|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**Лабораторная работа № 4**

**По курсу «Архитектура ЭВМ»**

|  |  |
| --- | --- |
| **Тема**  Взаимодействие между серверами, передача параметров скрипту и дочерние процессы  **Студент** Якуба Д. В.  **Группа** ИУ7-53Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Попов А. Ю. |  |

Москва

2020 г.

# Цели работы

* Изучение и организация взаимодействия между серверами.
* Освоение передачи параметров скрипту.
* Изучение работы с дочерними процессами.

# Отчёт по разделу №7

## Задание 1

### Условие

Создать сервер **А**. На стороне сервера хранится файл с содержимым в формате **JSON**. При получении запроса на **/insert/record** идёт добавление записи в файл. При получении запроса на **/select/record** идёт получение записи из файла. Каждая запись хранит информацию о машине (*название* и *стоимость*).

Создать сервер **Б**. На стороне сервера хранится файл с содержимым в формате **JSON**. Каждая запись в файле хранит информацию о складе и массиве машин, находящихся на данном складе. То есть каждая запись хранит в себе название склада (*строку*) и массив названий машин (*массив строк*). При получении запроса на **/insert/record** идёт добавление записи в файл. При получении запроса на **/select/record** идёт получение записи из файла.

Создать сервер **C**. Сервер выдаёт пользователю страницы с формами для ввода информации. При этом сервер взаимодействует с серверами **А** и **Б**. Реализовать для пользователя функции:

* создание нового типа машины
* получение информации о стоимости машины по её типу
* создание нового склада с находящимися в нём машинами
* получение информации о машинах на складе по названию склада

Реализовать удобный для пользователя интерфейс взаимодействия с системой (использовать поля ввода и кнопки).

### Код программы

Язык: JavaScript

**serverA/index.js**

"use strict";

const fileName = "cars.json";

const express = require("express");

const fs = require("fs");

const app = express();

const port = 5000;

app.listen(port);

console.log("Server on port " + port);

app.use(function(req, res, next) {

    res.header("Cache-Control", "no-cache, no-store, must-revalidate");

    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    res.header("Access-Control-Allow-Origin", "\*");

    next();

});

function loadBody(request, callback) {

    let body = [];

    request.on('data', (chunk) => {

        body.push(chunk);

    }).on('end', () => {

        body = Buffer.concat(body).toString();

        callback(body);

    });

}

app.post("/insert/record", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const carName = obj.carName;

        const price = obj.price;

        if (!fs.existsSync(fileName))

        {

            response.end(JSON.stringify({

                result: "Fail! Data file is not inited."

            }));

            return;

        }

        let gotJSON = fs.readFileSync(fileName, "utf-8");

        try {

            gotJSON = JSON.parse(gotJSON);

        } catch (error) {

            gotJSON = [];

        }

        gotJSON.push({carName: carName, price: price});

        gotJSON = JSON.stringify(gotJSON);

        fs.writeFileSync(fileName, gotJSON);

        response.end(JSON.stringify({

            result: "Success! Record was added."

        }));

    });

});

app.post("/select/record", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const carName = obj.carName;

        if (!fs.existsSync(fileName))

        {

            response.end(JSON.stringify({

                result: "Fail! Data file is not inited."

            }));

            return;

        }

        let gotJSON = fs.readFileSync(fileName, "utf-8");

        gotJSON = JSON.parse(gotJSON);

        let carPrice = -1;

        for (let i = 0; i < gotJSON.length && carPrice < 0; i++)

            if (gotJSON[i].carName === carName)

                carPrice = gotJSON[i].price;

        if (carPrice >= 0)

            response.end(JSON.stringify({

                result: "Found your car! It's " + carName + ' ' + carPrice + '$'

            }));

        else

            response.end(JSON.stringify({

                result: "No car with name " + carName

            }));

    });

});

**serverB/index.js**

"use strict";

const fileName = "stocks.json";

const express = require("express");

const fs = require("fs");

const app = express();

const port = 5001;

app.listen(port);

console.log("Server on port " + port);

app.use(function(req, res, next) {

    res.header("Cache-Control", "no-cache, no-store, must-revalidate");

    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    res.header("Access-Control-Allow-Origin", "\*");

    next();

});

function loadBody(request, callback) {

    let body = [];

    request.on('data', (chunk) => {

        body.push(chunk);

    }).on('end', () => {

        body = Buffer.concat(body).toString();

        callback(body);

    });

}

app.post("/insert/record", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const stockName = obj.stockName;

        const cars = obj.cars;

        if (!fs.existsSync(fileName))

        {

            response.end(JSON.stringify({

                result: "Fail! Data file is not inited."

