ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ Р.Е. АЛЕКСЕЕВА»

Институт радиоэлектроники и информационных технологий
Кафедра «Прикладная математика»

Отчёт по лабораторной работе курса:
«Распределённые вычислительные системы»

Лабораторная работа №1
“Создание классов при помощи fork() на ОС Linux”

Выполнил студент группы 18-ПМ:
Винокуров М.С.
Проверил:
Лобовиков П.В.

НИЖНИЙ НОВГОРОД
2022г.

Оглавление

Введение.

Цель работы.

Требуется создать указанное количество файлов с векторами, параметры которых будут переданы пользователем. После создания всех файлов с векторами, программа должна создать новый процесс для каждого, после обработки которого будет создан новый файл в который будет положена сумма векторов, созданных ранее. Для определения «до» и «после», нам нужно так же задать параметры «префикса» до работы и после выполнения.

Задание.

Создать файл с векторами с заданным префиксом.
Сложить вложенные данные уникальными процессами.
Положить данные в новый файл с указанным префиксом.

Данные на вход:

Аргументы входа	Описание	Пример ввода
-i	Префикс входных данных	-i input
-o	Префикс выходных данных	-o output
-p	Количество заданных процессов	-p 16
-m	Количество векторов в процессе	-m 32
-t	Размерность вектора	-p 3

При использовании программы, будем использовать следующие данные:
-i input -o output -p 16 -m 32 -t 3

Алгоритм программы.

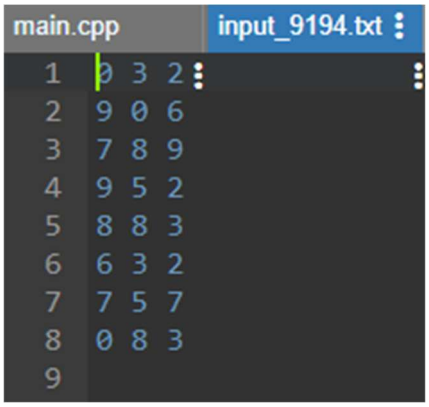
Создание файлов и векторов.

Для начала, по количеству заданных процессов, создаем файлы с положенными в них векторами. По начальным данным векторов будет 32, и их размерность будет равняться трём.

Код программы:

```
string createInFile(string prefixIn, int vectorCount, int dimension)
{
    srand(time(NULL) * getpid());
    string filename = prefixIn + "_" + to_string(getpid()) + ".txt";
    ofstream file(filename);
    if(!file.is_open())
    {
        cout << "Create file error!" << endl;
        return "";
    }
    for(int j = 0; j < vectorCount; j++)
    {
        for(int k = 0; k < dimension; k++)
        {
            file << rand() % 10 << " ";
        }
        file << endl;
    }
    file.close();
    return filename;
}
```

На выходе получаем определенное кол-во файлов, в нашем случае 16, вида:



Запись файлов и сумма векторов.

Передаем используемые в программе данные, а так же сгенерированные ранее вектора на функцию записи в файл и предварительно суммируем по математическим правилам вектора.

Код программы:

```
void createOutFile(string prefixOut, int vectorCount, int dimension, string
fileNameIn)
{
    ifstream fileIn(fileNameIn);
    if(!fileIn.is_open())
    {
        cout << "Create file error!" << endl;
        return;
    }
    vector<int> result(dimension, 0);
    int term;
    for(int i = 0; i < vectorCount; i++)
    {
        for(int j = 0; j < dimension; j++)
        {
            fileIn >> term;
            result[j] += term;
        }
    }
    fileIn.close();

    string filename = prefixOut + "_" + to_string(getpid()) + ".txt";
    ofstream fileOut(filename);
    if(!fileOut.is_open())
    {
        cout << "Create file error!" << endl;
        return;
    }
    for(int i = 0; i < dimension; i++)
        fileOut << result[i] << " ";

    fileOut.close();
}
```

На выход получаем файл:

Создание параллельных процессов.

Инициализируем и передадим параметры создания файлов при помощи fork().

Код блока программы:

```
pid_t processes[fileCount];
    pid_t mainProcess = getpid();
    int i = 0;
    while(i < fileCount)
    {
        if(getpid() == mainProcess)
            processes[i++] = fork();
        else
            break;
    }
    if(getpid() != mainProcess)
    {
        string fileNameIn = createInFile(prefixIn, vectorCount, dimension);
        createOutFile(prefixOut, vectorCount, dimension, fileNameIn);
    }

    int status;
    if(mainProcess == getpid())
    {
        for(int j = 0; j < fileCount;)
        {
            waitpid(processes[j], &status, 0);
            if(WIFEXITED(status))
                j++;
        }
    }
```

Заключение.

