**Лабораторные работы по курсу “Распределенные операционные системы и языки программирования”**

Лабораторные работы №1-4: выполнить задания с использованием следующих технологий:

* OS Inferno: Limbo и/или Inferno Shell
* Язык Go
* Библиотека OpenMPI (C, C++ или Python)
* .Net C# (F#, VB.Net) и облачные технологии Microsoft Azure.

В зависимости от используемой технологии необходимо использовать потоки (OS Inferno, Go) или процессы (OpenMPI, C#).

Задания выполняются в бригадах. Максимальный размер бригады указан в задании.

~~1~~) Дано два входных файла объемом в 1000 строк: In1.txt, In2.txt (создаются исполнителем из случайных данных). Необходим организовать два потока (нити 1 и 2), каждый из которых читает из файла In$номерпотока.txt и пишет в общий файл Out.txt. Потоки должны выполняться параллельно.

Размер бригады: 1 человек(Галимуллин)

2) Дан входной файл In.txt объемом в 2000 строк (создаются исполнителем из случайных данных). Необходим организовать два потока (нити 1 и 2), каждый из которых будет читать из файла In.txt и записывать в файл Out$номерпотока.txt каждую четную (нить 2) и каждую нечетную (нить 1) строку файла Out.txt. Потоки должны разделять один(!, иначе задание будет считаться невыполненным) файловый дескриптор(ресурс) для файла Out.txt и выполняться параллельно.

Размер бригады: 1 человек

~~3)~~ Реализовать параллельный алгоритм решения задачи коммивояжера.

Размер бригады: 1-2 человека (Райцин, Белов)

~~4)~~ Реализовать параллельный алгоритм решения задачи о клике (или задачи о независимом множестве).

Размер бригады: 1-2 человека (Рябиков)

~~5)~~ Разработайте программу с двумя процессами (**отдельные исполняемые файлы**) (процесс 1 и 2) обменивающимися сообщениями (1: Hello! 2: Hi! 1: How are you? 2: Fine, thanks, and you? 1: Fine. See you! 2: Bye). Процессы должны выполняться параллельно.

Размер бригады: 1 человек (Васильев)

6) Разработайте программу с N потоками (задается аргументом командной строки), где потоки 1 и N обмениваются сообщениями (1: Hello! 2: Hi! 1: How are you? 2: Fine, thanks, and you? 1: Fine. See you! 2: Bye). Все потоки соединены по кольцу (1->2->3->...->N->1), пересылать сообщения в противоположном стрелке направлении запрещено. Потоки должны выполняться параллельно.

Размер бригады: 1 человек

~~7)~~ Разработайте программу распределенного поиска всех “счастливых билетиков” (всех комбинаций 6 цифр, сумма первых трех равна сумме вторых трех, например 123015, 781952 и так далее). Программа должна иметь конфигурируемый параметр числа одновременно выполняющихся процессов.

Размер бригады: 1-2 человека (Лапин, Лозовик)

~~8)~~ Реализуйте параллельную версию алгоритма решения задачи о восьми ферзях: «Расставить на стандартной 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого».

Размер бригады: 1-2 человека (Раздьяконов, Смирнова)

9) Реализовать параллельную версию алгоритма перемножения двух разреженных матриц. Результат оформить в виде сетевого сервиса для одной из архитектур:

1. Клиент-сервер поверх UDP.
2. Клиент-сервер поверх TCP.
3. XMLRPC-сервис.
4. SOAP-сервис.

Каждая реализация должна иметь синхронную и асинхронную версию взаимодействия.

Размер бригады: 2-4 человека

10) Реализовать параллельную версию алгоритма Флойда — Уоршелла для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа. Результат оформить в виде сетевого сервиса для одной из архитектур:

1. Клиент-сервер поверх UDP.
2. Клиент-сервер поверх TCP.
3. ~~XMLRPC-сервис.~~ (Карвсавов, Фертиков)
4. SOAP-сервис.

Каждая реализация должна иметь синхронную и асинхронную версию взаимодействия.

Размер бригады: 2-4 человека

11) Реализовать параллельную версию алгоритма быстрой сортировки. Результат оформить в виде сетевого сервиса для одной из архитектур:

1. Клиент-сервер поверх UDP.
2. Клиент-сервер поверх TCP.
3. XMLRPC-сервис.
4. SOAP-сервис.

Каждая реализация должна иметь синхронную и асинхронную версию взаимодействия.

Размер бригады: 2-4 человека

~~12)~~ Разработать модель вокзала с двумя путями из разных направлений, но у которого есть только один путь, обслуживающий область погрузки. Если два поезда попытаются подойти к вокзалу в одно и то же время, они столкнуться. На каждом из входных путей (A, B) расположено по четыре (A1-4, B1-4, после A4 и B4 следует станция) семафора, которые могут быть в двух состояниях: поезд может идти (синий) и поезд не может идти (красный). Каждый поезд проезжает расстояние от семафора до семафора или станции за 1 секунду. Время погрузки занимает 3 секунды. Изначально, мы имеем два поезда перед семафорами B1 и B3 и поезд перед семафором A3. Не дайте поездам столкнуться. Программа должна иметь минимальный графический интерфейс, визуализирующий происходящие события.

Размер бригады: 2-3 человека (Алдонин, Любезников, Тихонова)

13) Разработать приложения для синхронизации работы внешних клиентов с хранилищем данных (хранилище – хэш-таблица). Условия:

1. Несколько клиентов **могут** одновременно читать из хранилища.
2. Запись в хранилище может одновременно производить **только один** клиент.
3. Необходимо реализовать механизм транзакций для возможности “отката” изменений в случае ошибки при операции записи.

Результат оформить в виде сетевого сервиса для одной из архитектур:

1. Клиент-сервер поверх UDP.
2. XMLRPC-сервис.
3. SOAP-сервис.
4. Сервис на основе Web-сокетов (Web-sockets).

Размер бригады: 2-3 человека

14) Реализовать параллельную версию алгоритма построения АВЛ-дерева по большому объему несортированных данных (словари >1Mb). Реализовать сервис (с использованием любой архитектуры) для быстрого поиска по построенному дереву.

Размер бригады: 2-3 человека