Теория вычислительных процессов

Задания

Задание 1. Вычисления в отдельном потоке

В приложении с GUI реализовать:

1. Запуск вычислений в основном потоке.
2. Запуск вычислений в отдельном потоке. При этом окно программы должно быть доступно для взаимодействия с пользователем.
3. *Дополнительно:*
    - *Показывать прогресс вычислений*
4. Предусмотреть остановку вычислений по желанию пользователя.

Примеры вычислительных задач:

- Задачи из численных методов: численное интегрирование, решение дифференциальных уравнений, задачи оптимизации; вычисление числа Пи, …
- Компьютерное моделирование, клеточные автоматы.
- Операции с большими массивами: сортировка, нахождение суммы, …
- Обработка изображений, применение фильтров.
- Фракталы, построение множеств Мандельброта или Жулиа.
- Генетический алгоритм.
- Загрузка большого числа файлов из Интернета или создание большого числа запросов.
- Алгоритмы и методы анализа данных и машинного обучения.
- Перебор текста по значению хеш-функции, перебор паролей.

Рекомендации к выполнению

Примеры работы с потоками: github.com/ivtipm/ProcessCalculus/tree/master/examples

В программе из этого задания поток с отдельными вычислениями должен быть запущен в методе – обработчике нажатия на кнопку. Сложность заключается в том,

что нужно после завершения вычислений в отдельном потоке обновить данные в окне программы. Однако обработчики событий окна программы находятся в основном потоке.

Qt

В Qt для этого выгодно использовать механизм сигналов и слотов. Создать отдельный класс – для вычислений в отдельном потоке, вызвать в нём сигнал `finished()`. Соединив (`QObject::connect`) этот сигнал с обработчиком – методом на форме можно обновить данные, когда вычисления завершатся. Пример: github.com/ivtipm/ProcessCalculus/tree/master/examples/example_qthread

JavaFX

Создавая интерфейс пользователя с помощью JavaFX можно обернуть обращение к элементам интерфейса из другого потока в производный от Runnable класс следующим образом:

```
// запуск потока в одном из методов контролёра
в new Thread( () -> {
    // продолжительные вычисления
    Platform.runLater(() -> {
    // обновление Lable в окне программы
    status_label.setText( "Done!" );          });
  }).start();
```

Статический метод `runLater`[1] поставит переданный в него объект типа Runnable в очередь обработки сообщений программы. При этом внутри runLater не должно быть продолжительных операций, так как этот код выполняется в основном потоке приложения.

Вопросы

1. Что такое блокирующий и не блокирующий вызовы?
2. Как в программе организован неблокирующий вызов вычислений?
3. Как в программе организовано обновление данных в окне после завершения работы потока с вычислениями?

Ссылки

- docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/async – C# Асинхронное программирование с использованием ключевых слов async и await

1 https://docs.oracle.com/javase/8/javafx/api/javafx/application/Platform.html#runLater-java.lang.Runnable

- github.com/ivtipm/ProcessCalculus/tree/master/examples/example_qthread – Пример работы с Qthread
- docs.oracle.com/javase/10/docs/api/java/lang/Thread.html – Thread (Java class)

Задание. Сеть Петри

Задана сеть Пери N = <P, T, I, O>.

Для своего варианта:

1. Построить входную и выходную расширенные функции.
2. Построить граф.
3. Построить матрицу инцидентности

```
Вариант 1
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P4}
I(t2)={ P2, P3, P5}
I(t3)={ P4, P5}
O(t1)={ P1, P5}
O(t2)={ P5}
O(t3)={ P3, P4}

Вариант 2
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P5}
I(t2)={ P1, P4, P4}
I(t3)={ P2, P3}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вариант 3
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P3, P5}
I(t2)={ P2, P3, P4}
I(t3)={ P2, P5}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вариант 4
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P4}
I(t2)={ P2, P3, P4}
I(t3)={ P2, P5}
O(t1)={ P2, P5}
O(t2)={ P2}
O(t3)={ P3, P4}

Вариант 5
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P2, P4}
I(t2)={ P2, P3, P4}
I(t3)={ P1, P5}
O(t1)={ P2, P5}
O(t2)={ P5}
O(t3)={ P3, P4}

Вариант 6
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P4}
I(t2)={ P2, P3, P5}
I(t3)={ P4, P5}
O(t1)={ P1, P5}
O(t2)={ P5}
O(t3)={ P3, P4}

Вариант 7
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P5}
I(t2)={ P1, P4, P4}
I(t3)={ P2, P3}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вариант 8
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P3, P5}
I(t2)={ P2, P3, P4}
I(t3)={ P2, P5}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вариант 9
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P4}
I(t2)={ P2, P3, P4}
I(t3)={ P2, P5}
O(t1)={ P2, P5}
O(t2)={ P2}
O(t3)={ P3, P4}
```

