Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Информатика |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №1**

|  |
| --- |
| Управление процессами в OC GNU/Linux |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | А. Н. Пупков |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-16/1б, 032155832 |  |  |  | Н. А. Терентьев |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Цели 3

2 Задачи 3

3 Описание варианта задания 3

4 Ход выполнения 4

4.1 Программа, запускающая дочерний процесс 4

4.2 Программа, реализующая основной алгоритм 5

4.3 Doxygen документация 6

4.4 Юнит-тестирование с помощью фреймворка CUnit 6

5 Выводы 7

Список использованных источников 8

Приложение А 9

**1 Цели**

Изучение особенностей программной реализации многозадачных приложений в ОС GNU/Linux.

**2 Задачи**

Для выполнения практической работы необходимо выполнить  
следующие задачи:

* ознакомиться с краткими теоретическими сведениями по управлению процессами в ОС GNU/Linux;
* получить у преподавателя индивидуальный вариант задания, который предусматривает разработку программы, являющейся дочерним процессом, которая запускается родительским процессом (вторая разрабатываемая программа) с обязательным использованием концепции fork-and-exec;
* произвести разработку юнит-тестов основных функциональных блоков код дочернего процесса;
* используя изученные механизмы, разработать и отладить программный код для родительского и дочернего процессов;
* составить общее описание результатов, инструкции по сборке и использованию программ, а также инструкцию по получению документации, сформировать архив формата tar.gz и представить на проверку с исходными текстами программы.

**3 Описание варианта задания**

Работа выполнена в соответствии с вариантом 21.

Программа принимает от пользователя две или три строки. При вводе трех строк первая и третья – это комплексные числа1 вида «±1.2.±i5.4» (моделируется программно двумя вещественными и/или целыми числами), причем первый знак необязателен, вторая строка – это знак операции вида «+», «-», «\*», «/». При вводе двух строк первая — комплексное число, вторая — операция, обозначенная литерами S (комплексно-сопряженное число), R (вещественная часть), I (мнимая часть). Затем программа выполняет требуемую операцию с полученными операндами, и выводит результат на экран.

**4 Ход выполнения**

**4.1 Программа, запускающая дочерний процесс**

В рамках практической работы необходимо разработать программу, являющуюся родительским процессом, которая будет запускать дочерний процесс с использованием концепции fork-and-exec. Код программы размещён в листинге 1.

Листинг 1 – Содержимое файла main.c

int spawn(char\* program, char\*\* argList)

{

pid\_t childPid;

childPid = fork();

if (0 != childPid)

{

return childPid;

}

else

{

execvp(program, argList);

fprintf(stderr, "случилась ошибка в execvp\n");

abort();

}

}

int main()

{

int childStatus;

char\* argList[] = {};

printf("Запуск дочернего процесса...\n")

spawn("./child", argList);

wait(&childStatus);

if(WIFEXITED(childStatus))

{

printf("Дочерний процесс завершился нормально с кодом %d.\n"

, WEXITSTATUS(childStatus)

);

}

else

{

printf("Дочерний процесс закончился ненормально.\n");

}

return 0;

}

**4.2 Программа, реализующая основной алгоритм**

Для выполнения поставленного задания была разработана программа, являющаяся дочерним процессом. Весь основной программы расположен в файле child.c.

Для работы с комплексными числами была написана структура complex. В листинге 2 размещен код структуры.

Листинг 2 – Содержимое файла structure.h

typedef struct complex complex; //!< Структура комплексного числа

struct complex {

float real; /\*!< Вещественная часть комплексного числа \*/

float imag; /\*!< Мнимая часть комплексного числа \*/

};

Программа получает от пользователя две или три строки (по выбору), проводит вычисления и выводит результат в консоль. Работа с программой представлена на рисунке 1.

