

# Проверка заданий.

Буленок В.Г.

Студент входит на сайт, используя учётную запись на github. При первом входе для него создаётся репозиторий для сохранения выполненных заданий и результатов тестирования заданий. После входа студент видит свои оценки, замечания преподавателя, результаты тестирования заданий. Использование github репозитория для сохранения результатов даёт возможность преподавателю и студенту совместно работать с выполняемыми заданиями, преподаватель и студент могут вносить изменения в код, преподаватель может ставить задания перед студентом, через механизм issues, отслеживать историю развития проекта. Описания действий при тестировании мы сохраняем в репозитории в формате JSON. В этой статье мы рассмотрим программу проводящую тестирование и сохраняющую результат тестирования в репозитории. Способ организации работы со студентами мы заимствовали из курса информатики университета Гарварда <https://cs50.harvard.edu>. Исходный код писали сами, и считаем, что всякий преподаватель должен сам писать код своих инструментов, а не пользоваться готовыми системами, например, moodle. Для написания статьи был выбран стиль литературного программирования. Цель этого показать, как мы реализовали наши инструменты, чтобы другие преподаватели могли использовать наш опыт при создании своих, мы считаем, что преподаватель должен сам создавать инструменты для автоматизации работы со студентами а не брать готовые.

## 1 Анализ задания

Цель программы проверить работу студента и загрузить её вместе с результатом тестирования в репозиторий на github.

### 1.1 Описание проверки

Описание проверки сохраняем в репозитории <https://github.com/{orgTest}/psets>. Например, для задний на 2021 года создаём ветку 2021, для каждого задания создаём в ней файл. Ниже приведён пример файла `hello.js`. Структура файла:

`{ "files":["hello.js"], "output":"Hello, World! "run": "node hello.js"}` Свойство `files` содержит список файлов, отправляемых на проверку. Свойство `output` содержит ожидаемый вывод из программы. Свойство `run` содержит команду запуска.

## 1.2 Запуск программы

Для запуска программы переходим в терминале в каталог с заданием и выполняем команду, например, `check204 2021/hello` Аргумент – имя ветки и название задания (имя файла без расширения).

## 2 Структура

2а  $\langle check204.js \ 2a \rangle \equiv$   
`#!/usr/bin/env node`  
 $\langle \text{загрузка модулей } 8a \rangle$   
 $\langle \text{константы } 8b \rangle$   
 $\langle \text{функции } 9 \rangle$   
`;(async function main()`  
`{`  
 $\langle \text{проверка аргументов командной строки } 2b \rangle$   
 $\langle \text{закачивание описания } 3b \rangle$   
 $\langle \text{тестирование задания } 4b \rangle$   
`if(local)`  
`process.exit(0);`  
 $\langle \text{отправка на проверку } 6b \rangle$   
`})();`

## 3 Реализация

### 3.1 Проверка аргументов

Если пользователь не указал ветку и название задания, то выдаётся сообщение об ошибке. Если пользователь указал неверный адрес, то выдаётся сообщение об ошибке.

2б  $\langle \text{проверка аргументов командной строки } 2b \rangle \equiv$  (2а)  
`await checkArgs();`

3a  $\langle \text{реализация проверки аргументов 3a} \rangle \equiv$  (9)

```

async function checkArgs()
{
    if(process.argv.length < 3 || process.argv.length > 4)
    {
        console.log(chalk.red('Использование: \n
                                check204 branch/problem \n
                                или\n
                                check204 branch/problem local\n'));
        process.exit(1);
    }
    const url2 = 'https://raw.githubusercontent.com/${orgTest}/pset
    const url_exist = await urlExist(url2);
    if(!url_exist)
    {
        console.log(chalk.red("Неверный адрес задания"));
        process.exit(2);
    }
    if(process.argv.length == 4)
    {
        if(process.argv[3] == 'local')
            local = true;
        else
        {
            console.log('Использование: \n
                        check204 branch/problem local\n')
            process.exit(2);
        }
    }
};

```

Uses orgTest 8b.

## 3.2 Закачивание описания

3b  $\langle \text{закачивание описания 3b} \rangle \equiv$  (2a)

```

const psetName = process.argv[2];
const test = await getTask(psetName);

```

4a  $\langle \text{реализация закичивания описания 4a} \rangle \equiv$  (9)

```

async function getTask(psetName)
{
    try
    {
        const url2 = 'https://raw.githubusercontent.com/${orgTest}/pset
        const res = await fetch(url2,
        {
            headers:
            {
                "Accept": "application/json"
            }
        })
        const res1 = await res.json();
        return res1;
    } catch (err)
    {
        console.log(err);
    }
};

```

Uses orgTest 8b.

