Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра «Информатики»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Управление процессами в ОС GNU/Linux

тема

Преподаватель \_\_\_\_ 31.09.2016 Пересунько Павел Викторович

подпись, дата инициалы, фамилия

Студент КИ15-17Б \_\_\_\_ 31.09.2016 Радионов Тимофей Вадимович

номер группы подпись, дата инициалы, фамилия

Красноярск 2016

**Цель работы с постановкой задачи.**

**Цель работы**: Изучение особенностей программной реализации многозадачных приложений в ОС GNU/Linux.

**Общая постановка задачи**: Требуется: разработать программу в виде Linux-приложения, которая представляет собой родительский процесс. Результат выполнения выводится на терминал/консоль. Также необходимо написать несколько тестов.

**Функционал программы представляющий собой дочерний процесс согласно варианту:** Программа принимает от пользователя элементы целочисленного одномерного массива, а также целочисленное значение для поиска, затем осуществляет поиск элемента двумя любыми известными методами, кроме прямого (линейного) поиска, и выводит результат на экран.

**Исходные тексты программ.**

Родительский процесс (fork.c):

/\*! \file fork.c

\* \brief Fork, exec and wait process.

\*

\* \details Forking child process with its executing followed by its waiting. Child process is a list of files via running "./main" command.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <signal.h>

#include <sys/wait.h>

#include <unistd.h>

/\*! \brief Spawn a child process

\*

\* \details Spawn a child process running a new program, as given by PROGRAM.

\*

\* \param program The name of the program to runs; the path will be searched for this program.

\* \param argList A NULL-terminated list of character strings to be passed as the program's argument list.

\* \return Identifier of the spawned process.

\*/

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

{

// Создание копии процесса процесса

pid\_t childPid;

childPid = fork();

if (0 != childPid) //Если процесс не совпадает с родительским \*/

{

return childPid;

}

else

{

execvp(program, argList);

fprintf(stderr, "Ошибка в процессе\n");

abort();

}

}

/\*! \brief Main function

\* \return Integer 0 upon exit success

\*/

int main()

{

int childStatus;

char\* argList[] = { NULL };

//Создание дочернего процесса по заданному пути

spawn("/home/timon72/LW/lab1/process.out", argList);

// Ожидание завершения работы дочернего процесса

wait(&childStatus);

if(WIFEXITED(childStatus))

{

printf("Дочерний процесс завершился благополучно с кодом %d.\n", WEXITSTATUS(childStatus));

}

else

{

printf("Дочерний процесс завершился неблагополучно\n");

}

return 0;

}

Дочерний процесс (process.cpp):

/\*

@file process.cpp

@mainpage Лабораторная работа №1

@author Радионов Тимофей, группа КИ15-17Б

@brief ВАРИАНТ 18. Задание: Выполнить поиск элемента введенного пользователем массива целочисленных чисел любыми алгоритмами, кроме прямого поиска

\*/

#include <iostream>

#include <string>

#include <vector>

#include <cstdlib>

using namespace std;

/\*! \brief Binary search in ordered array.

\*

\* \details Binary search in vector of integers.

\*

\* \param arr Ordered list of integers.

\* \param key Number which we will search.

\*

\* \return index of the element if found.

\* -1, otherwise. Index of nearest element via posBefore parameter.

\*/

int BinarySearch(vector<int>& arr)

{

int key;

int l = 0;

int u = arr.size() - 1;

cout << ">>> БИНАРНЫЙ ПОИСК <<<" << endl;

cout << "Введите число, которое вы хотите найти: ";

cin >> key;

while (l <= u)

{

int m = (l + u) / 2;

if (arr[m] < key)

{

l = m + 1;

}

if (arr[m] > key)

{

u = m - 1;

}

if (arr[m] == key)

{

return m;

}

}

return -1;

}

/\*! \brief Interpolation search in ordered array.

\*

\* \details Interpolation search in vector of integers.

\*

\* \param arr Ordered list of integers.

\* \param key Number which we will search.

\*

\* \return index of the element if found.

\* -1, otherwise. Index of nearest element via posBefore parameter.

