ВАРИАНТ 1.
Дан файл .txt с текстовой информацией. Составить программу, в которой в данном файле заменить все слова написанные с маленькой буквы на такие же слова, записанные с большой буквы. Посчитать количество слов в каждой строке. Параллелить по СТРОКАМ текстового файла (то есть: если в текстовом файле 10 строк, то сделать 10 потов/процессов; если в текстовом файле 5 строк, то сделать 5 потов/процессов).
Составить три программы:

первую: используя потоки
вторую:используя процессоры
третья: производит последовательные вычисления.

Провести сравнительную характеристику метрик (ускорение и эффективность) для трех программ. Результат сравнения трех программ изобразить графически.

ВАРИАНТ 2.
Написать программу нахождения массива K последовательных значений функции y[i]=sin(2*PI*i/N) (где i=0, 1, 2...K-1) с использованием ряда Тейлора. Пользователь задаёт значения K, N и количество n членов ряда Тейлора. Для расчета каждого члена ряда Тейлора запускается отдельный поток. Каждый поток выводит на экран свой pid и рассчитанное значение ряда. Головной процесс суммирует все члены ряда Тейлора, и полученное значение y[i] записывает в файл.

ВАРИАНТ 3. Написать программу синхронизации двух каталогов, например, Dir1 и Dir2. Пользователь задаёт имена Dir1 и Dir2. В результате работы программы файлы, имеющиеся в Dir1, но отсутствующие в Dir2, должны скопироваться в Dir2 вместе с правами доступа. Процедуры копирования должны запускаться в отдельном процессе для каждого копируемого файла. Каждый процесс выводит на экран свой pid, имя копируемого файла и число скопированных байт. Число одновременно работающих процессов не должно превышать N (вводится пользователем).

ВАРИАНТ 4.

ВАРИАНТ 5.
Создать дерево процессов по индивидуальному заданию:
Создать дерево процессов по индивидуальному заданию: распараллелить (параллелить по столбцам матрицы A) процесс умножение матрицы A произвольного размера на матрицу B. Умножение матриц провести классическим способом.Размер матрицы A и B вводит пользователь с клавиатуры. ОБЯЗАТЕЛЬНА проверка на корректность данных, введенных с клавиатуры.
Каждый процесс постоянно, через время t, выводит на экран следующую информацию: номер процесса/потока, pid, ppid текущее время (мсек). Время t=(номер процесса/потока по дереву)*200 (мсек).

ВАРИАНТ 6.
Составить программу для выполнения параллельного варианта метода Гаусса при ВЕРТИКАЛЬНОМ РАЗБИЕНИИ матрицы по столбцам. Постройте теоретические оценки времени работы этого алгоритма с учетом параметров используемой вычислительной системы. Проведите вычислительные эксперименты. Сравните результаты выполненных экспериментов с ранее полученными теоретическими оценками. Выполните анализ эффективности параллельных вычислений в отдельности для прямого и обратного этапов метода Гаусса. Оцените, на каком этапе происходит большее снижение показателей.

ВАРИАНТ 7. Написать программу поиска одинаковых по их содержимому файлов в двух каталогов, например, Dir1 и Dir2. Пользователь задаёт имена Dir1 и Dir2. В результате работы программы файлы, имеющиеся в Dir1, сравниваются с файлами в Dir2 по их содержимому. Процедуры сравнения должны запускаться в отдельном процессе для каждой пары сравниваемых файлов. Каждый процесс выводит на экран свой pid, имя файла, общее число просмотренных байт и результаты сравнения. Число одновременно работающих процессов не должно превышать N (вводится пользователем).

ВАРИАНТ 8.Написать программу поиска заданной пользователем комбинации из m байт (m < 255) во всех файлах текущего каталога. Пользователь задаёт имя каталога. Главный процесс открывает каталог и запускает для каждого файла каталога отдельный процесс поиска заданной комбинации из m байт. Каждый процесс выводит на экран свой pid, имя файла, общее число просмотренных байт и результаты поиска. Число одновременно работающих процессов не должно превышать N (вводится пользователем).

ВАРИАНТ 9. Написать программу, создающую два дочерних процесса с использованием двух вызовов. Родительский и два дочерних процесса должны выводить на экран свой pid и pid родительского процесса и текущее время в формате: часы : минуты : секунды : миллисекунды. Используя вызов system(), выполнить команду ps -x в родительском процессе. Найти свои процессы в списке запущенных процессов.

ВАРИАНТ 10.

Создать дерево процессов по индивидуальному заданию:

Создать дерево процессов по индивидуальному заданию: распараллелить (параллелить по столбцам матрицы A) процесс умножение матрицы A произвольного размера на матрицу B. Умножение матриц провести по алгоритму ВИНОГРАДА .Размер матрицы A и B вводит пользователь с клавиатуры. ОБЯЗАТЕЛЬНА проверка на корректность данных, введенных с клавиатуры.

Каждый процесс постоянно, через время t, выводит на экран следующую информацию: номер процесса/потока, pid, ppid текущее время (мсек).

Сравнить производительность этого алгоритма и алгоритма с использованием встроенных функций для больших матриц. Большая матрица- это матрица, размер которой, больше 1000x1000.

