Документация
nginx

FAQ

19.01.2009

Что означает ошибка "accept() failed (53: Software caused connection abort) while accepting new connection on 0.0.0.0:80" ?

Это некритическая ошибка, возникающая от из-за того, что клиент закрыл соединение до того, как nginx его получил из ядра. Такое может случиться, например, если пользователь, зайдя на страницу с картинками, перешёл по ссылке, не дожидаясь, пока загрузятся все картинки и браузер закрыл ставшие ненужными соединения.

Почему при использовании HTTPS для всех виртуальных серверов показывается сертификат только первого виртуального сервера ?

Потому что SSL-соединение устанавливается до того, как веб-сервер узнает, для какого именно виртуального сервера предназначен запрос, и поэтому выдаёт сертификат сервера по умолчанию.

На данный момент наиболее общим и надёжным решением этой проблемы является выделение каждому HTTPS-серверу отдельного IP-адреса. В частных случаях возможно использование сертификатов с альтернативными именами и wildcard'ами.

Наиболее вероятным решением этой проблемы в ближайшем будущем будет TLS extension Server Name Indication (SNI, RFC3546). На сегодняшний момент SNI поддерживается только со следующих версий браузеров:

- MSIE 7.0 (но только под Windows Vista),
- Firefox 2.0,
- Opera 8.0 (при включённой поддержке TLSv1.1),
- Safari 3.2.1 (Mac OS X 10.5.6),
- Chrome.

Как переписать апачевские правила для Drupal:

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
```

```
RewriteRule ^(.*)$ index.php?q=$1 [L,QSA]
```

> при использовании nginx/FastCGI ?

Вот так:

```
location / {
   try_files      $uri  $uri/  @drupal;
}

location @drupal {
   fastcgi_pass   ...;

   fastcgi_param  SCRIPT_FILENAME  /path/to/index.php;
   fastcgi_param  SCRIPT_NAME      /index.php;
   fastcgi_param  QUERY_STRING     q=$uri&$args;

   ... прочие fastcgi_param
}
```

Обычно практикуемая прямая трансляция правил:

```
location / {
   if (!-e $request_filename) {
       rewrite  ^(.*)  /index.php?q=$1  last;
   }
}

location = /index.php {
   fastcgi_pass  ...
   ... прочие fastcgi_param
}
```

- достойна всяческого порицания.

(C) Игорь Сысоев
http://sysoev.ru

Список рассылки

22.11.2009

Для обсуждения вопросов, связанных с nginx, создан русскоязычный список nginx-ru@nginx.org. Архивы списка: основной, у Алексея Тутубалина, Gmane и Форум.

> Для отправки писем в список нужно предварительно подписаться. Письма, отправленные через Gmane, в список не попадают.

(C) Игорь Сысоев
http://sysoev.ru

Установка nginx

18.07.2005

Конфигурация сборки осуществляется командой `configure`. Она определяет особенности системы и, в частности, методы, которые nginx может использовать для обработки соединений. В конце концов она создаёт `Makefile`. `configure` поддерживает следующие параметры:

`--prefix=<путь>` — задаёт каталог, в котором будут находиться файлы сервера. Этот же каталог будет использоваться для всех относительных путей, задаваемых `./configure` (кроме путей к исходным текстам библиотек) и в конфигурационном файле `nginx.conf`. По умолчанию — каталог `/usr/local/nginx`.

`--sbin-path=<путь>` — задаёт имя исполняемого файла nginx. Это имя используется только на стадии установки. По умолчанию файл называется `<prefix>/sbin/nginx`.

`--conf-path=<путь>` — задаёт имя конфигурационного файла `nginx.conf`. При желании nginx можно всегда запустить с другим конфигурационным файлом, указав его в параметре командной строки `-c <файл>`. По умолчанию файл называется `<prefix>/conf/nginx.conf`.

`--pid-path=<путь>` — задаёт имя файла nginx.pid, в котором будет хранится номер главного процесса. После установки имя файла можно всегда поменять в конфигурационном файле `nginx.conf` с помощью директивы pid. По умолчанию имя файла — `<prefix>/logs/nginx.pid`.

`--error-log-path=<путь>` — задаёт имя основного файла ошибок, предупреждений и диагностики. После установки имя файла можно всегда поменять в конфигурационном файле `nginx.conf` с помощью директивы error_log. По умолчанию имя файла — `<prefix>/logs/error.log`.

`--http-log-path=<путь>` — задаёт имя основного файла регистрации запросов http сервера. После установки имя файла можно всегда поменять в конфигурационном файле `nginx.conf` с помощью директивы access_log. По умолчанию имя файла — `<prefix>/logs/access.log`.

`--user=<имя>` — задаёт имя непривилегированного пользователя, с правами которого будут выполняться рабочие процессы. После установки это имя можно всегда поменять в конфигурационном файле `nginx.conf` с помощью директивы user. По умолчанию имя пользователя nobody.

`--group=<группа>` — задаёт группу, с правами которой будут выполняться рабочие процессы. После установки это имя можно всегда поменять в конфигурационном файле `nginx.conf` с помощью директивы user. По умолчанию группа совпадает с именем непривилегированного пользователя.

`--with-select_module`
`--without-select_module` — разрешает или запрещает сборку модуля для работы сервера с помощью метода select. Этот модуль собирается автоматически, если на платформе не обнаружено более подходящего метода — kqueue, epoll, rtsig или /dev/poll.

`--with-poll_module`
`--without-poll_module` — разрешает или запрещает сборку модуля для работы сервера с помощью метода poll. Этот модуль собирается автоматически, если на платформе не обнаружено более подходящего метода — kqueue, epoll, rtsig или /dev/poll.

`--without-http_gzip_module` — запрещает сборку модуля сжатия ответов http сервера. Для сборки и работы этого модуля нужна библиотека zlib.

`--without-http_rewrite_module` — запрещает сборку модуля http сервера, позволяющего делать редиректы и менять URI запросов. Для сборки и работы этого модуля нужна библиотека PCRE. Модуль экспериментальный — директивы модуля впоследствии могут измениться.

`--without-http_proxy_module` — запрещает сборку проксирующего модуля http сервера.

`--with-http_ssl_module` — разрешает сборку модуля для работы http сервера по протоколу HTTPS. По умолчанию модуль не собирается. Для сборки и работы этого модуля нужна библиотека OpenSSL.

`--with-pcre=<путь>` — задаёт путь к исходным текстам библиотеки PCRE. Дистрибутив библиотеки (версию 4.4 — 6.1) нужно взять на сайте PCRE и распаковать. Всё остальное сделают nginx'овские ./configure и make. Библиотека нужна для использования регулярных выражений в location и для модуля ngx_http_rewrite_module.

`--with-zlib=<путь>` — задаёт путь к исходным текстам библиотеки zlib. Дистрибутив библиотеки (версию 1.1.3 — 1.2.2) нужно взять на сайте zlib и распаковать. Всё остальное сделают nginx'овские ./configure и make. Библиотека нужна для модуля ngx_http_gzip_module.

`--with-cc-opt=<параметры>` — задаёт дополнительные параметры, которые будут добавлены к переменной CFLAGS. При использовании системной библиотеки

PCRE во FreeBSD, нужно указать `--with-cc-opt="-I /usr/local/include"`. Если нужно увеличить число файлов, с которыми может работать select(), то это тоже можно задать здесь же: `--with-cc-opt="-D FD_SETSIZE=2048"`.

`--with-ld-opt=<параметры>` — задаёт дополнительные параметры, которые будут использованы при линковке. При использовании системной библиотеки PCRE во FreeBSD, нужно указать `--with-ld-opt="-L /usr/local/lib"`.

Пример использования параметров (всё это нужно набрать в одной строке):

```
./configure
   --sbin-path=/usr/local/nginx/nginx
   --conf-path=/usr/local/nginx/nginx.conf
   --pid-path=/usr/local/nginx/nginx.pid
   --with-http_ssl_module
   --with-pcre=../pcre-4.4
   --with-zlib=../zlib-1.1.3
```

(C) Игорь Сысоев
http://sysoev.ru

Установка и использование под Windows

21.01.2011

nginx/Windows работает с Win32 API (не эмуляция Cygwin). В качестве метода обработки соединений используется select, поэтому не стоит ожидать высокой производительности и масштабируемости: пока это бета-версия. На данный момент доступна практически вся функциональность, что и в nginx/Unix, за исключением XSLT-фильтра, фильтра изображений, модуля geoip и встроенного perl'а.

Распаковываем дистрибутив на диск C:, переходим в каталог nginx-0.9.4 и запускаем `nginx`:

