



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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
***К КУРСОВОЙ РАБОТЕ***  
***НА ТЕМУ:***

***«Разработка статического сервера»***

Студент      ИУ7-76Б      \_\_\_\_\_ Ковель А.Д.

Руководитель курсовой работы      \_\_\_\_\_

2023 г.

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)

---

УТВЕРЖДАЮ

Заведующий кафедрой ИУ-7

И. В. Рудаков

«16» сентября 2023 г.

**ЗАДАНИЕ**  
**на выполнение курсовой работы**

по теме

**«Разработка статического сервера»**

Студент группы **ИУ7-76Б**

**Ковель Александр Денисович**

Направленность КР

**учебная**

Источник тематики

**Курсовая работа кафедры**

График выполнения КР: 25% к 6 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

***Техническое задание***

*Разработать статический веб-сервер для отдачи контента с диска. В качестве мультиплексора использовать poll. Сервер должен реализовывать многопоточную обработку запросов с использованием пула потоков.*

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на **12-20** листах формата А4.

Дата выдачи задания «16» сентября 2023 г.

**Руководитель курсовой работы**

\_\_\_\_\_  
(Подпись, дата)

\_\_\_\_\_  
(Фамилия И. О.)

**Студент**

\_\_\_\_\_  
(Подпись, дата)

**Ковель А. Д.**  
(Фамилия И. О.)

# СОДЕРЖАНИЕ

|   |           |
|---|-----------|
| <b>ВВЕДЕНИЕ</b>                             | <b>4</b>  |
| <b>1 Аналитические раздел</b>               | <b>5</b>  |
| 1.1 Thread pool . . . . .                   | 5         |
| 1.2 Сокет poll . . . . .                    | 5         |
| <b>2 Конструкторская часть</b>              | <b>8</b>  |
| 2.1 Разработка алгоритмов . . . . .         | 8         |
| <b>3 Технологическая часть</b>              | <b>10</b> |
| 3.1 Требования к программе . . . . .        | 10        |
| 3.2 Средства реализации . . . . .           | 10        |
| 3.3 Реализация сокета poll . . . . .        | 10        |
| 3.4 Реализация пуллинг потоков . . . . .    | 12        |
| <b>4 Исследовательская часть</b>            | <b>13</b> |
| 4.1 Технические характеристики . . . . .    | 13        |
| 4.2 Демонстрация работы программы . . . . . | 13        |
| 4.3 Результаты исследования . . . . .       | 13        |
| 4.4 Вывод . . . . .                         | 14        |
| <b>ЗАКЛЮЧЕНИЕ</b>                           | <b>15</b> |
| <b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>     | <b>16</b> |

## ВВЕДЕНИЕ

**Цель работы** — разработка написать статический сервер для отдачи файлов с диска на основе пула потоков и сокета poll.

Для достижения поставленной цели, необходимо решить следующие задачи:

- 1) формализовать задачу;
- 2) определить структуры, связанные с поставленной задачей;
- 3) разработать алгоритм poll сокета;
- 4) реализовать программное обеспечение;
- 5) провести исследование скорости работы статического сервера.

# 1 Аналитические раздел

В данном разделе рассмотрены технологии thread pool и сокет poll.

## 1.1 Thread pool

Параллельные вычисления — это тип вычислений, при котором одновременно выполняется множество операций или процессов. Пул потоков (Thread pool) [1] — это фиксированный набор потоков, одновременно выполняющих независимые друг от друга задачи, помещенные в массив. Массив задач представляется в виде очереди. Ключевым аспектом логики пула потоков является факт того, что все потоки запускаются один раз.

Число операций, которые можно поставить в очередь в пуле потоков, ограничено только доступной памятью. Однако пул потоков имеет ограничение на число потоков, которые можно активировать в процессе одновременно. Если все потоки в пуле заняты, дополнительные рабочие элементы помещаются в очередь и ожидают их освобождения.

