Відокремлений структурний підрозділ «**Волинський фаховий коледж Національного університету харчових технологій**»

Освітньо-професійна програма: **Інженерія програмного забезпечення**

**ОК Системне програмування**

**Звіт лабораторної роботи №4**  
**Тема “**Створення програми для запису в файл та читання з файла з використанням системних викликів open(), read(), write(), close()**”**

Студентки 4-того курсу   
Групи ІПЗ-41  
**Павліхи І.В**

**Луцьк 2025**

Лабораторна робота №4

**Тема:** Створення програми для запису в файл та читання з файлу з використанням системних викликів open(), read(), write(), close()

**Мета роботи**

Навчитися застосовувати системні виклики open(), read(), write(), close() для роботи з файлами. Відмітити переваги їх використання, порівнюючи з роботою з файлами засобами С/С++ у середовищі Windows для прямого і послідовного доступу.

***Завдання 1.***

Створити програму для одного із процесів, яка ілюструє запис і читання інформації в файл. Для ілюстрації виконання цього завдання розглянемо наступну програму.

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

int main() { int fd; char string[] = "Hello, world!\n"; ssize\_t size;

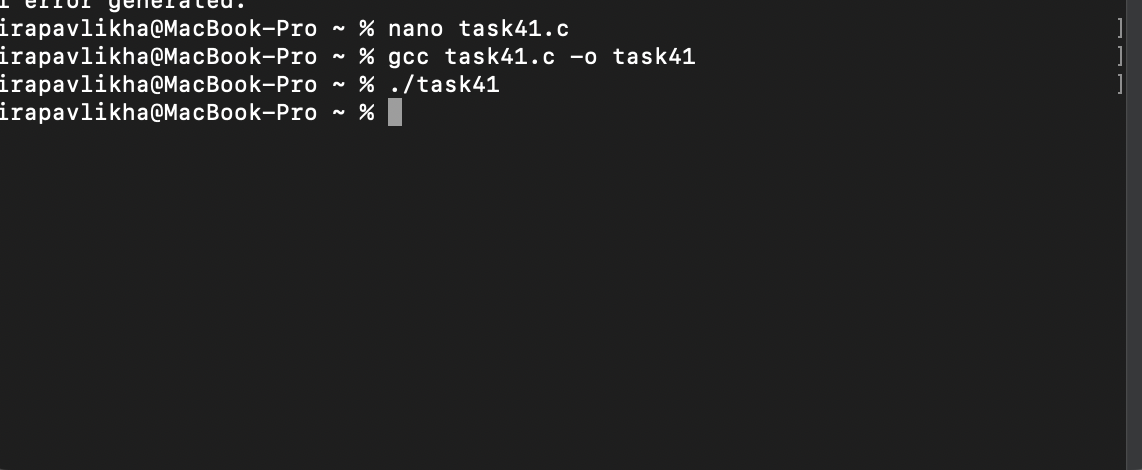
umask(0);  
  
fd = open("myfile", O\_WRONLY | O\_CREAT, 0666);  
if (fd < 0) {  
 printf("Can't open file!\n");  
 exit(-1);  
}  
  
size = write(fd, string, 14);  
if (size != 14) {  
 printf("Can't write all string!\n");  
 exit(-1);  
}  
  
if (close(fd) < 0) {  
 printf("Can't close file!\n");  
}  
  
return 0;

}

Використовується umask(0) для обнулення маски створення файлів поточного процесу, щоб права доступу точно відповідали параметру 0666. Файл myfile відкривається або створюється для запису (O\_WRONLY | O\_CREAT). Виконується запис рядка "Hello, world!\n" у файл за допомогою write(). Перевіряється, що записані всі 14 байт.

Файл закривається системним викликом close().

Програма передбачає тривіальний вихід при помилках відкриття, запису чи закриття файлу. Програма успішно записує рядок у файл, демонструючи базові операції з системними викликами.

Мал 1.1 - Виконання програми 1

**Завдання 2. (**Індивідуальне завдання**)**

Написати і відкомпілювати програму, яка б ілюструвала читання інформації з файла myfile, записаного в завданні 1. Для цього змінити програму з завдання 1 так, щоб вона читала інформацію з файла myfile і виводила її на екран, видаляючи з неї зайві оператори.

