



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра Інформаційної Безпеки

Операційні системи

Комп'ютерний практикум

Робота №10. Інтерфейс файлової системи в ОС Linux

Мета:

Ознайомитися з реалізацією файлових систем в Linux і основними структурами даних, що використовуються віртуальною файловою системою (VFS). Дослідити механізм доступу до файлів через інтерфейс віртуальної файлової системи в Linux.

Перевірив:

Виконав:

студент II курсу

групи ФБ-01

Сахній Н.Р.

Київ 2022

Завдання ДО ВИКОНАННЯ:

Варіант 4

Нехай N процесів здійснюють доступ до одного і того ж файлу на диску (але з різними режимами доступу). Розробити програму, яка демонструвала б динаміку формування таблиці відкритих файлів і зміни її елементів (при переміщенні покажчиків читання-запису, наприклад). Наприклад, сценарій програми може бути таким:

- відкриття файлу процесом 0 для читання;
- відкриття файлу процесом 1 для запису;
- відкриття файлу процесом 2 для додавання;
- читання зазначеного числа байт файлу процесом 0;
- запис зазначеного числа байт в файл процесом 1;
- додавання вказаного числа байт в файл процесом 2.

Після кожного з етапів друкуються таблиці файлів всіх процесів.

- Довільний текстовий файл (**Lorem ipsum dolor**)

```
nazar@ubuntu:~$ cd OS; mkdir lab_10; cd lab_10
nazar@ubuntu:~/OS/lab_10$ cat some_file.txt
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Maecenas ultricies mi eget mauris pharetra.
Consectetur purus ut faucibus pulvinar elementum.
Ante metus dictum at tempor commodo ullamcorper a lacus vestibulum.
Id venenatis a condimentum vitae sapien pellentesque habitant morbi.
nazar@ubuntu:~/OS/lab_10$
```

- Код програми:

```
nazar@ubuntu:~/OS/lab_10$ nano file_system.cpp
```

```
GNU nano 4.8 file_system.cpp
1 #include <iostream>
2 #include <fstream>
3 #include <cstring>
4
5 #include <sys/stat.h>
6
7 #include <unistd.h>
8 #include <fcntl.h>
9
10 using namespace std;
11
12
13 int main() {
14     struct stat buffer;
15     for (int num = 0; num < 3; num++) {
16         int fd;
17         pid_t child = fork();
```

```

18     if (child == 0) { // →Якщо це дочірній процес, то ...
19         sleep(1);
20         switch (num) { // Відповідно до номеру процесу обираємо вітку виконання
21             case 0:
22                 if ((fd = open("some_file.txt", O_RDONLY)) != -1) { // Відкриття файлу процесом 0 для читання ◀
23
24                     cout << '\n' << "0-ий процес та його PID = " << getpid() << endl;
25                     cout << "Файл відкрито 0-им процесом для читання." << endl;
26
27                     char buff[500];
28                     if (read(fd, buff, 300)) { // Читання зазначеного числа байт файлу процесом 0 •
29                         cout << "    Файл: " << '\n' << buff << endl;
30                     }
31                 }
32                 break;
33             case 1:
34                 if ((fd = open("some_file.txt", O_WRONLY)) != -1) { // Відкриття файлу процесом 1 для запису ◀
35                     sleep(2);
36
37                     cout << '\n' << "1-ий процес та його PID = " << getpid() << endl;
38                     cout << "Файл відкрито 1-им процесом для запису." << endl;
39                     string text = "Записуємо щось нове на початку файлу ... ";
40
41                     int size = text.size();
42                     char * buff = new char[size];
43                     strcpy(buff, text.c_str());
44
45                     if (write(fd, buff, size)) { // Запис зазначеного числа байт в файл процесом 1 •
46                         stat("some_file.txt", & buffer);
47
48                         cout << "У файлі {" << buffer.st_size << "} байтів інформації\n\n";
49                         cout << "Writing to the file ..." << endl;
50                         cout << "Новий рядок займає {" << size << "} байтів інформації\n\n" << endl;
51                     }
52                     delete[] buff;
53                 }
54                 break;
55             case 2:
56                 if ((fd = open("some_file.txt", O_RDWR | O_APPEND)) != -1) { // Відкриття файлу процесом 2 для додавання ◀
57                     sleep(4);
58
59                     cout << "2-ий процес та його PID = " << getpid() << endl;
60                     cout << "Файл відкрито 2-им процесом для додавання." << endl;
61                     string text = "Додаємо нову інфу в кінець файл\n";
62
63                     int size = text.size();
64                     char * buff = new char[size];
65                     strcpy(buff, text.c_str());
66
67                     if (write(fd, buff, size)) { // Додавання вказаного числа байт в файл процесом 2 •
68                         stat("some_file.txt", & buffer);
69
70                         cout << "У файлі {" << buffer.st_size << "} байтів інформації\n\n";
71                         cout << "Addition to the file ..." << endl;
72                         cout << "У файл було додано {" << size << "} байтів інформації\n\n" << endl;
73                     }
74                     delete[] buff;
75                 }
76                 break;
77             }
78             close(fd);
79             return 0;
80         }
81     }
82     return 0;
83 }
84

