# Модуль ngx_http_fastcgi_module

Модуль `ngx_http_fastcgi_module` позволяет передавать запросы FastCGI-серверу.

## Пример конфигурации

```
location / {
    fastcgi_pass  localhost:9000;
    fastcgi_index index.php;

    fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$f
    fastcgi_param QUERY_STRING    $query_string;
    fastcgi_param REQUEST_METHOD  $request_method;
    fastcgi_param CONTENT_TYPE    $content_type;
    fastcgi_param CONTENT_LENGTH  $content_length;
}
```

## Директивы

Синтаксис: **`fastcgi_allow_upstream`** *`строка`* `...;`
Умолчание: —
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.29.3.

Задаёт условия, при которых доступ к FastCGI-серверу будет разрешён или запрещён. Если все значения строковых параметров непустые и не равны “0”, то доступ разрешён. Условия проверяются каждый раз перед установлением соединения с FastCGI-сервером. В значении параметров допустимо использование переменных:

```
geo $upstream_last_addr $allow {
    volatile;
    10.10.0.0/24        1;
}

server {
    listen 127.0.0.1:8080;

    location / {
        fastcgi_pass           localhost:9000;
        fastcgi_allow_upstream $allow;
        ...
    }
}
```

Директива доступна как часть коммерческой подписки.

Синтаксис: **fastcgi_bind** *адрес* [transparent] | off;
Умолчание: —
Контекст: http, server, location

Эта директива появилась в версии 0.8.22.

Задаёт локальный IP-адрес с необязательным портом (1.11.2), который будет использоваться в исходящих соединениях с FastCGI-сервером. В значении параметра допустимо использование переменных (1.3.12). Специальное значение `off` (1.3.12) отменяет действие унаследованной с предыдущего уровня конфигурации директивы `fastcgi_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

Параметр `transparent` (1.11.0) позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с FastCGI-сервером, например, реальный IP-адрес клиента:

```
fastcgi_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы nginx с привилегиями суперпользователя. В Linux этого не требуется (1.13.8), так как если указан параметр `transparent`, то рабочие процессы наследуют capability `CAP_NET_RAW` из главного процесса. Также необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с FastCGI-сервера.

Синтаксис: **`fastcgi_bind_dynamic`** `on | off;`
Умолчание: `fastcgi_bind_dynamic off;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.29.3.

Если включено, операция bind осуществляется при каждой попытке соединения.

Директива доступна как часть коммерческой подписки.

Синтаксис: **`fastcgi_buffer_size`** *`размер`*`;`
Умолчание: `fastcgi_buffer_size 4k|8k;`
Контекст: `http`, `server`, `location`

Задаёт *`размер`* буфера, в который будет читаться первая часть ответа, получаемого от FastCGI-сервера. В этой части ответа находится, как правило, небольшой заголовок ответа; если его размер больше заданного, то такой ответ считается неверным. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4K, или 8K, однако его можно сделать меньше.

Синтаксис: **`fastcgi_buffering`** `on | off;`
Умолчание: `fastcgi_buffering on;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.5.6.

Разрешает или запрещает использовать буферизацию ответов FastCGI-сервера.

Если буферизация включена, то nginx принимает ответ FastCGI-сервера как можно быстрее, сохраняя его в буферы, заданные директивами fastcgi_buffer_size и fastcgi_buffers. Если ответ не вмещается целиком в память, то его часть может быть записана на диск во временный файл. Запись во

временные файлы контролируется директивами fastcgi_max_temp_file_size и fastcgi_temp_file_write_size.

Если буферизация выключена, то ответ синхронно передаётся клиенту сразу же по мере его поступления. nginx не пытается считать весь ответ FastCGI-сервера. Максимальный размер данных, который nginx может принять от сервера за один раз, задаётся директивой fastcgi_buffer_size.

Буферизация может быть также включена или выключена путём передачи значения “ `yes` ” или “ `no` ” в поле “X-Accel-Buffering” заголовка ответа. Эту возможность можно запретить с помощью директивы fastcgi_ignore_headers.

Синтаксис: **`fastcgi_buffers`** *`число размер`*`;`
Умолчание: `fastcgi_buffers 8 4k|8k;`
Контекст: `http`, `server`, `location`

Задаёт *`число`* и *`размер`* буферов для одного соединения, в которые будет читаться ответ, получаемый от FastCGI-сервера. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4K, или 8K.

Синтаксис: **`fastcgi_busy_buffers_size`** *`размер`*`;`
Умолчание: `fastcgi_busy_buffers_size 8k|16k;`
Контекст: `http`, `server`, `location`

При включённой буферизации ответов FastCGI-сервера, ограничивает суммарный *`размер`* буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ ещё не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл. По умолчанию *`размер`* ограничен двумя буферами, заданными директивами fastcgi_buffer_size и fastcgi_buffers.

Синтаксис: **`fastcgi_cache`** *`зона`* `| off;`
Умолчание: `fastcgi_cache off;`
Контекст: `http`, `server`, `location`

Задаёт зону разделяемой памяти, используемой для кэширования. Одна и та же зона может использоваться в нескольких местах. В значении параметра можно использовать переменные (1.7.9). Параметр `off` запрещает кэширование, унаследованное с предыдущего уровня конфигурации.

Синтаксис: **`fastcgi_cache_background_update`** `on | off;`
Умолчание: `fastcgi_cache_background_update off;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.11.10.

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший закэшированный ответ. Использование устаревшего закэшированного ответа в момент его обновления должно быть разрешено.

Синтаксис: **`fastcgi_cache_bypass`** *`строка ...;`*
Умолчание: —
Контекст: `http`, `server`, `location`

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно “0”, то ответ не берётся из кэша:

```
fastcgi_cache_bypass $cookie_nocache $arg_nocache$arg_com
fastcgi_cache_bypass $http_pragma    $http_authorization;
```

Можно использовать совместно с директивой fastcgi_no_cache.

Синтаксис: **`fastcgi_cache_key`** *`строка`*`;`
Умолчание: —
Контекст: `http`, `server`, `location`

Задаёт ключ для кэширования, например,

```
fastcgi_cache_key localhost:9000$request_uri;
```

Синтаксис: **`fastcgi_cache_lock`** `on | off;`
Умолчание: `fastcgi_cache_lock off;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.1.12.

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве fastcgi_cache_key, передав запрос на FastCGI-сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой fastcgi_cache_lock_timeout.

Синтаксис: **fastcgi_cache_lock_age** *время*;
Умолчание: fastcgi_cache_lock_age 5s;
Контекст: http, server, location

Эта директива появилась в версии 1.7.8.

Если последний запрос, переданный на FastCGI-сервер для заполнения нового элемента кэша, не завершился за указанное *время*, на FastCGI-сервер может быть передан ещё один запрос.

Синтаксис: **fastcgi_cache_lock_timeout** *время*;
Умолчание: fastcgi_cache_lock_timeout 5s;
Контекст: http, server, location

Эта директива появилась в версии 1.1.12.

Задаёт таймаут для fastcgi_cache_lock. По истечении указанного *времени* запрос будет передан на FastCGI-сервер, однако ответ не будет закэширован.

До версии 1.7.8 такой ответ мог быть закэширован.

Синтаксис: **fastcgi_cache_max_range_offset** *число*;
Умолчание: —
Контекст: http, server, location

Эта директива появилась в версии 1.11.6.

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на FastCGI-сервер и ответ не будет закэширован.

Синтаксис: **fastcgi_cache_methods** GET | HEAD | POST ...;

Умолчание: `fastcgi_cache_methods GET HEAD;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 0.7.59.

Если метод запроса клиента указан в этой директиве, то ответ будет закэширован. Методы “ `GET` ” и “ `HEAD` ” всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву fastcgi_no_cache.

Синтаксис: **`fastcgi_cache_min_uses`** *`число`*;
Умолчание: `fastcgi_cache_min_uses 1;`
Контекст: `http`, `server`, `location`

Задаёт *`число`* запросов, после которого ответ будет закэширован.

Синтаксис: **`fastcgi_cache_path`** *`путь`* [levels=*`уровни`*]
[use_temp_path=on|off] keys_zone=*`имя`*:*`размер`*
[inactive=*`время`*] [max_size=*`размер`*]
[min_free=*`размер`*] [manager_files=*`число`*]
[manager_sleep=*`время`*]
[manager_threshold=*`время`*] [loader_files=*`число`*]
[loader_sleep=*`время`*] [loader_threshold=*`время`*]
[purger=on|off] [purger_files=*`число`*]
[purger_sleep=*`время`*] [purger_threshold=*`время`*];
Умолчание: —
Контекст: `http`

Задаёт путь и другие параметры кэша. Данные кэша хранятся в файлах. Ключом и именем файла в кэше является результат функции MD5 от проксированного URL. Параметр `levels` задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2. Например, при использовании