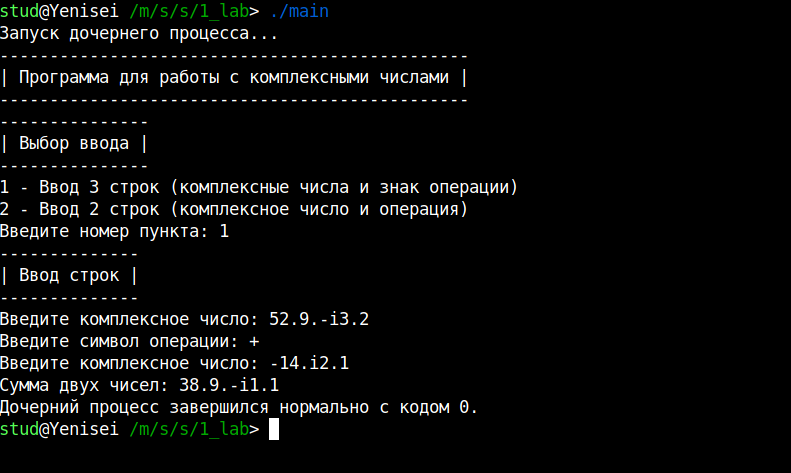


Рисунок 1 – Работа с программой

**4.3 Doxygen документация**

Все файлы содержат комментарии в стиле системы doxygen. Благодаря конфигурационному файлу была сгенерирована документация. Её внешний вид представлен на рисунке 2.

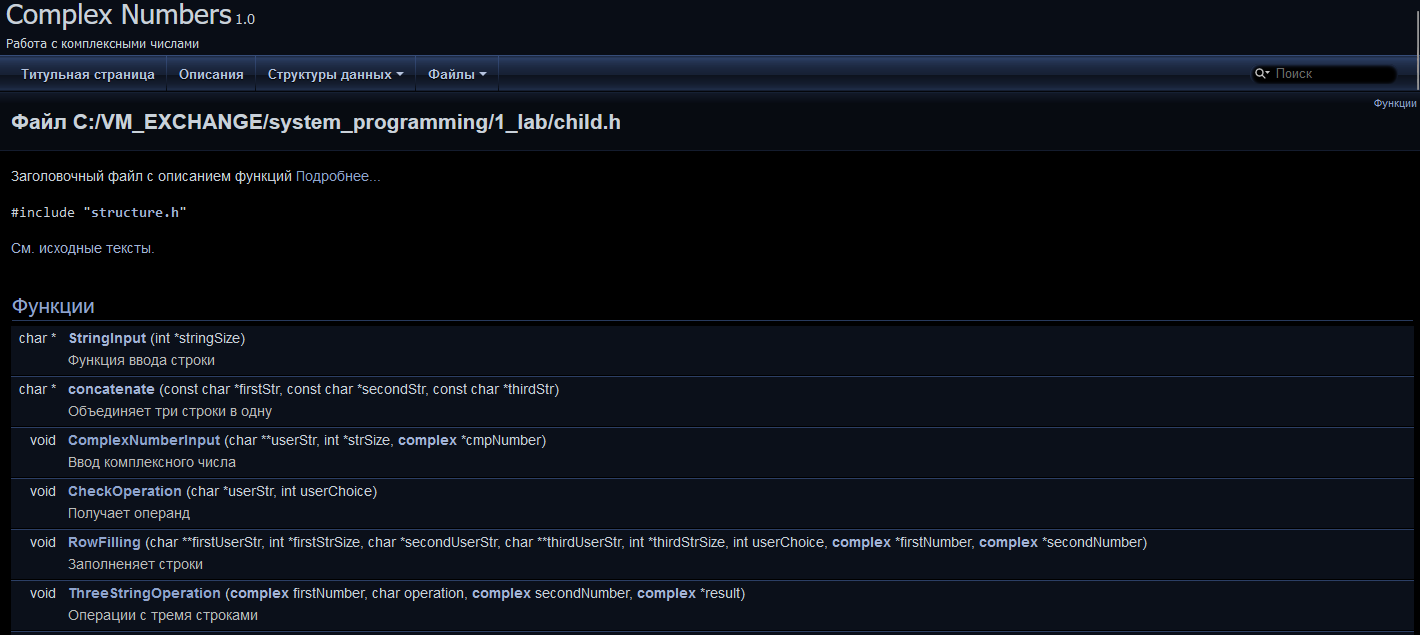


Рисунок 2 – Сгенерированная документация

**4.4 Юнит-тестирование с помощью фреймворка CUnit**

Для основных функциональных блоков код дочернего процесса были разработаны юнит-тесты. Пример такого теста представлен в листинге 3.