### 3.3 Тестирование программы

4b  $\langle \text{тестирование задания 4b} \rangle \equiv$  (2a)

$\langle \text{проверка наличия файлов 4c} \rangle$

$\langle \text{проверка правильности вывода из программы 5b} \rangle$

#### 3.3.1 Проверка наличия файлов

4c  $\langle \text{проверка наличия файлов 4c} \rangle \equiv$  (4b)

```

await fileExists(test);

```

5a  $\langle$ реализация проверки наличия файлов 5a $\rangle \equiv$  (9)

```

async function filesExists(test)
{
    const files = await readdir(process.cwd());
    for(const file of test.files)
    {
        if(files.includes(file))
        {
            console.log(chalk.green('Файл ${file} присутствует'));
            resTest += 'Файл ${file} присутствует.\n';
        }
        else
        {
            console.log(chalk.red('Файл ${file} отсутствует'));
            process.exit(1);
        }
    }
}

```

### 3.3.2 Тест вывода из программы

5b  $\langle$ проверка правильности вывода из программы 5b $\rangle \equiv$  (4b)

```

await testOutput(test);

```

6a  $\langle \text{реализация проверки правильности вывода из программы 6a} \rangle \equiv$  (9)

```

async function testOutput(test)
{
    const { stdout, stderr } = await exec(`${test.run}`);
    if(!stdout.localeCompare(test.output))
    {
        console.log(chalk.green('Вывод из программы верный'));
        resTest += '# Вывод из программы верный\n';
    }
    else
    {
        console.log(chalk.red('Ошибка'));
        resTest += "# Ошибка\n";
        console.log(chalk.red('Программа выдаёт'));
        resTest += '# Программа выдаёт\n';
        resTest += '<pre>\n';
        resTest += stdout.toString();
        resTest += '</pre>\n';
        console.log(stdout);
        console.log(stderr);
        console.log(chalk.red('Правильный вывод'));
        resTest += '# Правильный вывод\n';
        resTest += "<pre>";
        resTest += test.output;
        resTest += "</pre>"
        console.log(test.output);
        console.log(stderr);
    }
}

```

Uses `exec` 8a.

## 3.4 Отправка на проверку

6b  $\langle \text{отправка на проверку 6b} \rangle \equiv$  (2a)

$\langle \text{создание временного каталога 6c} \rangle$   
 $\langle \text{клонирование каталога 7b} \rangle$

### 3.4.1 Создание временного каталога

6c  $\langle \text{создание временного каталога 6c} \rangle \equiv$  (6b)

```

await createDir(test);

```

7a  $\langle \text{реализация создания временного каталога } 7a \rangle \equiv$  (9)

```

async function createDir(test)
{
    await rm('/tmp/test', {force: true, recursive: true});
    console.log("Folder /tmp/test deleted");
    await mkdir('/tmp/test', {recursive: true});
    console.log("Folder create dir");
}

```

### 3.4.2 Клонирование каталога

7b  $\langle \text{клонирование каталога } 7b \rangle \equiv$  (6b)

```

    await cloneRepo(orgTest, psetName, test);

```

Uses orgTest 8b.

7c  $\langle \text{реализация клонирования каталога } 7c \rangle \equiv$  (9)

```

async function cloneRepo(org, psetName, test)
{
    try{
        var username = readlineSync.question('Github username:');
        const origin = `git@github.com:${org}/${username}.git`;
        // const origin = `https://${username}@github.com/${org}/${username}`;
        const {stdout, stderr} = await exec(`git clone ${origin} -b main`);
        console.log(stdout);
        console.log(stderr);
        await exec(`git checkout -b ${psetName}`, {cwd: '/tmp/test'})
        for(let file of test.files)
        {
            await copyFile(path.join(process.cwd(), file), path.join('/tmp/test', file));
        }
        const fileName = `/tmp/test/${psetName.split('/')[1]}.md`;
        console.log(fileName);
        await writeFile(fileName, resTest);
        await exec(`git config user.name test && git config user.email test@example.com`);
        await exec(`git push -f ${origin} ${psetName}`, {cwd: '/tmp/test'})
    }catch(error)
    {
        console.error(error)
    }
}

```

Uses exec 8a.