```
fastcgi_cache_path /data/nginx/cache levels=1:2 keys_zone
```

имена файлов в кэше будут такого вида:

```
/data/nginx/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Начиная с версии 0.8.9 временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешёвой операции переименовывания

в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами. Какой из каталогов будет использоваться для временных файлов определяется параметром `use_temp_path` (1.7.10). Если параметр не задан или установлен в значение “ `on` ”, то будет использоваться каталог, задаваемый директивой fastcgi_temp_path для данного location. Если параметр установлен в значение “ `off` ”, то временные файлы будут располагаться непосредственно в каталоге кэша.

Кроме того, все активные ключи и информация о данных хранятся в зоне разделяемой памяти, *`имя`* и *`размер`* которой задаются параметром `keys_zone` . Зоны размером в 1 мегабайт достаточно для хранения около 8 тысяч ключей.

> Как часть коммерческой подписки в зоне разделяемой памяти также хранится расширенная информация о кэше, поэтому для хранения аналогичного количества ключей необходимо указывать больший размер зоны. Например зоны размером в 1 мегабайт достаточно для хранения около 4 тысяч ключей.

Если к данным кэша не обращаются в течение времени, заданного параметром `inactive` , то данные удаляются, независимо от их свежести. По умолчанию `inactive` равен 10 минутам.

Специальный процесс “cache manager” следит за максимальным размером кэша, заданным параметром `max_size` , а также за минимальным объёмом свободного места на файловой системе с кэшем, заданным параметром `min_free` (1.19.1). При превышении максимального размера кэша или недостаточном объёме свободного места процесс удаляет наименее востребованные данные. Удаление данных происходит итерациями, настраиваемыми параметрами (1.11.5) `manager_files` , `manager_threshold` и `manager_sleep` . За одну итерацию загружается не более `manager_files` элементов (по умолчанию 100). Время работы одной итерации ограничено параметром `manager_threshold` (по умолчанию 200 миллисекунд). Между итерациями делается пауза на время, заданное параметром `manager_sleep` (по умолчанию 50 миллисекунд).

Через минуту после старта активируется специальный процесс “cache loader”, который загружает в зону кэша информацию о ранее закэшированных данных, хранящихся на файловой системе. Загрузка также происходит итерациями. За одну итерацию загружается не более `loader_files` элементов (по умолчанию 100). Кроме того, время работы одной итерации ограничено параметром `loader_threshold` (по умолчанию 200 миллисекунд). Между итерациями делается пауза на время, заданное параметром `loader_sleep` (по умолчанию 50 миллисекунд).

Кроме того, следующие параметры доступны как часть коммерческой подписки:

`purger` = `on` | `off`

Указывает, будут ли записи в кэше, соответствующие маске, удалены с диска при помощи процесса “cache purger” (1.7.12). Установка параметра в значение `on` (по умолчанию `off`) активирует процесс “cache purger”, который проходит по всем записям в кэше и удаляет записи, соответствующие этой маске.

`purger_files` = *`число`*

Задаёт число элементов, которые будут сканироваться за одну итерацию (1.7.12). По умолчанию `purger_files` равен 10.

`purger_threshold` = *`время`*

Задаёт продолжительность одной итерации (1.7.12). По умолчанию `purger_threshold` равен 50 миллисекундам.

`purger_sleep` = *`время`*

Задаёт паузу между итерациями (1.7.12). По умолчанию `purger_sleep` равен 50 миллисекундам.

В версиях 1.7.3, 1.7.7 и 1.11.10 формат заголовка кэша был изменён. При обновлении на более новую версию nginx ранее закэшированные ответы будут считаться недействительными.

Синтаксис: **`fastcgi_cache_purge`** `строка ...;`
Умолчание: —
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.5.7.

Задаёт условия, при которых запрос будет считаться запросом на очистку кэша. Если значение хотя бы одного из строковых параметров непустое и не равно “0”, то запись в кэше с соответствующим ключом кэширования удаляется. В результате успешной операции возвращается ответ с кодом 204 (No Content).