Вариант 10
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P2, P4}
I(t2)={ P2, P3, P4}
I(t3)={ P1, P5}
O(t1)={ P2, P5}
O(t2)={ P5}
O(t3)={ P3, P4}

Вариант 11
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P4}
I(t2)={ P2, P3, P5}
I(t3)={ P4, P5}
O>(t1)={ P1, P5}
O>(t2)={ P5}
O>(t3)={ P3, P4}

Вариант 12
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P5}
I(t2)={ P1, P4, P4}
I(t3)={ P2, P3}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вариант 13
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P3, P5}
I(t2)={ P2, P3, P4}
I(t3)={ P2, P5}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вариант 14
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P2, P4}
I(t2)={ P2, P3, P4}
I(t3)={ P1, P5}
O(t1)={ P2, P5}
O(t2)={ P5}
O(t3)={ P3, P4}

Вариант 15
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P4}
I(t2)={ P2, P3, P5}
I(t3)={ P4, P5}
O(t1)={ P1, P5}
O(t2)={ P5}
O(t3)={ P3, P4}

Вариант 16
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P5}
I(t2)={ P1, P4, P4}
I(t3)={ P2, P3}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вариант 17
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P3, P5}
I(t2)={ P2, P3, P4}
I(t3)={ P2, P5}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вариант 18
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P4}
I(t2)={ P2, P3, P4}
I(t3)={ P2, P5}
O(t1)={ P2, P5}
O(t2)={ P2}
O(t3)={ P3, P4}

Вариант 19
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P2, P4}
I(t2)={ P2, P3, P4}
I(t3)={ P1, P5}
O(t1)={ P2, P5}
O(t2)={ P5}
O(t3)={ P3, P4}

Вариант 20
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P4}
I(t2)={ P2, P3, P5}
I(t3)={ P4, P5}
O>(t1)={ P1, P5}
O>(t2)={ P5}
O>(t3)={ P3, P4}

Вариант 21
P={P1, P2, P3, P4, P5}
T={t1, t2, t3}
I(t1)={ P1, P5}
I(t2)={ P1, P4, P4}
I(t3)={ P2, P3}
O(t1)={ P2, P3}
O(t2)={ P2}
O(t3)={ P4, P5}

Вопросы

1. Как определяется сеть Петри?
2. Для чего используются сети Петри?
3. Что такое расширенная входная(выходная) функция?
4. Что такое кратность позиции?
5. Что такое маркировка сети Петри?
6. Как выполняется сеть Петри? Когда переход разрешён? В каком порядке запускаются переходы?
7. Опишите выполнение сети Петри на примере.

Ссылки

- Моделирование параллельных процессов. Сети Петри: курс для системных архитекторов, программистов, системных аналитиков, проектировщиков сложных систем управления / В. Б. Мараховский, Л. Я. Розенблюм, А. В. Яковлев. - Санкт-Петербург : Профессиональная литература, Санкт-Петербург : АйТи-Подготовка, 2014. - 398
- Теория сетей Петри и моделирование систем. (Petri Net Theory and the Modeling of Systems, 1981). Перевод с английского М.В. Горбатовой, В.Л. Торхова, В.Н. Четверикова под редакцией В.А. Горбатова. (Москва: Издательство «Мир». Редакция литературы по новой технике, 1984)
- petri.hp102.ru/pnet.html – изображение графа сети, моделирование.

Задание. Несколько потоков

Создать параллельный алгоритм решения некоторой задачи (см. задание 1), в котором отдельные потоки используют общую память. Организовать синхронизацию потоков с помощью встроенных средств языка программирования (мьютексы). Параметры задачи (число подзадач) и число потоков задаются пользователем.

Оцените время выполнения программы для разного числа потоков. Постройте график. Определите оптимальное число потоков.

Задание. OpenMP

Создать параллельный алгоритм решения некоторой задачи (см. задание 1), в котором отдельные потоки используют общую память. Организовать синхронизацию потоков с помощью встроенных средств языка программирования (мьютексы). Параметры задачи (число подзадач) и число потоков задаются пользователем.

Оцените время выполнения программы для разного числа потоков. Постройте график. Определите оптимальное число потоков.

Задание. MPI

Создать параллельный алгоритм решения некоторой задачи (см. задание 1) на распределённой системе. Использовать MPI для синхронизации потоков.

Дополнительно: продемонстрировать работу алгоритма задействовав несколько компьютеров в сети.

Оцените время выполнения программы для разного числа процессов. Постройте график. Определите оптимальное число процессов.