```

```

nazar@ubuntu:~/OS/lab_10$ g++ file_system.cpp -o file_system
nazar@ubuntu:~/OS/lab_10$ ls -l
total 24
-rwxrwxr-x 1 nazar nazar 19008 May 17 20:55 file_system
-rw-rw-r-- 1 nazar nazar 2430 May 17 20:53 file_system.cpp
nazar@ubuntu:~/OS/lab_10$

```

- Результат виконання програми:

```
nazar@ubuntu:~/OS/lab_10$ ./file_system
nazar@ubuntu:~/OS/lab_10$
0-ий процес та його PID = 398003
Файл відкрито 0-им процесом для читання.
Файл:
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    Maecenas ultricies mi eget mauris pharetra.
    Consectetur purus ut faucibus pulvinar elementum.
    Ante metus dictum at tempor commodo ullamcorper a lacus vestibulum.
    Id venenatis a condimentum vitae sapien pellentesque habitant morbi.
***
1-ий процес та його PID = 398004
Файл відкрито 1-им процесом для запису.
У файлі {297} байтів інформації

Writing to the file ...
Новий рядок займає {72} байтів інформації

2-ий процес та його PID = 398005
Файл відкрито 2-им процесом для додавання.
У файлі {355} байтів інформації

Addition to the file ...
У файл було додано {58} байтів інформації

nazar@ubuntu:~/OS/lab_10$ cat some_file.txt
Записуємо щось нове на початку файлу ... ricies mi eget mauris pharetra.
    Consectetur purus ut faucibus pulvinar elementum.
    Ante metus dictum at tempor commodo ullamcorper a lacus vestibulum.
    Id venenatis a condimentum vitae sapien pellentesque habitant morbi.
Додаємо нову інфу в кінець файл
nazar@ubuntu:~/OS/lab_10$
```

Висновки:

У результаті виконання даного комп'ютерного практикуму я навчився відкривати файли із різними режимами доступу: читати файл з зазначеним числом байт, додавати до нього довільний текст на початок та в кінець.

Я ознайомився з реалізацією файлових систем в Linux і основними структурами даних, що використовуються віртуальною файловою системою (VFS). Дослідив механізм доступу до файлів через інтерфейс віртуальної файлової системи в Linux. Також я дізнався як використовувати такі функції для роботи з файловою системою, як `mount()` й `unmount()`, `stat()` і `fstat()`, `link()` й `unlink()`, `pipe()` та `dup()` та інші стандартні системні функції: `open()`, `read()`, `write()`, `lseek()`, `close()`, які забезпечують звернення до файлів, що існують.

Контрольні запитання

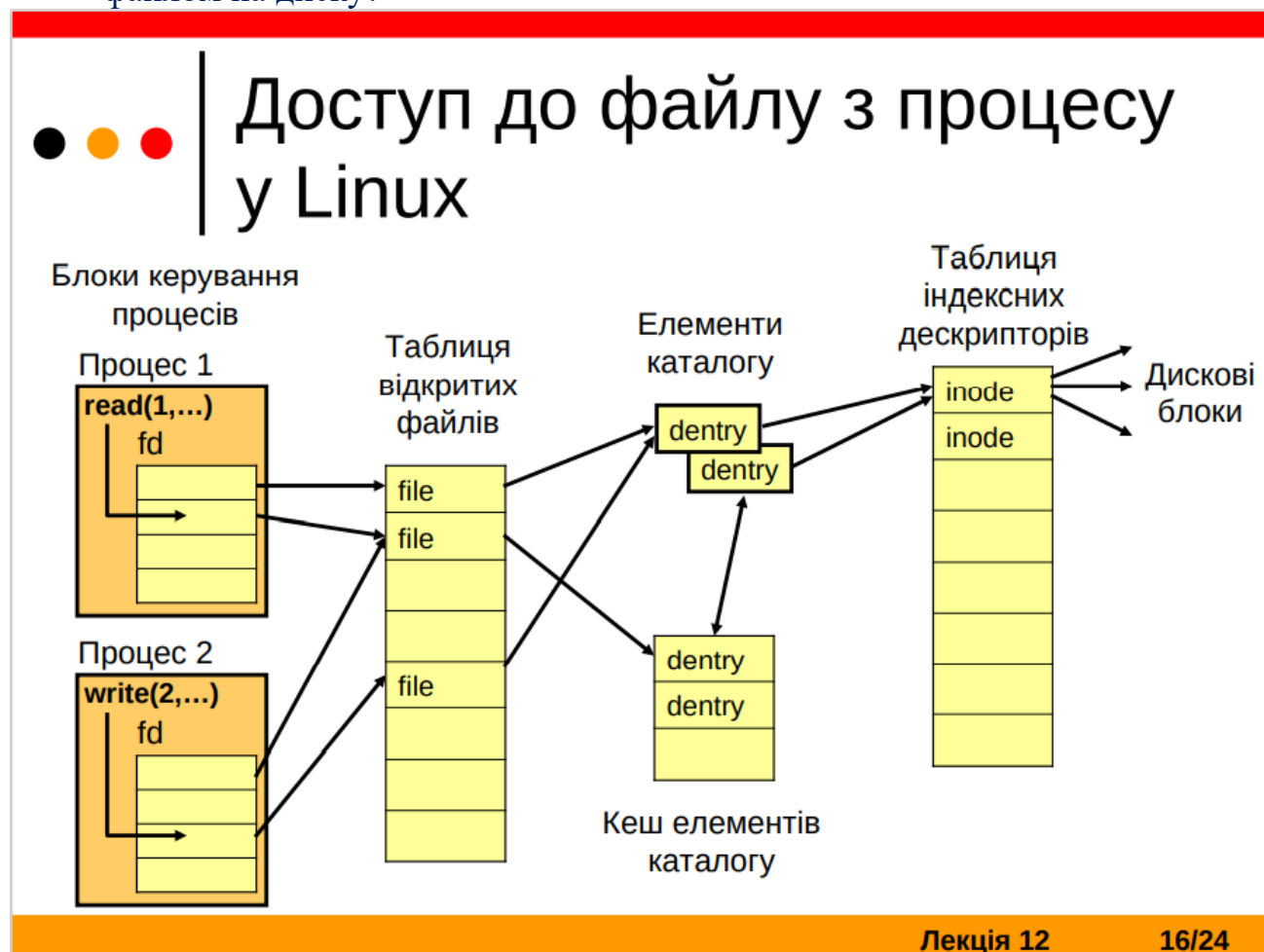
1. Яка структура дескрипторів файлів, таблиці відкритих файлів, таблиці відкритих файлів процесу?

