**Лабораторная работа №4,** «Создание HTTP-сервера»  
Группа: Б23-534  
ФИО: Калашников Владимир Алексеевич  
Номер в журнале: 5  
Год в Москве: 2025

**Описание лабораторной работы**

[Ссылка на GitHub](https://github.com/SD5th/Informatika_Labs_MIFI/tree/main/4Sem/LW_4sem-4)

Работа выполнена на языке Си и представляет из себя файл *main.c*, в котором прописана работа HTTP-сервера.

Также проект оснащён небольшим скриптиком *Build.sh* для сборки с помощью CMake и последующего запуска оболочки. И favicon.ico, если кто-то решит обратиться к серверу с помощью браузера.

**Функция main**

int main(int argc, char\* argv[]) {

  struct sigaction sa;

  sa.sa\_handler = handle\_signal;

  sigemptyset(&sa.sa\_mask);

  sa.sa\_flags = 0;

  sigaction(SIGINT, &sa, NULL);

  sigaction(SIGTERM, &sa, NULL);

  init\_server(argc, argv);

  init\_server\_dir();

  char buffer[BUFFER\_SIZE];

  while (KEEP\_RUNNING == 1) {

    int client\_fd = accept(SERVER\_FD, NULL, NULL);

    if (client\_fd < 0) {

      if (KEEP\_RUNNING != 1) {

        break;

      }

      perror("accept");

      continue;

    }

    ssize\_t n = recv(client\_fd, buffer, sizeof(buffer)-1, 0);

    if (n > 0) {

      buffer[n] = '\0';

      printf("\n\nReceived:\n%s\n", buffer);

      if (strncmp(buffer, "GET /", 5) == 0) {

        char \*path\_start = buffer + 5;

        char \*path\_end = strchr(path\_start, ' ');

        if (path\_end && path\_start != path\_end) {

          \*path\_end = '\0';

          char requested\_path[PATH\_SIZE];

          strcpy(requested\_path, SERVER\_DIR);

          strcat(requested\_path, path\_start);

          printf("Requested file: '%s'\n", path\_start);

          printf("Requested path: '%s'\n", requested\_path);

          send\_file(client\_fd, requested\_path);

        }

        else {

          send\_response(client\_fd, "404 Not Found", "text/plain", "404 Not Found\n");

        }

      }

      else {

        send\_response(client\_fd, "400 Bad Request", "text/plain", "Only GET method is supported\n");

      }

    }

    close(client\_fd);

  }

  close(SERVER\_FD);

  return 0;

}

Первый блок отвечает за переопределение сигналов killpid и ctrl+c. При их вызове системный вызов accept() будет прерван, переменная KEEP\_RUNNING приравняется к нулю функцией handle\_signal, а программа завершится.

Следующими идут инициализации сервера и рабочей папки.

Затем цикл, принимающий запрос и перенаправляющий его в функции обработки.

**Остальные функции слишком велики,**

..чтобы их сюда вставить. Увидеть их можно по [Ссылке на GitHub](https://github.com/SD5th/Informatika_Labs_MIFI/tree/main/4Sem/LW_4sem-4).

void handle\_signal(int sig);

Вызывается при срабатывании вышеописанных сигналов. Ставит переменную KEEP\_RUNNING в единицу.

void init\_server(int argc, char\* argv[]);

Проверяет argv и argc, создаёт сокет и привязывает его к порту из argv.

void init\_server\_dir();

Вычисляет директорию, из которой запустился процесс.

void send\_response(int client\_fd, const char \*status, const char \*content\_type,

const char \*body);

Отправляет ответик в формате HTTP 1.1

void send\_file(int client\_fd, const char \* requested\_path);

Проверяет, существует ли файл, можно ли к нему обеспечить доступ, отправляет его.