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

int main() {

int fd;

char buffer[100];

ssize\_t size;

fd = open("myfile", O\_RDONLY);

if (fd < 0) {

printf("Can't open file!\n");

exit(-1);

}

size = read(fd, buffer, sizeof(buffer) - 1);

if (size < 0) {

printf("Can't read file!\n");

exit(-1);

}

buffer[size] = '\0';

printf("%s", buffer);

if (close(fd) < 0) {

printf("Can't close file!\n");

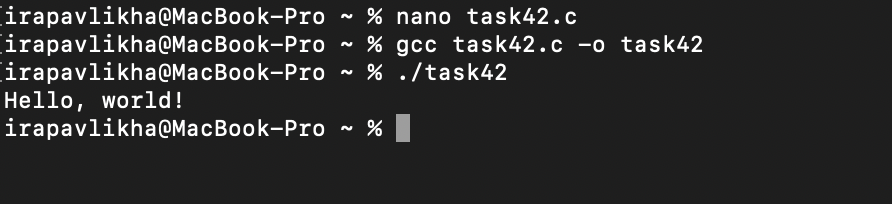
}

return 0;

}

Відкриття файлу myfile тільки для читання (O\_RDONLY). Зчитування вмісту файлу у буфер розміром 100 байт. Контроль реальної кількості зчитаних байт (size) та додавання кінця рядка '\0' для безпечного виводу. Вивід зчитаного рядка на екран. Коректне закриття файлу і тривіальний вихід у разі помилки.

Програма успішно читає інформацію з файлу і демонструє використання системного виклику read().

Мал 1.2 - Виконання програми 2

**Завдання 3. (**Творче завдання**)**

Написати і відкомпілювати програму, яка б ілюструвала читання інформації з файла myfile, записаного в завданні 1 і її запис у файл myfile1. Врахувати, що всі робочі файли повинні знаходитися в робочій директорії.

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

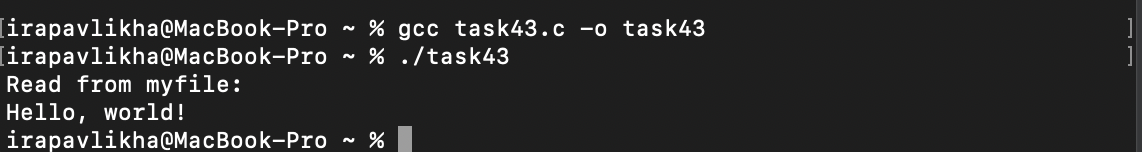
int main() { int fd1, fd2; char buffer[100]; ssize\_t size\_read, size\_write;

fd1 = open("myfile", O\_RDONLY);  
if (fd1 < 0) {  
 printf("Can't open myfile!\n");  
 exit(-1);  
}  
  
fd2 = open("myfile1", O\_WRONLY | O\_CREAT, 0666);  
if (fd2 < 0) {  
 printf("Can't open myfile1!\n");  
 close(fd1);  
 exit(-1);  
}  
  
size\_read = read(fd1, buffer, sizeof(buffer));  
if (size\_read < 0) {  
 printf("Can't read myfile!\n");  
 close(fd1);  
 close(fd2);  
 exit(-1);  
}  
  
buffer[size\_read] = '\0';  
printf("Read from myfile:\n%s", buffer);  
  
size\_write = write(fd2, buffer, size\_read);  
if (size\_write != size\_read) {  
 printf("Can't write all data!\n");  
}  
  
close(fd1);  
close(fd2);  
  
return 0;

}

Відкриття файлу myfile для читання і файлу myfile1 для запису (створюється за потреби). Зчитування інформації з myfile у буфер і контроль реальної кількості байт. Вивід зчитаної інформації на екран для підтвердження. Запис інформації у myfile1 з контролем кількості записаних байт. Закриття обох файлів та передбачення тривіальних виходів при помилках.

Програма демонструє роботу з двома файлами одночасно, перевірку кількості зчитаних та записаних байт, а також коректне завершення роботи.

Мал 1.3 - Виконання програми 3

**Пояснення до питань самоконтролю**

1. **open() / close()** – відкривають і закривають файл.
2. **read() / write()** – читають/записують байти; повертають кількість оброблених байт.
3. **Прапори open()** – O\_RDONLY, O\_WRONLY, O\_RDWR, O\_CREAT, O\_APPEND, O\_TRUNC.
4. **open()** – дескриптор файлу або -1 при помилці.
5. **read() / write()** – кількість реально зчитаних/записаних байт або -1.
6. **sizeof()** – так, для визначення розміру масиву або структури.
7. **fcntl.h** – містить константи та функції для роботи з файлами.
8. **read()** – адреса буфера для зчитування.
9. **write()** – адреса буфера для запису.
10. **close()** – 1 параметр (дескриптор файлу).
11. **umask(0)** – обнулює маску, щоб права доступу точно відповідали заданим.
12. **Прапори відкриття/створення** – ті ж, що й у пункті 3.
13. **Формула прав доступу** – r=4, w=2, x=1 для користувача, групи, інших.
14. **Комбінування вісьміркових кодів** – побітове OR (|).
15. **Комбінування прапорів** – також побітове OR (|).

**Висновок**

Виконано лабораторну роботу №4, освоєно системні виклики open(), read(), write(), close() для роботи з файлами. Було створено програми для запису (myfile), читання (myfile) та копіювання у новий файл (myfile1). Показано контроль кількості байт, тривіальні виходи при помилках і роботу з правами доступу через umask(0) та прапори O\_CREAT, O\_RDONLY, O\_WRONLY.