Листинг 3 – Юнит-тест

void FirstTest()

{

complex number;

complex result;

number.real = 2.54;

number.imag = -34;

TwoStringOperation(number, 'R', &result);

bool isValid = false;

if (number.real == result.real)

{

isValid = true;

}

CU\_ASSERT\_EQUAL(isValid, true)

}

**5** **Выводы**

В ходе работы были изучены особенности программной реализации многозадачных приложений в ОС GNU/Linux. Была разработана программа в соответствии с вариантом.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 СТО 4.2-07-2014 Система менеджмента качества. Общие требования  
к построению, изложении и оформлению документов учебной деятельности.  
Дата введения – 30.12.2-13.

**ПРИЛОЖЕНИЕ А**

**Исходный код файла child.c**

/\*! Код дочернего процесса программы \*/

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <math.h>

#include "structure.h"

#define CORRECT\_INPUT 2

#define END\_STRING '\n'

#define THREE\_LINES 1

#define TWO\_LINES 2

#define BACKSPACE\_KEY 8

#define START\_CHAR\_RANGE 32

#define END\_CHAR\_RANGE 126

char\* StringInput(int\* stringSize)

{

char\* userStr = (char\*)malloc(1 \* sizeof(char));

userStr[0] = '\0';

char curChar = 0;

int curSize = 1;

printf("Введите комплексное число: ");

while(curChar != '\n')

{

curChar = getchar();

int deltaVal = 0; // Определяет, на сколько изменится длина массива

int lengthDif = 0;

// Если мы стираем символы, а не пишем их,

if (curChar == BACKSPACE\_KEY)

{

deltaVal = -1; // то будем уменьшать длину массива

lengthDif = 1; // и копировать строку до предпоследнего символа

}

// Иначе проверяем, входит ли введённый символ в диапазон печатных

else

{

if (curChar >= START\_CHAR\_RANGE && curChar <= END\_CHAR\_RANGE)

{

deltaVal = 1; // Если да, то будем увеличивать длину на 1

lengthDif = 2; // Не заполняем последние 2 символа -

// оставляем место для введённого символа и '\0'

}

else

{

continue; // Если это не печатный символ, то пропускаем его

}

}

// Если стирать больше нечего, но пользователь

// всё равно жмёт Backspace.

int newSize = curSize + deltaVal;

if (newSize == 0)

{

continue; // то мы переходим на следующую итерацию - ждём '\n'

}

char\* tmpStr = (char\*)malloc(newSize \* sizeof(char));

if (tmpStr) // Проверяем, выделилась ли память

{

// Идём до предпоследнего символа, т.к. надо в конец записать '\0'

for (int i = 0;

i < newSize - lengthDif;

++i)

{

tmpStr[i] = userStr[i];

}

if (curChar != BACKSPACE\_KEY) // Если введён печатный символ,

{

tmpStr[newSize - 2] = curChar; // Добавляем его в строку

tmpStr[newSize - 1] = '\0';

}

free(userStr);

userStr = tmpStr;

curSize = newSize;

}

else

{

printf("Не могу выделить память под обновлённую строку!");

break;

}

}

\*stringSize = curSize - 1;

return userStr;

}

char\* concatenate(const char\* firstStr, const char\* secondStr,

const char\* thirdStr) {

size\_t firstLength = strlen(firstStr);

size\_t secondLength = strlen(secondStr);

size\_t thirdLength = strlen(thirdStr);

char \*res = malloc(firstLength + secondLength + thirdLength + 1);

if (res) {

memcpy(res, firstStr, firstLength);

memcpy(res + firstLength, secondStr, secondLength);

memcpy(res + firstLength + secondLength, thirdStr,

thirdLength + 1);

}

return res;

}

void ComplexNumberInput(char\*\* userStr, int\* strSize, complex\* cmpNumber)

{

char\* message = "Неверный ввод. Попробуйте снова.\n";

char\* findSearch = NULL;

bool isValid = false;

int result = 0;

float firstNumberF = 0.0, secondNumberF = 0.0;

int firstNumberI = 0;

while (!isValid)

{

\*userStr = StringInput(strSize);

if (strstr(\*userStr, ".i") != NULL)

{

findSearch = ".i";

}

else if (strstr(\*userStr, ".+i") != NULL)