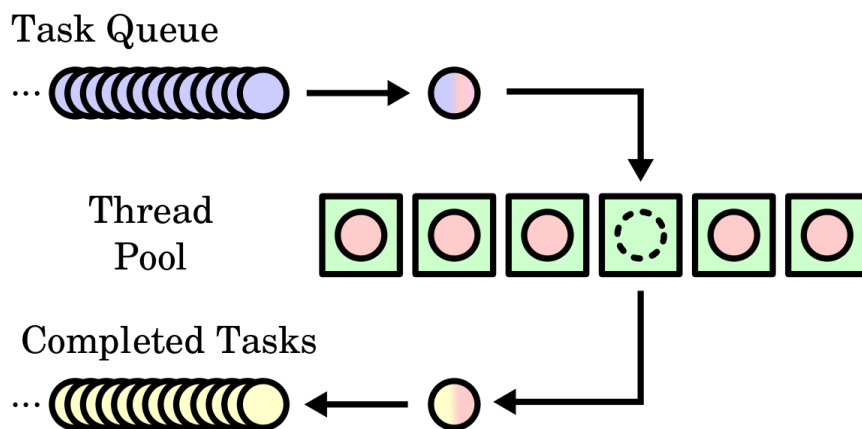


Рисунок 1 – Thread pool

## 1.2 Сокет poll

Сокеты — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных, связанных между собой сетью. Сокет — это абстрактный объект, представляющий конечную точку соединения.

Каждый сокет имеет свой адрес. ОС семейства UNIX могут поддерживать много типов адресов, но обязательным являются INET-адрес и UNIX-адрес.

Poll — это метод опроса сокетов, созданный после того как ресурс select оказался недостаточен. Преимуществами poll сокетов являются:

- 1) нет никакого лимита количества наблюдаемых дескрипторов;
- 2) не модифицируется структура pollfd, что дает возможность ее переиспользования между вызовами poll();
- 3) для идентификации отключения клиента, нет необходимости чтения данных из сокета.

Сокеты poll используют блокировку ввода-вывода с мультиплексированием, алгоритм работы представлен на рисунке 2.

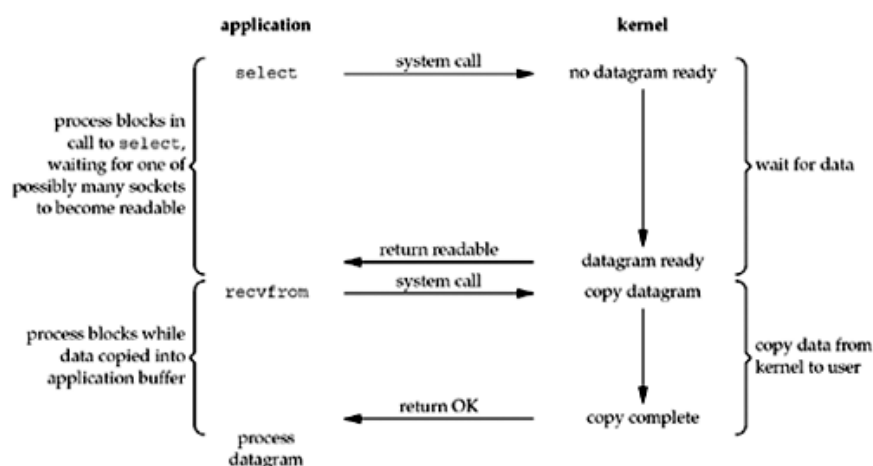


Рисунок 2 – Ввод-вывод с мультиплексированием

Основной функцией сокетов является:

### Листинг 1 – Функция poll

```
#include <poll.h>

int poll (struct pollfd *fdarray, unsigned long nfds, int timeout);

/* Returns: count of ready descriptors, 0 on timeout, -1 on error */
```

Первым аргументом этой функции является структура pollfd:

### Листинг 2 – Структура pollfd

```
struct pollfd {  
    int     fd;          /* descriptor to check */  
    short   events;      /* events of interest on fd */  
    short   revents;     /* events that occurred on fd */  
};
```

Проверяемые условия задаются членом `events`, а функция возвращает статус дескриптора в соответствующим члене `revents`. Такая структура данных (с двумя переменными на дескриптор, одна из которых является значением, а другая результатом) позволяет избежать аргументов «значение-результат». Каждые из этих двух членов состоит из одного или нескольких битов, которые задают определенное условие.

