1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 3**

1. «МЕХАНИЗМЫ МНОГОПОТОЧНОСТИ»
2. по дисциплине «Операционные системы»
3. Выполнил
4. студент гр. 4851003/00002 Вологдин М.В.

<*подпись*>

1. Преподаватель Крундышев В. М.

<*подпись*>

1. Санкт-Петербург
2. 2022
3. **Цель работы**

Изучить принципы разработки многопоточных программ, изучить программный интерфейс операционных систем для организации многопоточности, получить навыки организации взаимодействия потоков в многопоточных программах

1. **Ход работы**

1)Вычисления

Количество разложений находится путём полного перебора всевозможных пар, которые образуют это число, причём повторяющиеся пары не учитываются.

Перед началом параллельной обработки выделяются равные задачи, которые будут выполнять все потоки.

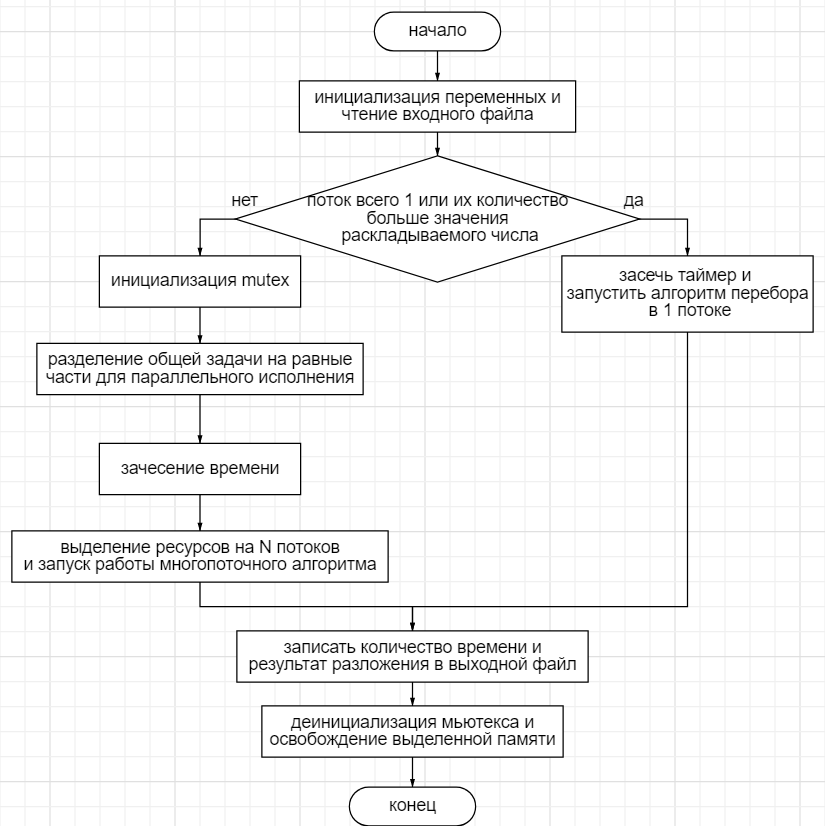


Рисунок 1 – Блок-схема алгоритма программы expr.cpp

Таблица 1 – Функции программы expr.cpp

|  |  |
| --- | --- |
| Функция | Описание |
| Partition(void\*) | Основная функция параллельного вычисления |
| checkNum() | Функция проверки для избежания повторений |
| prePartition() | Функция, разделяющая задачи для параллельного выполнения |
| soloThread(int n) | Функция полного перебора в одном потоке |

Таблица 2 – Временнáя диаграмма взаимодейтвия потоков в моменты синхронизации

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Поток 1 | Г | И | И | Г | Г | Г | И | И | И | Г | Г | Г | И | - |
| Поток 2 | Г | И | Г | И | Г | Г | И | И | И | Г | Г | И | - | - |
| Поток 3 | Г | И | Г | Г | И | Г | И | И | И | И | - | - | - | - |
| Поток 4 | Г | И | Г | Г | Г | И | И | И | И | Г | И | - | - | - |

Таблица 3 – Результаты тестирования программы expr.cpp(по горизонтали – число, по вертикали – количество потоков)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 60 | 100 | 110 |
| 1 | 99 мс | 10,6 с | 34,4 с |
| 2 | 98 мс | 13,3 с | 40,8 с |
| 5 | 121 мс | 14,9 с | 41,9 с |
| 10 | 116 мс | 14,1 с | 41,0 с |
| 50 | 58 мс | 12,9 с | 44,8 с |

2) Быстрая сортировка

Массив разбивается на равные части для каждого рабочего потока, и каждый поток далее сортирует свою часть массива.



