



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»

---

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К КУРСОВОЙ РАБОТЕ*

### *НА ТЕМУ:*

*«Разработка сервера для отдачи статического  
содержимого с диска»*

Студент ИУ7-71Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

Постнов С. А.  
(Фамилия И. О.)

Руководитель курсовой работы

\_\_\_\_\_  
(Подпись, дата)

Клочков М. С.  
(Фамилия И. О.)

*2024 г.*

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>3</b>
<b>1 Аналитический раздел</b>	<b>4</b>
1.1 Архитектура thread-pool . . . . .	4
1.2 Системный вызов pselect . . . . .	5
<b>2 Конструкторский раздел</b>	<b>7</b>
2.1 Схема алгоритма работы сервера . . . . .	7
<b>3 Технологический раздел</b>	<b>8</b>
3.1 Требования к разрабатываемой программе . . . . .	8
3.2 Средства реализации . . . . .	8
3.3 Примеры работы программы . . . . .	8
<b>4 Исследовательский раздел</b>	<b>11</b>
4.1 Технические характеристики . . . . .	11
4.2 Описание исследования . . . . .	11
4.3 Результаты исследования . . . . .	12
<b>ЗАКЛЮЧЕНИЕ</b>	<b>13</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>14</b>

## ВВЕДЕНИЕ

Статический веб-сервер — сервер, который обслуживает статические файлы по запросу клиента. Статические файлы — файлы, содержимое которых не изменяется динамически на стороне сервера. Статический сервер предназначен для чтения файлов из диска и отправления их клиенту, как правило, по протоколу HTTP.

Целью курсовой работы является разработка сервера для отдачи статического содержимого с диска. Архитектура сервера должна быть основана на пуле потоков (thread-pool) совместно `cpselect()`.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) описать предметную область;
- 2) спроектировать схему алгоритма работы сервера;
- 3) выбрать средства реализации сервера;
- 4) провести сравнение реализованного сервера с известными аналогами.

# 1 Аналитический раздел

## 1.1 Архитектура thread-pool

Архитектура пул потоков (thread-pool) — модель управления потоками выполнения (threads), которая предусматривает предварительное создание фиксированного или динамически изменяемого пула потоков, используемых для обработки задач из общей очереди. Такая архитектура обеспечивает эффективное распределение вычислительных ресурсов и уменьшение накладных расходов, связанных с созданием и уничтожением потоков. Она является одной из самых распространенных архитектур многопоточности Object Request Broker Architecture (CORBA), используемых в реализациях Object Request Broker (ORB), и была принята веб-серверами, такими как Microsoft Internet Information Server (IIS) [1].

Основными компонентами архитектуры thread-pool являются:

- 1) пул потоков (thread-pool);
- 2) очередь задач (task queue);
- 3) менеджер пула (pool manager);
- 4) задачи (tasks).

Архитектура пула потоков представлена на рисунке 1.1.

