void myFunction() {

// 线程执行的代码

}

int main() {

std::thread myThread(myFunction); // 创建线程对象，并将myFunction作为线程的执行体

// 其他代码

myThread.join(); // 等待线程执行完毕

return 0;

}

等待线程执行完毕 在主线程中，可以使用join()方法等待线程执行完毕。join()会阻塞主线程，直到被调用的线程执行完毕。

例如，上面的示例代码中的myThread.join()会等待myThread线程执行完毕后才会继续执行主线程。

其他线程操作方法 std::thread还提供了其他一些方法，例如detach()方法可以将线程与主线程分离，使得线程可以独立运行。

下面是一个使用std::thread创建多个线程的示例：

cpp

void printHello(int id) {

std::cout << "Hello from thread " << id << std::endl;

}

int main() {

std::vector<std::thread> threads;

for (int i = 0; i < 5; ++i) {

threads.push\_back(std::thread(printHello, i)); // 创建线程并传入参数

}

// 等待所有线程执行完毕

for (auto& thread : threads) {

thread.join();

}

return 0;

}

在上面的示例中，创建了5个线程，每个线程都会调用printHello函数，并传入不同的参数。最后，使用循环等待所有线程执行完毕。

在计算机科学中，线程的调度方式可以使用线程端光比（thread fairness ratio）来衡量。线程端光比是指一个线程被调度执行的时间与它应该被执行的时间之比。具体来说，如果一个线程被执行的时间与它应该被执行的时间相等，那么线程端光比为1；如果一个线程被执行的时间少于它应该被执行的时间，那么线程端光比小于1；如果一个线程被执行的时间超过它应该被执行的时间，那么线程端光比大于1。

线程端光比可以用来评估一个调度算法对线程的公平性。一个高度公平的调度算法应该尽量使线程端光比接近1，以保证每个线程都能得到合理的执行时间。反之，如果线程端光比偏离1很远，可能意味着某些线程被偏好执行，而其他线程则得不到足够的执行时间。

线程端光比在实际应用中很有用，特别是在多线程编程和并发控制方面。它可以帮助开发者了解线程调度的公平性，并进行调优和优化。