Если ключ кэширования запроса на очистку заканчивается звёздочкой (“`*`”), то все записи в кэше, соответствующие этой маске, будут удалены из кэша. Тем не менее, эти записи будут оставаться на диске или до момента удаления из-за отсутствия обращения к данным, или до обработки их процессом “cache purger” (1.7.12), или до попытки клиента получить к ним доступ.

Пример конфигурации:

```
fastcgi_cache_path /data/nginx/cache keys_zone=cache_zone

map $request_method $purge_method {
    PURGE   1;
    default 0;
}

server {
    ...
    location / {
        fastcgi_pass        http://backend;
        fastcgi_cache       cache_zone;
        fastcgi_cache_key   $uri;
        fastcgi_cache_purge $purge_method;
    }
}
```

Функциональность доступна как часть коммерческой подписки.

Синтаксис: **`fastcgi_cache_revalidate`** `on | off;`
Умолчание: `fastcgi_cache_revalidate off;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.5.7.

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка “If-Modified-Since” и “If-None-Match”.

Синтаксис: **fastcgi_cache_use_stale** error | timeout | invalid_header | updating | http_500 | http_503 | http_403 | http_404 | http_429 | off ...;
Умолчание: fastcgi_cache_use_stale off;
Контекст: http, server, location

Определяет, в каких случаях можно использовать устаревший закэшированный ответ. Параметры директивы совпадают с параметрами директивы fastcgi_next_upstream.

Параметр `error` также позволяет использовать устаревший закэшированный ответ при невозможности выбора FastCGI-сервера для обработки запроса.

Кроме того, дополнительный параметр `updating` разрешает использовать устаревший закэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к FastCGI-серверам при обновлении закэшированных данных.

Использование устаревшего закэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определённое количество секунд после того, как ответ устарел (1.11.10). Такой способ менее приоритетен, чем задание параметров директивы.

- Расширение “stale-while-revalidate” поля заголовка “Cache-Control” разрешает использовать устаревший закэшированный ответ, если на данный момент он уже обновляется.
- Расширение “stale-if-error” поля заголовка “Cache-Control” разрешает использовать устаревший закэшированный ответ в случае ошибки.

Чтобы минимизировать число обращений к FastCGI-серверам при заполнении нового элемента кэша, можно воспользоваться директивой fastcgi_cache_lock.

Синтаксис: **fastcgi_cache_valid** [*код* ...] *время*;
Умолчание: —
Контекст: http, server, location

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
fastcgi_cache_valid 200 302 10m;
fastcgi_cache_valid 404      1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только `время` кэширования,

```
fastcgi_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра `any`:

```
fastcgi_cache_valid 200 302 10m;
fastcgi_cache_valid 301      1h;
fastcgi_cache_valid any      1m;
```

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка “X-Accel-Expires” задаёт время кэширования ответа в секундах. Значение 0 запрещает кэшировать ответ. Если значение начинается с префикса `@`, оно задаёт абсолютное время в секундах с начала эпохи, до которого ответ может быть закэширован.
- Если в заголовке нет поля “X-Accel-Expires”, параметры кэширования определяются по полям заголовка “Expires” или “Cache-Control”.
- Ответ, в заголовке которого есть поле “Set-Cookie”, не будет кэшироваться.
- Ответ, в заголовке которого есть поле “Vary” со специальным значением “`*`”, не будет кэшироваться (1.7.7). Ответ, в заголовке которого есть поле “Vary” с другим значением, будет закэширован с учётом соответствующих полей заголовка запроса (1.7.7).

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы fastcgi_ignore_headers.

Синтаксис: **fastcgi_catch_stderr** *строка*;
Умолчание: —
Контекст: http, server, location

Задаёт строку для поиска в потоке ошибок ответа, полученного от FastCGI-сервера. Если *строка* найдена, то считается, что FastCGI-сервер вернул неверный ответ. Это позволяет обрабатывать ошибки приложений в nginx, например:

```
location /php/ {
    fastcgi_pass backend:9000;
    ...
    fastcgi_catch_stderr "PHP Fatal error";
    fastcgi_next_upstream error timeout invalid_header;
}
```

Синтаксис: **fastcgi_connect_timeout** *время*;
Умолчание: fastcgi_connect_timeout 60s;
Контекст: http, server, location

Задаёт таймаут для установления соединения с FastCGI-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

Синтаксис: **fastcgi_force_ranges** on | off;
Умолчание: fastcgi_force_ranges off;
Контекст: http, server, location

Эта директива появилась в версии 1.7.7.

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некэшированных ответов FastCGI-сервера вне зависимости от наличия поля “Accept-Ranges” в заголовках этих ответов.

Синтаксис: **fastcgi_hide_header** *поле*;
Умолчание: —
Контекст: http, server, location

По умолчанию nginx не передаёт клиенту поля заголовка “Status” и “X-Accel-...” из ответа FastCGI-сервера. Директива fastcgi_hide_header задаёт дополнительные поля, которые не будут передаваться. Если же передачу полей нужно

разрешить, можно воспользоваться директивой fastcgi_pass_header.

Синтаксис: **fastcgi_ignore_client_abort** on | off;
Умолчание: fastcgi_ignore_client_abort off;
Контекст: http, server, location

Определяет, закрывать ли соединение с FastCGI-сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

Синтаксис: **fastcgi_ignore_headers** *поле* ...;
Умолчание: —
Контекст: http, server, location

Запрещает обработку некоторых полей заголовка из ответа FastCGI-сервера. В директиве можно указать поля “X-Accel-Redirect”, “X-Accel-Expires”, “X-Accel-Limit-Rate” (1.1.6), “X-Accel-Buffering” (1.1.6), “X-Accel-Charset” (1.1.6), “Expires”, “Cache-Control”, “Set-Cookie” (0.8.44) и “Vary” (1.7.7).

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- “X-Accel-Expires”, “Expires”, “Cache-Control”, “Set-Cookie” и “Vary” задают параметры кэширования ответа;
- “X-Accel-Redirect” производит внутреннее перенаправление на указанный URI;
- “X-Accel-Limit-Rate” задаёт ограничение скорости передачи ответа клиенту;
- “X-Accel-Buffering” включает или выключает буферизацию ответа;
- “X-Accel-Charset” задаёт желаемую кодировку ответа.

Синтаксис: **fastcgi_index** *имя*;
Умолчание: —
Контекст: http, server, location

Задаёт имя файла, который при создании переменной `$fastcgi_script_name` будет добавляться после URI, если URI заканчивается слэшом. Например, при таких настройках