### **Вывод**

В данном разделе дано определение пула потоков и сокета `poll`. Также был представлен алгоритм их работы.

## 2 Конструкторская часть

В данном разделе представлены схемы алгоритмов сокетов poll и thread poll.

### 2.1 Разработка алгоритмов

На рисунке представлена схемка алгоритма сокетов poll 3.

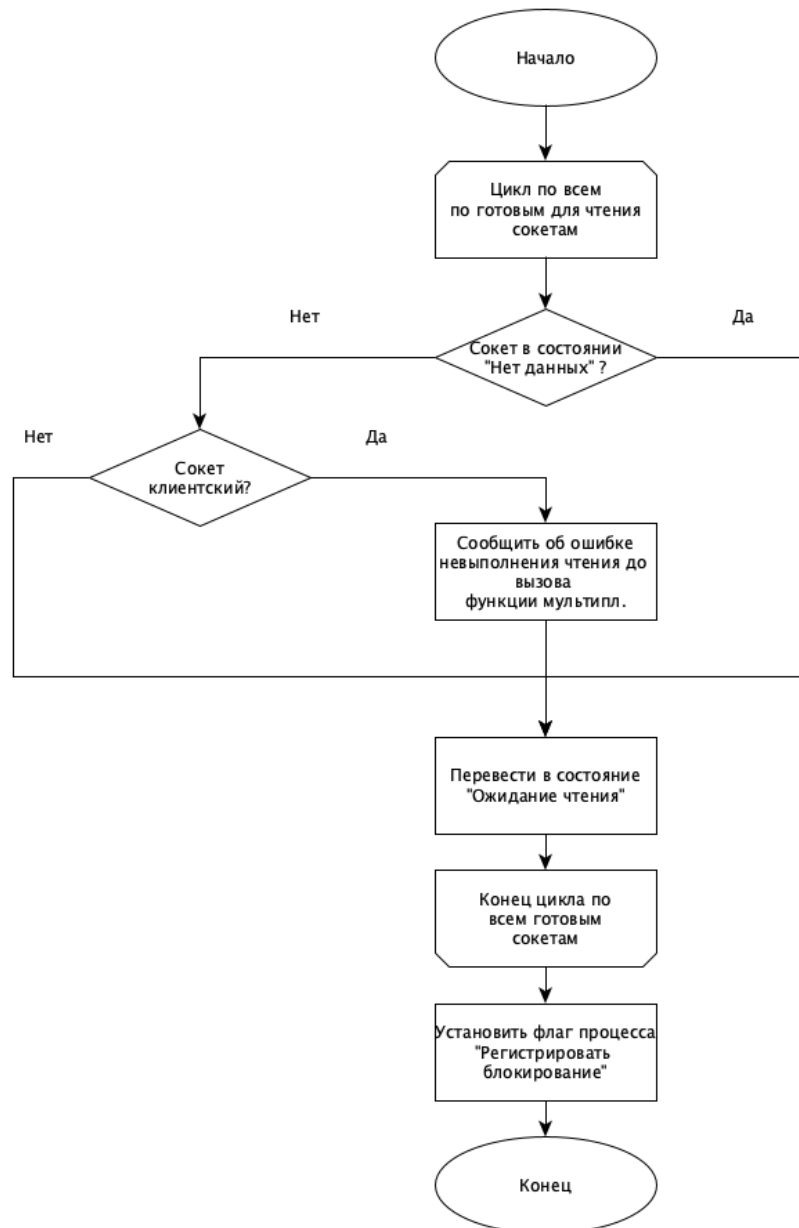


Рисунок 3 – Сокет poll

На рисунке 4 представлена схема алгоритма thread poll.