Таблиця дескрипторів файлів являє собою структуру даних, що зберігається в оперативній пам'яті комп'ютера, елементами якої є копії дескрипторів файлів `inode`, по одній на кожен файл ОС Linux, до якого була здійснена спроба доступу.

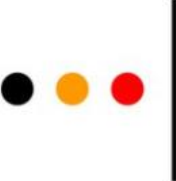
Кожен елемент `file` таблиці відкритих файлів містить інформацію про режим відкриття файлу, специфікований при відкритті файлу, а також інформацію про становище покажчика читання-запису.

У структурі масиву файлових дескрипторів процесу (`fd`) кожен елемент містить покажчик місця розташування відповідного елементу таблиці відкритих файлів (`file`), який у свою чергу містить посилання на елемент таблиці дескрипторів файлу (`inode`).

2. Яким є ланцюжок відповідності дескриптора файлу, відкритого процесом, і файлом на диску?



3. Опишіть функціональну структуру операції введення-виведення (пули, асоціація їх з драйверами, способи передачі інформації і т.д.).
4. Яким чином здійснюється підтримка пристроїв введення-виведення в ОС Linux?



Введення-виведення в UNIX і Linux

- Введення-виведення здійснюється через файлову систему
- Кожному драйверу пристрою відповідає один або кілька спеціальних файлів пристроїв
- Файли пристроїв традиційно розміщені у каталозі /dev
- Кожний файл пристрою характеризується чотирма параметрами
 - Ім'я файлу
 - застосовується для доступу до пристрою з прикладних програм
 - Тип пристрою (символьний чи блоковий)
 - фактично вказує на таблицю – одна таблиця для символьних пристроїв, друга – для блокових
 - Major number – номер драйвера у таблиці
 - ціле число, як правило, 1 байт (може 2)
 - Minor number – номер пристрою
 - це число передають драйверу, драйвер може працювати з кількома пристроями, у тому числі різними

Лекція 113/19

5. Яка структура таблиці відкритих файлів і масиву файлових дескрипторів процесу після відкриття файлу?

Під час відкриття файла системний виклик `open()` виконує такі дії: створює новий об'єкт `file`; зберігає адресу цього об'єкта у першому вільному елементі масиву файлових дескрипторів `fd`; повертає індекс цього елемента (файловий дескриптор), який можна використати для роботи з цим файлом.

6. Яка структура таблиці відкритих файлів і масиву файлових дескрипторів процесу після закриття файлу?
7. Яка структура таблиці відкритих файлів і масиву файлових дескрипторів процесу після створення каналу?

8. Яка структура таблиці відкритих файлів і масиву файлових дескрипторів процесу після створення нового процесу?

За замовчуванням під час створення процесу виділяють пам'ять під масив `fd` з 32 елементів. Якщо процес відкриє більше файлів, масив автоматично розширюється.

Перші три елементи масиву `fd` задаються автоматично під час запуску процесу, їм відповідають визначені файлові дескриптори. Кілька елементів масиву `fd` можуть вказувати на один і той самий об'єкт `file`. Доступ через кожний із них призводить до роботи з одним і тим самим відкритим файлом.