Системні виклики в UNIX/POSIX (файлові операції, fork(), qsort(), write(), read(), lseek() тощо

Загальні завдання

Завдання 8.1

Чи може виклик count = write(fd, buffer, nbytes); повернути в змінній count значення, відмінне від nbytes? Якщо так, то чому? Наведіть робочий приклад програми, яка демонструє вашу відповідь.

Завдання 8.2

Є файл, дескриптор якого — fd. Файл містить таку послідовність байтів: 4, 5, 2, 2, 3, 3, 7, 9, 1, 5. У програмі виконується наступна послідовність системних викликів:

```
lseek(fd, 3, SEEK_SET);

read(fd, &buffer, 4);
```

де виклик lseek переміщує покажчик на третій байт файлу. Що буде містити буфер після завершення виклику read? Наведіть робочий приклад програми, яка демонструє вашу відповідь.

Завдання 8.3

Бібліотечна функція qsort призначена для сортування даних будь-якого типу. Для її роботи необхідно підготувати функцію порівняння, яка викликається з qsort кожного разу, коли потрібно порівняти два значення.

Оскільки значення можуть мати будь-який тип, у функцію порівняння передаються два вказівники типу void* на елементи, що порівнюються.

- Напишіть програму, яка досліджує, які вхідні дані є найгіршими для алгоритму швидкого сортування. Спробуйте знайти кілька масивів даних, які змушують qsort працювати якнайповільніше. Автоматизуйте процес експериментування так, щоб підбір і аналіз вхідних даних виконувалися самостійно.

- Придумайте і реалізуйте набір тестів для перевірки правильності функції qsort.

Завдання 8.4

Виконайте наступну програму на мові програмування C:

```
int main() {

  int pid;

  pid = fork();

  printf("%d\n", pid);

}
```

Завершіть цю програму. Припускаючи, що виклик fork() був успішним, яким може бути результат виконання цієї програми?

Варіанти завдань:

1. Дослідіть поведінку write() при записі у FIFO, коли читачів немає, і поясніть результат.

2. Реалізуйте інтерпретатор команд, який підтримує черги читання-запису між процесами через pipe, не використовуючи shell.

3. Створіть програму, яка змінює вміст відкритого файлу без переміщення вказівника позиції читання/запису.

4. Проведіть експеримент і визначте, при яких умовах read() повертає менше байтів, ніж було запрошено.

5. Напишіть програму, яка симулює збій у середині операції write() і спробуйте зберегти цілісність даних.

6. Створіть експеримент, який показує вплив використання O_APPEND на поведінку write() у багатопроцесній програмі.

7. Розробіть програму, яка ілюструє різницю між dup() та dup2() у контексті перенаправлення потоків.

8. Напишіть тестовий фреймворк, який досліджує, як працює lseek() з файлами /dev/null та /dev/zero.

9. Проведіть тестування: що буде, якщо виконати fork() усередині циклу for із затримкою між викликами?

10. Побудуйте таблицю з часами виконання qsort() для різних схем вхідних даних і поясніть сплески в часі.

11. Зробіть самостійну реалізацію qsort() і порівняйте її поведінку з системною на масиві об’єктів з нестабільним порівнянням.

12. Напишіть програму, яка симулює часткове зчитування з файлу з паралельним записом з іншого процесу.

13. Побудуйте систему, яка перезаписує файл з одночасним використанням mmap() і write() та порівняйте результати.

14. Створіть програму, яка досліджує ефект різних позицій у файлі при читанні через pread().

15. Розробіть утиліту, яка виконує атомарний обмін вмістом двох файлів без створення тимчасових.

16. Напишіть код, який навмисно викликає race condition під час спільного доступу до файлу і зафіксуйте наслідки.

17. Продемонструйте ситуацію, в якій fork() і exec() можуть порушити логіку програми через неправильне управління дескрипторами.

18. Реалізуйте команду, яка виводить зсуви (offset) усіх відкритих файлів у процесі, який ви не створювали.

19. Дослідіть, як працює write() у ситуації нестачі дискового простору, і зафіксуйте поведінку змінної errno.

20. Напишіть програму, яка демонструє неочевидну поведінку при відкритті одного файлу кілька разів з різними прапорами.

21. Реалізуйте аналог tail -f без використання select(), poll() або inotify.

22. Створіть систему, де fork() створює дерево процесів, яке завершується каскадно через умови, задані у файлі.

23. Побудуйте схему аналізу фрагментів пам’яті, виділених через malloc(), при одночасному fork() і зміні значень.

24. Змоделюйте затримку у write() при записі на повільний пристрій, щоб виміряти ефективність буферизації.

25. Реалізуйте програму, яка демонструє взаємодію qsort() з неочевидними конфігураціями пам’яті (наприклад, сегментованими

масивами).

26. Напишіть утиліту, яка перевіряє, чи впливає порядок викликів lseek() та read() на вміст буфера.

27. Розробіть експеримент, який виявляє, чи можна отримати race condition між fork() і write() у реальній файловій системі.