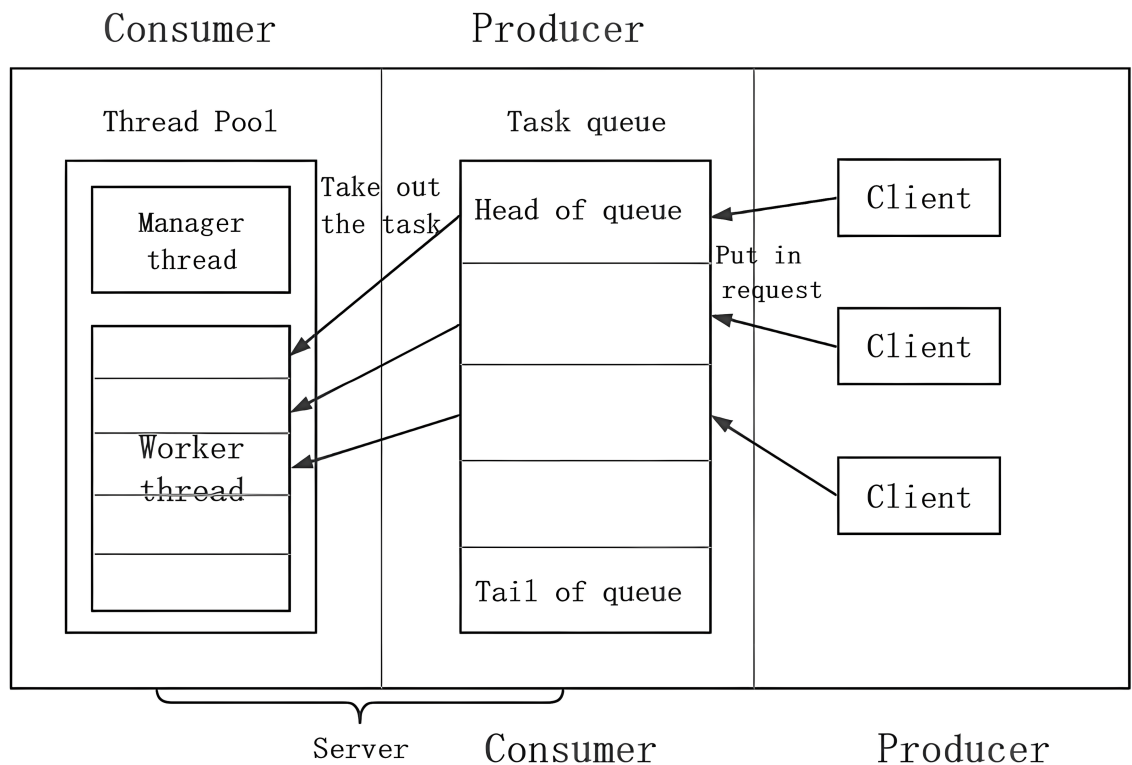


Рисунок 1.1 – Архитектура пула потоков

## 1.2 Системный вызов pselect

В Unix процесс выполняет ввод-вывод по одному файловому дескриптору за раз, поэтому происходит блокировка и снижение производительности программы. Чтобы избежать этой проблемы, необходимо использовать системный вызов **pselect()**, который позволяет программе отслеживать несколько файловых дескрипторов. Программа ожидает, пока один или несколько файловых дескрипторов не станут готовы к определённому классу операций ввода-вывода, не блокируя их и обеспечивая многопоточный синхронный ввод-вывод [2].

Модель неблокирующего синхронного ввода-вывода, в которой применяется `pselect()`, представлена на рисунке 1.2.

