Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа № 1

по дисциплине «Проектирование и разработка веб-приложений»

Выполнил:

ст. гр. ИРсп-121

Орлова С. В.

Приняла:

Максимова А. С.

Владимир 2023 г.

**Цель работы**

Решение простейших задач на программирование с вводом данных из консоли и выводом в консоль, работа с аргументами командной строки. Работа с файлами и папками средствами NodeJS, создание, удаление и редактирование файлов.

**Вывод данных на консоль**

console.log("Hi!!");

Результат работы представлен на рисунке 1.

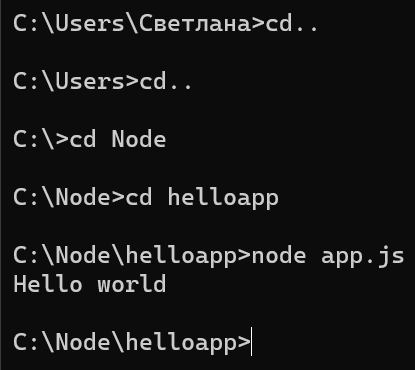


Рисунок 1 - Результат

**Ввод данных с клавиатуры**

const readline = require('readline')

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

})

rl.question(`Как вас зовут? `, name => {

console.log(`Привет,${name}!!`)

rl.close()

})

Результат работы представлен на рисунке 2.

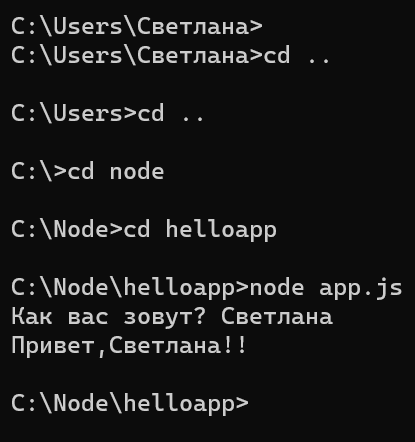


Рисунок 2 - Результат

**Работа с аргументами командной строки**

'use strict';

for (let j = 0; j < process.argv.length; j++) {

console.log(j + ' -> ' + (process.argv[j]));

}

Результат работы представлен на рисунке 3

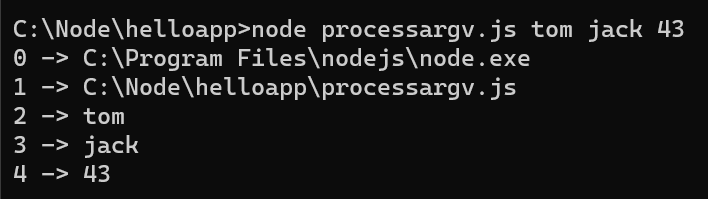


Рисунок 3 – Результат

**Использование модуля Minimist**

Результат работы представлен на рисунках 4 и 5.

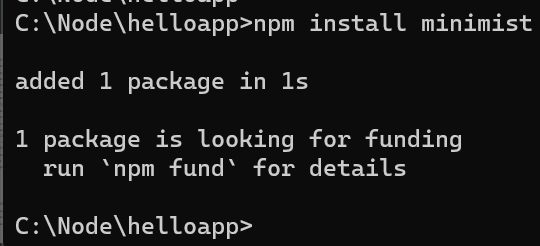


Рисунок 4 – Результат

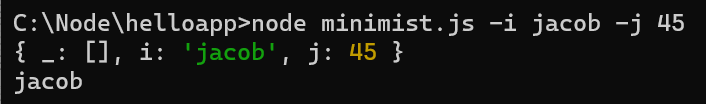


Рисунок 5 – Результат

'use strict';

const minimist = require('minimist');

let args = minimist(process.argv.slice(2), {

alias: {

h: 'help',

v: 'version'

}

});

console.log('args:', args);

Результат работы представлен на рисунке 6.

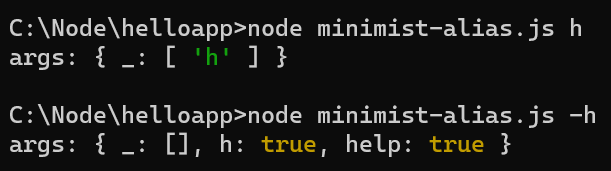


Рисунок 6 – Результат

'use strict';

const minimist = require('minimist');

let args = minimist(process.argv.slice(2), {

default: {

port: 8080

},

});

console.log('args:', args);

Результат работы представлен на рисунке 7.



Рисунок 7 – Результат

**Использование модуля yargs**

'use strict';

const args = require('yargs').argv;

console.log('Name: ' + args.name);

console.log('Age: ' + args.age);

Результат работы представлен на рисунке 8 и 9.