## 3.5 Загрузка модулей

8a  $\langle \text{загрузка модулей } 8a \rangle \equiv$  (2a)

```
import chalk from 'chalk';
import urlExist from 'url-exists-nodejs';
import fetch from 'node-fetch';
import {readdir, rm, mkdir, copyFile, writeFile } from 'node:fs/promises';
import readlineSync from 'readline-sync';
import * as child from 'node:child_process';
import * as util from 'node:util';
const exec = util.promisify(child.exec);
import * as path from 'node:path';
```

Defines:

`exec`, used in chunks 6a and 7c.

## 3.6 Задание констант

`orgTest` - имя организации в которой репозиторий описания задания.

`orgCheck` - имя организации где сохраняем результат.

8b  $\langle \text{константы } 8b \rangle \equiv$  (2a)

```
const orgTest = 'cs204';
const orgCheck = 'cs204check';
var resTest = '';
var local = false;
```

Defines:

`orgCheck`, never used.

`orgTest`, used in chunks 3a, 4a, and 7b.

## 4 Установка в домашний каталог

Настроим на установку в каталог `/npm`.

8c  $\langle \text{установка } 8c \rangle \equiv$  8d>

```
mkdir ~/npm
npm config set prefix ~/npm
export PATH="$HOME/npm/bin:PATH"
```

Последняя строчка в `.profile`, чтобы система находила программы.  
Перезагрузить.

8d  $\langle \text{установка } 8c \rangle + \equiv$  <8c

```
npm -global install check204
```



## 4.1 Функции

$$\begin{aligned} 9 \quad \langle \text{функции } 9 \rangle \equiv & \quad (2a) \\ & \langle \text{реализация проверки аргументов } 3a \rangle \\ & \langle \text{реализация закичивания описания } 4a \rangle \\ & \langle \text{реализация проверки наличия файлов } 5a \rangle \\ & \langle \text{реализация проверки правильности вывода из программы } 6a \rangle \\ & \langle \text{реализация создания временного каталога } 7a \rangle \\ & \langle \text{реализация клонирования каталога } 7c \rangle \end{aligned}$$

## 4.2 Makefile

В этой части мы создаем Makefile, управляющий сборкой. Утилита make начинает читать этот файл с конца, поэтому такой порядок действий выбран в файле. Результат работы утилиты make – это текст check204.pdf, и файл – javascript программы check204, эти цели заданы в первой строке. Здесь используются утилиты noweave, notangle из пакета noweb, для генерации tex и js файлов соответственно, pdflatex генерирует pdf файл из tex файлов.

```
10 <Makefile 10>≡
    res: doc/check204.pdf  bin/check204

    bin/check204: src/main.nw
        notangle -Rcheck204.js src/main.nw > bin/check204

    doc/check204.pdf: src/check204.tex  tmp/main.tex
        pdflatex -output-directory=doc src/check204.tex
        pdflatex -output-directory=doc src/check204.tex

    # Утилита noweave генерирует файл main.tex из отдельных файлов
    # формата noweb.
    tmp/main.tex: src/main.nw  src/Makefile.nw
        noweave -n -latex -index -autodefs c  src/main.nw \
            src/Makefile.nw > tmp/main.tex

    # Эту команду выполняем в случае, если были изменения в файле
    # Makefile.nw, notangle генерирует новый Makefile

    new: src/Makefile.nw
        notangle -t8 -RMakefile src/Makefile.nw > Makefile

    view:
        evince doc/check204.pdf &

    clean:
        rm doc/check204.pdf bin/check204 tmp/main.tex
```

# Содержание

<b>1</b>	<b>Анализ задания</b>	<b>1</b>
1.1	Описание проверки . . . . .	1
1.2	Запуск программы . . . . .	2
<b>2</b>	<b>Структура</b>	<b>2</b>
<b>3</b>	<b>Реализация</b>	<b>2</b>
3.1	Проверка аргументов . . . . .	2
3.2	Закачивание описания . . . . .	3
3.3	Тестирование программы . . . . .	4
3.3.1	Проверка наличия файлов . . . . .	4
3.3.2	Тест вывода из программы . . . . .	5
3.4	Отправка на проверку . . . . .	6
3.4.1	Создание временного каталога . . . . .	6
3.4.2	Клонирование каталога . . . . .	7
3.5	Загрузка модулей . . . . .	8
3.6	Задание констант . . . . .	8
<b>4</b>	<b>Установка в домашний каталог</b>	<b>8</b>
4.1	Функции . . . . .	9
4.2	Makefile . . . . .	10