```
cd c:\
unzip nginx-0.9.4.zip
cd nginx-0.9.4
start nginx
```

Если nginx не запустился, нужно смотреть причины в error_log. Если же error_log не создался, то об этом сообщается в Event Log.

nginx/Windows работает как обычное приложение (не сервис) и управляется следующим образом:

nginx -s stop	быстрое завершение
nginx -s quit	плавное завершение
nginx -s reload	изменение конфигурации, запуск новых рабочих процессов с новой конфигурацией, плавное завершение старых рабочих процессов
nginx -s reopen	переоткрытие лог-файлов

(C) Игорь Сысоев
http://sysoev.ru

Методы обработки соединений

26.09.2006

nginx поддерживает следующие методы обработки соединений, которые можно задать директивой use:

`select` — стандартный метод. Модуль для поддержки этого метода собирается автоматически, если на платформе не обнаружено более эффективного метода. Можно принудительно разрешить или запретить сборку этого модуля с помощью параметров `--with-select_module` или `--without-select_module`.

`poll` — стандартный метод. Модуль для поддержки этого метода собирается автоматически, если на платформе не обнаружено более эффективного метода. Можно принудительно разрешить или запретить сборку этого модуля с помощью параметров `--with-poll_module` или `--without-poll_module`.

`kqueue` — эффективный метод, используемый во FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 и MacOS X. На двух-процессорных машинах под управлением MacOS X использование kqueue может привести к kernel panic.

`epoll` — эффективный метод, используемый в Linux 2.6+. В некоторых дистрибутивах, например SuSE 8.2, есть патчи для поддержки epoll ядром 2.4.

`rtsig` — real time signals, эффективный метод, используемый в Linux 2.2.19+. По умолчанию в очереди может находиться не более 1024 сигналов для всей системы. Этого недостаточно для нагруженных серверов, поэтому нужно увеличить размер очереди с помощью параметра ядра `/proc/sys/kernel/rtsig-max`. Однако, начиная с Linux 2.6.6-mm2, этого параметра уже нет и для каждого процесса существует отдельная очередь сигналов, размер которой задаётся с помощью RLIMIT_SIGPENDING.

При переполнении очереди nginx сбрасывает её и начинает обрабатывать соединения с помощью метода poll до тех пор, пока ситуация не нормализуется.

`/dev/poll` — эффективный метод, используемый в Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ и Tru64 UNIX 5.1A+.

`eventport` — event ports, эффективный метод, используемый в Solaris 10. Во избежания kernel panic, нужно установить патч.

Синтаксис конфигурационного файла

24.07.2005

Размеры можно указывать в килобайтах и мегабайтах:

k,K	килобайты
m,M	мегабайты

например, "8k", "1m". По умолчанию размер в байтах.

Время можно указывать в минутах, часах, днях и так далее:

ms	миллисекунды
s	секунды
m	минуты
h	часы
d	дни
w	недели
M	месяцы, 30 дней
y	годы, 365 дней

например, "1h 30m", "1y 6M". По умолчанию время в секундах.

(C) Игорь Сысоев
http://sysoev.ru

Пример конфигурации nginx

26.11.2005

Пример конфигурации сайта, который передаёт все запросы бэкенду, кроме картинок и запросов, начинающихся с "/download/".

```
user  www www;

worker_processes  2;

pid /var/run/nginx.pid;

#                          [ debug | info | notice | warn | error | crit ]

error_log  /var/log/nginx.error_log  info;

events {
   connections   2000;

   # use [ kqueue | rtsig | epoll | /dev/poll | select | poll ];
   use kqueue;
}

http {

   include       conf/mime.types;
   default_type  application/octet-stream;


   log_format main      '$remote_addr - $remote_user [$time_local] '
                        '"$request" $status $bytes_sent '
                        '"$http_referer" "$http_user_agent" '
                        '"$gzip_ratio"';

   log_format download  '$remote_addr - $remote_user [$time_local] '
                        '"$request" $status $bytes_sent '
                        '"$http_referer" "$http_user_agent" '
                        '"$http_range" "$sent_http_content_range"';

   client_header_timeout  3m;
   client_body_timeout    3m;
   send_timeout           3m;

   client_header_buffer_size    1k;
   large_client_header_buffers  4 4k;

   gzip on;
   gzip_min_length  1100;
   gzip_buffers     4 8k;
   gzip_types       text/plain;

   output_buffers   1 32k;
   postpone_output  1460;
```

```
sendfile         on;
tcp_nopush       on;
tcp_nodelay      on;
send_lowat       12000;

keepalive_timeout  75 20;

#lingering_time     30;
#lingering_timeout  10;
#reset_timedout_connection  on;


server {
    listen        one.example.com;
    server_name   one.example.com  www.one.example.com;

    access_log   /var/log/nginx.access_log  main;

    location / {
        proxy_pass         http://127.0.0.1/;
        proxy_redirect     off;

        proxy_set_header   Host             $host;
        proxy_set_header   X-Real-IP        $remote_addr;
        #proxy_set_header  X-Forwarded-For  $proxy_add_x_forwarded_for;

        client_max_body_size       10m;
        client_body_buffer_size    128k;

        client_body_temp_path      /var/nginx/client_body_temp;

        proxy_connect_timeout      70;
        proxy_send_timeout         90;
        proxy_read_timeout         90;
        proxy_send_lowat           12000;

        proxy_buffer_size          4k;
        proxy_buffers              4 32k;
        proxy_busy_buffers_size    64k;
        proxy_temp_file_write_size 64k;

        proxy_temp_path            /var/nginx/proxy_temp;

        charset  koi8-r;
    }

    error_page  404  /404.html;

    location /404.html {
        root  /spool/www;

        charset         on;
        source_charset  koi8-r;
    }

    location /old_stuff/ {
        rewrite   ^/old_stuff/(.*)$  /new_stuff/$1  permanent;
    }
```

```
        location /download/ {

            valid_referers  none  blocked  server_names  *.example.com;

            if ($invalid_referer) {
                #rewrite   ^/   http://www.example.com/;
                return   403;
            }

            #rewrite_log  on;

            # rewrite /download/*/mp3/*.any_ext to /download/*/mp3/*.mp3
            rewrite ^/(download/.*)/mp3/(.*)\..*$
                    /$1/mp3/$2.mp3                   break;

            root         /spool/www;
            #autoindex    on;
            access_log   /var/log/nginx-download.access_log  download;
        }

        location ~* \.(jpg|jpeg|gif)$ {
            root         /spool/www;
            access_log   off;
            expires      30d;
        }
    }
}
```

(C) Игорь Сысоев
http://sysoev.ru

Параметры командной строки nginx

30.06.2008

nginx поддерживает следующие параметры:

- `-c <файл>` — указывает использовать конфигурационный файл <файл> вместо файла по умолчанию.

`-g` — задаёт глобальные директивы конфигурации, например,

```
nginx -g "pid /var/run/nginx.pid; worker_processes `sysctl -n hw.ncpu`;"
```

- `-t` — тестировать конфигурацию. nginx проверяет синтаксическую правильность конфигурации, а затем пытается открыть файлы, описанные в конфигурации.
- `-v` — показать версию nginx.
- `-V` — показать версию nginx, версию компилятора и параметры конфигурации сборки.

Настройка виртуальных серверов

15.06.2005

Настраивать виртуальные сервера очень просто. В каждом сервере нужно описать все адреса и порты, на которых нужно принимать соединения для этого сервера, и все имена серверов. Рассмотрим следующую конфигурацию:

```
http {

   server {
       listen  192.168.10.1;
       listen  192.168.10.1:8000;

       server_name   one.example.com  www.one.example.com;

       ...
   }

   server {
       listen  192.168.10.1;
       listen  192.168.10.2:8000;
       listen  9000;

       server_name   two.example.com  www.two.example.com
                     three.example.com  www.three.example.com;

       ...
   }

   server {
       listen  9000;

       server_name   four.example.com  www.four.example.com;