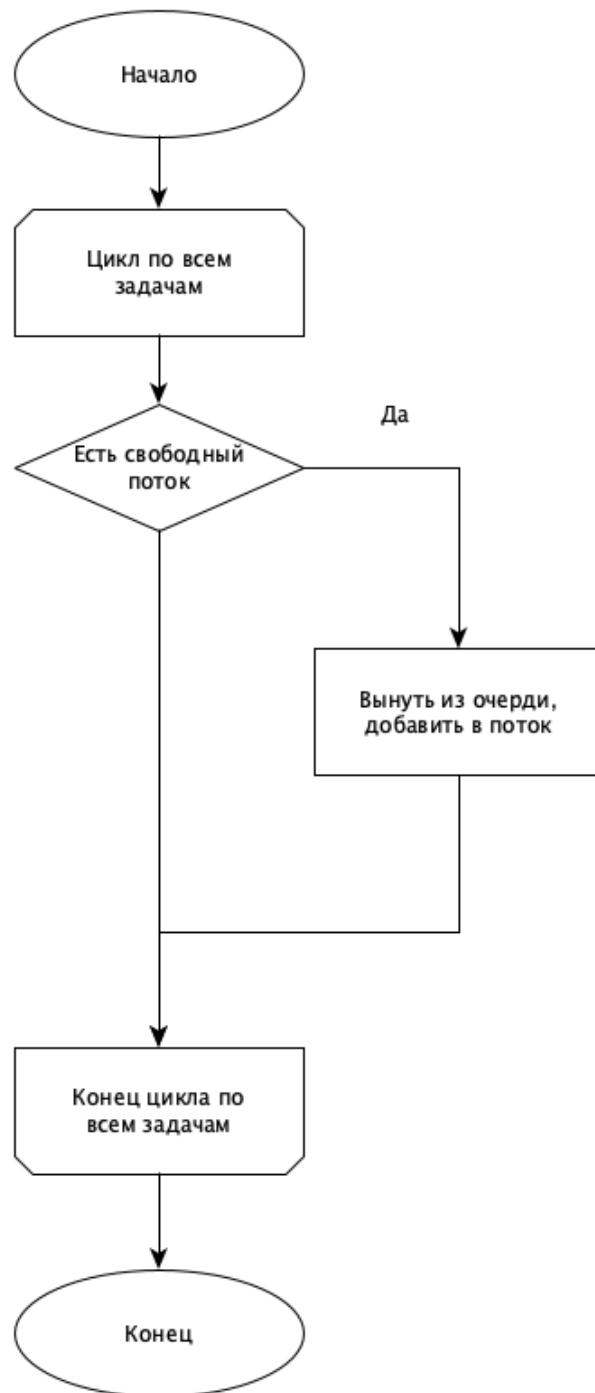


Рисунок 4 – Пул потоков

### Вывод

В данном разделе разработаны и представлены схемы алгоритмов сокетов poll и пул потоков.

### 3 Технологическая часть

В данном разделе представлены требования к программе и реализованы спроектированные методы.

#### 3.1 Требования к программе

Для того чтобы программное обеспечение удовлетворяло требованиям, необходимо определить их заранее и придерживаться их в процессе разработки. Программное обеспечение должно удовлетворять требованиям, которые необходимы для работы спроектированной системы:

- 1) поддержка запросов GET и HEAD;
- 2) корректная передача файлов размером в 100мб;
- 3) поддержка многопоточности.

#### 3.2 Средства реализации

Для реализации ПО был выбран язык C [2]. В данном языке есть все требующиеся инструменты для данной курсовой работы. В качестве среды разработки была выбрана среда Clion [3].

#### 3.3 Реализация сокета poll

Листинг 3 – Сокета poll

```
while (1)
{
    numfds = poll(server->clients, maxcl + 1, -1);
    if (numfds < 0)
    {
        LOG_ERROR("poll error");
        continue;
    }
    if (server->clients[0].revents & POLLIN)
    {
        int client_sock = accept(server->listen_sock, NULL, 0);
        if (client_sock < 0) continue;
        long i = 0;
        for (i = 1; i < server->cl_num; ++i)
        {
```

```

        if (server->clients[i].fd < 0)
        {
            server->clients[i].fd = client_sock;
            server->clients[i].events = POLLIN | POLLPRI;
            break;
        }
    }
    if (i == server->cl_num) {
        LOG_ERROR("too many connections");
        continue;
    }
    if (i > maxcl)
    {
        maxcl = i;
        LOG_INFO("Max clients: %d", maxcl);
    }
    if (--numfds <= 0)
    {
        continue;
    }
}
for (int i = 1; i <= maxcl; ++i)
{
    if (server->clients[i].fd >= 0 && server->clients[i].revents & (POLLIN |
        POLLERR))
    {
        worker_sock_t worker_sock;
        worker_sock.clientfd = &server->clients[i].fd;
        worker_sock.wd = server->wd;
        tpool_add_work(server->pool, worker, &worker_sock);

        if (--numfds < 0)
        {
            break;
        }
    }
}
tpool_wait(server->pool);
}