```
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastc
```

и запросе “ `/page.php` ” параметр `SCRIPT_FILENAME` будет равен “ `/home/www/scripts/php/page.php` ”, а при запросе “ `/` ” - “ `/home/www/scripts/php/index.php` ”.

Синтаксис: **`fastcgi_intercept_errors`** `on | off;`
Умолчание: `fastcgi_intercept_errors off;`
Контекст: `http`, `server`, `location`

Определяет, передавать ли клиенту ответы FastCGI-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку nginx’у с помощью директивы error_page.

Синтаксис: **`fastcgi_keep_conn`** `on | off;`
Умолчание: `fastcgi_keep_conn off;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.1.4.

По умолчанию FastCGI-сервер будет закрывать соединение сразу же после отправки ответа. При установке значения `on` nginx указывает FastCGI-серверу оставлять соединения открытыми. Это в частности требуется для функционирования постоянных соединений с FastCGI-серверами.

Синтаксис: **`fastcgi_limit_rate`** *`скорость`*`;`
Умолчание: `fastcgi_limit_rate 0;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.7.7.

Ограничивает скорость чтения ответа от FastCGI-сервера. *`Скорость`* задаётся в байтах в секунду. Значение 0 отключает ограничение скорости. Ограничение устанавливается на запрос, поэтому, если nginx одновременно откроет два соединения к FastCGI-серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена буферизация ответов FastCGI-сервера. В значении параметра можно использовать переменные (1.27.0).

Синтаксис: **fastcgi_max_temp_file_size** *размер*;
Умолчание: fastcgi_max_temp_file_size 1024m;
Контекст: http, server, location

Если включена буферизация ответов FastCGI-сервера, и ответ не вмещается целиком в буферы, заданные директивами fastcgi_buffer_size и fastcgi_buffers, часть ответа может быть записана во временный файл. Эта директива задаёт максимальный *размер* временного файла. Размер данных, сбрасываемых во временный файл за один раз, задаётся директивой fastcgi_temp_file_write_size.

Значение 0 отключает возможность буферизации ответов во временные файлы.

Данное ограничение не распространяется на ответы, которые будут закэшированы или сохранены на диске.

Синтаксис: **fastcgi_next_upstream** error | timeout | denied | invalid_header | http_500 | http_503 | http_403 | http_404 | http_429 | non_idempotent | off ...;
Умолчание: fastcgi_next_upstream error timeout;
Контекст: http, server, location

Определяет, в каких случаях запрос будет передан следующему серверу:

`error`
: произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;

`timeout`
: произошёл таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;

`denied`
: сервер отклонил соединение (1.29.3);

Параметр доступен как часть коммерческой подписки.

`invalid_header`

сервер вернул пустой или неверный ответ;

`http_500`
: сервер вернул ответ с кодом 500;

`http_503`
: сервер вернул ответ с кодом 503;

`http_403`
: сервер вернул ответ с кодом 403;

`http_404`
: сервер вернул ответ с кодом 404;

`http_429`
: сервер вернул ответ с кодом 429 (1.11.13);

`non_idempotent`
: обычно запросы с неидемпотентным методом (`POST`, `LOCK`, `PATCH`) не передаются на другой сервер, если запрос серверу группы уже был отправлен (1.9.13); включение параметра явно разрешает повторять подобные запросы;

`off`
: запрещает передачу запроса следующему серверу.

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту ещё ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа, то исправить это уже невозможно.

Директива также определяет, что считается неудачной попыткой работы с сервером. Случаи `error`, `timeout`, `denied` и `invalid_header` всегда считаются неудачными попытками, даже если они не указаны в директиве. Случаи `http_500`, `http_503` и `http_429` считаются неудачными попытками, только если они указаны в директиве. Случаи `http_403` и `http_404` никогда не считаются неудачными попытками.

Передача запроса следующему серверу может быть ограничена по количеству попыток и по времени.

Синтаксис: **fastcgi_next_upstream_timeout** *время*;
Умолчание: fastcgi_next_upstream_timeout 0;
Контекст: http, server, location

Эта директива появилась в версии 1.7.5.

Ограничивает время, в течение которого возможна передача запроса следующему серверу. Значение `0` отключает это ограничение.

Синтаксис: **fastcgi_next_upstream_tries** *число*;
Умолчание: fastcgi_next_upstream_tries 0;
Контекст: http, server, location

Эта директива появилась в версии 1.7.5.

Ограничивает число допустимых попыток для передачи запроса следующему серверу. Значение `0` отключает это ограничение.

Синтаксис: **fastcgi_no_cache** *строка* ...;
Умолчание: —
Контекст: http, server, location

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно “0”, то ответ не будет сохранён:

```
fastcgi_no_cache $cookie_nocache $arg_nocache$arg_comment
fastcgi_no_cache $http_pragma    $http_authorization;
```

Можно использовать совместно с директивой fastcgi_cache_bypass.

Синтаксис: **fastcgi_param** *параметр значение* [if_not_empty];
Умолчание: fastcgi_param HTTP_HOST $host$is_request_port$request_port;
Контекст: http, server, location

Задаёт `параметр`, который будет передаваться FastCGI-серверу. В качестве значения можно использовать текст, переменные и их комбинации. Директивы наследуются с

предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `fastcgi_param`.

Ниже приведён пример минимально необходимых параметров для PHP:

```
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastc
fastcgi_param QUERY_STRING    $query_string;
```

Параметр `SCRIPT_FILENAME` используется в PHP для определения имени скрипта, а в параметре `QUERY_STRING` передаются параметры запроса.

Если скрипты обрабатывают запросы `POST`, то нужны ещё три параметра:

```
fastcgi_param REQUEST_METHOD  $request_method;
fastcgi_param CONTENT_TYPE    $content_type;
fastcgi_param CONTENT_LENGTH  $content_length;
```

Если PHP был собран с параметром конфигурации `--enable-force-cgi-redirect`, то нужно передавать параметр `REDIRECT_STATUS` со значением “200”:

```
fastcgi_param REDIRECT_STATUS 200;
```

Если директива указана с `if_not_empty` (1.1.11), то такой параметр с пустым значением передаваться на сервер не будет:

```
fastcgi_param HTTPS           $https if_not_empty;
```

Синтаксис: **`fastcgi_pass`** *`адрес`*`;`
Умолчание: —
Контекст: `location`, `if в location`

Задаёт адрес FastCGI-сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
fastcgi_pass localhost:9000;
```