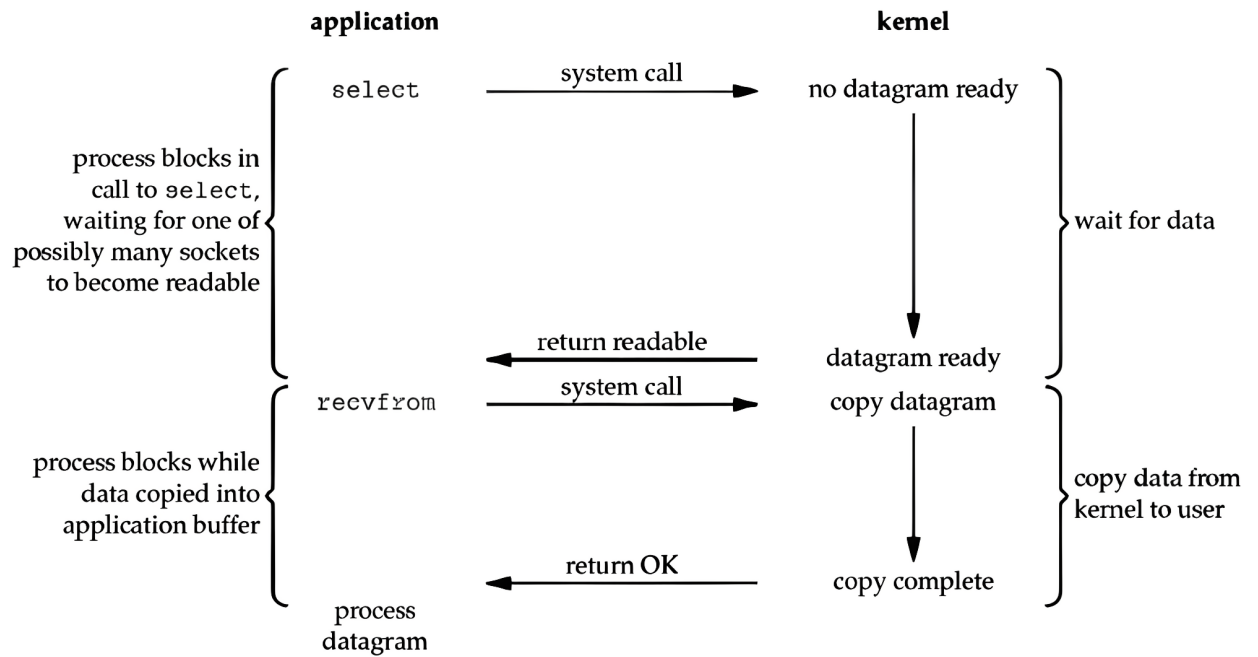


Рисунок 1.2 – Модель неблокирующего синхронного ввода-вывода

## 2 Конструкторский раздел

### 2.1 Схема алгоритма работы сервера

На рисунке 2.1 представлена схема алгоритма работы сервера.



Рисунок 2.1 – Схема алгоритма работы сервера

## 3 Технологический раздел

### 3.1 Требования к разрабатываемой программе

Разрабатываемое программное обеспечение должно удовлетворять следующим требованиям:

- 1) поддержка запросов GET и HEAD;
- 2) поддержка статусов 200, 403, 404 и 405 (на неподдерживаемые запросы);
- 3) поддержка корректной передачи файлов размером до 128 Мб;
- 4) возврат по умолчанию html-страницы с css-стилем;
- 5) запись информации о событиях;
- 6) минимальные требования к безопасности серверов статического содержимого.

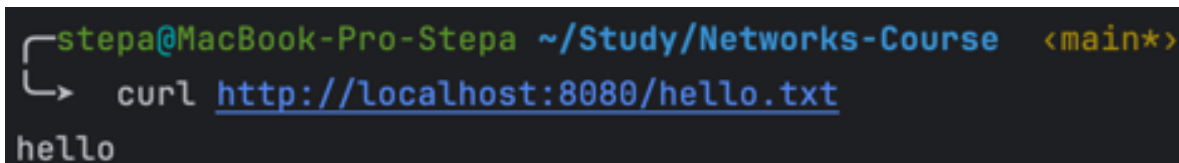
### 3.2 Средства реализации

В качестве языка программирования для реализации веб-сервера был выбран язык C [3].

В качестве среды для разработки была выбрана среда CLion [4].

### 3.3 Примеры работы программы

На рисунке 3.1 представлен пример ответа на GET-запрос.



```
stepa@MacBook-Pro-Stepa ~/Study/Networks-Course <main*>  
└─┬ curl http://localhost:8080/hello.txt  
hello
```

Рисунок 3.1 – Пример ответа на GET-запрос



На рисунке 3.2 представлен пример ответа на HEAD-запрос.

```
stepa@MacBook-Pro-Stepa ~/Study/Networks-Course <main*>  
└─> curl -I http://localhost:8080/index.html  
HTTP/1.1 200 OK  
Content-Length: 9336  
Content-Type: text/html  
Connection: close
```

Рисунок 3.2 – Пример ответа на HEAD-запрос

На рисунке 3.3 представлен пример ответа на GET-запрос несуществующего файла.

```
stepa@MacBook-Pro-Stepa ~/Study/Networks-Course <main*>  
└─> curl -I http://localhost:8080/indexxx.html  
HTTP/1.1 404 Not Found  
Content-Length: 22  
Content-Type: text/html  
Connection: close
```

Рисунок 3.3 – Пример ответа на GET-запрос несуществующего файла

На рисунке 3.4 представлен пример ответа на неразрешенный POST-запрос.

```
stepa@MacBook-Pro-Stepa ~/Study/Networks-Course <main*>  
└─> curl -X POST http://localhost:8080/1.png  
<h1>405 Method Not Allowed</h1>%
```

Рисунок 3.4 – Пример ответа на неразрешенный POST-запрос

На рисунке 3.5 представлен пример ответа на GET-запрос без прав на доступ.

```
stepa@MacBook-Pro-Stepa ~/Study/Networks-Course <main*>
└─> chmod 0 static/hello.txt

stepa@MacBook-Pro-Stepa ~/Study/Networks-Course <main*>
└─> curl http://localhost:8080/hello.txt
<h1>403 Forbidden</h1>%
```

