Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра цифрових технологій в енергетиці

ЗВІТ   
з виконання лабораторної роботи №1  
з дисципліни «Технології паралельних обчислень в енергетичних комплексах

»

«Оброблення сигналів в багатопотокових застосунках»

Варіант 21

Виконав: студент групи ТР-12 Руденко В.І

Завдання:

Пересвідчитись у тому, що потоки в ОС Linux реалізовані як група процесів, яку очолює головний потік. Функція getpgrp. Напишіть багатопотокову програму та з’ясуйте, якому потоку буде доставлений сигнал, якщо декілька потоків одночасно очікують надходження того самого сигналу за допомогою функції sigwait. Оброблення сигналів в багатопотокових застосунках 3 Напишіть багатопотокову програму та з’ясуйте, в який спосіб буде оброблений доставлений сигнал у випадку одночасного використання деяким потоком функцій sigwait та sigaction.

Хід Роботи:

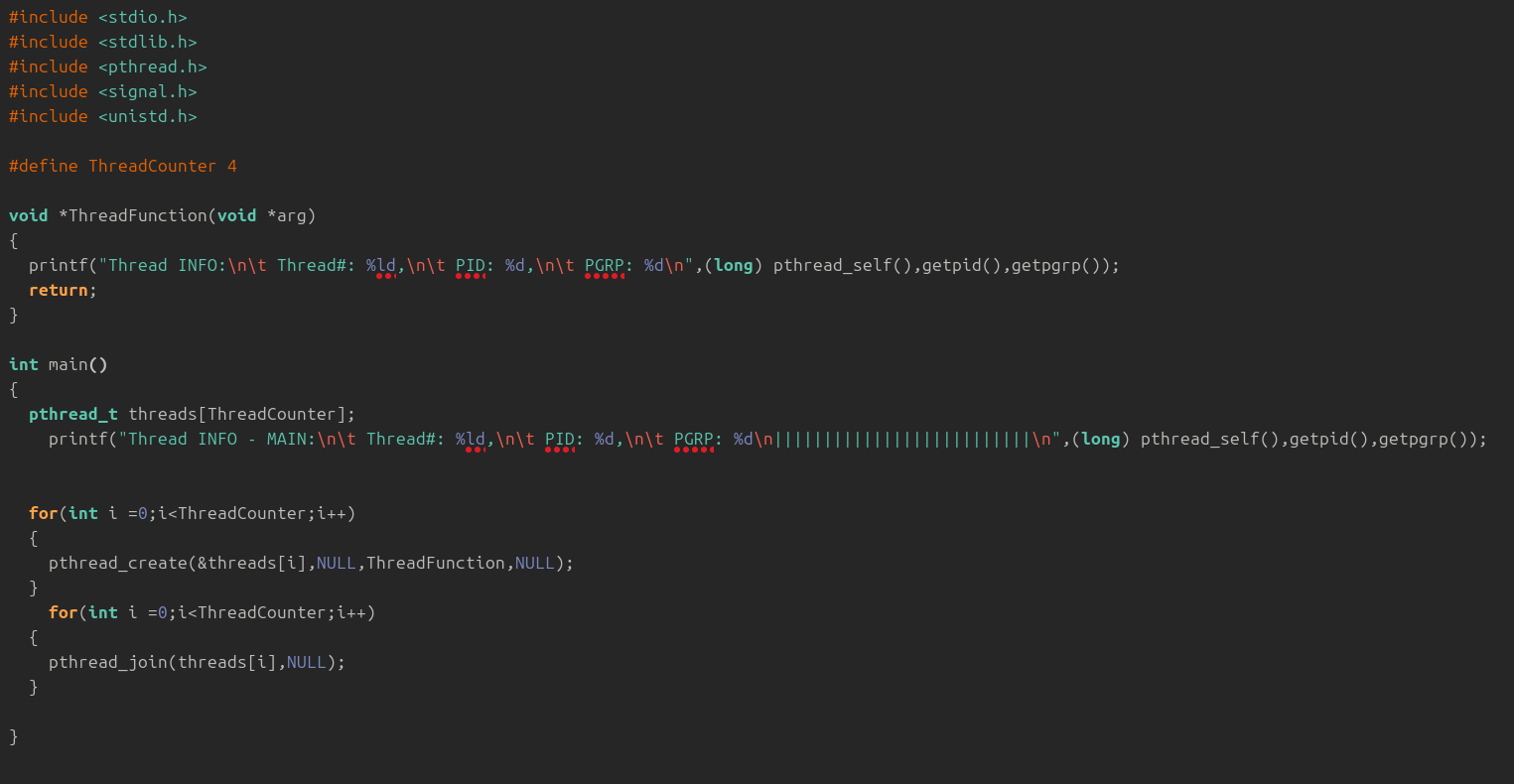


Table 1 Код програми Завдання 1

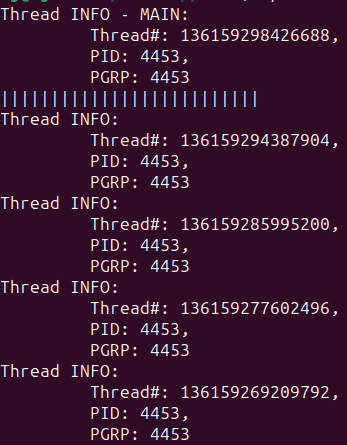


Table 2 Виконання завдання 1: Вивід інформації про потоки



Table 3 Код Завдання 2

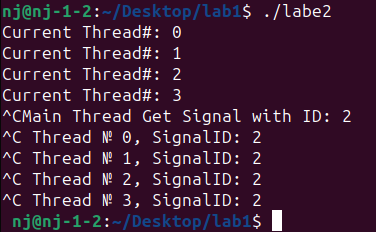


Table 4 Виконання Завдання 2, Отримання сигналу потоками

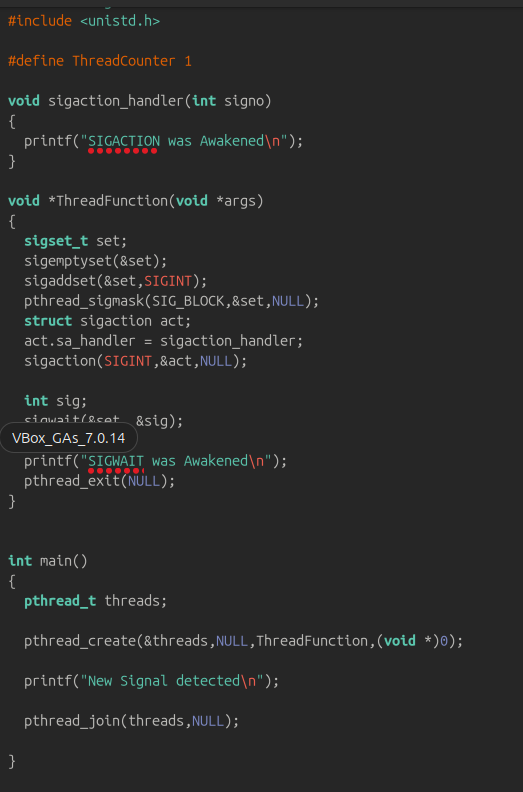


Table 5 Код виконання завдання 3

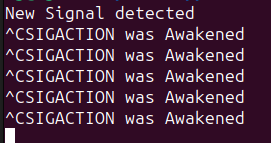


Table 6 Результат виконання завдання 3, сигнал перехвачено директивою SIGACTION

**Контрольні запитання**

1. Чим є сигнали та з якою метою їх використовують в ОС UNIX?

Сигнали – це спосіб взаємодії між процесами та ядром операційної системи в UNIX-подібних системах.

Основне призначення сигналів – це сповіщення про події або помилки. Вони також використовуються для управління процесами, наприклад, для відправлення сигналу, який змушує процес завершити роботу або перезавантажитися.

2. Які види сигналів існують в ОС UNIX та яким чином можна на них реагувати в разі надходження?

В UNIX існують різноманітні сигнали, наприклад:

SIGINT: Виникає при натисканні комбінації клавіш Ctrl+C і зазвичай використовується для відміни виконання програми.

SIGHUP: Викликається при закритті терміналу або розриві з’єднання і може бути використаний, наприклад, для перезапуску програми або переведення її в фоновий режим.

