***Лабораторна робота №21***

***«Програмування потоків у Linux»***

***Теоретичні відомості***

Основна мета використання потоків - це поділ програми на підзадачі, які можуть виконуватись паралельно. У порівнянні з процесами взаємодія і синхронізація потоків вимагає менше часу, оскільки потоки одного процесу виконуються в од-ном адресному просторі.

В ОС UNIX/Linux є API для потоків стандарту POSIX (Portable Operating System Interface) - ***pthreads*** ("P" - від POSIX). Прототипи функцій роботи з потоками і необхідні типи даних містяться в заголовку ***<pthread.h>***. Ці функції не включені в стандартну бібліотеку мови **С**, вони знаходяться в бібліотеці ***libthread***. Тому в командний рядок для компонування необхідно додати опцію (див. лабораторну роботу №14) ***-lpthread***

***Створення та завершення потоків***

Створення потоку. Потік створюється функцією ***pthtead\_create***:

***#include <pthread.h>***

***int pthread\_create( pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine)( void\*), void \*arg);***

Дана функція має 4 параметри:

1. Вказівник на змінну типу ***pthread\_t***, в неї буде записаний ID нового потоку;
2. Вказівник на-об'єкт атрибут потоку. Цей об'єкт управляє деталями взаємодії потоку з іншою програмою. Якщо параметр дорівнює ***NULL***, то потік буде створений з атрибутами за замовчуванням.
3. Вказівник на функцію потоку. Це звичайний вказівник на функцію типу ***void \* (\*) (void \*)***, тобто функція потоку приймає один параметр типу вказівник на ***void*** і повертає значення типу вказівник на ***void*** .
4. Значення атрибута потоку типу ***void \****. Це значення передається потоку як аргумент у функцію потоку. Через нього можна передати новому потоку параметри.

Після створення кожний потік виконує функцію потоку - звичайну функцію в програмі користувача. При завершенні цієї функції потік завершується.

Повернення з функції ***pthread\_create*** відбувається негайно, і вихідний потік продовжує виконання команд, що настають за викликом ***pthread\_create***. Одночасно новий потік починає виконання функції потоку. Функція ***pthread\_create*** повертає нуль у випадку успіху або нуль в разі помилки.

При нормальних умовах потік завершується двома способами:

1. Звичайне повернення з функції потоку. Змінна, яка повертається у ***return***, буде значенням, що повертається потоком.

2. Повернення за допомогою функції ***pthread\_exit***. Вона може бути викликана з будь-якої функції даного потоку. Аргумент цієї функції буде значенням, що повертається потоком.

**Приклад 1.** Програма створює потік, який безперервно друкує ***'x'*** на стандартний потік помилок. Після створення потоку головний потік безперервно друкує ***'o'*** на стандартний потік помилок. Призупинити виконання програми можна за допомогою <***Ctrl>-<S>***; відновити - будь-якою клавішею. Перервати програму можна за допомогою <***Ctrl>-<C>***.

***#include <pthread.h>***

***#include <stdio.h>***

***#include <stdlib.h>***

***void\* print\_xs (void\* unused)***

***{***

***while (1)***

***fputc (‘x’, stderr);***

***return NULL;***

***}***

***int main ()***

***{***

***int p;***

***pthread\_t thread\_id;***

***p = pthread\_create(&thread\_id, NULL, &print\_xs, NULL);***

***if (p != 0)***

***{***

***perror("Thread problem");***

***exit(1);***

***}***

***while (1)***

***fputc (‘o’, stderr);***

***return 0;***

***}***

***Передача даних в потік***

Оскільки тип аргументу, переданого в потік, - ***void \****, то для передачі одного параметра типу ***int*** його слід перетворити до типу ***(void \*)*** Для передачі більшої кількості параметрів аргумент потоку повинен бути вказівником на структуру або область даних, що містить передані параметри.

Необхідно, щоб дані, що передаються новому потоку, були доступні потоку, при цьому не слід передавати стекові змінні.

**Приклад 2.** Програма створює два нових потоки: один друкує ***'x'***, інший ***'o'*** на стандартний потік помилок. Кожний потік друкує певну кількість символів і потім завершується поверненням з функції потоку. Обидва потоки використовують одну і ту ж функцію, ***char\_print***, але викликають її з різними значеннями параметрів.

***#include <pthread.h>***

***#include <stdio.h>***

***struct char\_print\_parms***

***{***

***char character;*** /\* Символ, який друкувати \*/

***int count;*** /\* Скільки разів друкувати символ \*/

***};***

***void\* char\_print (void\* parameters)***

***{***

/\* Перетворити вказівник до потрібного типу \*/

***char\_print\_parms\* p = (char\_print\_parms\*) parameters;***

***int i;***

***for (i = 0; i < p->count; ++i)***

***fputc (p->character, stderr);***

***return NULL;***

***}***

***int main ()***

***{***

***pthread\_t thread1\_id;***

***pthread\_t thread2\_id;***

***char\_print\_parms thread1\_args;***

***char\_print\_parms thread2\_args;***

***thread1\_args.character = ’x’;***

***thread1\_args.count = 30;*** /\* Друкувати 'x' 30 разів \*/

***pthread\_create (&thread1\_id, NULL, &char\_print, &thread1\_args);***

***thread2\_args.character = ’o’;***

***thread2\_args.count = 20;*** /\* Друкувати 'о' 20 разів \*/

***pthread\_create (&thread2\_id, NULL, &char\_print, &thread2\_args);***

***return 0;***

***}***

***Об'єднання потоків***

У разі потоків аналогом функції ***wait*** є функція ***pthread\_join***: потік, який викликав цю функцію, буде очікувати завершення зазначеного потоку. Функція повертає нуль у випадку нормального виконання, і не нуль в разі помилки. Функція має два параметри: ID потоку, завершення якого слід очікувати, і змінну типу вказівник на ***void***, куди буде записано значення, що повертається потоком. Якщо це значення не потрібно, то другий параметр функції ***pthread\_join*** може бути ***NULL***.

***#include <pthread.h>***

***int pthread\_join (pthread\_t thread, void \*\*status\_addr);***

***Завдання***

1. Написати програму, яка створює два дочірніх потоки. Батьківський процес і два дочірніх потоки повинні виводити на екран свій ***id*** і ***pid*** батьківського процесу і поточний час в форматі: година: хвилини: секунди: мілісекунди.
2. Знайдіть помилку у Прикладі 2 лабораторної роботи. Для налагодження програми додайте в функцію потоку друк переданих параметрів. Виправте помилку і поясніть її причину.
3. Модифікуйте Приклад 2 наступним чином:

* додайте в головний потік виклик функцій ***pthread\_join*** для очікування завершити обидва дочірніх потоків;
* поверніть з дочірніх потоків якісь значення (різні!) і роздрукуйте їх в головному потоці.

***Контрольні питання***

1. Як програмним чином створити новий потік?
2. Як програмним чином завершити потік?
3. Як програмним чином приєднати потік?
4. Як програмним чином від’єднати потік?