Рисунок 3.5 – Пример ответа на GET-запрос без прав на доступ

На рисунке 3.6 представлен пример логирования событий в системе.

```
stepa@MacBook-Pro-Stepa ~/Study/Networks-Course <main*>
└─> cat server.log
[Sun Dec 8 20:39:45 2024] Server starting on :8080
[Sun Dec 8 20:39:59 2024] Request: GET on /
[Sun Dec 8 20:39:59 2024] Response: 200 OK
[Sun Dec 8 20:40:19 2024] Request: GET on /
[Sun Dec 8 20:40:19 2024] Response: 200 OK
[Sun Dec 8 20:40:23 2024] Request: GET on /hello.txt
[Sun Dec 8 20:40:23 2024] Response: 200 OK
[Sun Dec 8 20:40:41 2024] Request: GET on /hello.txt
[Sun Dec 8 20:40:41 2024] Response: 200 OK
[Sun Dec 8 20:40:49 2024] Request: GET on /hello.txt
[Sun Dec 8 20:40:49 2024] Response: 200 OK
[Sun Dec 8 20:41:23 2024] Request: HEAD on /1.png
[Sun Dec 8 20:41:23 2024] Response: 200 OK
[Sun Dec 8 20:41:32 2024] Request: HEAD on /index.html
[Sun Dec 8 20:41:32 2024] Response: 200 OK
[Sun Dec 8 20:42:03 2024] Request: HEAD on /indexxx.html
[Sun Dec 8 20:42:03 2024] Response: 404 Not Found
```

Рисунок 3.6 – Пример логирования событий в системе

## 4 Исследовательский раздел

В данном разделе будут представлены технические характеристики, а также будет проведено нагрузочное тестирование разработанного программного обеспечения и сравнение значений с известным аналогом.

### 4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- операционная система: Ubuntu 22.04.4 LTS [**ubuntu**];
- объем оперативной памяти: 8 Гбайт;
- процессор: AMD Ryzen 5 5500U – 2.10 ГГц [**amd**].

Во время тестирования ноутбук был подключен к сети электропитания. Также ноутбук был нагружен только встроенными приложениями и системой тестирования.

### 4.2 Описание исследования

Для замеров метрик производительности сравниваемых веб - серверов использовалась утилита Apache Benchmark (ab) [**ab**]. Для сравнения производительности был выбран nginx — это веб - сервер, созданный работать под высокой нагрузкой, чаще всего используемый для отдачи статического контента [**nginx**]. Конфигурация nginx представлена ниже 4.1.

Листинг 4.1 – Конфигурация nginx

```
server {  
    listen 80;  
  
    location / {  
        root /etc/nginx/static;  
        index index.html;  
    }  
}
```

### 4.3 Результаты исследования

Для оценки производительности измерялось общее время обработки запросов в секундах. Замеры времени производились на следующих значениях: 10, 50, 100, 500, 1000, 50000, 10000. Количество параллельных запросов: 6, что соответствует числу ядер ноутбука, на котором производилось тестирование.

Результаты нагрузочного тестирования разработанного сервера и nginx представлены в таблице 4.1.

Таблица 4.1 – RPS при обработке 10000 запросов на файл размером 2 Кб

Число запросов	Разработанный сервер	nginx
10	414.97	406.74
50	1878.45	1823.23
100	1798.11	1155.56
500	1798.11	1155.56
1000	1798.11	1155.56
5000	1798.11	1155.56
10000	1798.11	1155.56

### Выводы

## ЗАКЛЮЧЕНИЕ

Цель курсовой работы — разработать сервер для отдачи статического содержимого с диска — была достигнута.

Для достижения поставленной цели были выполнены следующие задачи:

- описана предметная область;
- формализованы требования к серверу;
- спроектирована структура программного обеспечения;
- выбраны средства реализации и реализовано программное обеспечение;
- проведено нагрузочное тестирование разработанного сервера и сравнение с известным аналогом.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Yibei Ling, Tracy Mullen, Xiaola Lin*. Analysis of Optimal Thread Pool Size. — 2000.
2. MAN pselect. — [Электронный ресурс]. — Режим доступа: <https://www.opennet.ru/man.shtml?topic=pselect&category=2&russian=0> (дата обращения: 23.11.24).
3. C Language Reference. — [Электронный ресурс]. — Режим доступа: <https://learn.microsoft.com/en-us/cpp/c-language/c-language-reference?view=msvc-170> (дата обращения: 24.11.24).
4. A cross-platform IDE for C and C++. — [Электронный ресурс]. — Режим доступа: <https://www.jetbrains.com/clion/> (дата обращения: 24.11.24).