            }));

            return;

        }

        let gotJSON = fs.readFileSync(fileName, "utf-8");

        try {

            gotJSON = JSON.parse(gotJSON);

        } catch (error) {

            gotJSON = [];

        }

        gotJSON.push({stockName: stockName, cars: cars});

        gotJSON = JSON.stringify(gotJSON);

        fs.writeFileSync(fileName, gotJSON);

        response.end(JSON.stringify({

            result: "Success! Record was added."

        }));

    });

});

app.post("/select/record", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const stockName = obj.stockName;

        if (!fs.existsSync(fileName))

        {

            response.end(JSON.stringify({

                result: "Fail! Data file is not inited."

            }));

            return;

        }

        let gotJSON = fs.readFileSync(fileName, "utf-8");

        gotJSON = JSON.parse(gotJSON);

        let cars = null;

        for (let i = 0; i < gotJSON.length && !cars; i++)

            if (gotJSON[i].stockName === stockName)

                cars = gotJSON[i].cars;

        response.end(JSON.stringify({

                result: cars

            }));

    });

});

**serverC/index.js**

"use strict";

// http://localhost:5002/cars.html

const express = require("express");

const request = require("request");

const app = express();

const port = 5002;

app.listen(port);

console.log(`Server on port ${port}`);

app.use(express.static(\_\_dirname));

app.use(function(req, res, next) {

    res.header("Cache-Control", "no-cache, no-store, must-revalidate");

    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    res.header("Access-Control-Allow-Origin", "\*");

    next();

});

function sendPost(url, body, callback) {

    const headers = {};

    headers["Cache-Control"] = "no-cache, no-store, must-revalidate";

    headers["Connection"] = "close";

    request.post({

        url: url,

        body: body,

        headers: headers,

    }, function (error, response, body) {

        if(error) {

            callback(null);

        } else {

            callback(body);

        }

    });

}

app.get("/insCar", function(request, response) {

    const carName = request.query.carName;

    const price = request.query.price;

    sendPost("http://localhost:5000/insert/record", JSON.stringify({

        carName: carName,

        price: price

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const result = answerObject.result;

        response.end(result);

    });

});

app.get("/getCar", function(request, response) {

    const carName = request.query.carName;

    sendPost("http://localhost:5000/select/record", JSON.stringify({

        carName: carName

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const result = answerObject.result;

        response.end(result);

    });

});

app.get("/insStock", function(request, response) {

    const stockName = request.query.stockName;

    const cars = request.query.cars;

    sendPost("http://localhost:5001/insert/record", JSON.stringify({

        stockName: stockName,

        cars: cars

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const result = answerObject.result;

        response.end(result);

    });

});

app.get("/getStock", function(request, response) {

    const stockName = request.query.stockName;

    sendPost("http://localhost:5001/select/record", JSON.stringify({

        stockName: stockName

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const result = answerObject.result;

        response.end(result);

    });

});

**serverC/cars.html**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Я ОЧЕНЬ ХОЧУ УМЕРЕТЬ</title>

    <link rel="stylesheet" href="/style.css">

</head>

<body>

    <h1>Создание нового типа машины</h1>

    <p>Тип новой машины:</p>

    <input class="inp" id="carNameIns" type="text" spellcheck="false" autocomplete="off">

    <p>Стоимость новой машины:</p>

    <input class="inp" id="priceIns" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <button class="button" onclick="addCar()">Отправить данные о машине!</button>

    <br>

    <br>

    <br>

    <br>

    <h1>Получение информации о стоимости машины по её типу</h1>

    <p>Тип машины:</p>

    <input class="inp" id="carNameGet" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <button class="button" onclick="getCar()">Получить данные о машине!</button>