Рисунок 2 – Блок-схема алгоритма программы qsort.cpp

Таблица 4 – Функции программы qsort.cpp

|  |  |
| --- | --- |
| Функция | Описание |
| thread() | Основная функция параллельного вычисления |
| partition() | Функция разделения границ для параллельного вычисления |
| qsort() | Основная функция сортировки |

Таблица 5 – Временнáя диаграмма взаимодейтвия потоков в моменты синхронизации

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Поток 1 | Г | И | И | Г | Г | Г | И | И | И | И | И | И | И | - |
| Поток 2 | Г | И | Г | И | Г | Г | И | И | И | И | И | И | - | - |
| Поток 3 | Г | И | Г | Г | И | Г | И | И | И | И | - | - | - | - |
| Поток 4 | Г | И | Г | Г | Г | И | И | И | И | И | И | - | - | - |

Таблица 6 – Результаты тестирования программы qsort.cpp

|  |  |
| --- | --- |
|  | 50000 |
| 1 | 25 мс |
| 2 | 15 мс |
| 4 | 11 мс |
| 12 | 13 мс |
| 25 | 2 мс |

3)Сортировка слиянием

Массив разбивается на равные части для каждого рабочего потока, и каждый поток далее сортирует свою часть массива.



Рисунок 3 – Блок-схема алгоритма программы msort.cpp

Таблица 7 – Функции программы msort.cpp

|  |  |
| --- | --- |
| Функция | Описание |
| thread() | Основная функция параллельного вычисления |
| msort() | Функция сортировки двух подмассивов |
| msort\_ext() | Функция объединения двух отсортированных подмассивов |

Таблица 8 – Временнáя диаграмма взаимодейтвия потоков в моменты синхронизации

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Поток 1 | Г | И | И | Г | Г | Г | И | И | И | И | И | И | И | - |
| Поток 2 | Г | И | Г | И | Г | Г | И | И | И | И | И | И | - | - |
| Поток 3 | Г | И | Г | Г | И | Г | И | И | И | И | - | - | - | - |
| Поток 4 | Г | И | Г | Г | Г | И | И | И | И | И | И | - | - | - |

Таблица 9 – Результаты тестирования программы msort.cpp

|  |  |
| --- | --- |
|  | 50000 |
| 1 | 38 мс |
| 2 | 92 мс |
| 4 | 117 мс |
| 12 | 184 мс |
| 25 | 182 мс |

4) Философы

Программе подаётся на вход два числа – общее время выполнения и минимальное время ожидания. Далее создаётся необходимое количество потоков и в соответствии с алгоритмом философы меняют своё состояние через определённое время. Алгоритм состоит в том, что программа даёт кушать двум философам с разницей в номерах равной 3 по модулю 5, так как всего их 5. Таким образом последовательность перехода философов к еде будет: 1, 4, 2, 5, 3, …. Переходят к еде философы попарно, так как из пяти вилок максимум могут быть задействованы только 4.

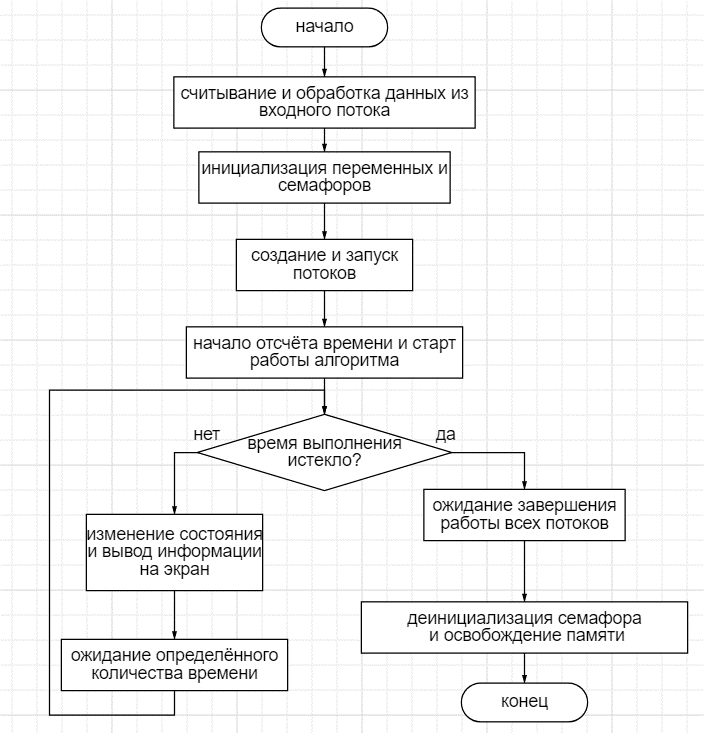


Рисунок 4 – Блок-схема алгоритма программы phil.cpp

Таблица 10 – Функции программы phil.cpp

|  |  |
| --- | --- |
| Функция | Описание |
| srvThreadExecution() | Функция исполнения слуги |
| philThreadExecution() | Функция исполнения философа |

