Параллельное программирование. Потоки и средства их синхронизации.

№ урока: 5 **Курс:** C++ Advanced

Средства обучения: Qt Creator

Обзор, цель и назначение урока

Научить студентов понимать и применять на практике базовые механизмы параллельного программирования из стандартной библиотеки потоков, разобрать примитивы синхронизации, основы параллелизма и конкурентности.

Изучив материал данного занятия, учащийся сможет

- Понимать, что такое параллельное программирование.
- Уметь объяснить разницу между параллелизмом и конкурентностью.
- Понимать, что такое поток, процесс, как происходит взаимодействие между ними.
- Знать основные механизмы в std::thread.
- Применять мьютексы и условные переменные должным образом.

Содержание урока

1. Основные принципы параллельного программирования
2. Использование потоков std::thread
3. Средства синхронизации потоков
4. mutex, recursive_mutex, timed_mutex, recursive_timed_mutex
5. std::lock_guard, unique_lock, condition_variable

Резюме

Для работы со стандартными потоками необходимо в конце использовать либо метод join(), либо метод detach().

Для того, чтобы узнать количество реальных потоков на вычислительной машине, необходимо вызвать метод hardware_concurrency()

Не передавайте указатели и ссылки на защищенные данные за пределы области видимости блокировки никаким способом, будь то возврат из функции,
сохранение в видимой извне памяти или передача в виде аргумента пользовательской функции.

Для избегания взаимоблокировок захватывайте мьютексы в одном и том же порядке.

Иерархия блокировок, грануляция.

recursive_mutex: может войти «сам в себя»

timed_mutex: в отличие от обычного мьютекса, имеет еще два метода: try_lock_for() и try_lock_until()

CyberBionic systematics

CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

recursive_timed_mutex: это комбинация timed_mutex и recursive_mutex

Закрепление материала

- Чем отличается параллелизм от конкурентности?
- Чем отличается поток от процесса?
- Для чего необходим мьютекс? Какую проблему он решает?
- Как передать сообщение от одного потока к другому?
- Что такое deadlock?

Дополнительное задание

Задание
Изучите различные проблемы, которые могут возникать при работе с параллелизмом: livelock, критическая секция, и т.п.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Реализуйте примитивное клиент-серверное приложение, где будет 3 клиента и 1 сервер.
Каждый из них работает в своем потоке, клиенты отсылают серверу сообщения каждые 2 секунды.
Сервер по получению сообщения отправляет ответ, что принял, после этого клиент отключается.

Задание 3
Зайдите на сайт MSDN.
Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

Рекомендуемые ресурсы

https://en.cppreference.com/w/cpp/thread/thread

https://en.cppreference.com/w/cpp/thread/unique_lock

http://www.cplusplus.com/reference/mutex/recursive_mutex/

CyberBionic systematics
CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com