или в виде пути UNIX-сокета:

```
fastcgi_pass unix:/tmp/fastcgi.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). И, кроме того, адрес может быть группой серверов.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью resolver'а.

Синтаксис: **fastcgi_pass_header** *поле*;
Умолчание: —
Контекст: http, server, location

Разрешает передавать от FastCGI-сервера клиенту запрещённые для передачи поля заголовка.

Синтаксис: **fastcgi_pass_request_body** on | off;
Умолчание: fastcgi_pass_request_body on;
Контекст: http, server, location

Позволяет запретить передачу исходного тела запроса на FastCGI-сервер. См. также директиву fastcgi_pass_request_headers.

Синтаксис: **fastcgi_pass_request_headers** on | off;
Умолчание: fastcgi_pass_request_headers on;
Контекст: http, server, location

Позволяет запретить передачу полей заголовка исходного запроса на FastCGI-сервер. См. также директивы fastcgi_pass_request_body.

Синтаксис: **fastcgi_read_timeout** *время*;
Умолчание: fastcgi_read_timeout 60s;
Контекст: http, server, location

Задаёт таймаут при чтении ответа FastCGI-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени FastCGI-сервер ничего не передаст, соединение закрывается.

Синтаксис: **fastcgi_request_buffering** on | off;

Умолчание: `fastcgi_request_buffering on;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.7.11.

Разрешает или запрещает использовать буферизацию тела запроса клиента.

Если буферизация включена, то тело запроса полностью читается от клиента перед отправкой запроса на FastCGI-сервер.

Если буферизация выключена, то тело запроса отправляется на FastCGI-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан следующему серверу, если nginx уже начал отправку тела запроса.

Синтаксис: **`fastcgi_request_dynamic`** `on | off;`
Умолчание: `fastcgi_request_dynamic off;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.29.3.

Разрешает или запрещает создание отдельного экземпляра запроса для каждого FastCGI-сервера. По умолчанию для всех FastCGI-серверов используется единый запрос. Если разрешено, то для каждого сервера создаётся отдельный экземпляр запроса, что позволяет кастомизировать запрос для конкретного сервера.

Директива доступна как часть коммерческой подписки.

Синтаксис: **`fastcgi_send_lowat`** *`размер`*`;`
Умолчание: `fastcgi_send_lowat 0;`
Контекст: `http`, `server`, `location`

При установке директивы в ненулевое значение nginx будет пытаться минимизировать число операций отправки на исходящих соединениях с FastCGI-сервером либо при помощи флага `NOTE_LOWAT` метода kqueue, либо при помощи параметра сокета `SO_SNDLOWAT`, с указанным *`размером`*.

Эта директива игнорируется на Linux, Solaris и Windows.

Синтаксис: **`fastcgi_send_timeout`** *`время`*`;`
Умолчание: `fastcgi_send_timeout 60s;`

Контекст: `http`, `server`, `location`

Задаёт таймаут при передаче запроса FastCGI-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени FastCGI-сервер не примет новых данных, соединение закрывается.

Синтаксис: **`fastcgi_socket_keepalive`** `on | off;`
Умолчание: `fastcgi_socket_keepalive off;`
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.15.6.

Конфигурирует поведение “TCP keepalive” для исходящих соединений к FastCGI-серверу. По умолчанию для сокета действуют настройки операционной системы. Если указано значение “`on`”, то для сокета включается параметр `SO_KEEPALIVE`.

Синтаксис: **`fastcgi_socket_rcvbuf`** *`size`*`;`
Умолчание: —
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.31.3.

Задаёт размер буфера приёма (параметр `SO_RCVBUF`) для исходящих соединений к FastCGI-серверу. Специальное значение `0` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `fastcgi_socket_rcvbuf`, позволяя использовать настройки операционной системы.

Синтаксис: **`fastcgi_socket_sndbuf`** *`size`*`;`
Умолчание: —
Контекст: `http`, `server`, `location`

Эта директива появилась в версии 1.31.3.

Задаёт размер буфера передачи (параметр `SO_SNDBUF`) для исходящих соединений к FastCGI-серверу. Специальное значение `0` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `fastcgi_socket_sndbuf`, позволяя использовать настройки операционной системы.

Синтаксис: **fastcgi_split_path_info** *regex*;
Умолчание: —
Контекст: location

Задаёт регулярное выражение, выделяющее значение для переменной $fastcgi_path_info. Регулярное выражение должно иметь два выделения, из которых первое становится значением переменной $fastcgi_script_name, а второе - значением переменной $fastcgi_path_info. Например, при таких настройках

