Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Лабораторна робота №7

з дисципліни «Операційна система UNIX»

«Потоки, перенаправлення потоків»

Виконав:

студент 2-го курсу, НН ІАТЕ

групи ТР-23

Ровний Григорій Олександрович

Перевірила:

проф. Левченко Л.О.

КИЇВ 2023

**Мета роботи:**

ознайомитись із системними викликами для управління потоками в ОС Linux, та перенаправлення потоків при роботі файлами і командами.

**Теоретичні відомості**

**Управління потоками.** Процес розбивається на виконувані одиниці - потоки (threads).Потік – набір послідовностей виконуваних команд процесора, які використовуютьзагальний адресний простір процесу, це елемент виконання всередині процесу:віртуальний процесор, стек або статус програми. У порівнянні з процесами взаємодія ісинхронізація потоків вимагає менше часу, оскільки потоки одного процесувиконуються в одному адресному просторі.

Процес містить один або кілька потоків. Якщо процес містить тільки один потік, такий процес називається однопоточними. Це класичні процеси UNIX. Якщо процес містить більше одного потоку, такі процеси називаються багатопоточними. Існує дві основні категорії реалізації потоків: користувацькі потоки та потоки ядра - потоки, що реалізуються через системні виклики і працюють в просторі ядра.

В ОС Linux для потоків реалізований стандарт Р-потоків - POSIX - pthreads. Для написання багатопотокової програми API для роботи з Р-потоками надає біля 100 інтерфейсів. Кожна функція в API забезпечена префіксом pthread\_.

Прототипи функцій роботи з потоками і необхідні типи даних містяться в заголовки <pthread.h>. Ці функції не включені в стандартну бібліотеку мови С, вони знаходяться в бібліотеці libthread. Тому в командному рядку для gсс необхідно додати опцію «- pthread»: *gcc -Wall -Werror -pthread beard.c -o beard.*

Кожнен потік має свій ID потоку. В програмах на С/С++ для ID потоків слід використовувати тип pthread\_t з <sys/types.h>. При роботі з потоками використовуються основні функції: створення потоку; блокування роботи потоку в очікування завершення іншого; дострокове та звичайне завершення роботи потоків;

Створення потоку. Потоки створюються функцією pthread\_create. Ця функція визначена в заголовки <pthread.h>. Перший параметр цієї функції є вказівником на змінну типу pthread\_t, в яку буде записано адресу ідентифікатора створюваного потоку. Другий параметр є вказівником на змінну типу pthread\_attr\_t, використовується для установки атрибутів потоку. Цей об'єкт управляє деталями взаємодії потоку з іншою програмою. Третім параметром функції pthread\_create повинна бути адреса функції потоку – вказівник на функцію потоку. Ця функція відіграє для потоку ту ж роль, що функція main для головної програми. Функція потоку приймає один параметр типу покажчик на void і повертає значення типу вказівник на void. Четвертий параметр функції pthread\_create має тип void\*. Цей параметр може використовуватися для передачі значення як аргумент у функцію потоку. Після виклику pthread\_create функція потоку буде запущена на виконання паралельно з іншими потоками програми. Функція повертає 0 в разі успіху або код помилки – в разі невдачі.

Ідентифікатори потоків (Thread ID) для потоків є аналогами ідентифікаторів процесів (PID). У той час як PID призначаються ядром Linux, TID призначаються бібліотекою Р-потоків. Цей тип представлений pthread\_t, і POSIX не вимагає, щоб він був арифметичним. TID нового потоку визначається за допомогою аргументу thread при успішному виклику pthread\_create().

**Блокування роботи потоку.** Приєднання дозволяє одному з потоків блокуватися в очікуванні завершення іншого.

**Дострокове завершення потоків.** Р-потоки викликають завершення інших потоків через їх скасування. Це забезпечує функція pthread\_cancel(). Успішний виклик pthread\_cancel() надсилає запит на скасування потоку, представленому через ідентифікатор потоку thread. Чи може потік бути скасований і коли, залежить від його стану відміни і типу скасування. Стан відміни потоку може бути доступно або недоступно. Потоки можуть змінювати свій стан через pthread\_setcancelstate():

Тип скасування потоку може бути асинхронним або відкладеним. З асинхронним типом скасування потік може бути убитий в будь-якій точці після отримання команди на скасування. З відкладеним типом потік може бути убитий тільки в спеціальних точках скасування, які є функціями Р- потоків або бібліотеки С і являють собою безпечні моменти

**Завершення роботи потоків.** Завершення роботи потоків схоже на завершення роботи процесів, за винятком, коли потік завершується, інші потоки в процесі продовжують виконуватися. Потоки можуть перериватися:

- якщо потік повертається зі стартової процедури, він переривається;

- якщо потік викликає функцію pthread\_exit(), він завершується;

- якщо потік скасовується іншим потоком через функцію pthread\_cancel(), він завершується; це аналог відправки сигналу SIGKILL через kill(). Найпростіший шлях потоку для завершення самого себе, - це «вихід за границі» своєї початкової процедури. Однак часто потрібно завершити потік десь в глибині стека виклику функції, достатньо далеко від стартової процедури. Для таких випадків в Р- потоках є виклик pthread\_exit(), потоковий еквівалент exit().

**Синхронізація потоків**

Усі потоки виконуються в одному адресному просторі. У зв'язку з цим постає проблема спільного використання загальних змінних, доступу до певного ресурсу, оскільки в один момент часу тільки єдиний потік повинен працювати з певним розділюваним ресурсом. Така задача має назву забезпечення взаємовиключення, а ділянки програмного коду, в яких потоки виконують операції з розділюваними ресурсами, називаються критичними секціями. З іншого боку можлива ситуація, коли потоку для продовження своєї роботи потрібно результат виконання іншого потоку, що потребує синхронізації дій. Для узгодженості взаємодії потоків розроблені засоби синхронізації потоків - м'ютекси (взаємні виключення), семафори і умовні змінні.

М’ютекс дозволяє потокам управляти доступом до даних. При використанні м’ютекса тільки один потік в певний момент часу може заблокувати м’ютекс і отримати доступ до ресурсу (право на використання). По завершенні роботи з ресурсом потік повинен повернути це право, розблокувавши м’ютекс. Якщо будь-який потік звернеться до вже заблокованого м’ютексу, то він буде змушений чекати розблокування м’ютекса потоком, який їм володіє.

Семафор призначений для синхронізації потоків щодо дій та даних. Семафор – це захищена змінна, значення якої можна опитувати і міняти тільки за допомогою спеціальних операцій P і V і операції ініціалізації. Семафор може приймати ціле невід'ємне значення. При виконанні потоком операції P над семафором S значення семафора зменшується на 1 при S> 0 або потік блокується, «чекаючи на семафорі», при S=0. При виконанні операції V(S) відбувається пробудження одного з потоків, які очікують на семафорі S, а якщо таких немає - значення семафора збільшується на 1. При вході в критичну секцію потік повинен виконувати операцію P, а при виході з критичної секції операцію V. Прототипи функцій для маніпуляції з семафора описуються у файлі semaphore.h.

**Поставлене завдання:**

1. Використати вихідні тексти, наведених прикладів для потоків, м’ютексів та семафорів, для створення програм.
2. Пояснити та прокоментувати результати роботи програм та їх особливості.
3. Створити файл з особистими даними (група, ПІБ), використовуючи стандартне введення з клавіатури. Вивести дані на екран. Додати у файл інформацію про ваше хоббі. Створити канал.

**Результат виконання роботи**

**Висновок:**

В результаті виконання лабораторної роботи…