МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації проектування енергетичних процесів і систем

Лабораторна робота №3

з дисципліни «Системи моніторингу стану мережі об'єктів у реальному часі»

**Виконав:**

студент 5-го курсу

ТЕФ, гр. ТР-02мп

Круглий Д.В.

**Перевірив:**

проф. Сліпченко В. Г.

**Київ-2021**

**Тема:** модель призначення однорідних ресурсів.

**Мета:** побудова моделюючого комплексу для моделі однорідних ресурсів.

**Постановка задачі:** потрібно запрограмувати завдання, написавши монітор "Ресурс", який реалізує виділення запитуваної кількості ресурсів і вільний від нескінченного очікування.

Процеси-клієнти створюються за допомогою запуску скрипта client.php. Процес-сервер запускається за допомогою скрипта server.php. Взаємодія між процесами реалізована за допомогою іменованих каналів (FIFO). На скріншотах у верхній секції терміналу запускається процес-сервер, у нижніх процеси-клієнти. Спочатку запускається процес-сервер і чекає запитів на виділення ресурсів. Протягом усієї роботи програми процеси виводять в консоль інформацію про запити до серверу, дії, кількість та статус ресурсів.

**Представлення коду програми**

**Client.php**

<?php

require '../vendor/autoload.php';

require 'define.php';

use Jenner\SimpleFork\Queue\PipeQueue;

$queueForReserveResource = new PipeQueue();

$queueForFreeResorce = new PipeQueue('/tmp/free.pipe');

$pid = getmypid();

$requiredResources = random\_int(1, MAX\_RESOURCES);

$clientQueue = new PipeQueue(getClientPipePath($pid));

$queueForReserveResource->put($pid . ',' . $requiredResources);

echo "Процес $pid запитує $requiredResources ресурсів. Очікування...\n";

while (1) {

    $dataFromQueue = $clientQueue->get();

    if ($dataFromQueue) {

        if ($dataFromQueue == ERROR) {

            echo $dataFromQueue . "\n";

            die();

        } else {

            $useResourcesTime = random\_int(5, 10);

            echo $dataFromQueue . "\n";

            echo "Процес використовує $requiredResources ресурсів. Звільнення ресурсів через $useResourcesTime с.\n";

            //процес деякий час використовує ресурси

            sleep($useResourcesTime);

            //звільнення ресурсів

            echo "Ресурси звільнено\n";

            $queueForFreeResorce->put($requiredResources);

            die();

        }

    }

}

**Server.php**

<?php

require '../vendor/autoload.php';

require 'define.php';

use Jenner\SimpleFork\Queue\PipeQueue;

$queueForReserveResource = new PipeQueue();

$queueForFreeResorce = new PipeQueue('/tmp/free.pipe');

$freeResources = MAX\_RESOURCES;

while (1) {

    //звільнюємо ресурси

    $dataFromQueue = $queueForFreeResorce->get();

    if ($dataFromQueue) {

        $freedResources = $dataFromQueue;

        $freeResources += $freedResources;

        echo "Звільнилось $freedResources ресурсів. Вільних ресурсів: $freeResources\n";

    }

    //надаємо ресурси

    $dataFromQueue = $queueForReserveResource->get();

    if ($dataFromQueue) {

        [$pid, $askedResources] = explode(',', $dataFromQueue);

        $clientQueue = new PipeQueue(getClientPipePath($pid));

        if ($askedResources > MAX\_RESOURCES) {

            echo "Помилка. Процес запитує кількість ресурсів більшу ніж максимальна\n";

            $clientQueue->put(ERROR);

        } elseif ($askedResources > $freeResources) {

            //якщо ресурсів недостатньо - клієнт чекає

            $q = [$dataFromQueue];

            while ($p = $queueForReserveResource->get()) {

                $q[] = $p;

            }

            foreach ($q as $item) {

                $queueForReserveResource->put($item);

            }

        } else {

            //виділяємо ресурси

            echo "Процес $pid отримав $askedResources ресурсів\n";

            $clientQueue->put('Отримано ' . $askedResources . ' ресурсів');

            $freeResources -= $askedResources;

            echo "Залишилось вільних $freeResources ресурсів\n";

        }

    }

};

**Приклад роботи програми**



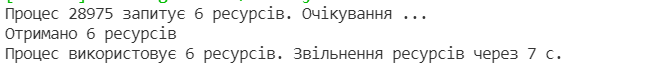
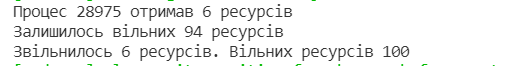


Рис. 1 - Виділення ресурсів для одного процесу. Процес-клієнт запитує виділення 6-ти одиниць ресурсів. Сервер має в наявності всі ресурси(100 одиниць) тому надає ресурси клієнту.



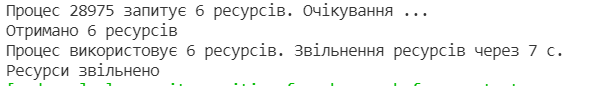


Рис. 2 - Після отримання ресурсів, клієнт певний час( генерується випадково) їх використовує, і потім повертає їх серверу назад.

Розглянемо що відбувається коли сервер не має достатньої кількості ресурсів, необхідної для клієнта.



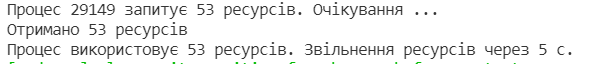
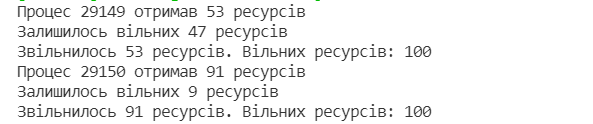
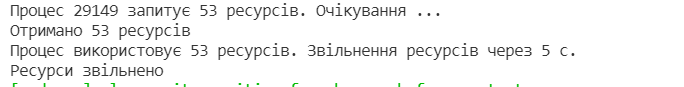




Рис. 3 - Процес 29149 запитує 53 одиниць ресурсів. Сервер надає ресурси процесу. Після цього сервер має вільних 47 ресурсів. Далі процес 29150 запитує 91 ресурс і так як сервер не має такої кількості вільних ресурсів, процес 29150 стає у чергу та чекає коли ресурси звільняться.





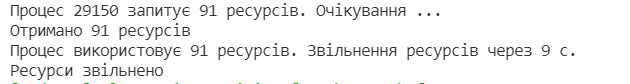


Рис. 4 - Процес 29149 звільнив ресурси і сервер відразу надав їх в розпорядження процесу 29150.

**Висновок:** виконавши дану лабораторну роботу, було запрограмовано модель виділення ресурсів на мові програмування php. Процес-клієнт запитує ресурси у процесу-сервера. Сервер виділяє ресурси процесам, що запросили, у порядку черги.