```

### 3.4 Реализация пуллинг потоков

#### Листинг 4 – Сокета poll

```
static void* tpool_worker(void* arg)
{
    tpool_t* tm = arg;
    tpool_work_t* work;

    while (1)
    {
        pthread_mutex_lock(&(tm->work_mutex));
        while (tm->work_first == NULL && !tm->stop)
            pthread_cond_wait(&(tm->work_cond), &(tm->work_mutex));
        if (tm->stop)
            break;
        work = tpool_work_get(tm);
        tm->working_cnt++;
        pthread_mutex_unlock(&(tm->work_mutex));
        if (work != NULL)
        {
            work->func(work->arg);
            tpool_work_destroy(work);
        }
        pthread_mutex_lock(&(tm->work_mutex));
        tm->working_cnt--;
        if (!tm->stop && tm->working_cnt == 0 && tm->work_first == NULL)
            pthread_cond_signal(&(tm->working_cond));
        pthread_mutex_unlock(&(tm->work_mutex));
    }
    tm->thread_cnt--;
    pthread_cond_signal(&(tm->working_cond));
    pthread_mutex_unlock(&(tm->work_mutex));
    return NULL;
}
```

#### Вывод

В данном разделе представлены требования к программе и реализованы спроектированные методы.

## 4 Исследовательская часть

### 4.1 Технические характеристики

Тестирование выполнялось на устройстве со следующими техническими характеристиками:

- 1) Операционная система Mac Os[4];
- 2) Оперативная память 16 Гбайт;
- 3) Процессор Mac M1 Pro [5].

Устройство было подключено к сети во времени исследования времени работы программы.

### 4.2 Демонстрация работы программы

На рисунке 5 представлен пример работы программы. Программа выводит тело запрашиваемой страницы. Страница была получена с диска.

```
ics7-cn-cw/src on  main [!+?] via desktop-linux via C v15.0.0-clang via v3.27.6
) curl -X GET http://0.0.0.0:9990/Users/akovel/Documents/bmstu/ics7-cn-cw/src/input/doc.html
<meta charset="utf-8">

<h1>Лабораторная работа 1</h1>

<h2>Название</h2>

<p>Псих-Админ</p>

<h2>Цель</h2>

<p>Данная система администрирования предназначена для работы с пациентами.
Проект создан для персонала психбольниц.
Данная система будет содержать информацию о пациентах, заболеваниях и плана лечения.</p>

<h2>Use-Case</h2>

<p></p>

<h2>ER</h2>

<p></p>

<h2>BD</h2>

<p></p>

<h2>Пользовательские сценарии</h2>

<p>Пользователь открывает клиент и может:</p>
```

Рисунок 5 – Демонстрация работы программы

### 4.3 Результаты исследования

На рисунке 6 представлен график сравнения времени запросов.