       ...
   }

}
```

При такой настройке запрос, пришедший на 192.168.10.1:80 с заголовком "Host: www.three.example.com", будет обслужен вторым сервером. Если в запросе нет заголовка "Host" или же в нём указано имя, неописанное ни в одном сервере, слушающем на адресе и порту, на которые пришёл запрос, то запрос будет обслужен сервером, у которого первым описаны эти адрес и порт. Например, все запросы без заголовка "Host", пришедшие на 9000 порт, будут обслужены вторым сервером (two.example.com). То же самое произойдёт и с запросом с заголовком "Host: www.one.example.com", пришедшим на 9000 порт. Для гибкой настройки серверов по умолчанию можно использовать параметр default в

директиве listen.

Управление nginx

01.09.2008

Управлять nginx можно с помощью сигналов. Номер главного процесса по умолчанию записывается в файл `/usr/local/nginx/logs/nginx.pid`. Изменить имя этого файла можно при конфигурации сборки или же в `nginx.conf` директивой pid. Главный процесс поддерживает следующие сигналы:

TERM, INT	быстрое завершение
QUIT	плавное завершение
HUP	изменение конфигурации, обновление изменившейся временной зоны (только для FreeBSD и Linux), запуск новых рабочих процессов с новой конфигурацией, плавное завершение старых рабочих процессов
USR1	переоткрытие лог-файлов
USR2	обновление исполняемого файла
WINCH	плавное завершение рабочих процессов

Управлять рабочими процессами по отдельности не нужно. Тем не менее, они тоже поддерживают некоторые сигналы:

TERM, INT	быстрое завершение
QUIT	плавное завершение
USR1	переоткрытие лог-файлов

Изменение конфигурации

Для того, чтобы nginx перечитал файл конфигурации, нужно послать главному процессу сигнал HUP. Главный процесс сначала проверяет синтаксическую правильность конфигурации, а затем пытается применить новую конфигурацию, то есть, открыть лог-файлы и новые listen сокеты. Если ему это не удаётся, то он откатывает изменения и продолжает работать со старой конфигурацией. Если же удаётся, то он запускает новые рабочие процессы, а старым шлёт сообщение о плавном выходе. Старые рабочие процессы закрывают listen сокеты и продолжают обслуживать старых клиентов. После обслуживания всех клиентов старые рабочие процессы завершаются.

Предположим, на FreeBSD 4.x команда

```
ps ax -o pid,ppid,user,%cpu,vsz,wchan,command | egrep '(nginx|PID)'
```

> показывает примерно такую картину:

```
  PID  PPID USER    %CPU   VSZ WCHAN  COMMAND
33126     1 root     0.0  1148 pause  nginx: master process /usr/local/nginx/sb
33127 33126 nobody   0.0  1380 kqread nginx: worker process (nginx)
33128 33126 nobody   0.0  1364 kqread nginx: worker process (nginx)
33129 33126 nobody   0.0  1364 kqread nginx: worker process (nginx)
```

Если послать сигнал HUP главному процессу, то картина может быть такой:

```
  PID  PPID USER    %CPU   VSZ WCHAN  COMMAND
33126     1 root     0.0  1164 pause  nginx: master process /usr/local/nginx/sb
33129 33126 nobody   0.0  1380 kqread nginx: worker process is shutting down (n
33134 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
33135 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
33136 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
```

Один старый рабочий процесс 33129 всё ещё продолжает работать. По истечении некоторого времени он завершается:

```
  PID  PPID USER    %CPU   VSZ WCHAN  COMMAND
33126     1 root     0.0  1164 pause  nginx: master process /usr/local/nginx/sb
33134 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
33135 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
33136 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
```

Ротация лог-файлов

Лог-файлы нужно переименовать, а затем послать сигнал USR1 главному процессу. Он откроет заново все текущие открытые файлы и назначит им в качестве владельца непривилегированного пользователя, под которым работают рабочие процессы. После успешного открытия главный процесс закрывает все открытые файлы и посылает сообщение о переоткрытии файлов рабочим процессам. Они также открывают новые файлы и сразу же закрывают старые. В результате старые файлы практически сразу же готовы для дальнейшей обработки, например, их можно сжимать.

Обновление сервера на лету

Для обновления сервера нужно записать на место старого исполняемого файла новый. Затем нужно послать сигнал USR2 главному процессу он переименует свой файл с номером процесса в файл с суффиксом `.oldbin`, например, `/usr/local/nginx/logs/nginx.pid.oldbin`, после чего запустит новый исполняемый файл, а тот в свою очередь свои рабочие процессы:

```
  PID  PPID USER    %CPU   VSZ WCHAN  COMMAND
33126     1 root     0.0  1164 pause  nginx: master process /usr/local/nginx/sb
33134 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
33135 33126 nobody   0.0  1380 kqread nginx: worker process (nginx)
33136 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
36264 33126 root     0.0  1148 pause  nginx: master process /usr/local/nginx/sb
```

```
36265 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
36266 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
36267 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
```

Теперь все рабочие процессы наравне принимают запросы. Если послать сигнал WINCH первому главному процессу, то он пошлёт своим рабочим процессам сообщение о плавном выходе, и они будут постепенно выходить:

```
  PID  PPID USER    %CPU   VSZ WCHAN  COMMAND
33126     1 root     0.0  1164 pause  nginx: master process /usr/local/nginx/sb
33135 33126 nobody   0.0  1380 kqread nginx: worker process is shutting down (n
36264 33126 root     0.0  1148 pause  nginx: master process /usr/local/nginx/sb
36265 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
36266 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
36267 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
```

> При использовании метода rtsig новые процессы могут не принимать соединения даже после того, как старому главному процессу послан сигнал WINCH. В этом случае новому главному процессу нужно посылать сигнал USR1 до тех пор, пока новые процессы не начнут принимать соединения.

По истечении времени запросы будут обрабатывать только новые рабочие процессы:

```
  PID  PPID USER    %CPU   VSZ WCHAN  COMMAND
33126     1 root     0.0  1164 pause  nginx: master process /usr/local/nginx/sb
36264 33126 root     0.0  1148 pause  nginx: master process /usr/local/nginx/sb
36265 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
36266 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
36267 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
```

Нужно заметить, что старый процесс не закрывает свои listen сокеты и при необходимости ему можно сказать, чтобы он снова запустил свои рабочие процессы. Если работа нового исполняемого файла по каким-то причинам не устраивает, то можно сделать следующее:

Послать старому главному процессу сигнал HUP. Старый процесс, не перечитывая конфигурации, запустит новые рабочие процессы. После этого можно плавно завершить новые процессы, послав их главному процессу QUIT.

Послать новому главному процессу сигнал TERM, он пошлёт сообщение о немедленном выходе рабочим процессам и все они практически сразу же завершатся. По выходу нового главного процесса старый запустит новые рабочие процессы.

Если же новые процессы не завершаются, то нужно послать им сигнал KILL. По выходу нового главного процесса старый запустит свои рабочие процессы.

Если новый главный процесс выходит, то старый процесс убирает суффикс `.oldbin` из имени файла с номером процесса.

Если же обновление прошло удачно, то старому процессу нужно послать сигнал QUIT, и у нас остаются только новые процессы:

```
  PID  PPID USER    %CPU   VSZ WCHAN  COMMAND
36264     1 root     0.0  1148 pause  nginx: master process /usr/local/nginx/sb
36265 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
36266 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
36267 36264 nobody   0.0  1364 kqread nginx: worker process (nginx)
```

Настройка хэшей

28.09.2007

Для быстрой обработки статических наборов данных, таких как имена серверов, значения директивы map, mime-types, имена строк заголовков запроса, nginx использует хэш-таблицы. Во время старта и при каждой переконфигурации nginx подбирает минимально возможный размер хэш-таблиц с учётом того, чтобы размер корзины, куда попадают ключи с совпадающими хэш-значениями, не превышал заданного параметра (hash bucket size). Размер таблицы считается в корзинах. Подбор ведётся до тех пор, пока размер таблицы не превысит параметр hash max size. Для большинства хэшей есть директивы, которые позволяют менять эти параметры, например, для хэшей имён серверов директивы называются server_names_hash_max_size и server_names_hash_bucket_size.

Параметр hash bucket size всегда выравнивается до размера, кратного размеру строки кэша процессора. Это позволяет ускорить поиск ключа в хэше на современных процессорах, уменьшив число обращений к памяти. Если hash bucket size равен размеру одной строки кэша процессора, то во время поиска ключа число обращений к памяти в худшем случае будет равно двум первый раз для определения адреса корзины, а второй при поиске ключа внутри корзины. Соответственно, если nginx выдал сообщение о необходимости увеличить hash max size или hash bucket size, то сначала нужно увеличивать первый параметр.

(C) Игорь Сысоев
http://sysoev.ru

Глобальные директивы

01.09.2008

Содержание

Пример конфигурации