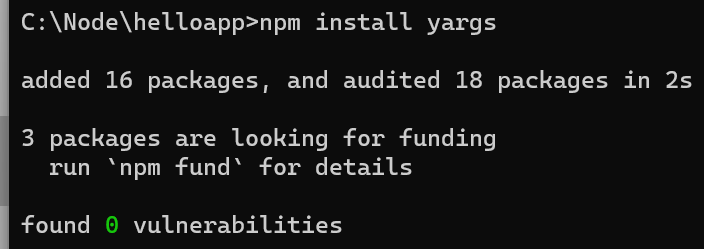


Рисунок 8 - Результат

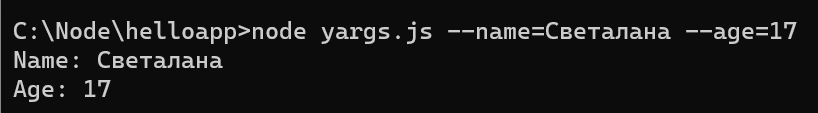


Рисунок 9 - Результат

**Cпособ использования yargs - это .command()**

'use strict';

const argv = require('yargs')

.command('upload', 'upload a file', (yargs) => {}, (argv) => {

console.log('Uploading your file now...');

// Do stuff here

}).argv;Результат работы представлен на рисунке 10.

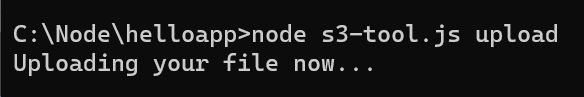


Рисунок 10 - Результат

**Работа с файлами Node.Js**

**Чтение из файла**

const fs = require('fs');

let text = fs.readFileSync("hello.txt", "utf8");

console.log(text)Результат работы представлен на рисунке 11.

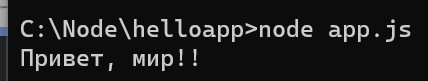


Рисунок 11 - Результат

**Синхронное и асинхронное чтение файла**

const fs = require('fs');

// асинхронное чтение

fs.readFile('hello.txt', 'utf8', function (error, data) {

console.log('Асинхронное чтение файла');

if (error) throw error; // если возникла ошибка

console.log(data); // выводим считанные данные

});

// синхронное чтение

console.log('Синхронное чтение файла');

let fileContent = fs.readFileSync('hello.txt', 'utf8');

console.log(fileContent);

Результат работы представлен на рисунке 12.

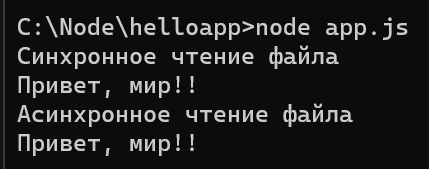


Рисунок 12 - Результат

**Запись файла**

const fs = require('fs');

fs.writeFile('hello.txt', Привет, МИР-2023!', function (error) {

if (error) throw error; // если возникла ошибка

console.log(

'Асинхронная запись файла завершена. Содержимое файла:'

);

let data = fs.readFileSync('hello.txt', 'utf8');

console.log(data); // выводим считанные данные

});

Результат работы представлен на рисунке 13.

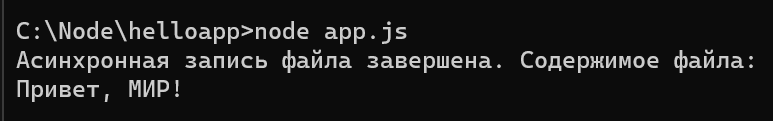


Рисунок 13 - Результат

**Применение методов fs.appendFile()/fs.appendFileSync()**

const fs = require('fs');

fs.appendFileSync('hello.txt', 'Привет, мир!!!');

fs.appendFile('hello.txt', 'Привет, МИР!!!', function (error) {

if (error) throw error; // если возникла ошибка

console.log(

'Запись файла завершена. Содержимое файла:'

);

let data = fs.readFileSync('hello.txt', 'utf8');

console.log(data); // выводим считанные данные

});

Результат работы представлен на рисунке 14.

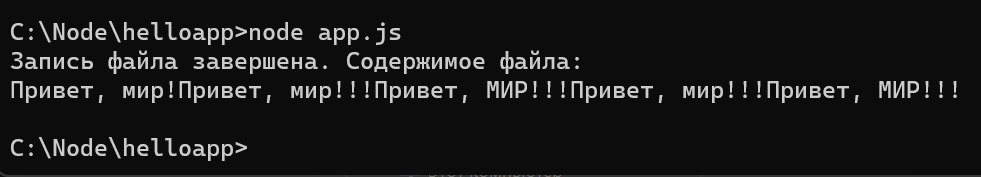


Рисунок 14 - Результат

**Удаление файла**

fs.unlink('hello.txt', (err) => {

if (err) console.log(err);

// если возникла ошибка

else console.log('hello.txt удален');

});

Результат работы представлен на рисунке 15.

