МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ ЦИФРОВЫХ ТЕХНОЛОГИЙ, ЭЛЕКТРОНИКИ И ФИЗИКИ (ИЦТЭФ)

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И ЭЛЕКТРОНИКИ (ВТиЭ)

**Отчет по лабораторной работе № 4**

по курсу “Системное программное обеспечение”

**Вариант №10**

Выполнил студент 506 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Осипенко

Проверил: преп. кафедры ВТиЭ.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Сеулеков

Лабораторная работа защищена

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2023

# Цель работы:

Получение навыков работы с процессами в Linux.

# **Задачи:**

1. Создать пару ссылок на файлы в директории input;

2. Разработать программу согласно заданному варианту;

# Задание:

Вариант 10(5) – Написать программу, определяющую номер inode, на который в заданной директории больше всего мягких ссылок и вывести на экран номер этого inode и имена всех файлов, которые на него ссылаются. Если таких inode несколько, вывести их все.

# **Выполнение работы:**

В директорию input помещаем 3 файла:

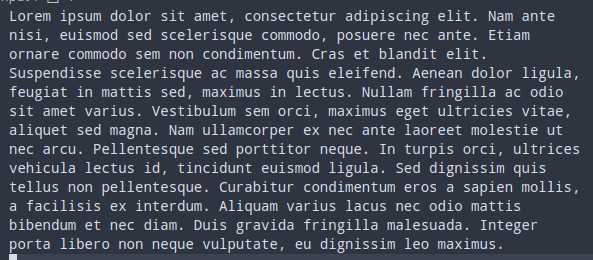


Рис 1.1 Содержимое файла 1.

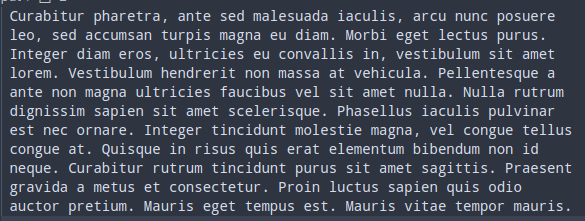


Рис 1.2 Содержимое файла 2.

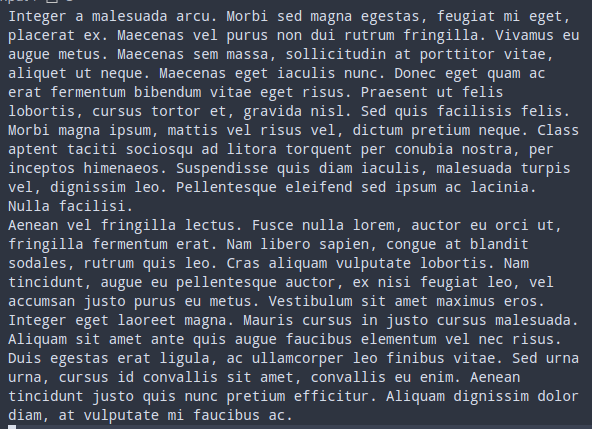


Рис 1.3 Содержимое файла 3.

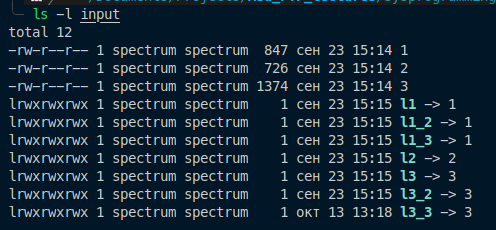


Рис 1.4 Ссылки на файлы

Программа разработана на языке ZIG. Для доступа к директориям, файлам и создания дочерних процессов используются функции из стандартной библиотеки, которые являются обертками системных вызовов определенной операционной системы.

Далее представлен код программы, считывающий ссылки в входной директории, выполняющей вычисления и выводящей результат в stdout:

const std = @import("std");

const fs = std.fs;

const cout = std.io.getStdOut().writer();

const string = []const u8;

pub fn main() !void {

var iterableDir = try fs.cwd().openIterableDir("input", .{});

defer iterableDir.close();

// создаем хеш таблицу

var inodeMap = std.AutoHashMap(u64, u64).init(std.heap.page\_allocator);

var dirIterator = iterableDir.iterate();

var maxInode: u64 = 0;

while (try dirIterator.next()) |dirEntry| { // проходим через каждый файл

if (dirEntry.kind == fs.File.Kind.sym\_link) {

var linkFile = try iterableDir.dir.openFile(dirEntry.name, .{});

var linkFileMeta = try linkFile.stat(); // читаем метаданные

// проверяем, есть ли данный inode в таблице

if (inodeMap.contains(linkFileMeta.inode)) {

// получаем его кол-во

var inodeCount = inodeMap.get(linkFileMeta.inode) orelse 0;

inodeCount += 1; // увеличиваем

try inodeMap.put(linkFileMeta.inode, inodeCount); // сохраняем

// проверяем, если оно максимально

if ((inodeMap.get(maxInode) orelse 0) < inodeCount) {

// сохраняем данный inode как максимальный

maxInode = linkFileMeta.inode;

}

} else { // если inode в таблице нет, то заносим его

try inodeMap.put(linkFileMeta.inode, 1);

}

}

}

try printResult(maxInode, inodeMap, iterableDir); // выводим результат на экран

}

pub fn printResult(maxInode: u64, inodeMap: std.HashMap(u64, u64, std.hash\_map.AutoContext(u64), 80), iterableDir: std.fs.IterableDir) !void {

var dirIterator = iterableDir.iterate();

var maxInodeCount: u64 = inodeMap.get(maxInode) orelse 0; // получаем макс значение

var mapKeyIterator = inodeMap.keyIterator();

while (mapKeyIterator.next()) |key| { // проходим таблицу по ключам

// если inode имеет макс кол-во файлов

if (inodeMap.get(key.\*) orelse 0 == maxInodeCount) {

// выводим inode и макс. число

try cout.print("{d} -> #{d}[ ", .{ key.\*, maxInodeCount });

dirIterator.reset(); // сбрасываем итератор входной директории

// проходимся по всем файлам в поисках с данным inode

while (try dirIterator.next()) |dirEntry| {

if (dirEntry.kind == fs.File.Kind.sym\_link) { // если файл это ссылка

var linkFile = try iterableDir.dir.openFile(dirEntry.name, .{});

var linkFileMeta = try linkFile.stat(); // получаем метаданные

// выводим, если нашли необходимый inode

if (linkFileMeta.inode == key.\*) {

try cout.print("{s} ", .{dirEntry.name});

}

}

}

try cout.print("]\n", .{}); // закрывающий вывод

}

}

}

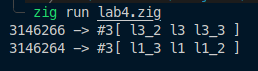


Рис. 1.5 Результат работы

# Вывод:

В ходе выполнения лабораторной работы были получили навыки работы с мягкими и жесткими ссылками.