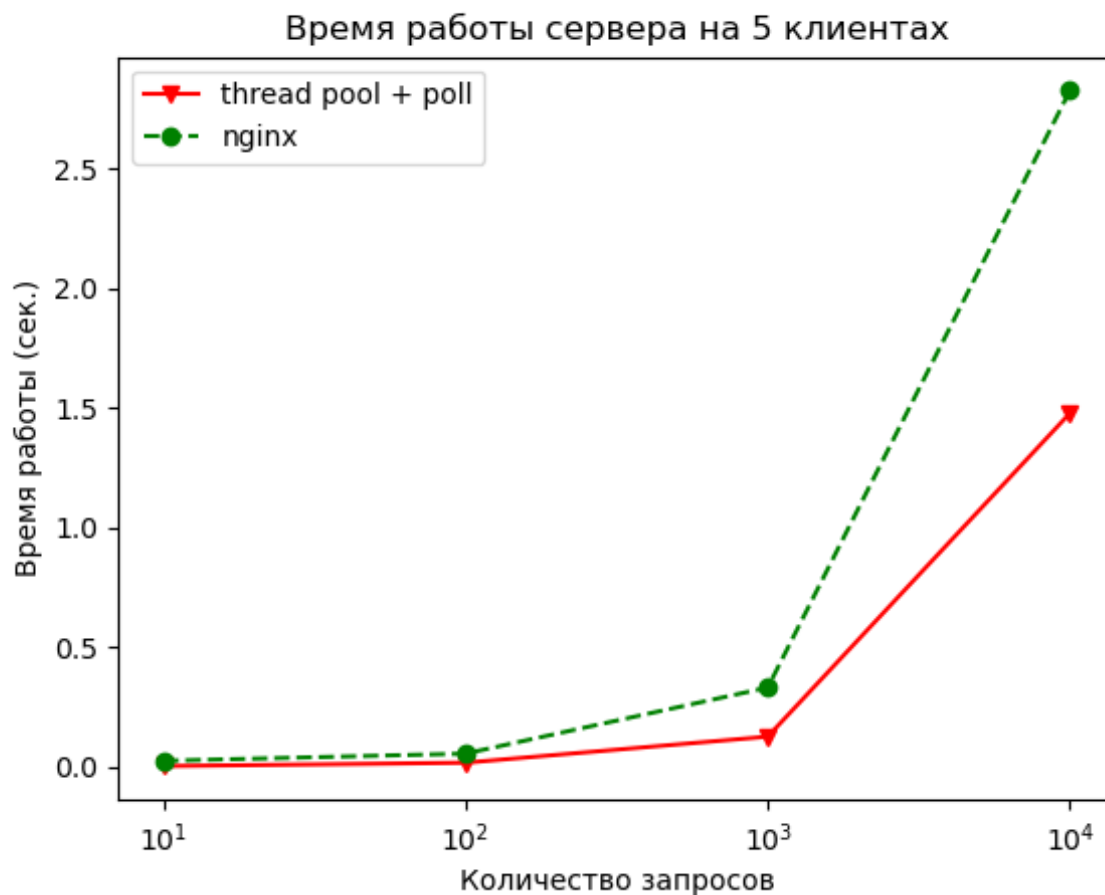


Рисунок 6 – Результаты времени запроса к серверу

Из графика можно сделать вывод, что nginx работает 2 раза медленнее на 5 клиентах.

#### 4.4 Вывод

В данном разделе был приведен анализ работы статического сервера.

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной работы были выполнены следующие задачи:

- 1) формализована задачу;
- 2) определены структуры, связанные с поставленной задачей;
- 3) разработан алгоритм poll сокета;
- 4) реализовано программное обеспечение;
- 5) проведено исследование скорости работы статического сервера.

Поставленная цель достигнута.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Thread Pools [Электронный ресурс]. Режим доступа: <https://www.iitk.ac.in/esc101/05Aug/tutorial/essential/threads/pool.html#:~:text=A%20thread%20pool%20is%20a,executing%20a%20collection%20of%20tasks> (дата обращения: 03.12.2023).
2. Язык программирования C [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ru-ru/cpp/c-language/?view=msvc-170>. дата обращения: 03.12.2023.
3. Vscode [Электронный ресурс]. <https://code.visualstudio.com/>. дата обращения: 03.12.2023.
4. Linux – Документация [Электронный ресурс]. Режим доступа: <https://docs.kernel.org> (дата обращения: 03.12.2023).
5. Процессор AMD® Ryzen 7 2700 eight-core processor × 16 [Электронный ресурс]. Режим доступа: <https://www.amd.com/en/products/cpu/amd-ryzen-7-2700> (дата обращения: 03.12.2023).