    <br>

    <br>

    <br>

    <br>

    <h1>Создание нового склада машин</h1>

    <p>Название склада:</p>

    <input class="inp" id="stockNameIns" type="text" spellcheck="false" autocomplete="off">

    <p>Машины на складе (через пробел):</p>

    <input class="inp" id="carsIns" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <button class="button" onclick="addStock()">Отправить данные о складе!</button>

    <br>

    <br>

    <br>

    <br>

    <h1>Получение информации о складе машин</h1>

    <p>Название склада:</p>

    <input class="inp" id="stockNameGet" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <button class="button" onclick="getStock()">Получить данные о складе!</button>

    <h1 id="result-label"></h1>

    <script src="/code.js"></script>

</body>

</html>

**serverC/style.css**

body {

    padding: 60px;

    background: rgb(0, 255, 234);

    font-family: sans-serif;

}

.button {

    padding: 10px;

    background: rgb(0, 255, 136);

    color: rgb(0, 0, 0);

    cursor: pointer;

    display: inline-block;

}

.inp {

    padding: 13px;

    background: rgb(0, 0, 0);

    color: rgb(255, 255, 255);

}

**serverC/code.js**

"use strict";

function ajaxGet(urlString, callback) {

    let r = new XMLHttpRequest();

    r.open("GET", urlString, true);

    r.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");

    r.send(null);

    r.onload = function() {

        callback(r.response);

    };

};

function addCar() {

    const carName = document.getElementById("carNameIns").value;

    const price = document.getElementById("priceIns").value;

    const url = `/insCar?carName=${carName}&&price=${price}`;

    ajaxGet(url, function(stringAnswer) {

        alert(stringAnswer);

    });

};

function getCar() {

    const carName = document.getElementById("carNameGet").value;

    const url = `/getCar?carName=${carName}`;

    ajaxGet(url, function(stringAnswer) {

        alert(stringAnswer);

    });

};

function addStock() {

    const stockName = document.getElementById("stockNameIns").value;

    const cars = JSON.stringify(document.getElementById("carsIns").value.split(' '));

    const url = `/insStock?stockName=${stockName}&&cars=${cars}`;

    ajaxGet(url, function(stringAnswer) {

        alert(stringAnswer);

    });

};

function getStock() {

    const stockName = document.getElementById("stockNameGet").value;

    const url = `/getStock?stockName=${stockName}`;

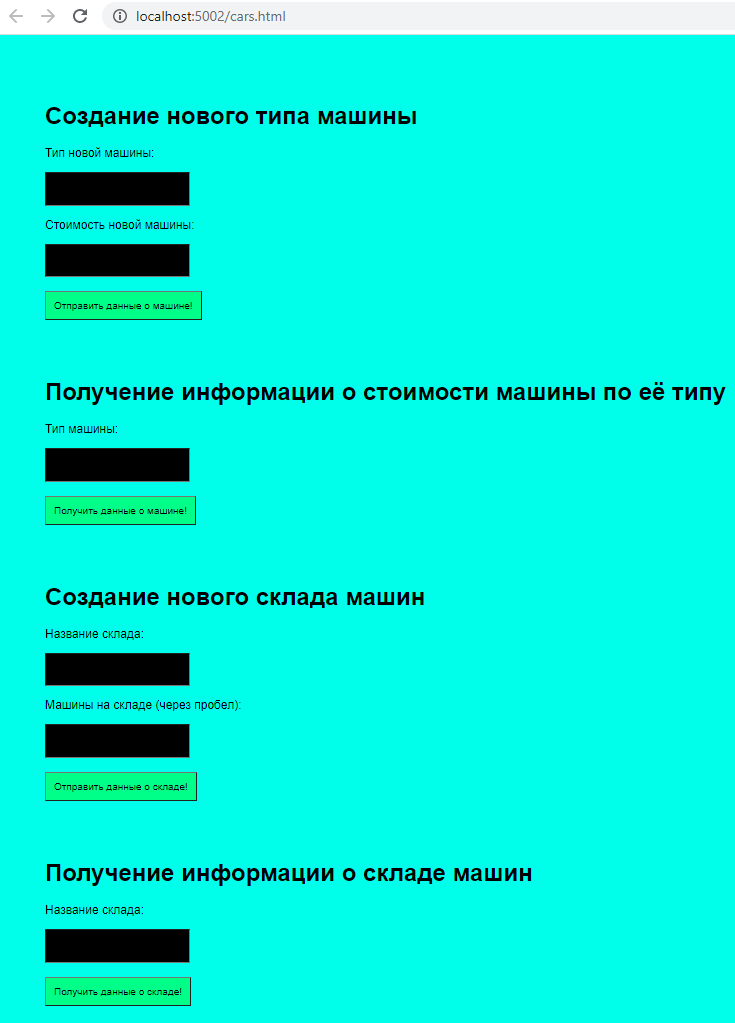
    ajaxGet(url, function(stringAnswer) {