Таблица 11 – Временнáя диаграмма взаимодейтвия потоков в моменты синхронизации

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Философ 1 | Г | Г | Г | Г | И | И | И | Г | Г | Г | Г | Г | Г | И | И | И |
| Философ 2 | Г | И | И | И | Г | Г | Г | Г | Г | Г | И | И | И | Г | Г | Г |
| Философ 3 | Г | Г | Г | Г | Г | Г | Г | И | И | И | Г | Г | Г | И | И | И |
| Философ 4 | Г | Г | Г | Г | И | И | И | Г | Г | Г | И | И | И | Г | Г | Г |
| Философ 5 | Г | И | И | И | Г | Г | Г | И | И | И | Г | Г | Г | Г | Г | Г |

Таблица 12 – Сравнительная таблица количества квантов поедания от общего времени выполнения и минимального времени поедания

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TOTAL=200 | TOTAL=400 | TOTAL=1000 | TOTAL=5000 | TOTAL=10000 |
| Phil=20 | 1-й = 3  2-й = 4  3-й = 3  4-й = 4  5-й = 4 | 1-й = 7  2-й = 8  3-й = 7  4-й = 8  5-й = 8 | 1-й = 19  2-й = 20  3-й = 19  4-й = 20  5-й = 20 | 1-й = 99  2-й = 100  3-й = 99  4-й = 100  5-й = 100 | 1-й = 199  2-й = 200  3-й = 199  4-й = 200  5-й = 200 |
| Phil=100 | 1-й = 0  2-й = 1  3-й = 0  4-й = 0  5-й = 1 | 1-й = 1  2-й = 1  3-й = 1  4-й = 1  5-й = 2 | 1-й = 3  2-й = 4  3-й = 3  4-й = 4  5-й = 4 | 1-й = 19  2-й = 20  3-й = 19  4-й = 20  5-й = 20 | 1-й = 39  2-й = 40  3-й = 39  4-й = 40  5-й = 40 |
| Phil=500 | 1-й = 0  2-й = 1  3-й = 0  4-й = 0  5-й = 1 | 1-й = 0  2-й = 1  3-й = 0  4-й = 0  5-й = 1 | 1-й = 0  2-й = 1  3-й = 0  4-й = 0  5-й = 1 | 1-й = 3  2-й = 4  3-й = 3  4-й = 4  5-й = 4 | 1-й = 7  2-й = 8  3-й = 7  4-й = 8  5-й = 8 |

5)Шахматы

Задача расстановки фигур решена рекурсивным перебором возможных вариантов. Каждый поток получает задачу перебора с определенной расстановкой фигур. Фигура – магараджа.

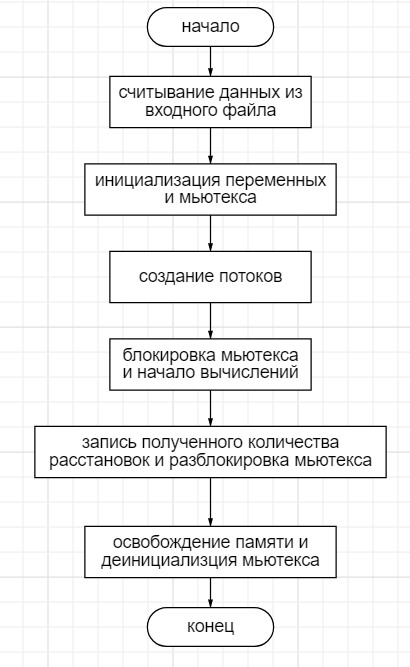


Рисунок 5 – Блок-схема алгоритма программы chess.cpp

Таблица 13 – Функции программы chess.cpp

|  |  |
| --- | --- |
| Функция | Описание |
| thread\_entry() | Основная поточная функция |
| SetMaharajaOnBoard() | Функция установки фигуры на поле |
| position\_check() | Функция, проверяющая свободна ли данная клетка для постановки фигуры |
| time() | Функция для подсчета времени |

Таблица 14 – Временнáя диаграмма взаимодейтвия потоков в моменты синхронизации

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Поток 1 | Г | И | И | Г | Г | Г | И | И | И | И | И | И | И | - |
| Поток 2 | Г | И | Г | И | Г | Г | И | И | И | И | И | И | - | - |
| Поток 3 | Г | И | Г | Г | И | Г | И | И | И | И | - | - | - | - |
| Поток 4 | Г | И | Г | Г | Г | И | И | И | И | И | И | - | - | - |

Таблица 15 – Результаты тестирование программы chess.cpp

|  |  |  |  |
| --- | --- | --- | --- |
|  | 8 | 10 | 11 |
| 1 | 12 мс | 628 мс | 6,3 с |
| 2 | 23 мс | 1429 мс | 12,5 с |
| 5 | 43 мс | 1728 мс | 30,6 с |
| 10 | 47 мс | 1717 мс | 15,5 с |
| 50 | 60 мс | 1986 мс | 16,4 с |

1. **Выводы**

В ходе лабораторной работы были получены знания об устройствемногопоточности. Также были получены навыки по работе с их синхронизацией. Исходя из проведённых тестов, можно сказать, что важно правильно выбирать количество потоков, не всегда больше – лучше.