Вывод.

В ходе решения поставленной задачи, была создана программа, способная сгенерировать по заданным параметрам командной строки определенное количество файлов с векторами, и последующим их суммированием. Все действия создавались отдельными процессами, тем самым, удалось сократить время исполнения, относительно последовательного решения задачи.

Исходный код.

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <string>
#include <fstream>
#include <time.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

using namespace std;

bool cmdOptionsExists(char** begin, char** end, const vector<string>& options)
{
    for(const string& option : options)
        if(find(begin, end, option) == end)
            return false;
    return true;
}

string cmdAfterKey(char** begin, char** end, const std::string& option)
{
    if(find(begin, end, option) + 1 != end && *(find(begin, end, option) + 1)[0]
!= '-')
        return *(find(begin, end, option) + 1);
    return "";
}

string createInFile(string prefixIn, int vectorCount, int dimension)
{
    srand(time(NULL) * getpid());
    string filename = prefixIn + "_" + to_string(getpid()) + ".txt";
    ofstream file(filename);
```

```
    if(!file.is_open())
    {
        cout << "Create file error!" << endl;
        return "";
    }
    for(int j = 0; j < vectorCount; j++)
    {
        for(int k = 0; k < dimension; k++)
        {
            file << rand() % 10 << " ";
        }
        file << endl;
    }
    file.close();
    return filename;
}

void createOutFile(string prefixOut, int vectorCount, int dimension, string
fileNameIn)
{
    ifstream fileIn(fileNameIn);
    if(!fileIn.is_open())
    {
        cout << "Create file error!" << endl;
        return;
    }
    vector<int> result(dimension, 0);
    int term;
    for(int i = 0; i < vectorCount; i++)
    {
        for(int j = 0; j < dimension; j++)
        {
            fileIn >> term;
            result[j] += term;
        }
    }
    fileIn.close();

    string filename = prefixOut + "_" + to_string(getpid()) + ".txt";
    ofstream fileOut(filename);
    if(!fileOut.is_open())
    {
        cout << "Create file error!" << endl;
        return;
    }
    for(int i = 0; i < dimension; i++)
        fileOut << result[i] << " ";

    fileOut.close();
}
```

```
int main(int argc, char * argv[])
{
    if(argc != 11 && !cmdOptionsExists(argv, argv+argc, {"-i", "-o", "-p", "-m",
"-t"}))
    {
        cout << "Input error!" << endl;
        return -1;
    }

    string prefixIn, prefixOut;
    int fileCount, vectorCount, dimension;
    if(prefixIn = cmdAfterKey(argv, argv+argc, "-i"); prefixIn == "")
    {
        cout << "Prefix In error!" << endl;
        return -1;
    }
    if(prefixOut = cmdAfterKey(argv, argv+argc, "-o"); prefixOut == "")
    {
        cout << "Prefix Out error!" << endl;
        return -1;
    }
    if(fileCount = atoi(cmdAfterKey(argv, argv+argc, "-p").c_str()); fileCount ==
0)
    {
        cout << "File count error!" << endl;
        return -1;

    }
    if(vectorCount = atoi(cmdAfterKey(argv, argv+argc, "-m").c_str());
vectorCount == 0)
    {
        cout << "Vector count error!" << endl;
        return -1;

    }
    if(dimension = atoi(cmdAfterKey(argv, argv+argc, "-t").c_str()); dimension ==
0)
    {
        cout << "Dimesion error!" << endl;
        return -1;

    }

    pid_t processes[fileCount];
    pid_t mainProcess = getpid();
    int i = 0;
    while(i < fileCount)
    {
        if(getpid() == mainProcess)
```

```
            processes[i++] = fork();
        else
            break;
    }
    if(getpid() != mainProcess)
    {
        string fileNameIn = createInFile(prefixIn, vectorCount, dimension);
        createOutFile(prefixOut, vectorCount, dimension, fileNameIn);
    }

    int status;
    if(mainProcess == getpid())
    {
        for(int j = 0; j < fileCount;)
        {
            waitpid(processes[j], &status, 0);
            if(WIFEXITED(status))
                j++;
        }
    }

    return 0;
}
```