



**МИНИСТЕРСТВО ЦИФРОВЫХ ТЕХНОЛОГИЙ
РЕСПУБЛИКИ УЗБЕКИСТАН**

**ТАШКЕНТСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ
МУХАММАДА АЛЬ-ХОРЕЗМИ**

САМОСТОЯТЕЛЬНАЯ РАБОТА № 2

По предмету

Система реального времени

Выполнил(а): __Примов Шохзод__

Проверил(а): __Караханова Ширин__

Ташкент – 2025

Задание 2: Работа с процессами, потоками и ресурсами

Описание задания:

Создайте приложение, в котором:

- запускается несколько потоков и процессов,
- происходит совместный доступ к общим ресурсам (например, файлам или памяти),
- реализуется защита доступа с помощью механизмов синхронизации (семафоры, мьютексы),
- используется механизм общей памяти для обмена данными между процессами.

Модули/темы: A4, A10, A11, A13

Node.js Пример: Потоки, Процессы и Общая Память

Описание проекта

Данный проект демонстрирует, как в Node.js можно организовать:

- **Многопоточность** с использованием `worker_threads`.
- **Многопроцессность** с использованием `child_process.fork`.
- **Совместный доступ к ресурсам**: потокам и процессам доступна общая память или файл.
- **Синхронизацию доступа** с помощью атомарных операций (`Atomics`) для потоков и блокировок (`proper-lockfile`) для процессов.
- **Обмен данными между процессами** через общий файл (`shm.txt`) без использования нативных модулей.

Проект работает **в командной строке**, не требует Visual Studio или сборки нативных модулей.

Структура проекта

project-root/

└─ package.json # зависимости и скрипты

└─ index.js # главный процесс

└─ worker_thread.js # код потоков (worker_threads)

└─ child_worker.js # код дочерних процессов (fork)

└─ shm.txt # файл для совместного доступа между процессами

Работа потоков (worker_threads)

- Потоки создаются внутри одного процесса Node.js.
- Используется **SharedArrayBuffer** + **Atomics** для безопасного инкремента счетчика.
- Потоки сообщают главному процессу о завершении работы через `parentPort.postMessage`.

Пример кода:

```
const { parentPort, workerData } = require('worker_threads');  
  
const view = new Int32Array(workerData.shared);  
  
for (let i = 0; i < 50; i++) {  
  Atomics.add(view, 0, 1);  
}  
  
parentPort.postMessage({ done: true });
```

Работа процессов (child_process.fork)

- Дочерние процессы запускаются как отдельные Node.js процессы.
- Для обмена данными используется **файл shm.txt**.
- Для безопасной записи используется **блокировка файла** через `proper-lockfile`.

Пример кода:

```
const fs = require('fs');  
  
const lockfile = require('proper-lockfile');  
  
await lockfile.lock(SHM_FILE);  
  
fs.writeFileSync(SHM_FILE, `PID:${process.pid} -i:${i}\n`, 'utf8');  
  
await lockfile.unlock(SHM_FILE);
```

Главный процесс (index.js)

- Создаёт потоки и процессы.
- Отслеживает их сообщения (message) и выводит текущее состояние:
 - Счётчик потоков (SharedArrayBuffer).
 - Содержимое файла процессов.

Пример чтения состояния:

```
setInterval(() => {  
  
  const counter = Atomics.load(sharedView, 0);  
  
  const fileContent = fs.readFileSync(SHM_FILE, 'utf8');  
  
  console.log('SharedArrayBuffer counter:', counter);  
  
  console.log('File content:', fileContent.trim());  
  
}, 2000);
```

- Опционально можно завершать программу автоматически, когда все потоки и процессы закончат работу.

Особенности реализации

1. **Потоки** используют атомарные операции через `Atomics` для синхронизации без блокировок.
2. **Процессы** используют файл и блокировки через `proper-lockfile`, чтобы избежать коллизий при записи.
3. **Общая память** между процессами реализована через обычный файл (`shm.txt`), без нативных модулей.
4. Проект работает на **Windows, Linux и Mac**.

```
Parent got from child: { wrote: 'PID:8688-i:0-2025-11-11T15:27:29.388Z' }
Parent got from child: { wrote: 'PID:8688-i:1-2025-11-11T15:27:30.039Z' }
Parent got from child: { wrote: 'PID:724-i:0-2025-11-11T15:27:30.397Z' }
Parent got from child: { wrote: 'PID:8688-i:2-2025-11-11T15:27:30.584Z' }
Parent got from child: { wrote: 'PID:8688-i:3-2025-11-11T15:27:30.959Z' }
SharedArrayBuffer counter: 100
File content: PID:8688-i:3-2025-11-11T15:27:30.959Z
Parent got from child: { wrote: 'PID:724-i:1-2025-11-11T15:27:31.349Z' }
Parent got from child: { wrote: 'PID:8688-i:4-2025-11-11T15:27:31.719Z' }
Parent got from child: { wrote: 'PID:8688-i:5-2025-11-11T15:27:32.292Z' }
Parent got from child: { wrote: 'PID:724-i:2-2025-11-11T15:27:32.338Z' }
Parent got from child: { wrote: 'PID:8688-i:6-2025-11-11T15:27:32.725Z' }
Parent got from child: { wrote: 'PID:724-i:3-2025-11-11T15:27:32.833Z' }
Parent got from child: { wrote: 'PID:8688-i:7-2025-11-11T15:27:33.111Z' }
SharedArrayBuffer counter: 100
File content: PID:8688-i:7-2025-11-11T15:27:33.111Z
Parent got from child: { wrote: 'PID:724-i:4-2025-11-11T15:27:33.547Z' }
Parent got from child: { wrote: 'PID:8688-i:8-2025-11-11T15:27:33.699Z' }
Parent got from child: { wrote: 'PID:724-i:5-2025-11-11T15:27:34.261Z' }
Parent got from child: { wrote: 'PID:8688-i:9-2025-11-11T15:27:34.587Z' }
Parent got from child: { wrote: 'PID:724-i:6-2025-11-11T15:27:34.850Z' }
```

Запуск проекта:

Перейдите по ссылке (<https://github.com/HTML-CSS-Dev/processes-potoki-resources.git>)

Там есть зеленная кнопка с названием <code> и там скачайте zip формат и нажмите на run.bat она автоматически запускается в командной строке.