```
user               www  www;
worker_processes   2;

error_log   /var/log/nginx-error.log  info;

events {
   use    kqueue;
   worker_connections   2048;
}

...
```

Директивы

---

syntax: daemon on|off
default: daemon on
context: main

Директива определяет, будет ли nginx запускаться в режиме демона. Используется в основном для разработки.

---

syntax: env VAR|VAR=VALUE
default: env TZ
context: main

Директива позволяет ограничить набор переменных среды, поменять им значения или же создать новые переменные для следующих случаев:

- наследование переменных во время обновления исполняемого файла на лету;
- использование переменных модулем ngx_http_perl_module;
- использование переменных рабочими процессами. Однако нужно иметь в виду, что управление поведением системных библиотек подобным образом возможно не всегда, поскольку зачастую библиотеки используют переменные только во время инициализации, то есть ещё до того, как их можно задать с помощью данной директивы. Исключением из этого является вышеописанное обновление исполняемого файла на лету.

Если переменная TZ не описана явно, то она всегда наследуется и всегда доступна модулю ngx_http_perl_module.

Пример использования:

```
env  MALLOC_OPTIONS;
env  PERL5LIB=/data/site/modules;
env  OPENSSL_ALLOW_PROXY_CERTS=1;
```

---

syntax: include файл|маска
default: нет
context: везде

Директива позволяет включить в конфигурацию другой файл. Файл должен содержать синтаксически законченные директивы и блоки.

Пример использования:

```
include  mime.types;
include  vhosts/*.conf;
```

---

syntax: master_process on|off
default: master_process on
context: main

Директива определяет, будут ли запускаться рабочие процессы. Используется только для разработки.

---

syntax: pid файл
default: pid nginx.pid
context: main

Директива задаёт файл, в котором хранится номер основго процесса.

---

syntax: ssl_engine устройство
default: нет
context: main

Директива задаёт название аппартного SSL-акселератора.

---

syntax: user пользователь [группа]
default: user nobody nobody
context: main

Директива задаёт пользователя и группу, с правами которого будут работать рабочие процессы. Если группа не задана, то используется группа, имя которой совпадает с именем пользователя.

---

syntax: timer_resolution время
default: нет
context: main

Директива уменьшает разрешение времени в рабочих процессах, за счёт чего уменьшается число системных вызовов gettimeofday(). По умолчанию gettimeofday() вызывается после каждой операции получения событий из ядра. С уменьшенным разрешением gettimeofday() вызывается только один раз за указанный интервал.

Пример использования:

```
timer_resolution   100ms;
```

Внутренняя реализация интервала зависит от используемого метода:

- фильтр EVFILT_TIMER при использовании kqueue;
- timer_create() при использовании eventport;
- и setitimer() во всех остальных случаях.

---

syntax: worker_rlimit_core размер
default: нет
context: main

Директива изменяет ограничение на размер core-файла RLIMIT_CORE для рабочего процесса. Используется для увеличения ограничения без перезапуска основного процесса.

---

syntax: worker_rlimit_nofile число
default: нет
context: main

Директива изменяет ограничение на число используемых файлов RLIMIT_NOFILE для рабочего процесса. Используется для увеличения ограничения без перезапуска основного процесса.

---

syntax: worker_priority число
default: worker_priority 0
context: main

Директива задаёт приоритет рабочих процессов подобно тому, как это делается командой `nice`: отрицательное число означает более высокий приоритет. Диапазон возможных значений, как правило, от -20 до 20.

Пример использования:

```
worker_priority  -10;
```

---

syntax: worker_processes число
default: worker_processes 1
context: main

Директива задаёт число рабочих процессов.

---

syntax: working_directory путь
default: нет
context: main

Директива задаёт каталог, который будет текущим для рабочего процесса. Основное применение запись core-файла, в этом случае рабочий процесс должен иметь права на запись в этот каталог.

Директивы модуля ngx_http_core_module

16.11.2009

Содержание

Директивы

---

syntax: aio [on|off|sendfile]
default: aio off
context: http, server, location

Директива (0.8.11) разрешает или запрещает использовать файловый AIO во FreeBSD и Linux.

Во FreeBSD AIO можно использовать, начиная с FreeBSD 4.3 версии. AIO можно собрать в ядре статически

```
options VFS_AIO
```

> или же подгрузить динамически

```
kldload aio
```

Во FreeBSD 5 и 6 при включении AIO статически или динамически на стадии загрузки ядра вся сетевая подсистема будет использовать GiantLock, что может негативно сказаться на производительности системы в целом. Эта зависимость устранена во FreeBSD-6.4 STABLE от 2009 года и во FreeBSD 7. Однако, начиная с FreeBSD 5.3, есть возможность включать AIO, не связывая сетевую подсистему GiantLock'ом для этого модуль AIO нужно подгружать уже после загрузки ядра. В этом случае в /var/log/messages появится сообщение

```
WARNING: Network stack Giant-free, but aio requires Giant.
Consider adding 'options NET_WITH_GIANT' or setting debug.mpsafenet=0
```

> которое можно смело проигнорировать.

> Требование использовать GiantLock в AIO связано с тем, что FreeBSD поддерживает асинхронные вызовы aio_read()/aio_write() для работы с сокетами. Но поскольку nginx использует AIO только для работы с диском, то проблем не возникает.

Для работы AIO нужно выключить sendfile:

```
location  /video/ {
    sendfile        off;
    aio             on;
    output_buffers  1 64k;
}
```

Кроме того, начиная с FreeBSD 5.2.1 и nginx-0.8.12, AIO также можно использовать для подгрузки данных для sendfile():

```
location  /video/ {
    sendfile        on;
    tcp_nopush      on;
    aio             sendfile;
}
```

В такой конфигурации используется флаг SF_NODISKIO и sendfile() не блокируется на диске, а сообщает об отсутствии данных в памяти, после чего nginx инициирует асинхронную подгрузку данных, читая только один байт. При этом ядро FreeBSD подгружает в память первые 128K файла, однако при последующих чтениях файл подгружается частями только по 16K. Изменить это можно с помощью директивы read_ahead.

В Linux AIO можно использовать, только начиная с версии ядра 2.6.22, и, кроме того, ещё необходимо дополнительно включать directio, иначе чтение будет блокирующимся:

```
location  /video/ {
    aio             on;
    directio        512;
    output_buffers  1 128k;
}
```

Поскольку directio в Linux можно использовать только для чтения блоков, выравненных по 512 байт (или 4K для XFS), то невыравненный конец файла будет читаться блокировано. То же относится к запросам части ответа byte-ranges и к запросам FLV не с начала файла: чтение невыровненных начала и конца ответа будет блокирующимся. sendfile выключать не нужно, так как при использовании directio он выключается сам.

---

syntax: alias путь
default: нет
context: location

Директива задаёт замену для указанного location'а. Например, при такой конфигурации

```
location  /i/ {
    alias  /data/w3/images/;
}
```

> на запрос "/i/top.gif" будет отдан файл "/data/w3/images/top.gif".

В значении пути можно использовать переменные.

Если директива alias используется внутри location'а, заданного регулярным выражением, то регулярное выражение должно содержать выделения, а директива alias ссылки на эти выделения (0.7.40), например:

```
location ~ ^/users/(.+\.(?:gif|jpe?g|png))$ {
    alias  /data/w3/images/$1;
}
```

Если location и последняя часть значения директивы совпадают:

```
location  /images/ {
    alias  /data/w3/images/;
}
```

то лучше воспользоваться директивой root:

```
location  /images/ {
   root   /data/w3;
}
```

---

syntax: client_body_in_file_only on|clean|off
default: client_body_in_file_only off
context: http, server, location

Директива определяет, сохранять ли всё тело запроса клиента в файл. Директиву можно использовать для отладки и при использовании переменной $request_body_file или метода $r->request_body_file модуля ngx_http_perl_module.

При использовании параметра "on" временные файлы по окончании обработки запроса не удаляется.

Параметр "clean" разрешает удалять временные файлы, оставшиеся по окончании обработки запроса.

---

syntax: client_body_in_single_buffer on|off
default: client_body_in_single_buffer off
context: http, server, location

Директива определяет, хранить ли всё тело запроса клиента в одном буфере. Директива рекомендуется при использовании переменной $request_body для уменьшения операций копирования.

---

syntax: client_body_buffer_size размер
default: client_body_buffer_size 8k/16k
context: http, server, location

Директива задаёт размер буфера для чтения тела запроса клиента. Если тело запроса больше заданного буфера, то всё тело запроса или только его часть записывается во временный файл. По умолчанию размер одного буфера равен двум размерам страницы, в зависимости от платформы это или 8K, или 16K.

---

syntax: client_body_temp_path путь [уровень1 [уровень2 [уровень3]]]
default: client_body_temp_path client_body_temp
context: http, server, location

Директива задаёт имя каталога для хранения временных файлов с телом запроса клиента. В каталоге может использоваться иерархия подкаталогов до трёх уровней. Например, при такой конфигурации

```
client_body_temp_path  /spool/nginx/client_temp 1 2;
```

> имя временного будет такого вида:

```
/spool/nginx/client_temp/7/45/00000123457
```

---

syntax: client_body_timeout время
default: client_body_timeout 60
context: http, server, location

Директива задаёт таймаут при чтении тела запроса клиента. Таймаут устанавливается не на всю передачу тела запроса, а только между двумя операциями чтения. Если по истечении этого времени клиент ничего не передаст, то ему возвращается ошибка "Request time out" (408).

---

syntax: client_header_buffer_size размер
default: client_header_buffer_size 1k
context: http, server

Директива задаёт размер буфера для чтения заголовка запроса клиента. Для подавляющего большинства запросов вполне достаточно буфера размером в 1K. Однако если в запросе есть большие cookies или же запрос пришёл от wap-клиента, то он может не поместиться в 1K. Поэтому, если строка запроса или строка заголовка запроса не помещается полностью в этот буфер, то выделяются большие буферы, задаваемые директивой large_client_header_buffers.

---

syntax: client_header_timeout время
default: client_header_timeout 60
context: http, server

Директива задаёт таймаут при чтении заголовка запроса клиента. Если по истечении этого времени клиент не передаст полностью заголовок запроса, то

ему возвращается ошибка "Request time out" (408).

---

syntax: client_max_body_size размер
default: client_max_body_size 1m
context: http, server, location

Директива задаёт максимально допустимый размер тела запроса клиента, указываемый в строке "Content-Length" в заголовке запроса. Если размер больше заданного, то клиенту возвращается ошибка "Request Entity Too Large" (413). Следует иметь в виду, что браузеры не умеют корректно показывать эту ошибку.

---

syntax: default_type MIME-тип
default: default_type text/plain
context: http, server, location

Директива задаёт MIME-тип ответов по умолчанию.

---

syntax: directio [размер|off]
default: directio off
context: http, server, location

Директива (0.7.7) разрешает использовать флаги O_DIRECT (FreeBSD, Linux), F_NOCACHE (Mac OS X) или функцию directio() (Solaris) при чтении файлов, размер которых больше либо равен указанному. Директива автоматически запрещает (0.7.15) использование sendfile'a для данного запроса. Рекомендуется использовать для больших файлов:

```
directio  4m;
```

или при использовании aio в Linux.

---

syntax: directio_alignment размер
default: directio_alignment 512
context: http, server, location

Директива (0.8.11) устанавливает выравнивание для directio. В большинстве случае достаточно выравнивания 512 байт, однако при использовании XFS под

Linux его нужно увеличить до 4K.

---

syntax: error_page код [код ...] [=|=ответ] uri
default: нет
context: http, server, location, if в location

Директива задаёт URI, который будет показываться для указанных ошибок. Директивы наследуются с предыдущего уровня при условии, что на данном уровне не описаны свои директивы error_page. В URI можно использовать переменные.

Пример использования:

```
error_page   404          /404.html;
error_page   502 503 504  /50x.html;
error_page   403          http://example.com/forbidden.html;
```

Кроме того, можно поменять код ответа на другой, например:

```
error_page   404  =200  /empty.gif;
```

Если ошибочный ответ обрабатывается проксированным сервером или FastCGI-сервером и этот сервер может вернуть разные коды ответов, например, 200, 302, 401 или 404, то можно выдавать возвращаемый код:

```
error_page   404  =  /404.php;
```

Если при перенаправлении не нужно менять URI, то можно перенаправить обработку ошибки в именованный location:

```
location / {
   error_page   404  =  @fallback;
}

location @fallback {
   proxy_pass   http://backend;
}
```

---

syntax: if_modified_since [off|exact|before]
default: if_modified_since exact
context: http, server, location

Директива (0.7.24) определяет, как сравнивать время модификации ответа и время в заголовке запроса "If-Modified-Since":

- off — не проверять заголовок запроса "If-Modified-Since" (0.7.34);
- exact — точно совпадение;
- before — время модификации ответа меньше или равно времени, заданному в заголовке запроса "If-Modified-Since".

---

syntax: internal
default: нет
context: location

Директива указывает, что данный location может использоваться только для внутренних запросов. Для внешних запросов будет возвращаться ошибка "Not found" (404). Внутренними запросами являются

- запросы, перенаправленные директивой error_page;
- подзапросы, формируемые командой include virtual модуля ngx_http_ssi_module;
- запросы, изменённые директивой rewrite модуля ngx_http_rewrite_module.

Пример использования:

```
error_page   404   /404.html;

location  /404.html {
   internal;
}
```

---

syntax: keepalive_requests число
default: keepalive_requests 100
context: http, server, location

Директива (0.8.0) задаёт максимальное число запросов, которые можно сделать по одному keep-alive соединению.

---

syntax: keepalive_timeout время [время]
default: keepalive_timeout 75
context: http, server, location

Директива задаёт таймаут, в течение которого keep-alive соединение с клиентом не будет закрыто со стороны сервера. Второй параметр задаёт

значение в строке "Keep-Alive: timeout=время" в заголовке ответа. Параметры могут отличаться друг от друга. Строку "Keep-Alive: timeout=время" понимают Mozilla и Konqueror. MSIE сам закрывает keep-alive соединение примерно через 60 секунд.

---

syntax: large_client_header_buffers число размер
default: large_client_header_buffers 4 4k/8k
context: http, server

Директива задаёт максимальное число и размер буферов для чтения большого заголовка запроса клиента. Строка запроса должна быть не больше размера одного буфера, иначе клиенту возвращается ошибка "Request URI too large" (414). Длинная строка заголовка запроса также должна быть не больше размера одного буфера, иначе клиенту возвращается ошибка "Bad request" (400). Буферы выделяются только по мере необходимости. По умолчанию размер одного буфера равен размеру страницы, в зависимости от платформы это или 4K, или 8K. Если по окончании обработки запроса соединение переходит в состояние keep-alive, то эти буферы освобождаются.

---

syntax: limit_except методы { ... }
default: нет
context: location

Директива ограничивает HTTP-методы, доступные внутри location. Метод GET также включает в себя метод HEAD. Для ограничения могут использоваться директивы модулей ngx_http_access_module и ngx_http_auth_basic_module:

```
limit_except  GET {
   allow  192.168.1.0/32;
   deny   all;
}
```

Обратите внимание, что данное ограничение будет выполняться для всех методов, кроме методов GET и HEAD.

---

syntax: limit_rate скорость
default: нет
context: http, server, location, if в location

Директива задаёт скорость передачи ответа клиенту. Скорость задаётся в байтах в секунду. Ограничение работает только для одного соединения, то есть, если клиент откроет 2 соединения, то суммарная скорость будет в 2 раза выше ограниченной.

Если необходимо ограничить скорость для части клиентов на уровне сервера, то директива limit_rate для этого не подходит. Вместо этого следует задать нужную скорость переменной $limit_rate:

```
server {

   if ($slow) {
       set $limit_rate  4k;
   }

   ...
}
```

---

syntax: limit_rate_after размер
default: нет
context: http, server, location, if в location

Директива (0.8.0) задаёт объём данных, после передачи которого начинает ограничиватся скорость передачи ответа клиенту, например:

```
location /flv/ {
   flv;
   limit_rate_after  500k;
   limit_rate        50k;
}
```

---

syntax: listen адрес:порт [default|default_server| [backlog=число | rcvbuf=размер | sndbuf=размер | accept_filter=фильтр | deferred | bind | ipv6only=[on|off] | ssl]]
default: listen *:80 | *:8000
context: server

Директива задаёт адрес и порт, на которых сервер принимает запросы. Можно указать только адрес или только порт, кроме того, адрес может быть именем сервера, например:

```
listen  127.0.0.1:8000;
listen  127.0.0.1;
listen  8000;
listen  *:8000;
listen  localhost:8000;
```

> адреса IPv6 (0.7.36) задаются в квадратных скобках:

```
listen  [::]:8000;
listen  [fe80::1];
```

Если указан только адрес, то используется порт 80.

Если директива не указана, то используется порт *:80, если nginx работает с правами пользователя root, или порт *:8000.

Если у директивы есть параметр default, то сервер, в котором описана эта директива, будет сервером по умолчанию для указанной пары адрес:порт. Если же директив с параметром default нет, то сервером по умолчанию будет первый сервер, в котором описана пара адрес:порт. Начиная с версии 0.8.21, можно использовать параметр default_server.

В директиве listen с параметром default можно также указать несколько параметров, специфичных для системных вызовов listen(2) и bind(2). Начиная с версии 0.8.21, эти параметры можно задать в любой директиве listen, но только один раз для указанной пары адрес:порт.

- backlog=число — задаёт параметр backlog в вызове listen(2). По умолчанию backlog равен -1 для FreeBSD и 511 для всех остальных платформ.
- rcvbuf=размер — задаёт параметр SO_RCVBUF для слушающего сокета.
- sndbuf=размер — задаёт параметр SO_SNDBUF для слушающего сокета.
- accept_filter=фильтр — задаёт название accept-фильтра. Работает только на FreeBSD, можно использовать два фильтра — dataready и httpready. По сигналу -HUP accept-фильтр можно менять только в последних версиях FreeBSD, начиная с 6.0, 5.4-STABLE и 4.11-STABLE.
- deferred — указывает использовать отложенный accept(2) на Linux с помощью опции TCP_DEFER_ACCEPT.
- bind — указывает, что для данной пары адрес:порт нужно делать bind(2) отдельно. Дело в том, что если описаны несколько директив listen с одинаковым портом, но разными адресами и одна из директив listen слушает на всех адресах для данного порта (*:порт), то nginx сделает bind(2) только на *:порт. Необходимо учитывать, что в этом случае для определения адреса, на которой пришло соединение, делается системный вызов getsockname(). Если же используются параметры backlog, rcvbuf, sndbuf, accept_filter или deferred, то для данной пары адрес:порт bind(2) всегда делается отдельно.
- ipv6only — параметр (0.7.42) задаёт значение параметра IPV6_V6ONLY для слушающего сокета. Установить этот параметр можно только один раз на старте.

• ssl — параметр (0.7.14) не имеет отношения к системным вызовам listen(2) и bind(2), а позволяет указать, что все соединения, принимаемые на этом порту, должны работать в режиме SSL. Это позволяет задать компактную конфигурацию для сервера, работающего сразу в двух режимах — HTTP и HTTPS.

```
listen  80;
listen  443 default ssl;
```

Пример использования параметров:

```
listen  127.0.0.1 default accept_filter=dataready backlog=1024;
```

---

syntax: location [=|~|~*|^~|@] /uri/ { ... }
default: нет
context: server

Директива устанавливает конфигурацию в зависимости от URI запроса. location можно задать обычной строкой или регулярным выражением. Регулярные выражения задаются префиксом "~*" — без учёта регистра символов, и "~" — с учётом. Для определения соответствия location'а и запроса сначала проверяются location'ы, заданные обычными строками. Среди них ищется максимальное совпадение. Затем проверяются регулярные выражения. В отличие от обычных строк, они не сортируются, а проверяются в порядке их следования в конфигурационном файле. Проверка регулярных выражений прекращается после первого же совпадения. Если совпадение с регулярным выражением не найдено, то используется конфигурация максимально совпавшего location'а.

Для операционных систем, не чувствительных к регистру символов, таких как Mac OS X и Cygwin, проверка обычных строк делается без учёта регистра (0.7.7). Однако, сравнение ограничено только однобайтными locale'ями.

Регулярное выражение может содержать выделения (0.7.40), которые могут затем использоваться в других директивах.

Если нужно запретить проверку регулярных выражений после проверки обычных строк, то это можно сделать с помощью префикса "^~". Если у максимально совпавшего location'а есть этот префикс, то регулярные выражения не проверяются.

Кроме того, с помощью префикса "=" можно задать точное совпадение URI и location. При совпадении поиск сразу же прекращается, так как дальше искать

не имеет смысла. Например, если запрос "/" очень частый, то указав "location = /", можно ускорить обработку этого запроса, так как поиск location прекратится после первого же сравнения.

В версиях с 0.7.1 по 0.8.41, если запрос точно совпал с обычным location'ом без префиксов "=" и "^~", то поиск тоже сразу же прекращается и регулярные выражения также не проверяются.

Проиллюстрируем вышесказанное примером:

```
location  = / {
   [ конфигурация A ]
}

location  / {
   [ конфигурация B ]
}

location ^~ /images/ {
   [ конфигурация C ]
}

location ~* \.(gif|jpg|jpeg)$ {
   [ конфигурация D ]
}
```

> Для запроса "/" будет выбрана конфигурация A, для запроса "/documents/document.html" □ конфигурация B, для запроса "/images/1.gif" □ конфигурация C, для запроса "/documents/1.jpg" □ конфигурация D.

Префикс "@" задаёт именованный location. Такой location не используется при обычной обработке запросов, а предназначен только для перенаправления в него запросов.

---

syntax: log_not_found [on|off]
default: log_not_found on
context: http, server, location

Директива разрешает или запрещает записывать в error_log ошибки о том, что файл не найден.

---

syntax: log_subrequest [on|off]
default: log_subrequest off
context: http, server, location

Директива разрешает или запрещает записывать в access_log подзапросы.

---

syntax: merge_slashes [on|off]
default: merge_slashes on
context: http, server

Директива разрешает или запрещает объединять в URI два и более слэшей в один.

Необходимо иметь ввиду, что это объединение необходимо для корректной проверки location'ов и регулярных выражений. Например, запрос "//scripts/one.php" не попадает в

```
location /scripts/ {
   ...
}
```

> и может быть обслужен как статический файл, поэтому он приводится в "/scripts/one.php".

Выключение объединения может понадобиться, если в URI используются имена, закодированные методом base64, который использует символ "/". Но по соображениям безопасности лучше избегать выключения объединения.

Если директива указана на уровне server в сервере по умолчанию, то её значение распространяется на все виртуальные сервера, слушающие на том же адресе и порту.

---

syntax: msie_padding [on|off]
default: msie_padding on
context: http, server, location

Директива разрешает или запрещает добавлять в ответы для MSIE со статусом больше 400 комментарий для увеличения размера ответа до 512 байт.

---

syntax: msie_refresh [on|off]
default: msie_refresh off
context: http, server, location

Директива разрешает или запрещает выдавать для MSIE refresh'ы вместо редиректов.

---

syntax: open_file_cache max=N [inactive=время]|off
default: open_file_cache off
context: http, server, location

Директива задаёт кэш, в котором могут хранится

- дескрипторы открытых файлов, информация об их размерах и времени модификации;
- информация о существовании каталогов;
- информация об ошибках поиска файла нет файла, нет прав на чтение и тому подобное. Кэширование ошибок нужно разрешить директивой open_file_cache_errors.

Параметры директивы:

- max задаёт максимальное число элементов в кэше; при переполнении кэша удаляются наиболее давно не используемые элементы (LRU);
- inactive задаёт время, после которого элемент кэша удаляется, если к нему не было обращений в течение этого времени; по умолчанию 60 секунд;
- off запрещает кэш.

Пример использования:

```
open_file_cache          max=1000  inactive=20s;
open_file_cache_valid    30s;
open_file_cache_min_uses 2;
open_file_cache_errors   on;
```

---

syntax: open_file_cache_errors on|off
default: open_file_cache_errors off
context: http, server, location

Директива определяет, кэшировать или нет ошибки поиска файлов в open_file_cache.

---

syntax: open_file_cache_min_uses число
default: open_file_cache_min_uses 1

context: http, server, location

Директива определяет минимальное число использований файла в течение времени, заданного параметром inactive в директиве open_file_cache, после которого дескриптор файла будет оставаться открытым в кэше.

---

syntax: open_file_cache_valid время
default: open_file_cache_valid 60
context: http, server, location

Директива определяет, через какое время нужно проверять актуальность информации об элементе в open_file_cache.

---

syntax: optimize_server_names [on|off]
default: optimize_server_names on
context: http, server

Устаревшая директива.

Директива разрешает или запрещает оптимизировать проверку имени хоста в name-based виртуальных серверах. Проверка в частности влияет на имя хоста, используемого в редиректах. Если оптимизация разрешена и все name-based сервера, слушающие на одной паре адрес:порт, имеют одинаковую конфигурацию, то во время исполнения запроса имена не проверяются и в редиректах используется первое имя сервера. Если в редиректе нужно использовать имя хоста, переданное клиентом, то оптимизацию нужно выключить.

---

syntax: port_in_redirect [on|off]
default: port_in_redirect on
context: http, server, location

Директива разрешает или запрещает указывать порт в редиректах, выдаваемых nginx'ом.

---

syntax: read_ahead размер
default: read_ahead 0

context: http, server, location

Директива задаёт ядру размер предчтения при работе с файлами. Под Линуксом используется системный вызов

```
posix_fadvise(0, 0, 0, POSIX_FADV_SEQUENTIAL);
```

> поэтому размер игнорируется.

Под FreeBSD используется fcntl(O_READAHEAD, размер), появившийся во FreeBSD-9 CURRENT. Для FreeBSD 7 нужно установить патч.

---

syntax: recursive_error_pages [on|off]
default: recursive_error_pages off
context: http, server, location

Директива разрешает или запрещает делать несколько перенаправлений через директиву error_page.

---

syntax: reset_timedout_connection [on|off]
default: reset_timedout_connection off
context: http, server, location

Директива разрешает или запрещает сбрасывать соединение по таймауту. Сброс делается следующим образом перед закрытием сокета для него ставится опция SO_LINGER с таймаутом 0. После чего при закрытии сокета клиенту отсылается пакет RST, а всё память, связанная с этим сокетом, освобождается. Это позволяет избежать длительного нахождения уже закрытого сокета в состоянии FIN_WAIT1 с заполненными буферами.

Необходимо отметить, что соединения, находящиеся в состоянии keepalive, по истечении таймаута закрываются обычным образом.

---

syntax: resolver адрес
default: нет
context: http, server, location

Директива задаёт адрес name-сервера, например:

```
    resolver    127.0.0.1;
```

---

syntax: resolver_timeout время
default: resolver_timeout 30s
context: http, server, location

Директива задаёт таймаут для определения имени, например:

```
    resolver_timeout    5s;
```

---

syntax: root путь
default: root html
context: http, server, location, if в location

Директива задаёт корневой каталог для запросов. Например, при такой конфигурации

```
    location  /i/ {
        root  /data/w3;
    }
```

> на запрос "/i/top.gif" будет отдан файл "/data/w3/i/top.gif".

В значении пути можно использовать переменные.

Путь к файлу формируется как простое добавление URI к значению директивы root. Если же необходима модификация URI, то нужно воспользоваться директивой alias.

---

syntax: satisfy all|any
default: satisfy all
context: location

Директива разрешает доступ при хотя бы одной успешной проверке, выполненной модулями ngx_http_access_module или ngx_http_auth_basic_module:

```
location / {
    satisfy  any;

    allow  192.168.1.0/32;
    deny   all;

    auth_basic            "closed site";
```

```
    auth_basic_user_file  conf/htpasswd;
}
```

---

syntax: satisfy_any on|off
default: satisfy_any off
context: location

Директива переименована в директиву satisfy.

---

syntax: send_timeout время
default: send_timeout 60
context: http, server, location

Директива задаёт таймаут при передаче ответа клиенту. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями записями. Если по истечении этого времени клиент ничего не примет, то nginx закрывает соединение.

---

syntax: sendfile [on|off]
default: sendfile off
context: http, server, location

Директива разрешает или запрещает использовать sendfile().

---

syntax: server { ... }
default: нет
context: http

Директива задаёт конфигурацию для виртуального сервера. Чёткого разделения виртуальных серверов ip-based (на основании ip-адреса) и name-based (на основании имени, передаваемого в строке "Host" заголовка запроса), нет. Вместо этого директивами listen описываются все адреса и порты, на которых нужно принимать соединения для этого сервера, и в директиве server_name указываются все имена серверов. Пример конфигурации описан в настройке виртуальных серверов.

---

syntax: server_name имя [...]
default: server_name hostname
context: server

Директива задаёт имена виртуального сервера, например:

```
server {
   server_name   example.com  www.example.com;
}
```

Первое имя становится основным именем сервера. По умолчанию используется имя машины (hostname). В именах серверов можно использовать "*" для замены первой или последней части имени:

```
server {
   server_name   example.com  *.example.com  www.example.*;
}
```

Два первых вышеприведённых имени можно объединить в одно:

```
server {
   server_name   .example.com;
}
```

Кроме того, в качестве имени сервера можно использовать регулярное выражение, указав перед ним "~":

```
server {
   server_name   www.example.com   ~^www\d+\.example\.com$;
}
```

Регулярное выражение может содержать выделения (0.7.40), которые могут затем использоваться в других директивах:

```
server {
   server_name   ~^(www\.)?(.+)$;

   location / {
       root  /sites/$2;
   }
}

server {
   server_name   _;

   location / {
       root  /sites/default;
   }
}
```

Начиная с 0.8.25, именованные выделения в регулярном выражении создают переменные, которые могут затем использоваться в других директивах:

```
server {
   server_name   ~^(www\.)?(?<domain>.+)$;

   location / {
       root  /sites/$domain;
   }
}

server {
   server_name   _;

   location / {
       root  /sites/default;
   }
}
```

Начиная с 0.7.11, можно использовать пустое имя "":

```
server {
   server_name   www.example.com   "";
}
```

> что позволяет обрабатывать запросы без строки "Host" в заголовке запроса в этом сервере, а не в сервере по умолчанию для данной пары адрес:порт.

Порядок проверки имён следующий:

- полные имена,
- имена с маской в начале имени *.example.com,
- имена с маской в конце имени mail.*,
- регулярные выражения.

---

syntax: server_name_in_redirect [on|off]
default: server_name_in_redirect on
context: http, server, location

Директива разрешает или запрещает использовать в редиректах, выдаваемых nginx'ом, основное имя сервера, задаваемое директивой server_name. Если использование основного имени запрещено, то используется имя, указанного в строке "Host" в заголовке запроса. Если же этой строки нет, то используется IP-адрес сервера.

---

syntax: server_names_hash_max_size число
default: server_names_hash_max_size 512
context: http

Директива задаёт максимальный размер хэш-таблиц имён серверов. Подробнее смотри в описании настройки хэшей.

---

syntax: server_names_hash_bucket_size число
default: server_names_hash_bucket_size 32/64/128
context: http

Директива задаёт размер корзины в хэш-таблицах имён серверов. Значение по умолчанию зависит от размера строки кэша процессора. Подробнее смотри в описании настройки хэшей.

---

syntax: server_tokens [on|off]
default: server_tokens on
context: http, server, location

Директива разрешает или запрещает выдавать версию nginx'а в сообщениях об ошибках и в строке заголовка ответа "Server".

---

syntax: tcp_nodelay [on|off]
default: tcp_nodelay on
context: http, server, location

Директива разрешает или запрещает использовать опцию TCP_NODELAY. Опция включаются только при переходе соединения в состояние keep-alive.

---

syntax: tcp_nopush [on|off]
default: tcp_nopush off
context: http, server, location

Директива разрешает или запрещает использовать опции TCP_NOPUSH во FreeBSD или TCP_CORK в Linux. Опции включаются только при использовании sendfile. Включение опции позволяет

- передавать заголовок ответа и начало файла в одном пакете в Linux и во FreeBSD 4.x;
- передавать файл в полных пакетах.

---

syntax: try_files файл [файл ...] (uri|=код)
default: нет
context: location

Директива проверяет существование файлов в заданном порядке и использует для обработки запроса первый найденный файл, причём обработка делается в контексте этого же location'а. С помощью слэша в конце имени можно задать проверку существования каталога, например, так "$uri/". В случае, если ни один файл не найден, то делается внутренний редирект на последний параметр. Последний параметр может быть кодом (0.7.51):

```
location / {
   try_files      $uri  $uri/index.html  $uri.html  =404;
}
```

Пример использования при проксировании Mongrel:

