***Лабораторна робота №18***

***«Робота з файлами та каталогами Linux на мові C»***

***Теоретичні відомості***

Для виконання операцій запису і читання даних в існуючому файлі програмним шляхом мовою програмування **С/C++** його слід відкрити за допомогою виклику функції ***open()***.

***int open(const char \*pathname, int flags, [mode\_t mode]);***

***int fopen(const char \*pathname, int flags, [mode\_t mode]);***

Другий аргумент ***flags*** у системному виклику ***open()*** має цілочисельний тип і визначає метод доступу. Параметр ***flags*** приймає одне із значень, заданих постійними в заголовки ***fcnt1.h***. У файлі визначені три константи:

* ***O\_RDONLY*** - відкрити файл лише для читання;
* ***O\_WRONLY*** - відкрити файл лише для запису;
* ***O\_RDWR*** - відкрити файл для читання і запису;

або ***"r"***, ***"w"***, ***"rw"*** для ***fopen()***.

Третій параметр ***mode*** встановлює права доступу до файлу і є необов'язковим, він використовується тільки разом з прапором ***O\_CREAT***. Приклад створення нового файлу:

***#include <sys/types.h>***

***#include <sys/stat.h>***

***#include <fcnt1.h>***

***int Fd1;***

***FILE \*F1;***

***F1=fopen(“Myfile2.txt”, “w”, 644);***

***Fd1=open(“Myfile1.txt”, O\_CREAT, 644);***

Системні виклики ***stat*** і ***fstat*** дозволяють процесу визначити значення властивостей в існуючому файлі.

***#include <sys/types.h>***

***#include <sys/stat.h>***

***int stat(const char \*pathname, struct stat \*buf);***

***int fstat(int filedes, struct stat \*buf);***

Приклад: ***stat(“1.exe”, &st1);***

Аргументи функції ***stat***:

* ***pathname*** - повне ім'я файлу;
* ***buf*** -Структура типу ***stat***, яка після успішного виклику буде містити пов'язану з файлом інформацію.

У свою чергу структура ***stat*** включає наступні елементи:

***struct stat {***

***dev\_t st\_dev;*** /\* логічний пристрій, де знаходиться файл \*/

***ino\_t st\_ino;*** /\* номер індексного дескриптора \*/

***mode\_t st\_mode;*** /\* права доступу до файлу \*/

***nlink\_t st\_nlink;*** /\* кількість жорстких посилань на файл \*/

***uid\_t st\_uid;*** /\* ID користувача-власника \*/

***gid\_t st\_gid;*** /\* ID групи- власника \*/

***dev\_t st\_rdev;*** /\* тип пристрою \*/

***off\_t st\_size;*** /\* загальний розмір в байтах \*/

***unsigned long st\_blksize;*** /\* розмір блоку введення-виведення \*/

***unsigned long st\_blocks;*** /\* кількість блоків, що займає файл \*/

***time\_t st\_atime;*** /\* час останнього доступу \*/

***time\_t st\_mtime;*** /\* час останньої модифікації \*/

***time\_t st\_ctime;*** /\* час останньої зміни \*/

***};***

***Призначення прав доступу в Linux***

Права доступу до файлів представлені у вигляді послідовності бітів, де кожен біт означає дозвіл на **запис (w)**, **читання (r)** або **виконання (x)**. Права доступу записуються для власника (створювача) файлу **(owner)**; групи, до якої належить власник файлу **(group)**; і всіх інших **(other)** (див. лабораторну роботу №5). Для перегляду прав доступу можна використати функцію ***stat***.

