) == -1) {
    fprintf(stderr, "Помилка закриття input.txt: %s\n", strerror(errno));
  }
  if (close(dest_fd) == -1) {
    fprintf(stderr, "Помилка закриття output.txt: %s\n", strerror(errno));
  }
  time_t finish_time = time(NULL);
  fprintf(stdout, “Час роботи програми: %ld s\n”, finish_time - start_time);
  exit(0);
}
```

Додаток 2

Список студентів по курсу «Операційні системи» для визнання номеру індівідуального завдання.

1	Адамов Ярослав Родіонович	larchistrike@gmail.com
2	Астапенко Кирило Тарасович	kirill.ast98@gmail.com
3	Башлай Владислав Віталійович	vladbashlay123@gmail.com
4	Бурмагін Нікіта Сергійович	nnburmagin@gmail.com
5	Дереча Олексій Іванович	lollexa215@gmail.com
6	Доценко Денис Олексійович	denis.dotsenko.al@gmail.com
7	Дячко Діана Євгенівна	dianadiachko@gmail.com
8	Євстігнєєв Ростислав Денисович	yevstihnieiev.uni@gmail.com
9	Згурський Данило Олександрович	danilzgurs@gmail.com
10	Здор Віталій Віталійович	azot587@gmail.com
11	Іванченко Софія Олександрівна	sonyaivanchenko1@gmail.com
12	Кабанов Артем Іванович	artemkabanov108@gmail.com
13	Каптюх Костянтин Денисович	ravolut1onr@gmail.com
14	Караваєв Олександр Олександрович	koshak327@gmail.com
15	Колєснік Максим Олександрович	maksimkolesnik99@gmail.com
16	Крутько Дмитро Геннадійович	durkomom735@gmail.com
17	Кузін Іван Олексійович	ivankuzin3049@gmail.com
18	Лукіна Дарія Дмитрівна	lukinadariia@gmail.com
19	Матяш Ярослав Олександрович	matyhya23@gmail.com
20	Мірошниченко Богдан Володимирович	bogdanmirosnicenko@gmail.com
21	Могилін Владислав Олександрович	mogilinvlad@gmail.com
22	Мороко Ярослав Владиславович	legionemptys@gmail.com
23	Петрик Денис Сергійович	denispetrik714@gmail.com
24	Пригарін Богдан Сергійович	pryharin.bohdan@gmail.com
25	Сандак Поліна Андріївна	charasailar@gmail.com
26	Сапа Олександр Олександрович	sahasapa2000@gmail.com
27	Сапа Сергій Олександрович	serchie228@gmail.com
28	Тарасенко Нікіта Сергійович	tarasenkonikita27@gmail.com
29	Темченко Станіслав Андрійович	temchenkosv8@gmail.com
30	Хоренженко Іван Сергійович	ivankhorenzhenko22@gmail.com
31	Черевичний Ярослав Сергійович	yarik16122006@gmail.com
32	Черненко Едуард Кирилович	edikchernenko133@gmail.com
33	Шарій Юрій Андрійович	shariyya777@gmail.com
34	Шевченко Богдан Миколайович	ylua5555@gmail.com