Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий Фізико-технічний інститут

ОПЕРАЦІЙНІ СИСТЕМИ

Комп'ютерний практикум
Робота №10

Виконав:
студент групи ФІ-12
Завалій Олександр
Перевірив:
Кірієнко О.В.

Київ-2023

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.

Варіант №5

Зміст індивідуального завдання:

Розробити програму, яка демонструвала б роботу ОС Linux при відкритті файлу процесом і читанні-записи в нього. При цьому досить показати тільки динаміку створення таблиць, пов'язаних з цією подією (таблиця дескрипторів файлу, таблиця відкритих файлів, масив файлових дескрипторів процесу). Наприклад, сценарій програми може бути таким:

- неявне відкриття стандартного файлу введення;
- неявне відкриття стандартного файлу виведення;
- неявне відкриття стандартного файлу виведення помилок;
- відкриття першого призначеного для користувача файлу;
- відкриття другого призначеного для користувача файлу;
- записування 20 байт в перший файл;
- зчитування 15 байт з другого файлу;
- записування 45 байт в перший файл.

Після кожного з етапів друкуються таблиця дескрипторів файлів, таблиця відкритих файлів, таблиця відкритих файлів процесів.

```
alex@Oleksandr:~/labs/lab_10$ cat lab_10.cpp
#include <iostream>
#include <fcntl.h>
#include <unistd.h>
#include <cstring>
#include <cstdlib>

using namespace std;

void displayOpenFilesTable() {
    FILE* stream = popen("lsof -p $$", "r");
    if (stream) {
        char buffer[256];
        while (!feof(stream)) {
            if (fgets(buffer, sizeof(buffer), stream) != NULL) {
                cout << buffer;
            }
        }
        pclose(stream);
    }
}

int main() {
    int fileDescriptor = open("test_file.txt", O_RDWR | O_CREAT, S_IRUSR | S_IWUSR);
    if (fileDescriptor == -1) {
        cerr << "Failed to open file.\n";
        return 1;
    }

    cout << "Open file table after opening a file:\n";
    displayOpenFilesTable();

    char buffer[100];
    ssize_t bytesRead = read(fileDescriptor, buffer, sizeof(buffer));
    if (bytesRead == -1) {
        cerr << "Error reading from file.\n";
        close(fileDescriptor);
        return 1;
    }
    cout << endl << bytesRead << " bytes have been read from the file." << endl;

    cout << "Open file table after reading from file:\n";
    displayOpenFilesTable();

    const char* message = "Hello, World!";
    ssize_t bytesWritten = write(fileDescriptor, message, strlen(message));
    if (bytesWritten == -1) {
        cerr << "Error writing to file.\n";
        close(fileDescriptor);
        return 1;
    }
    cout << endl << bytesWritten << " bytes are written to the file." << endl;
```

```
    cout << endl << bytesWritten << " bytes are written to the file." << endl;

    cout << "Open file table after writing to a file:\n";
    displayOpenFilesTable();

    if (close(fileDescriptor) == -1) {
        cerr << "Error closing file.\n";
        return 1;
    }

    cout << endl << "Open file table after closing the file:\n";
    displayOpenFilesTable();

    return 0;
}alex@Oleksandr:~/labs/lab_10$
```

```
alex@Oleksandr:~/labs/lab_10$ g++ -pthread -o lab_10 lab_10.cpp
alex@Oleksandr:~/labs/lab_10$ ./lab_10
Open file table after opening a file:
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
sh      6034 alex  cwd    DIR    8,3     4096  789353 /home/alex/labs/lab_10
sh      6034 alex  rtd    DIR    8,3     4096       2 /
sh      6034 alex  txt    REG    8,3   125688 1048777 /usr/bin/dash
sh      6034 alex  mem    REG    8,3  2216304 1056798 /usr/lib/x86_64-linux-gnu/libc.so.6
sh      6034 alex  mem    REG    8,3   240936 1056463 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
sh      6034 alex    0u   CHR  136,0      0t0       3 /dev/pts/0
sh      6034 alex    1w  FIFO   0,13      0t0   51482 pipe
