**Отчёт по дисциплине «Операционные Системы»**

**Лабораторная работа №4**

Выполнил: *Лилик Глеб Егорович студент группы 8И32*

ЗАДАНИЯ:

1. Разработать программу, которая запускает параллельно все процессы.

2. Каждый процесс при запуске должен получать порядковый номер внутри программного кода.

3. Внутри каждого процесса необходимо установить задержку выполнения. Время задержки выполнения процесса является случайно величиной и для каждого процесса определяется отдельно.

4. Каждый процесс в ходе выполнения должен выводить свой номер и результаты процесса в соответствии с заданием.

5. Рассмотреть выполнение представленной программы и определить количество процессов запускаемых данной программой.

6. Выполнить прогон программы несколько раз и определить последовательность выполнения параллельных процессов для каждого прогона. Сделать выводы.

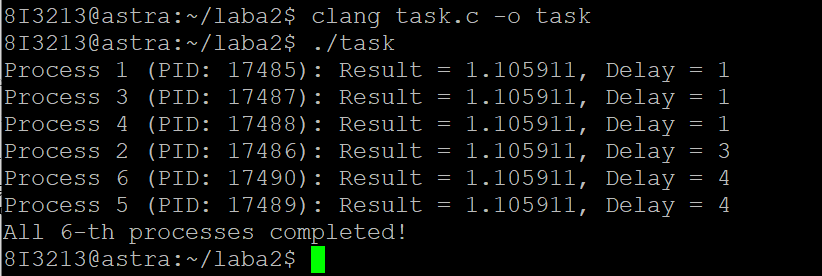
7. Провести анализ и определить результат работы программы при отсутствии вызова функции wait(&st).

Для компиляции программы воспользуйтесь компилятором clang.

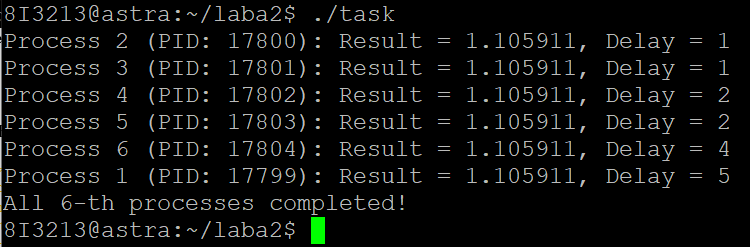
ВАРИАНТ ЗАДАНИЯ:

17-ый Вариант: (ab + c)/(d + e/f) + g/(h + km)

ВЫПОЛНЕНИЕ:

****

**Выполнение компиляции используя clang и первый запуск программы**

****

**Второй запуск программы**

ПРОГРАММНЫЙ КОД:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <time.h>

double get\_ans(double a, double b, double c, double d, double e, double f, double g, double h, double k, double m){

return (a \* b + c) / (d + e / f) + g / (h + k \* m);

}

int main(){

int p\_count = 6; //переменная управления количеством дочерних процессов

pid\_t pid; //Тип данных pid\_t хранит в себе данные о процессе или о группе процессов

for(int i = 0; i < p\_count; i++){

pid = fork();

if(pid < 0){

perror("Failed");

return 1;

} else if (pid == 0){

int p\_num = i + 1; //Задание номера дочернего процесса

srand(time(NULL) ^ (getpid() << 16)); //XOR и сдвиг - частый выбор основы для инициализации генератора случайных чисел

int p\_delay = rand() % 5 + 1; //Установка времени ожидания для процесса от 1 до 5 секунд

double a = 1.0, b = 2.0, c = 3.0, d = 4.0, e = 5.0, f = 6.0, g = 7.0, h = 8.0, k = 9.0, m = 10.0;

//При данный значениях переменных от a до m результат составляет 1,10591133...87..., данный результат был посчитан на калькуляторе заранее для упрощения отладки

double res = get\_ans(a, b, c, d, e, f, g, h, k, m);

sleep(p\_delay); //Ожидание процесса определённого времени

printf("Process %d (PID: %d): Result = %lf, Delay = %d\n", p\_num, getpid(), res, p\_delay);

exit(0); // Обязательный элемент процесса, без него процессы не завершаться никогда

}

}

while(wait(NULL) > 0) {} //Конструкция для бесконечного ожидания завершения ВСЕХ дочерних процессов

printf("All %d-th processes completed!", p\_count);

return 0;

}

ВЫВОД**:**

По результатам данной лабораторной работы, мною были изучены и приобретены навыки компиляции программ, написанных на C++ используя clang. Процессы создаются в одном кластере процессов, т.е. их PID очень близки друг к другу. Кроме прочего, если убрать конструкцию wait для синхронизации дочерних процессов с родительскими, родительский процесс завершится раньше, а после уже завершатся дочерние.