Алгоритм Винограда

Одним из алгоритмов умножения матриц является алгоритм Винограда (Shmuel Winograd). Он даёт небольшой выигрыш в скорости вычисления произведения, но может быть эффективно распараллелен. Основная идея алгоритма весьма проста и заключается в преобразовании суммы $\sum_{j=1}^{m} a_i^j b_j^l$ следующим образом:

$$c_i^l = \underbrace{\sum_{j=1}^{m/2}(a_i^{2j-1} + b_{2j}^l)(a_i^{2j} + b_{2j-1}^l)}_{(1)} - \underbrace{\sum_{j=1}^{m/2} a_i^{2j-1} a_i^{2j}}_{(2)} - \underbrace{\sum_{j=1}^{m/2} b_{2j-1}^l b_{2j}^l}_{(3)}.$$

ВАРИАНТ 11. Разработать программу «интерпретатор команд», которая воспринимает команды, вводимые с клавиатуры, (например, ls -l /bin/bash) и осуществляет их корректное выполнение. Для этого каждая вводимая команда должна

выполняться в отдельном процессе с использованием вызова exec(). Предусмотреть контроль ошибок.

ВАРИАНТ 12. Реализовать параллельный алгоритм сложения двух векторов. Составить три программы:

первую: используя потоки
вторую:используя процессоры
третья: производит последовательные вычисления.

Провести сравнительную характеристику метрик (ускорение и эффективность) для трех программ. Результат сравнения трех программ изобразить графически.

ВАРИАНТ 13. Реализовать параллельный алгоритм нахождения скалярного произведения двух векторов. Составить три программы:

первую: используя потоки
вторую:используя процессоры
третья: производит последовательные вычисления.

Провести сравнительную характеристику метрик (ускорение и эффективность) для трех программ. Результат сравнения трех программ изобразить графически.

ВАРИАНТ 14. Создать дерево процессов по индивидуальному заданию: распараллелить (параллелить по строкам матрицы A) процесс решения системы линейных уравнений методом Гаусса.. Размер матрицы A вводится пользователь с клавиатуры. ОБЯЗАТЕЛЬНА проверка на корректность данных, введенных с клавиатуры.
Каждый процесс постоянно, через время t, выводит на экран следующую информацию: номер процесса/потока, pid, ppid текущее время (мсек). Время t=(номер процесса/потока по дереву)*100 (мсек)

ВАРИАНТ 15. Написать программу нахождения массива K последовательных значений функции y[i]=cos(2*PI*i/N)+sin(2*PI*i/N) (где i=0, 1, 2...K-1) с использованием ряда Тейлора. Пользователь задаёт значения K, N и количество n членов ряда Тейлора. Для расчета каждого члена ряда Тейлора запускается отдельный поток. Каждый поток выводит на экран свой pid и рассчитанное значение ряда. Головной процесс суммирует все члены ряда Тейлора, и полученное значение y[i] записывает в файл.

ВАРИАНТ 16. Написать программу, создающую два дочерних процесса с использованием двух вызовов. Родительский и два дочерних процесса должны выводить на экран свой pid и pid родительского процесса и текущее время в формате: часы : минуты : секунды : миллисекунды. Используя вызов system(), выполнить команду ps -x в родительском процессе. Найти свои процессы в списке запущенных процессов.

ВАРИАНТ 17. Написать программу поиска одинаковых по их содержимому файлов в двух каталогов, например, Dir1 и Dir2. Пользователь задаёт имена Dir1 и Dir2. В результате работы программы файлы, имеющиеся в Dir1, сравниваются с файлами в Dir2 по их содержимому. Процедуры сравнения должны запускаться в отдельном процессе для каждой пары сравниваемых файлов. Каждый процесс выводит на экран свой pid, имя файла, общее число просмотренных байт и результаты сравнения. Число одновременно работающих процессов не должно превышать N (вводится пользователем).

ВАРИАНТ 18.Написать программу поиска заданной пользователем комбинации из m байт (m < 255) во всех файлах текущего каталога. Пользователь задаёт имя каталога. Главный процесс открывает каталог и запускает для каждого файла каталога отдельный процесс поиска заданной комбинации из m байт. Каждый процесс выводит на экран свой pid, имя файла, общее число просмотренных байт и результаты поиска. Число одновременно работающих процессов не должно превышать N (вводится пользователем).

ВАРИАНТ 19. Разработать программу «интерпретатор команд», которая воспринимает команды, вводимые с клавиатуры, (например, ls -l /bin/bash) и осуществляет их корректное выполнение. Для этого каждая вводимая команда должна выполняться в отдельном процессе с использованием вызова exec(). Предусмотреть контроль ошибок.

ВАРИАНТ 18. Создать дерево процессов по индивидуальному заданию: распараллелить (параллелить по строкам матрицы A) процесс умножение матрицы A произвольного размера на вектор-столбец B. Размер матрицы вводит пользователь с клавиатуры. ОБЯЗАТЕЛЬНа проверка на корректность введенных данных.

Каждый процесс постоянно, через время t, выводит на экран следующую информацию: номер процесса/потока, pid, ppid текущее время (мсек). Время t=(номер процесса/потока по дереву)*200 (мсек).

ВАРИАНТ 20. Разработайте две параллельных программ, которые для диапазона целых чисел [1; N], используя

- первая программа: M потоков
- вторая программа: P процессов

вычисляет суммы для каждого из M отрезков, на которые делится исходный диапазон.

Например, для N=20 и M=5 выводимые строки должны иметь вид
Поток 0 Числа 1 2 3 4 сумма 10
Поток 1 Числа 5 6 7 8 сумма 26
Поток 2 Числа 9 10 11 12 сумма 42
Поток 3 Числа 13 14 15 16 сумма 58
Поток 4 Числа 17 18 19 20 сумма 74
Строки выводятся в случайной последовательности