\*/

int InterpolationSearch(vector<int>& arr)

{

int key;

int l = 0;

int u = arr.size() - 1;

cout << ">>> ИНТЕРПОЛЯЦИОННЫЙ ПОИСК <<<" << endl;

cout << "Введите число, которое вы хотите найти: ";

cin >> key;

if (arr[l] == key)

{

return l;

}

while (arr[l] < key && arr[u] >= key)

{

int m = l + ((u - l) \* (key - arr[l])) / (arr[u] - arr[l]);

if (key < arr[m])

{

u = m - 1;

}

if (key > arr[m])

{

l = m + 1;

}

if (key == arr[m])

{

return m;

}

}

return -1;

}

/\*! \brief Main function

\* \return Integer 0 - success

\*/

int main()

{

vector<int> arr;

string buffer = "";

// Считывание массива и конвертация в тип int

cout << "Заполните ваш массив (через пробел): ";

getline(cin, buffer);

int end = buffer.find(' ');

int size = 0;

while (end != -1)

{

arr.push\_back(atoi(buffer.substr(0, end).c\_str()));

buffer = buffer.substr(end + 1, buffer.length() - 1);

end = buffer.find(' ');

size++;

}

arr.push\_back(atoi(buffer.substr(0, buffer.length()).c\_str()));

size++;

// Вывод массива на экран

for (int i = 0; i < size; i++)

{

cout << arr[i] << endl;

}

cout << "======================================" << endl;

cout << endl;

// Осуществление бинарного поиска элемента

int x = BinarySearch(arr);

cout << "Индекс искомого числа: " << x << endl;

// Осуществление интерполяционного поиска элемента

int y = InterpolationSearch(arr);

cout << "Индекс искомого числа: " << y << endl;

return 0;

}

Тесты CUnit (test.cpp):

/\*! \file test.cpp

\* \brief Library of functions to operate on sets.

\*

\* \details Unit-testing for library of functions to operate on sets.

\* CUnit library to be linked ($g++ <filename> -lcunit).

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <signal.h>

#include <sys/wait.h>

#include <unistd.h>

#include <iostream>

#include <string>

#include <vector>

#include <cstdlib>

#include <CUnit/CUnit.h>

#include <CUnit/Console.h>

#include <CUnit/Basic.h>

using namespace std;

/\*! \brief Binary search in ordered array.

\*

\* \details Binary search in vector of integers.

\*

\* \param arr Ordered list of integers.

\* \param key Number which we will search.

\*

\* \return index of the element if found.

\* -1, otherwise. Index of nearest element via posBefore parameter.

\*/

int BinarySearch(vector<int>& arr, int key)

{

int l = 0;

int u = arr.size() - 1;

while (l <= u)

{

int m = (l + u) / 2;

if (arr[m] < key)

{

l = m + 1;

}

if (arr[m] > key)

{

u = m - 1;

}

if (arr[m] == key)

{

return m;

}

}

return -1;

}

/\*! \brief Interpolation search in ordered array.

\*

\* \details Interpolation search in vector of integers.

\*

\* \param arr Ordered list of integers.

\* \param key Number which we will search.

\*

\* \return index of the element if found.

\* -1, otherwise. Index of nearest element via posBefore parameter.

\*/

int InterpolationSearch(vector<int>& arr, int key)

{

int l = 0;

int u = arr.size() - 1;

if (arr[l] == key)

{

return l;

}

while (arr[l] < key && arr[u] >= key)

{

int m = l + ((u - l) \* (key - arr[l])) / (arr[u] - arr[l]);

if (key < arr[m])

{

u = m - 1;

}

if (key > arr[m])

{

l = m + 1;

}

if (key == arr[m])

{

return m;

}

}

return -1;

}

/\*! \brief Unit test for binary searching. \*/

void test11()

{

vector<int> arr(3);

arr[0] = 1;

arr[1] = 2;

arr[2] = 3;

int key = 3;

int result = BinarySearch(arr, key);

if (result == 2)

{

CU\_ASSERT(1);

}

else

{

CU\_ASSERT(0);

}

}

/\*! \brief Unit test for interpolation searching. \*/

void test12()

{

vector<int> arr(3);

arr[0] = 1;

arr[1] = 2;

arr[2] = 3;

int key = 2;

int result = InterpolationSearch(arr, key);

if (result == 1)

{

CU\_ASSERT(1);

}

else

{

CU\_ASSERT(0);

}

}

/\*! \brief Main function

\* \return Unit testing error code

\*/

int main()

{

CU\_pSuite suite;

CU\_initialize\_registry();

suite = CU\_add\_suite("main\_suite", NULL, NULL);

CU\_ADD\_TEST(suite, test11);

CU\_ADD\_TEST(suite, test12);

CU\_console\_run\_tests();

CU\_cleanup\_registry();

return 0;

}