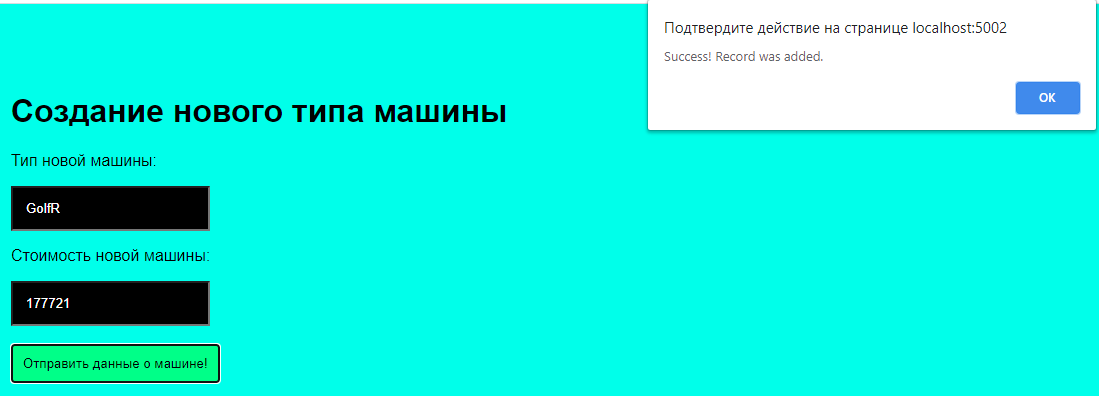
        alert(stringAnswer);

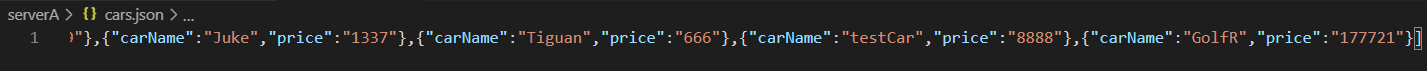
    });

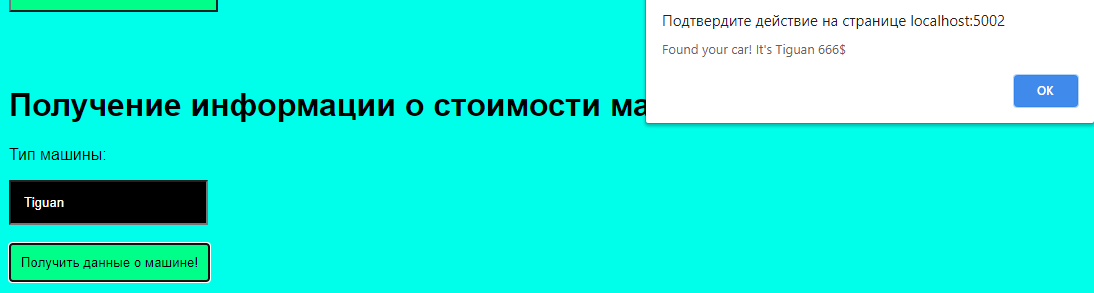
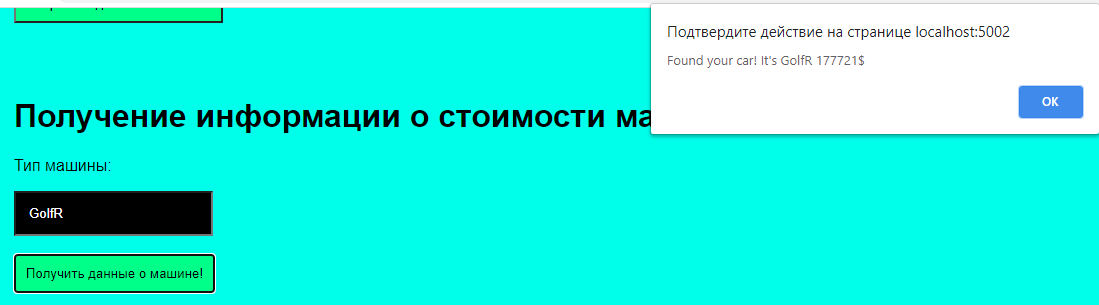
};

