# **Report: Thread Pool Implementation – Assignment 2**

**Student Name:** Olesia Mykhailyshyn

**Group:** 1

**Chosen Variant:** 10

**Github Link:** [**https://github.com/VY-Assignments/parallel-assignment-2-olesia-mykhailyshyn.git**](https://github.com/VY-Assignments/parallel-assignment-2-olesia-mykhailyshyn.git)

### **Task**

*Implement a program in which its parts will be synchronized. which includes the implementation of a Thread pool in order not to create threads, then they perform work, are deleted and other new threads are created, and so on in a circle. And here the work was to reduce the load on the central processor using the condition variable and mutex.*

### **System Model**

**Insert a photo or screenshot of the system model here:**

Зображення, що містить текст, ескіз, малюнок, почерк

Автоматично згенерований опис

### **Solution Description**

**Provide a description of the implemented software solution, explaining how the system was designed and developed:***.)*

*The thread pool implementation in this code consists of a General task queue (tasksQueue):*

*A queue to which new tasks are first added in FIFO order.*

*A priority queue (firstPriorityQueue):*

*A second queue (secondQueue):*

*Used exclusively for slow tasks that have been running too long in the priority queue.*

*Two threads (TaskGenerator) create tasks and add them to the tasksQueue.*

*A separate thread that monitors the tasksQueue and passes the tasks to the firstPriorityQueue.*

*First-queue workers Process tasks from the firstPriorityQueue.*

*Second-queue workers Process only "slow tasks" from the secondQueue.*

*Synchronization techniques*

*a. Mutexes (std::mutex)*

*Provide exclusive access to shared resources:*

*b. Condition variables (std::condition\_variable)*

*Used to communicate between threads, allowing them to efficiently wait and send signals*

*c. Atomic variables (std::atomic)*

*Manage global states of threads without using locks:*

*stop: Signals threads to stop processing.*

*immediateStop: Allows all operations to be stopped immediately.*

*paused: Pauses task generation if necessary.*

*activeThreads: Tracks the number of active threads.*

### **Testing**

* **Describe how the testing was conducted:** *(Explain how you tested the thread pool's functionality, including task addition, execution, and safe shutdown. Mention any specific tests you performed to ensure thread safety and the correct handling of synchronization primitives.) It was just debugging and following the consol output.*
* **Performance Evaluation:** *(Perform time-limited testing to monitor and record the number of threads created and the average time a thread spends in the waiting state. Evaluate the performance of your thread pool by determining the average length of each queue and the average task execution time for unlimited queues, or ascertain the maximum and minimum times until a queue is filled and the number of rejected tasks for limited queues*

*Metrics such as wait time, execution time, and number of jobs processed are tracked using variables (totalWaitTimeFirstQueue, totalExecutionTimeFirstQueue, etc.). Metrics are calculated using timestamps. Results include: Average wait and execution time for each queue. Total number of jobs processed in each queue.*

**Insert screenshots of the program during testing:** *(Show the console input/output that verifies the thread pool's behavior under various scenarios, like loading tasks, pausing, and stopping the pool.) Pausing and stopping threads*

*Paused:*

*Task generators stop running if the system temporarily pauses processing.*

*Threads resume when paused changes to false.*

*Stop and immediateStop:*

*A soft stop allows you to complete all tasks that are in the queue.*

*An immediate stop stops all operations immediately.*

*The pause mechanism allows you to temporarily stop task generation without terminating the thread.*

*The code uses the paused flag and the taskQueueCV conditional variable in the TaskGenerator function. Here's how it works:*

*if (paused) {*

*std::cout << "Generator paused by thread " << threadId << std::endl;*

*taskQueueCV.wait(lock, [] { return !paused || immediateStop; });*

*if (immediateStop) {*

*return; // Exit the thread if paused.*

*}*

*}*

*If paused == true, task generation is stopped.*

*The thread waits for the paused flag to change state via taskQueueCV.*

*When paused == false, the thread resumes.*

*The pool is resumed by setting paused = false and notifying all threads via taskQueueCV:*

*paused = false;*

*taskQueueCV.notify\_all(); // Resume all threads.*

*Things to consider:*

*Before resuming work, you need to make sure that the system is ready to accept tasks (for example, not all queues are full).*

*Threads waiting in wait will immediately continue their work.*

*Smooth stop*

*The stop flag is used to terminate the thread pool. This allows you to:*

*Wait for all active threads to complete their work.*

*Make sure that all tasks remaining in the queues have been completed.*

*In the code:*

*stop = true;*

*taskQueueCV.notify\_all(); // Notify threads that the queues will no longer be populated.*

*firstPriorityQueueCV.notify\_all(); // Notify workers.*

*secondQueueCV.notify\_all(); // Notify second queue.*

*Immediate stop*

*To stop work immediately, set immediateStop = true. All threads check this condition and terminate immediately:*

*if (immediateStop) {*

*return; // The thread terminates execution immediately.*

1 task with 50%:

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

5 tasks with 40%:

Зображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

20 tasks with 50%

Зображення, що містить текст, знімок екрана, Шрифт, меню

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, меню, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, меню

Автоматично згенерований опис