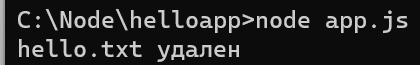


Рисунок 15 - Результат

const fs = require("fs").promises;

const fsConstants = require("fs").constants;

const path = require("path");

const folderName = "testdir";

const fileName = "testfile.txt";

const folderPath = path.join(\_\_dirname, folderName);

const filePath = path.join(\_\_dirname, folderName, fileName);

const checkWriteAccess = async (entityPath) => {

try {

await fs.access(entityPath, fsConstants.F\_OK);

return true;

} catch (err) {

return false

}

}

(async () => {

const folderExists = await checkWriteAccess(folderName)

if (!folderExists) {

await fs.mkdir(folderPath);

}

const fileExists = await checkWriteAccess(filePath);

if (!fileExists) {

await fs.writeFile(filePath, "TIME LOG\n", { encoding: 'utf8', flag: 'w+' });

}

await fs.appendFile(filePath, `A record has been added at ${new Date()}.\n`);

})()

Результат работы представлен на рисунке 16.

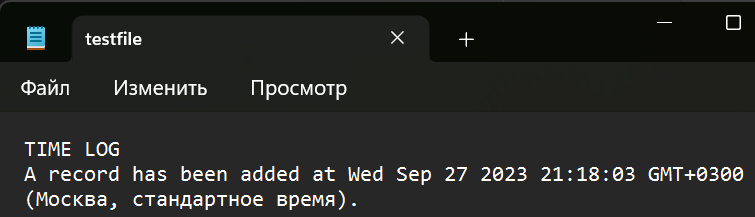


Рисунок 16 – Результат

**Задание**

Создать программу, которая будет выполнять ряд задач:

- Создание нескольких файлов с разным расширением;

- Запись в некоторые из них любые данные;

- Удаление файлов определенного разрешения;

**Программный код**

const fs = require("fs").promises;

const fsConstants = require("fs").constants;

const path = require("path");

const folderName = "werlamma";

const fileName\_1 = "fileName\_1.txt";

const fileName\_2 = "fileName\_2.docx";

const folderPath = path.join(\_\_dirname, folderName);

const filePath\_1 = path.join(folderPath, fileName\_1);

const filePath\_2 = path.join(folderPath, fileName\_2);

const checkWriteAccess = async (entityPath) => {

try {

await fs.access(entityPath, fsConstants.F\_OK);

return true;

} catch (err) {

return false;

}

};

(async () => {

const folderExists = await checkWriteAccess(folderPath);

if (!folderExists) {

await fs.mkdir(folderPath);

}

const fileExists\_1 = await checkWriteAccess(filePath\_1);

if (!fileExists\_1) {

await fs.writeFile(filePath\_1, " Я бежал без передышки. \n Почему вокруг все еще темно? \n Скажи, разве время всё решит? \n Я просто боюсь.");

}

const fileExists\_2 = await checkWriteAccess(filePath\_2);

if (!fileExists\_2) {

await fs.writeFile(filePath\_2, "Владимирский государственный университет");

}

await fs.appendFile(filePath\_2, "высшее учебное заведение Владимира, крупнейший вуз Владимирской области и один из крупнейших в ЦФО. В апреле 2017 года стал одним из региональных опорных университетов");

try {

await fs.unlink(filePath\_2);

console.log(`Файл ${filePath\_2} успешно удален.`);

} catch (err) {

console.error(`Ошибка при удалении ${filePath\_2}:`, err);

}

})();

Результат работы представлен на рисунке 17-19.

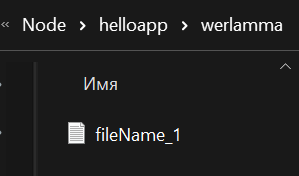


Рисунок 17 – Результат

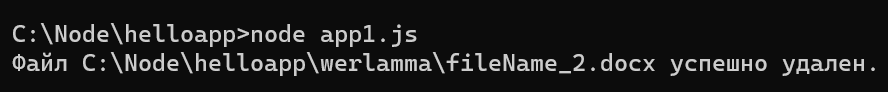


Рисунок 18 – Результат

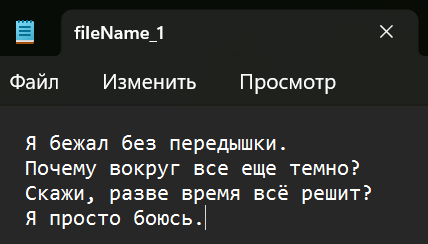


Рисунок 19 – Результат

**Вывод**

В ходе данной лабораторной работы осуществлено решение простейших задач на программирование с вводом данных из консоли и выводом в консоль, работа с аргументами командной строки. Работа с файлами и папками средствами NodeJS, создание, удаление и редактирование файлов.