{

findSearch = ".+i";

}

else if (strstr(\*userStr, ".-i") != NULL)

{

findSearch = ".-i";

}

if (findSearch != NULL)

{

result = sscanf(\*userStr,

concatenate("%f", findSearch,

"%f"),

&firstNumberF, &secondNumberF);

if (result != 2)

{

firstNumberF = secondNumberF = 0.0;

result = sscanf(\*userStr, concatenate("%d",

findSearch,

"%f"),

&firstNumberI, &secondNumberF);

}

}

if (result == 2)

{

if (firstNumberI != 0)

{

cmpNumber -> real = firstNumberI;

}

else

{

cmpNumber -> real = firstNumberF;

}

if (findSearch[1] == '-')

{

cmpNumber -> imag = -secondNumberF;

}

else

{

cmpNumber -> imag = secondNumberF;

}

isValid = true;

}

else

{

printf("%s", message);

}

}

}

void CheckOperation(char\* userStr, int userChoice)

{

const char message[] = "Введите символ операции: ";

char inputChar = '\0';

char\* pch;

char\* symbols;

printf("%s", message);

int input = scanf("%c%c", &\*userStr, &inputChar);

if (userChoice == TWO\_LINES)

{

symbols = "IRS";

}

else

{

symbols = "+-\*/";

}

pch=strchr(symbols, \*userStr);

while (input != CORRECT\_INPUT || inputChar != END\_STRING ||

pch == NULL)

{

if (pch == NULL && inputChar != END\_STRING)

{

while ((inputChar = getchar()) != '\n');

}

\*userStr = 0;

printf("Неверный ввод. Попробуйте снова.\n%s", message);

input = scanf("%c%c", &\*userStr, &inputChar);

pch=strchr(symbols, \*userStr);

}

}

void RowFilling(char\*\* firstUserStr, int\* firstStrSize, char\* secondUserStr,

char\*\* thirdUserStr, int\* thirdStrSize, int userChoice,

complex\* firstNumber, complex\* secondNumber)

{

printf("--------------\n");

printf("| Ввод строк |\n");

printf("--------------\n");

ComplexNumberInput(firstUserStr, firstStrSize,

firstNumber);

CheckOperation(secondUserStr, userChoice);

if (userChoice == THREE\_LINES)

{

ComplexNumberInput(thirdUserStr, thirdStrSize,

secondNumber);

}

}

void ThreeStringOperation(complex firstNumber, char operation,

complex secondNumber, complex\* result)

{

if (operation == '+')

{

result -> real = firstNumber.real + secondNumber.real;

result -> imag = firstNumber.imag + secondNumber.imag;

if (result -> imag >= 0)

{

printf("Сумма двух чисел: %.1f.i%.1f\n", result -> real,

result -> imag);

}

else

{

printf("Сумма двух чисел: %.1f.-i%.1f\n", result -> real,

fabsf(result -> imag));

}

}

else if (operation == '-')

{

result -> real = firstNumber.real - secondNumber.real;

result -> imag = firstNumber.imag - secondNumber.imag;

if (result -> imag >= 0)

{

printf("Разность двух чисел: %.1f.i%.1f\n", result -> real,

result -> imag);

}

else

{

printf("Разность двух чисел: %.1f.-i%.1f\n", result -> real,

fabsf(result -> imag));

}

}

else if (operation == '\*')

{

result -> real = firstNumber.real \* secondNumber.real -

firstNumber.imag \* secondNumber.imag;

result -> imag = firstNumber.imag \* secondNumber.real +

firstNumber.real \* secondNumber.imag;

if (result -> imag >= 0)

{

printf("Произведение двух чисел: %.1f.i%.1f\n", result -> real,

result -> imag);

}

else

{

printf("Произведение двух чисел: %.1f.-i%.1f\n", result -> real,

fabsf(result -> imag));

}

}

else if (operation == '/')

{

float x, y, z;

if (secondNumber.real == 0 && secondNumber.imag == 0)

printf("Деление на 0 + 0i не разрешено.");

else

{

x = firstNumber.real \* secondNumber.real +

firstNumber.imag \* secondNumber.imag;

y = firstNumber.imag \* secondNumber.real -

firstNumber.real \* secondNumber.imag;

z = secondNumber.real \* secondNumber.real +

secondNumber.imag \* secondNumber.imag;

if (fmodf(x, z) == 0.0 && fmodf(y, z) == 0.0)