```
location ~ ^(.+\.php)(.*)$ {
    fastcgi_split_path_info       ^(.+\.php)(.*)$;
    fastcgi_param SCRIPT_FILENAME /path/to/php$fastcgi_sc
    fastcgi_param PATH_INFO       $fastcgi_path_info;
```

и запросе "/show.php/article/0001" параметр SCRIPT_FILENAME будет равен "/path/to/php/show.php", а параметр PATH_INFO - "/article/0001".

Синтаксис: **fastcgi_store** on | off | *строка*;
Умолчание: fastcgi_store off;
Контекст: http, server, location

Разрешает сохранение на диск файлов. Параметр on сохраняет файлы в соответствии с путями, указанными в директивах alias или root. Параметр off запрещает сохранение файлов. Кроме того, имя файла можно задать явно с помощью строки с переменными:

```
fastcgi_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю "Last-Modified" в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Начиная с версии 0.8.9 временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешёвой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными

файлами, задаваемый директивой fastcgi_temp_path для данного location.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов, например, так:

```
location /images/ {
    root                /data/www;
    error_page          404 = /fetch$uri;
}

location /fetch/ {
    internal;

    fastcgi_pass        backend:9000;
    ...

    fastcgi_store       on;
    fastcgi_store_access user:rw group:rw all:r;
    fastcgi_temp_path   /data/temp;

    alias               /data/www/;
}
```

Синтаксис: **fastcgi_store_access** *пользователи:права ...;*
Умолчание: `fastcgi_store_access user:rw;`
Контекст: `http`, `server`, `location`

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
fastcgi_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для `group` или `all`, то права для `user` указывать необязательно:

```
fastcgi_store_access group:rw all:r;
```

Синтаксис: **fastcgi_temp_file_write_size** *размер*;
Умолчание: `fastcgi_temp_file_write_size 8k|16k;`
Контекст: `http`, `server`, `location`

Ограничивает *размер* данных, сбрасываемых во временный файл за один раз, при включённой буферизации ответов FastCGI-сервера во временные файлы. По умолчанию *размер* ограничен двумя буферами, заданными директивами fastcgi_buffer_size и fastcgi_buffers.

Максимальный размер временного файла задаётся директивой fastcgi_max_temp_file_size.

Синтаксис: **fastcgi_temp_path** *путь* [*уровень1* [*уровень2* [*уровень3*]]];
Умолчание: fastcgi_temp_path fastcgi_temp;
Контекст: http, server, location

Задаёт имя каталога для хранения временных файлов с данными, полученными от FastCGI-серверов. В каталоге может использоваться иерархия подкаталогов до трёх уровней. Например, при такой конфигурации

```
fastcgi_temp_path /spool/nginx/fastcgi_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/nginx/fastcgi_temp/7/45/00000123457
```

См. также параметр `use_temp_path` директивы fastcgi_cache_path.

## Параметры, передаваемые FastCGI-серверу

Поля заголовка HTTP-запроса передаются FastCGI-серверу в виде параметров. В приложениях и скриптах, запущенных в виде FastCGI-сервера, эти параметры обычно доступны в виде переменных среды. Например, поле заголовка “User-Agent” передаётся как параметр `HTTP_USER_AGENT`. Кроме полей заголовка HTTP-запроса можно передавать произвольные параметры с помощью директивы fastcgi_param.

## Встроенные переменные

В модуле `ngx_http_fastcgi_module` есть встроенные переменные, которые можно использовать для формирования параметров с помощью директивы fastcgi_param:

`$fastcgi_script_name`

URI запроса или же, если URI заканчивается слэшом, то URI запроса, дополненное именем индексного файла,

задаваемого директивой fastcgi_index. Эту переменную можно использовать для задания параметров `SCRIPT_FILENAME` и `PATH_TRANSLATED`, используемых, в частности, для определения имени скрипта в PHP. Например, для запроса “ `/info/` ” и при использовании директив

```
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fas
```

параметр `SCRIPT_FILENAME` будет равен “ `/home/www/scripts/php/info/index.php` ”.

При использовании директивы fastcgi_split_path_info переменная `$fastcgi_script_name` равна значению первого выделения, задаваемого этой директивой.

`$fastcgi_path_info`

значение второго выделения, задаваемого директивой fastcgi_split_path_info. Эту переменную можно использовать для задания параметра `PATH_INFO`.