МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет автоматики и вычислительной техники

Кафедра автоматики



**ОТЧЁТ**

**по лабораторной работе №3**

***«****Многозадачное программирование****»***

по дисциплине: «Программирование»

Вариант № 2

Выполнили:Проверил:

студенты гр. АВТ-019 Нерлих Максим, с.п. Ядрышников О.Д.

Владислав Иманов, Владимир Рожнов

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г.«\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка, подпись)

Новосибирск

2021

Оглавление

[Цель работы 3](#_Toc84443194)

[Методические указания 3](#_Toc84443195)

[Порядок выполнения работы 3](#_Toc84443196)

[Задание 3](#_Toc84443197)

[Исходный код 3](#_Toc84443198)

[Алгоритм работы 5](#_Toc84443199)

[Вывод 6](#_Toc84443200)

# Цель работы

Изучить способы и средства реализации параллельно выполняющихся процессов.

# Методические указания

1. Проект может быть реализован на Visual C++ 6.0 или в среде Borland C++ 5.0 и выше. В первом случае выбирается консольное приложение Win32 без дополнительных библиотек. В любом случае в программу
2. Выбор функции для ожидания завершения порожденных процессов зависит от логики работы программы, определяемой вариантом задания. Уничтожение порожденных процессов применяется лишь в тех вариантах, где это действительно необходимо.
3. Для обмена информацией между процессами рекомендуется использовать аргументы командной строки и коды завершения процессов.
4. Примеры кода дочерней (file\_new.cpp) и родительской (spaces\_new.cpp) программ доступны по адресу: http://gun.cs.nstu.ru/ssw/API OC.

# Порядок выполнения работы

1. В качестве программы, реализующей порожденный (дочерний) процесс, использовать программу из лабораторной работы.
2. Написать и отладить программу, реализующую родительский процесс, порождающий столько дочерних процессов, сколько аргументов командной строки им получено, ожидающий завершения порожденных процессов, получающий результаты выполнения порожденных процессов.

# Задание

|  |  |  |
| --- | --- | --- |
| Вариант | Задание | Параметры командной строки |
| 2 | В конце каждой строки вставить заданный символ | 1. Имя входного файла 2. Заданный символ |

# Исходный код

main.cpp

#include "lib.h"

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <sys/wait.h>

#include <unistd.h>

using namespace std;

int main(int argc, char\* argv[]) {

const vector<string> params(argv, argv + argc);

if (params.size() < 3) {

cerr << "Too few arguments" << endl;

return -1;

}

const string filename = params.at(1);

const string symbol = params.at(2);

pid\_t pid = fork();

if (pid == 0) {

appendFile(filename, symbol);

} else {

waitpid(pid, nullptr, 0);

cout << "child process exited" << endl;

}

return 0;

}

lib.h

#include <string>

void appendFile(std::string filename, std::string symbol);

lib.cpp

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

void appendFile(string filename, string symbol) {

ifstream input(filename);

if (!input) {

cerr << "Can't open " << filename << endl;

}

string outFilename = filename;

int dot = filename.find('.');

outFilename.replace(

outFilename.begin() + dot,

outFilename.end(),

".out"

);

ofstream output(outFilename, ios::app);

if (!output) {

cerr << "Can't write to " << outFilename << endl;

}

while (input) {

string buff;

input >> buff;

output << buff << symbol << endl;

}

}

Makefile

all: lib main

lib: lib.cpp

clang++ -std=c++17 -dynamiclib -o liblab.dylib lib.cpp

main: main.cpp

clang++ -std=c++17 -o main -L./ -llab main.cpp

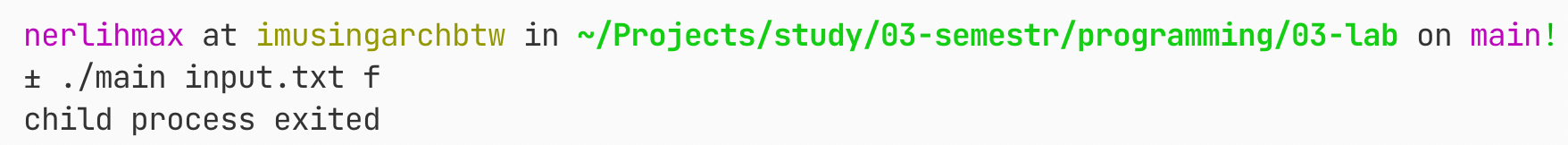
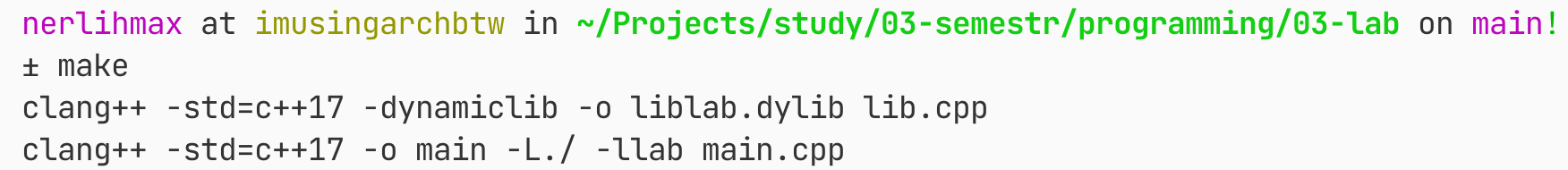
clean:

rm \*.dylib main

# Алгоритм работы

1. В папке с программой создаем текстовый документ input.txt с исходными данными

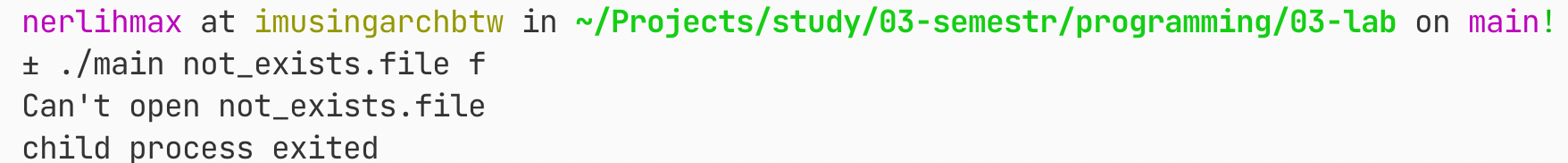


1. Запускаем программу следующим набором команд: 
2. После отработки программы, вывод записывается в файл с таким же названием, как и входной файл, но с расширением out, а также сообщит в консоль о завершении дочернего процесса, который выполнял манипуляции.



В результате программа добавила к концу каждой строки заданный символ

1. Если программа не смогла открыть файл с исходными данными, выводится ошибка:



1. Если выходной файл уже существует, программа создаст и заменит прежний

# Вывод

В результате работы над лабораторной работой изучили особенности написания программ с многозадачностью путем использования дополнительных дочерних процессов.