sh      6034 alex    2u   CHR  136,0      0t0       3 /dev/pts/0
sh      6034 alex    3u   REG    8,3      113  815778 /home/alex/labs/lab_10/test_file.txt

100 bytes have been read from the file.
Open file table after reading from file:
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
sh      6037 alex  cwd    DIR    8,3     4096  789353 /home/alex/labs/lab_10
sh      6037 alex  rtd    DIR    8,3     4096       2 /
sh      6037 alex  txt    REG    8,3   125688 1048777 /usr/bin/dash
sh      6037 alex  mem    REG    8,3  2216304 1056798 /usr/lib/x86_64-linux-gnu/libc.so.6
sh      6037 alex  mem    REG    8,3   240936 1056463 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
sh      6037 alex    0u   CHR  136,0      0t0       3 /dev/pts/0
sh      6037 alex    1w  FIFO   0,13      0t0   51503 pipe
sh      6037 alex    2u   CHR  136,0      0t0       3 /dev/pts/0
sh      6037 alex    3u   REG    8,3      113  815778 /home/alex/labs/lab_10/test_file.txt

13 bytes are written to the file.
Open file table after writing to a file:
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
sh      6040 alex  cwd    DIR    8,3     4096  789353 /home/alex/labs/lab_10
sh      6040 alex  rtd    DIR    8,3     4096       2 /
sh      6040 alex  txt    REG    8,3   125688 1048777 /usr/bin/dash
sh      6040 alex  mem    REG    8,3  2216304 1056798 /usr/lib/x86_64-linux-gnu/libc.so.6
sh      6040 alex  mem    REG    8,3   240936 1056463 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
sh      6040 alex    0u   CHR  136,0      0t0       3 /dev/pts/0
sh      6040 alex    1w  FIFO   0,13      0t0   55396 pipe
sh      6040 alex    2u   CHR  136,0      0t0       3 /dev/pts/0
sh      6040 alex    3u   REG    8,3      113  815778 /home/alex/labs/lab_10/test_file.txt

Open file table after closing the file:
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
sh      6043 alex  cwd    DIR    8,3     4096  789353 /home/alex/labs/lab_10
sh      6043 alex  rtd    DIR    8,3     4096       2 /
sh      6043 alex  txt    REG    8,3   125688 1048777 /usr/bin/dash
sh      6043 alex  mem    REG    8,3  2216304 1056798 /usr/lib/x86_64-linux-gnu/libc.so.6
sh      6043 alex  mem    REG    8,3   240936 1056463 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
sh      6043 alex    0u   CHR  136,0      0t0       3 /dev/pts/0
sh      6043 alex    1w  FIFO   0,13      0t0   55418 pipe
sh      6043 alex    2u   CHR  136,0      0t0       3 /dev/pts/0
alex@Oleksandr:~/labs/lab_10$
```

Висновки

Файлові системи забезпечують організацію та керування збереженням даних на диску. Linux підтримує різноманітні типи файлових систем, такі як ext4, XFS, Btrfs та інші.

Віртуальна файлова система (VFS) забезпечує спільний інтерфейс для взаємодії з різними файловими системами. Вона дозволяє програмам працювати з файлами та каталогами незалежно від конкретної файлової системи, що використовується. Це робить розробку програм, які працюють з файловою системою, більш універсальною та зручною.

Основними структурами даних для VFS є іноді (inode), файлова таблиця (file table) та опис процесу (process descriptor). Іноди є структурою даних, яка містить інформацію про кожен файл або каталог. Файлова таблиця зберігає відкриті файлові дескриптори та іншу інформацію про відкриті файли. Опис процесу містить інформацію про кожен активний процес в системі.

Доступ до файлів здійснюється за допомогою системних викликів, таких як open, read, write, close і т.д. Ці системні виклики дозволяють програмам взаємодіяти з файлами. Вони забезпечують стандартизований спосіб доступу до файлів незалежно від конкретної файлової системи.

Тобто файлові системи в Linux є ключовим компонентом ОС і виконують важливі завдання з організації та управління файлами.