Для запису прав доступу служить функція ***chmod***:

***#include <sys/types.h>***

***#include <sys/stat.h>***

***int chmod(const char \*pathname, mode\_t mode);***

Приклад: ***chmod(“1.exe”, 0777);***

***Робота з каталогами***

Каталоги в **Linux** - це особливі файли. Для відкриття або закриття каталогів існують виклики:

***#include <dirent.h>***

***DIR \*opendir(const char \*dirname);***

***int closedir(DIR \*dirptr);***

Для роботи з каталогами існують системні виклики:

* ***int mkdir(const char \*pathname, mode\_t mode)*** – створення нового каталогу;
* ***int rmdir(const char \*pathname)*** – видалення каталогу.

Перший параметр функції ***mkdir*** - ім'я створюваного каталогу, другий - права доступу:

Приклади:

***retval=mkdir(“/home/gleb/test”,0777);***

***retval=rmdir(“/home/gleb/test”);***

Зауважимо, що виклик ***rmdir("/home/gleb/test")*** буде успішним, тільки якщо каталог, що видаляється, є порожнім, тобто містить записи "точка" (***.***) і "подвійна точка" (***..***). Для читання записів каталогу існує виклик:

Для читання записів каталогу існує виклик:

***struct dirent \*readdir(DIR \*dirptr);***

Структура ***dirent*** є наступною:

***struct dirent {***

***long d\_ino;***

***off\_t d\_off;***

***unsigned short d\_reclen;***

***char d\_name [1];***

***};***

Поле ***d\_ino*** - це число, яке унікально для кожного файлу в файловій системі. Значенням поля ***d\_off*** служить зсув даного елемента в реальному каталозі. Поле ***d\_name*** є початком масиву символів, що задає ім'я елементу каталогу. Дане ім'я обмежене нульовим байтом і може містити не більше ***MAXNAMLEN*** символів. Тим самим описувана структура має змінну довжину, що зберігається в поле ***d\_reclen***.

Приклад виклику:

***DIR \*dp;***

***struct dirent \*d;***

***d=readdir(dp);***

При першому виклику функція ***readdir*** в структуру ***dirent*** буде зчитаний перший запис каталогу. Після прочитання всього каталогу в результаті наступних викликів ***readdir*** буде повернуто значення ***NULL***. Для повернення покажчика в початок каталогу на перший запис існує виклик:

***void rewindir(DIR \*dirptr);***

Щоб отримати ім'я поточного робочого каталогу існує функція:

***char \*getcwd(char \*name, size\_t size);***

***Завдання***

1. Написати програму введення символів з клавіатури і запису їх в файл (в якості аргументу при запуску програми вводиться ім'я файлу). Для читання або запису файлу використовувати тільки функції посимвольного введення-виведення (***getc()***, ***putc()***, ***fgetc()***, ***fputc()***. Передбачити вихід після введення певного символу та контроль помилок відкриття/закриття/читання файлу.

2. Написати програму виведення вмісту текстового файлу на екран (як аргумент при запуску програми передається ім'я файлу, другий аргумент (***N***) встановлює виведення по групах рядків (по ***N*** рядків) або суцільним текстом (***N=0***)). Для виведення чергової групи рядків необхідно очікувати натискання користувачем будь-якої клавіші. Для читання або запису файлу використовувати тільки функції посимвольного введення-виведення. Передбачити контроль помилок відкриття/закриття/читання/запису файлу.

3. Написати програму копіювання одного файлу в інший. Як параметри при виклику програми передаються імена першого і другого файлів. Для читання або запису файлу використовувати тільки функції посимвольного введення-виведення. Передбачити копіювання прав доступу до файлу і контроль помилок відкриття/закриття/читання/запису файлу.

4. Написати програму виведення на екран вмісту поточного та кореневого каталогів. Передбачити контроль помилок відкриття/закриття/читання каталогу.

***Контрольні питання***

1. Яка інформація визначена у структурі ***stat***?
2. Які системні виклики використовуються для зміни доступу до файлів?
3. Як програмним чином отримати поточний робочий каталог?
4. Як програмним чином змінити поточний робочий каталог?
5. Як програмним чином створити та видалити каталоги?
6. Як програмним чином прочитати вміст каталогу?