***#include <sys/stat.h>  
# Include <unistd.h>  
int mkfifo (char \* path, int mode);***

Функція ***mkfifo*** призначена для створення **FIFO** в операційній системі. Параметр ***path*** є покажчиком на рядок, що містить повне або відносне ім'я файлу, який буде міткою **FIFO** на диску. Для успішного створення **FIFO** файлу з таким ім'ям перед викликом функції не повинно існувати. Параметр mode встановлює атрибути прав доступу різних категорій користувачів до **FIFO**. Цей параметр задається як деяка сума наступних вісімкових значень:

Важливо розуміти, що файл типу **FIFO** не служить для розміщення на диску інформації, яка записується в іменований **pipe**. Ця інформація розташовується усередині адресного простору операційної системи, а файл є тільки міткою, що створює передумови для її розміщення. Не намагайтеся переглянути вміст цього файлу за допомогою **Midnight Commander (mc)!** Це призведе до його глибоке зависання!

Системні виклики **read ()** і **write ()** при роботі з **FIFO** мають ті ж особливості поведінки, що і при роботі з pip6 ом. Системний виклик ***open ()*** при відкритті **FIFO** також веде себе трохи інакше, ніж при відкритті інших типів файлів, що пов'язано з можливістю блокування виконують його процесів. Якщо **FIFO** відкривається тільки для читання, і прапор **O\_NDELAY** не заданий, то процес, який здійснив системний виклик, блокується до тих пір, поки який-небудь інший процес не відкриє **FIFO** на запис. Якщо прапор **O\_NDELAY** заданий, то повертається значення файлового дескриптора, асоційованого з **FIFO**. Якщо **FIFO** відкривається тільки для запису, і прапор **O\_NDELAY** не заданий, то процес, який здійснив системний виклик, блокується до тих пір, поки який-небудь інший процес не відкриє **FIFO** на читання. Якщо прапор **O\_NDELAY** заданий, то констатується виникнення помилки і повертається значення -1. Завдання прапора **O\_NDELAY** в параметрах системного виклику **open ()** призводить і до того, що процесу, який відкриває **FIFO**, забороняється блокування при виконанні подальших операцій читання з цього потоку даних і запису в нього.

***# include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <unistd.h>  
#include <stdio.h>  
int main () {  
int fd, result;  
size\_t size;  
char resstring [14];  
char name [] = "aaa.fifo";  
/ \* Обнуляємо маску створення файлів поточного процесу для того, щоб права доступу у створюваного FIFO точно відповідали параметру виклику mknod () \* /  
(void) umask (0);  
/ \* Спробуємо створити FIFO з ім'ям aaa.fifo в поточній директорії \* /  
if (mknod (name, S\_IFIFO | 0666, 0) <0) {  
/ \* Якщо створити FIFO не вдалося, друкуємо про це повідомлення і припиняємо роботу \* /  
printf ("Can \ 't create FIFO \ n");  
exit (-1);  
}  
/ \* Породжує новий процес \* /  
if ((result = fork ()) <0) {  
/\* Якщо створити процес не вдалося, повідомляємо про це та завершуємо роботу \*/***

***printf("Can\'t fork child\n");***

***exit(-1);***

***} else if (result > 0) {***

***/\* Ми знаходимося в батьківському процесі, який буде передавати інформацію процесу-дитини. У цьому процесі***

***відкриваємо FIFO на запис.\*/***

***if((fd = open(name, O\_WRONLY)) < 0){***

**Іменовані канали (FIFO)**

Як ми з'ясували, доступ до інформації про розташування ***pip'а*** в операційній системі і його стані може бути здійснений тільки через таблицю відкритих файлів процесу, який створив pipe, і через успадковані від нього таблиці відкритих файлів процесів-нащадків.

Тому викладений вище механізм обміну інформацією через pipe справедливий лише для споріднених процесів, що мають загального прабатька, що ініціював системний виклик pipe (), або для таких процесів і самого прабатька і не може використовуватися для потокового спілкування з іншими процесами. В операційній системі UNIX існує можливість використання ***pip'а*** для взаємодії інших процесів, але її реалізація досить складна і лежить далеко за межами наших занять.

Для організації потокового взаємодії будь-яких процесів в операційній системі UNIX застосовується засіб зв'язку, що одержало назву ***FIFO (від First Input First Output***) або іменований ***pipe. FIFO*** в усьому подібний pip6 у, за одним винятком: дані про розташування FIFO в адресному просторі ядра і його стані процеси можуть отримувати не через родинні зв'язки, а через файлову систему. Для цього при створенні іменованого pip6 а на диску заводиться файл спеціального типу, звертаючись до якого процеси можуть отримати цікаву для них інформацію. Для створення ***FIFO*** використовується системний виклик ***mknod ()*** або існуюча в деяких версіях UNIX функція **mkfifo ().**

Слід зазначити, що при їх роботі не відбувається дійсного виділення області адресного простору операційної системи під іменований ***pipe***, а тільки заводиться файл-мітка, існування якої дозволяє здійснити реальну організацію ***FIFO*** в пам'яті при його відкритті за допомогою вже відомого нам сітемного виклику ***open ().***

Після відкриття іменований pipe поводиться точно так само, як і неіменованого. Для подальшої роботи з ним застосовуються системні виклики ***read (),*** ***write ()*** і ***close ().*** Час існування **FIFO** в адресному просторі ядра операційної системи, як і у випадку з ***pip6*** ом, не може перевищувати час життя останнього з використали його процесів. Коли всі процеси, що працюють з **FIFO**, закривають всі файлові дескриптори, асоційовані з ним, система звільняє ресурси, виділені під **FIFO**. Вся непрочитана інформація втрачається. У той же час файл-мітка залишається на диску і може використовуватися для нової реальної організації **FIFO** надалі.

***# include <sys/stat.h>  
# include <unistd.h>  
int mknod (char \* path, int mode, int dev);***

Нашою метою є не повний опис системного виклику ***mknod***, а тільки опис його використання для створення **FIFO**. Тому ми будемо розглядати не всі можливі варіанти завдання параметрів, а тільки ті з них, які відповідають цій специфічній діяльності. Параметр ***dev*** є несуттєвим у нашій ситуації, і ми будемо завжди задавати його рівним 0. Параметр ***path*** є покажчиком на рядок, що містить повне або відносне ім'я файлу, який буде міткою **FIFO** на диску. Для успішного створення **FIFO** файлу з таким ім'ям перед викликом існувати не повинно. Параметр ***mode*** встановлює атрибути прав доступу різних категорій користувачів до FIFO. Цей параметр задається як результат побітової операції "або" значення **S\_IFIFO**, що вказує, що системний виклик повинен створити **FIFO**, і деякої суми наступних вісімкових значень:

**. 0400 0** дозволено читання для користувача, який створив **FIFO**;

. **0200 0** дозволена запис для користувача, який створив **FIFO**;

. **0040 0** дозволено читання для групи користувача, який створив **FIFO**;

. **0020 0** дозволена запис для групи користувача, який створив **FIFO**;

**. 0004 0** дозволено читання для всіх інших користувачів;

. **0002 0** дозволена запис для всіх інших користувачів

При створенні **FIFO** реально встановлюються права доступу виходять зі стандартної комбінації параметра **mode** і маски створення файлів поточного процесу **umask**, а саме 0 вони рівні **(0777 & mode) & ~ umask.**

При успішному створенні **FIFO** системний виклик повертає значення 0, при неуспішному 0 від'ємне значення.

5.5