{

if (y/z >= 0.0)

printf("Деление двух чисел: %.1f.i%.1f\n",

x/z, y/z);

else

printf("Деление двух чисел: %.1f.-i%.1f\n",

x/z, fabsf(y/z));

}

else if (fmodf(x, z) == 0.0 && fmodf(y, z) != 0.0)

{

if (y/z >= 0.0)

printf("Деление двух чисел: %.1f.i%.1f/%.1f\n",

x/z, y, z);

else

printf("Деление двух чисел: %.1f.-i%.1f/%.1f\n",

x/z, fabsf(y), z);

}

else if (fmodf(x, z) != 0.0 && fmodf(y, z) == 0.0)

{

if (y/z >= 0)

printf("Деление двух чисел: %.1f/%.1f.i%.1f\n",

x, z, y/z);

else

printf("Деление двух чисел: %.1f.-i%.1f/%.1f\n",

x, fabsf(z), y/z);

}

else

{

if (y/z >= 0)

printf("Деление двух чисел: %.1f/%.1f.i%.1f/%.1f\n",

x, z, y, z);

else

printf("Деление двух чисел: %.1f/%.1f.-i%.1f/%.1f\n",

x, z, fabsf(y), z);

}

}

}

}

void TwoStringOperation(complex number, char operation, complex\* result)

{

if (operation == 'S')

{

result -> imag = -number.imag;

if (number.imag >= 0)

{

printf("Комплексно-сопряженное число: %.1f.-i%.1f\n",

number.real, number.imag);

}

else

{

printf("Комплексно-сопряженное число: %.1f.i%.1f\n",

number.real, fabsf(number.imag));

}

}

else if (operation == 'R')

{

result -> real = number.real;

printf("Вещественная часть: %.1f\n", number.real);

}

else

{

result -> imag = number.imag;

printf("Мнимая часть: %.1f\n", number.imag);

}

}

int CheckingInput(const char message[], int lowerBound, int count)

{

int userInput = 1;

char inputChar = '\0';

printf("%s", message);

int input = scanf("%d%c", &userInput, &inputChar);

while (input != CORRECT\_INPUT || inputChar != END\_STRING

|| userInput < lowerBound || userInput > count)

{

if (userInput >= lowerBound && userInput <= count)

{

while ((inputChar = getchar()) != '\n');

}

userInput = 1;

printf("Неверный ввод. Попробуйте снова.\n%s", message);

input = scanf("%d%c", &userInput, &inputChar);

}

return userInput;

}

void PrintMenu(void)

{

printf("---------------\n");

printf("| Выбор ввода |\n");

printf("---------------\n");

printf("1 - Ввод 3 строк (комплексные числа и знак операции)\n");

printf("2 - Ввод 2 строк (комплексное число и операция)\n");

}

#ifndef DOING\_UNIT\_TESTS

int main(void)

{

printf("-----------------------------------------------\n");

printf("| Программа для работы с комплексными числами |\n");

printf("-----------------------------------------------\n");

const char firstMessage[] = "Введите номер пункта: ";

int userChoice, lowerBound = 1;

int menuItemsCount = 2;

char\* firstUserStr = 0;

int firstStrSize = 0;

char secondUserStr = 0;

char\* thirdUserStr = 0;

int thirdStrSize = 0;

complex firstNumber;

complex secondNumber;

complex result;

PrintMenu(); // выводим меню на экран

// получаем номер выбранного пункта меню

userChoice = CheckingInput(firstMessage, lowerBound,

menuItemsCount);

RowFilling(&firstUserStr, &firstStrSize, &secondUserStr,

&thirdUserStr, &thirdStrSize, userChoice, &firstNumber,

&secondNumber);

switch (userChoice)

{

case THREE\_LINES:

ThreeStringOperation(firstNumber, secondUserStr,

secondNumber, &result);

break;

case TWO\_LINES:

TwoStringOperation(firstNumber, secondUserStr,

&result);

break;

default:

free(firstUserStr);

free(thirdUserStr);

printf("Выход...");

}

return 0;

}

#endif