Титульный лист

Домашнее задание №5

по дисциплине «Архитектура вычислительных систем»

Тема: «Многопоточность»

Пояснительная записка

Исполнитель: студент 2 курса Якунюшкин Никита Олегович

Группа: БПИ206

ТЕКСТ ЗАДАНИЯ

Вариант 32. Пляшущие человечки. На тайном собрании глав преступного мира города Лондона председатель собрания профессор Мориарти постановил: отныне вся переписка между преступниками должна вестись тайнописью. В качестве стандарта были выбраны «пляшущие человечки», шифр, в котором каждой букве латинского алфавита соответствует хитроумный значок. Реализовать многопоточное приложение, шифрующее исходный текст (в качестве ключа используется кодовая таблица, устанавливающая однозначное соответствие между каждой буквой и каким-нибудь числом). Каждый поток шифрует свои кусочки текста. При решении использовать парадигму портфеля задач.

Структура проекта

В папке “Task5” находятся:

- папка “code” с кодом программы;
- папка “tests” с файлами тестов;
- папка “note” с пояснительной запиской.

ФОРМАТ ВВОДА ПАРАМЕТРОВ ЗАПУСКА

<полный путь к вводному файлу с текстом> <полный путь к выходному файлу> <полный путь к файлу с алфавитом или же “rnd”, если нужно сгенерировать алфавит рандомно> <количество потоков>

МОДЕЛЬ ВЫЧИСЛЕНИЯ

Сначала считывается текст из входного файла. Потом в зависимости от количества потоков и количества символов во входном тексте формируется

количество символов, которое обрабатывает один поток. В случае, если потоков больше, чем символов в тексте, количество потоков автоматически заменяется на количество символов в тексте.

При разработке приложения была использована парадигма параллельного программирования взаимодействующие равные и парадигма портфеля задач.

Взаимодействующие равные – модель, в которой исключен не занимающийся непосредственными вычислениями управляющий поток.

Распределение работ в таком приложении либо фиксировано заранее, либо динамически определяется во время выполнения. Одним из распространенных способов динамического распределения работ является «портфель задач». Портфель задач, как правило, реализуется с помощью разделяемой переменной, доступ к которой в один момент времени имеет только один процесс.

Вычислительная задача делится на конечное число подзадач. Как правило, каждая подзадача должна выполнить однотипные действия над разными данными. Подзадачи нумеруются, и каждому номеру определяется функция, которая однозначно отражает номер задачи на соответствующий ему набор данных.

В случае данного приложения портфелем задач является массив **auto *threads = new std::thread[number_of_threads];** каждой подзадачей является кодирование определенного кусочка текста. Во избежание кодирования одной и той же части текста различными потоками, предусмотрена переменная **int num_of_thread**, увеличивающаяся после того, как поток «взял» подзадачу. Закодированный под определенным номером символ будет помещен в другой массив (с выходными данными) под тем же номером, что позволяет избежать накладок.

ТЕСТИРОВАНИЕ

Программа тестируется на 12-и тестах.

1. Первый тест проверяет ввод пустого файла.
2. Второй тест проверяет слитное/раздельное написание букв, перенос строки, использование спец. символов и цифр.
3. Проверяет задание через файл собственного закодированного алфавита.
4. Четвертый тест разделён на 3 подтеста. Во всех них используется случайный текст на примерно 5к строк в среднем по 85 символов каждая.
    a. Первый подтест работает на одном ядре и выполняет задачу за 38 миллисекунд
    b. Второй подтест работает на двух ядрах и выполняет задачу за 31 миллисекунду
    c. Третий подтест работает на четырех ядрах и выполняет задачу за 20 миллисекунд
5. Пятый тест разделён на 3 подтеста. Во всех них используется случайный текст на примерно 10к строк в среднем по 85 символов каждая.
    a. Первый подтест работает на одном ядре и выполняет задачу за 78 миллисекунд
    b. Второй подтест работает на двух ядрах и выполняет задачу за 56 миллисекунд
    c. Третий подтест работает на четырех ядрах и выполняет задачу за 36 миллисекунд
6. Шестой тест разделён на 3 подтеста. Во всех них используется случайный текст на примерно 20к строк в среднем по 85 символов каждая.
    a. Первый подтест работает на одном ядре и выполняет задачу за 150 миллисекунд
    b. Второй подтест работает на двух ядрах и выполняет задачу за 82 миллисекунды
    c. Третий подтест работает на четырех ядрах и выполняет задачу за 41 миллисекунду

Источники

[1] Парадигмы параллельного программирования
https://pro-prof.com/forums/topic/parallel-programming-paradigms
[2] Парадигмы параллельных приложений
https://studref.com/702392/informatika/paradigmy_parallelnyh_prilozheniy