Ch11: Multi Threading

Multi Threading

Threads (std::thread)

In C++, a thread is a basic unit of CPU utilization. It is a way for a program to perform multiple tasks simultaneously or in parallel, improving the efficiency and performance.

```cpp
#include <iostream>
#include <thread>
// Function that performs Task 1
void computeTask1(int n) {
    std::cout << "Computing Task 1: " << n << std::endl;
}
// Function that performs Task 2
void computeTask2(double n) {
    std::cout << "Computing Task 2: " << n << std::endl;
}
int main() {
    std::thread t1{computeTask1, 7};
    std::thread t2{computeTask2, 4.4};

    // Wait for both threads to finish
    t1.join();
    t2.join();

    std::cout << "Both tasks have finished executing." << std::endl;
}
```

Example of creating two threads.

A thread is created using std::thread (defined in thread header) and initialized with name of a function and its input arguments. Each thread starts executing the function immediately after creation.

Note: std::thread is recommended when the functions do not return a value (i.e, void).

Tasks (std::async)

If the functions return a value, std::async (defined in future header) is recommended.

```
#include <iostream>
#include <future>
// Function that performs Task 1
int computeTask1(int n) {
    std::cout << "Computing Task 1: " << n << std::endl;
    return n*n;
}
// Function that performs Task 2
double computeTask2(double n) {
    std::cout << "Computing Task 2: " << n << std::endl;
    return n*n;
}
int main() {
    auto t1 = std::async(std::launch::async, computeTask1, 7);
    auto t2 = std::async(std::launch::async, computeTask2, 4.4);

    // Wait for both threads to finish and get the results
    auto result1 = t1.get();
    auto result2 = t2.get();

    std::cout << "Both tasks have finished executing." << std::endl;
}
```

std::async(policy, function, arguments)

Policy std::launch::async ensures the function runs on a new thread (but std::launch::deferred defers the execution of the function until get() is called.)

auto is std::future<int>.

auto is std::future<double>.

auto is int.

auto is double.