SIGQUIT: Відправляється при натисканні комбінації клавіш Ctrl+\ і зазвичай використовується для завершення програми з виходом, може вивести інформацію про помилку або важливі дані.

SIGKILL: Це безумовний сигнал для завершення процесу, неможливо перехопити або заблокувати.

SIGTERM: Використовується для закриття програми та може бути перехоплений або заблокований. Програма може встановити власний обробник сигналу для здійснення чистого завершення та виконання додаткових дій перед закриттям.

SIGUSR1, SIGUSR2: Призначені для користувача і можуть бути використані для будь-яких цілей, наприклад, для перезавантаження або відладки програми.

Для керування обробкою сигналів у програмах в UNIX існують різні функції як signal, sigaction, sigprocmask, sigwait. Ці функції дозволяють ефективно керувати обробкою сигналів програмах, встановлюючи обробники, блокуючи сигнали, очікуючи їх надходження та надсилаючи їх іншим процесам або потокам.

3. Окресліть основні принципи розроблення оброблювачів сигналів.

При розробці обробників сигналів в UNIX важливо створювати функції для обробки конкретних сигналів, враховуючи їхню ціль використання. Обробники сигналів повинні бути якомога більш простими та швидкими. Вони повинні виконувати мінімальний набір операцій, який потрібний для коректної обробки сигналу, оскільки вони виконуються у контексті основного потоку виконання програми і можуть призвести до небажаних побічних ефектів або навіть блокування програми. Для безпечної обробки необхідно використовувати операції, які гарантують атомарність. Важливо уникати блокування сигналів, які можуть мати критичне значення для програми, і ефективно керувати їхнім блокуванням у багатозадачному середовищі.

4. Як можна реалізувати синхронне оброблення асинхронних сигналів у багатопотоковій програмі?

У багатопотоковій програмі синхронне оброблення асинхронних сигналів можна здійснити, використовуючи функцію sigwait(). Ця функція дозволяє програмі чекати на надходження сигналу і продовжувати виконання лише після його отримання, тим самим реалізуючи синхронну обробку асинхронних подій, які можуть виникнути в будь-якому потоці програми.

Такий підхід дозволяє програмі реагувати на асинхронні події, такі як сигнали, у синхронному режимі, не перериваючи виконання основних операцій програми у інших потоках.