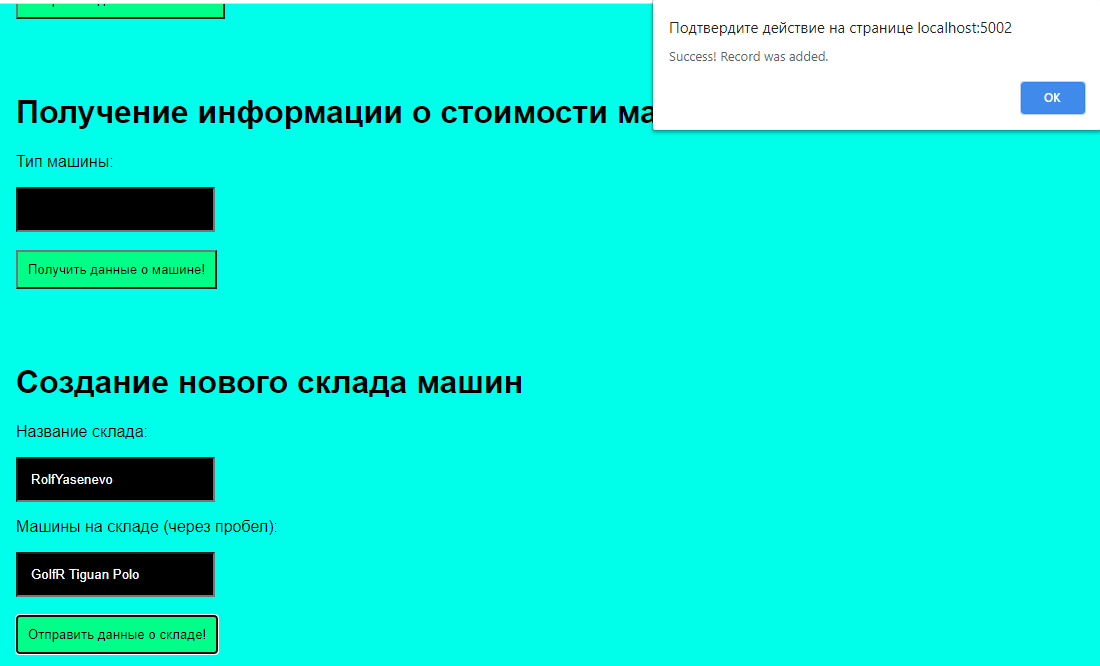
### Результаты тестирования

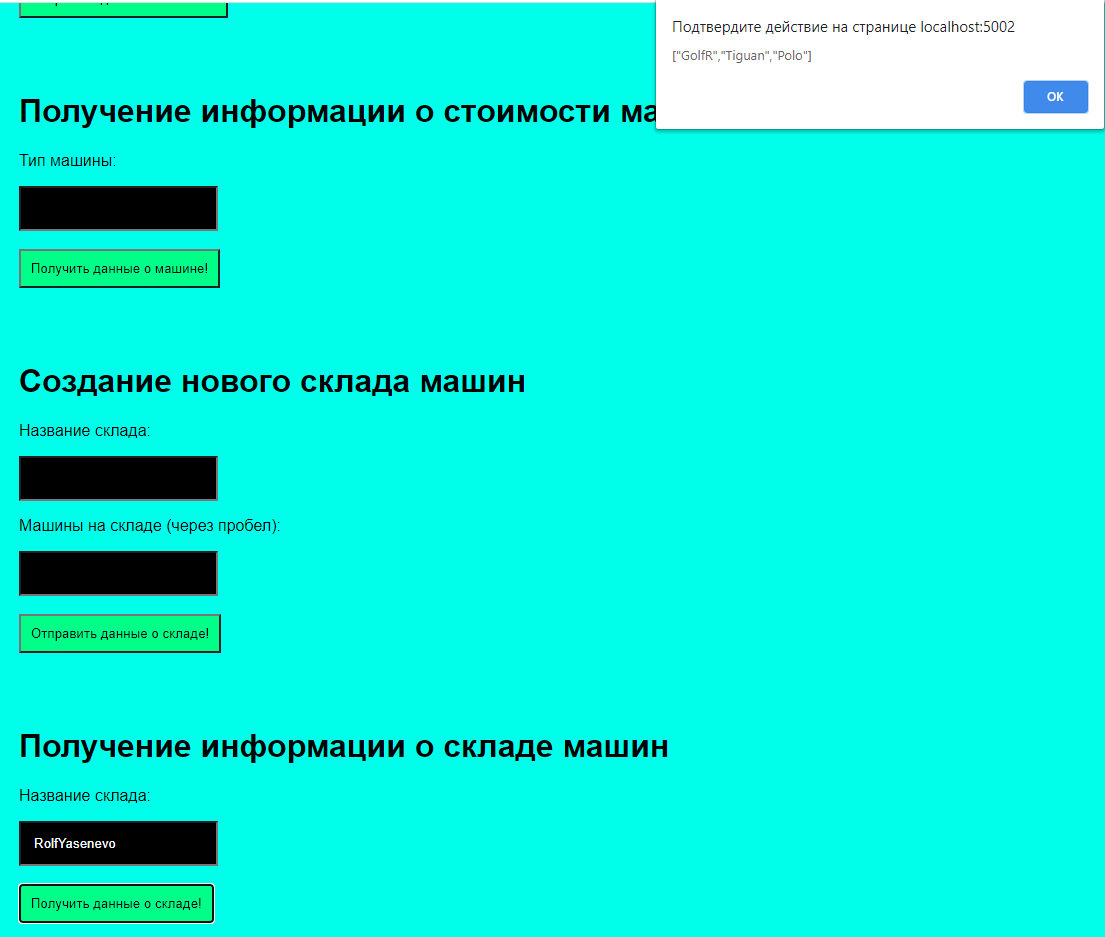












## Задание 2

### Условие

Написать скрипт, который принимает на вход число и считает его факториал. Скрипт должен получать параметр через **process.argv**.

Написать скрипт, который принимает на вход массив чисел и выводит на экран факториал каждого числа из массива. Скрипт принимает параметры через **process.argv**.

При решении задачи вызывать скрипт вычисления факториала через **execSync**.

### Код программы

Язык: JavaScript

**fact.js**

"use strict";

const maxNum = process.argv[2];

let fact = 1;

for (let i = 1; i <= maxNum; i++)

    fact \*= i;

console.log(fact);

**allFact.js**

"use strict";

const execSync = require('child\_process').execSync;

function useCmd(s) {

    const options = {encoding: 'utf8'};

    const cmd = s.toString();

    const answer = execSync(cmd, options);

    return answer.toString();

}

let factArr = process.argv.slice(2, process.argc);

let factCommand = "";

let curFact = 1;

for (let num of factArr)

{

    factCommand = 'node fact.js ' + num;

    curFact = useCmd(factCommand);

    console.log(curFact);

}

### Результаты тестирования

node allFacts.js 1 3 6 2 3

1

6

720

2

6

# Вывод

В результате выполнения работы:

* Было изучено и организовано взаимодействие между серверами.
* Была освоена передача параметров скрипту.
* Была изучена работа с дочерними процессами.