```
location / {
   try_files      /system/maintenance.html
                  $uri  $uri/index.html  $uri.html
                  @mongrel;
}

location @mongrel {
   proxy_pass     http://mongrel;
}
```

Пример использования вместе с Drupal/FastCGI:

```
location / {
   try_files      $uri  $uri/  @drupal;
}

location ~ \.php$ {
   try_files      $uri  @drupal;

   fastcgi_pass   ...;

   fastcgi_param  SCRIPT_FILENAME  /path/to$fastcgi_script_name;
   fastcgi_param  SCRIPT_NAME      $fastcgi_script_name;
   fastcgi_param  QUERY_STRING     $args;

   ... прочие fastcgi_param
}

location @drupal {
```

```
    fastcgi_pass   ...;

    fastcgi_param  SCRIPT_FILENAME  /path/to/index.php;
    fastcgi_param  SCRIPT_NAME      /index.php;
    fastcgi_param  QUERY_STRING     q=$uri&$args;

    ... прочие fastcgi_param
}
```

> В этом примере директива try_files

```
location / {
    try_files      $uri  $uri/  @drupal;
}
```

> аналогична директивам

```
location / {
    error_page     404 = @drupal;
    log_not_found  off;
}
```

> А здесь

```
location ~ \.php$ {
    try_files      $uri  @drupal;

    fastcgi_pass   ...;

    fastcgi_param  SCRIPT_FILENAME  /path/to$fastcgi_script_name;

    ...
}
```

> try_files тестирует существование PHP-файла, прежде чем передать запрос FastCGI-серверу.

Пример использования вместе с Wordpress и Joomla:

```
location / {
    try_files      $uri  $uri/  @wordpress;
}

location ~ \.php$ {
    try_files      $uri  @wordpress;

    fastcgi_pass   ...;

    fastcgi_param  SCRIPT_FILENAME  /path/to$fastcgi_script_name;
    ... прочие fastcgi_param
}

location @wordpress {
    fastcgi_pass   ...;

    fastcgi_param  SCRIPT_FILENAME  /path/to/index.php;
    ... прочие fastcgi_param
```

```
}
```

---

syntax: types { ... }
context: http, server, location

Директива задаёт соответствие расширения и MIME-типов ответов. Одному MIME-типу может соответствовать несколько расширений. По умолчанию используется такие соответствия:

```
types {
    text/html    html;
    image/gif    gif;
    image/jpeg   jpg;
}
```

Достаточно полная таблица соответствий входит в дистрибутив и находится в файле conf/mime.types.

Для того, чтобы для определённого location'а для всех ответов выдавался MIME-тип "application/octet-stream", можно использовать следующее:

```
location /download/ {
    types         { }
    default_type  application/octet-stream;
}
```

---

syntax: underscores_in_headers [on|off]
default: underscores_in_headers off
context: http, server

Директива разрешает или запрещает использование символов подчёркивания в строках заголовка запроса клиента.

Встроенные переменные

Модуль ngx_http_core_module поддерживает встроенные переменные, имена которых совпадают с именами переменных в Apache. Прежде всего, это переменные, представляющие из себя строки заголовка запроса клиента, например, $http_user_agent, $http_cookie и тому подобное. Кроме того, есть и другие переменные:

- $args, эта переменная равна аргументам в строке запроса;
- $arg_name, эта переменная равна аргументу name в строке запроса;

- $binary_remote_addr, эта переменная равна адресу клиента в бинарном виде, длина её значения всегда 4 байта;
- $content_length, эта переменная равна строке "Content-Length" в заголовке запроса;
- $content_type, эта переменная равна строке "Content-Type" в заголовке запроса;
- $cookie_name, эта переменная равна cookie name;
- $document_root, эта переменная равна значению директивы root для текущего запроса;
- $document_uri, то же самое, что и $uri;
- $host, эта переменная равна строке "Host" в заголовке запроса или имени сервера, на который пришёл запрос, если этой строки нет;
- $hostname, эта переменная равна имени хоста;
- $http_name, эта переменная равна строке name в заголовке запроса;
- $is_args, эта переменная равна "?", если в строке запроса есть аргументы, и пустой строке, если их нет;
- $limit_rate, эта переменная позволяет установить ограничение скорости соединения;
- $pid, эта переменная равна номеру рабочего процесса;
- $request_method, эта переменная равна методу запроса, обычно это "GET" или "POST";
- $remote_addr, эта переменная равна адресу клиента;
- $remote_port, эта переменная равна порту клиента;
- $remote_user, эта переменная равна имени пользователя, используемого в Basic аутентификации;
- $realpath_root, эта переменная равна значению директивы root для текущего запроса, при этом все символические ссылки преобразованы в реальные путь;
- $request_filename, эта переменная равна пути к файлу для текущего запроса, формируемому из директив root или alias и URI запроса;
- $request_body, эта переменная содержит тело запроса. Значение переменной появляется в location'ах, обрабатываемых директивами proxy_pass и fastcgi_pass.
- $request_body_file, эта переменная равна имени временного файла, в котором хранится тело запроса. По завершению работы файл необходимо удалить. Для того, чтобы тело запроса клиента всегда записывалось в файл, нужно указать client_body_in_file_only on. При передаче имени в проксированном запросе или в запросе к FastCGI-серверу следует запретить передачу самого тела директивами "proxy_pass_request_body off" или "fastcgi_pass_request_body off" соответственно.

- $request_uri, эта переменная равна полному первоначальному URI вместе с аргументами;
- $query_string, то же самое, что и $args;
- $scheme, эта переменная равна схеме запроса "http" или "https";
- $server_protocol, эта переменная равна протоколу запроса, обычно это "HTTP/1.0" или "HTTP/1.1";
- $server_addr, эта переменная равна адресу сервера, на который пришёл запрос. Как правило, для получения значения этой переменной делается один системный вызов. Для того, чтобы избежать системного вызова, нужно указывать адреса в директивах listen и использовать параметр bind;
- $server_name, эта переменная равна имени сервера, на который пришёл запрос;
- $server_port, эта переменная равна порту сервера, на который пришёл запрос;
- $uri, эта переменная равна текущему URI в запросе, он может отличаться от первоначального, например, при внутренних редиректах или при использовании индексных файлов.

Директивы модуля ngx_http_upstream

15.08.2007

Модуль позволяет описывать группы серверов, которые могут использоваться в директивах proxy_pass и fastcgi_pass.

Содержание

Пример конфигурации

```
upstream  backend  {
   server   backend1.example.com        weight=5;
   server   backend2.example.com:8080;
   server   unix:/tmp/backend3;

   server   backup1.example.com:8080    backup;
   server   backup2.example.com:8080    backup;
}

server {
   location / {
       proxy_pass  http://backend;
   }
}
```

Директивы

---

syntax: ip_hash
default: нет
context: upstream

Директива задаёт метод распределения запросов по серверам на основе IP-адресов клиентов. В качестве ключа для хеширования используется сеть класса C, в которой находится адрес клиента. Метод гарантирует, что запросы клиента будут передаваться на один и тот же сервер. Если же этот сервер

будет считаться неработающим, то запросы этого клиента будут передаваться на другой сервер. С большой долей вероятности это также будет один и тот же сервер.

Для серверов, использующих метод распределения ip_hash, нельзя задать вес. Если один из серверов нужно убрать на некоторое время, то для сохранения текущего хеширования IP-адресов клиентов этот сервер нужно пометить параметром down.

Пример конфигурации:

```
upstream  backend  {
   ip_hash;

   server   backend1.example.com;
   server   backend2.example.com;
   server   backend3.example.com  down;
   server   backend4.example.com;
}
```

---

syntax: server название [параметры]
default: нет
context: upstream

Директива задаёт имя и параметры сервера. В качестве имени можно использовать доменное имя, адрес, порт или путь unix-сокета. Если доменное имя резолвится в несколько адресов, то используются все.

- weight=число задаёт вес сервера, по умолчанию вес равен одному.
- max_fails=число задаёт число неудачных попыток работы с сервером в течение времени, заданного параметром fail_timeout, после которых он считается неработающим также в течение времени заданного параметром fail_timeout. По умолчанию число попыток равно одной. Нулевое значение запрещает учёт попыток. Что считается неудачной попыткой, задаётся директивами proxy_next_upstream и fastcgi_next_upstream. Состояние http_404 не считается неудачной попыткой.
- fail_timeout=время задаёт
- время, в течение которого должно произойти заданное число неудачных попыток работы с сервером для того, чтобы сервер считался неработающим;
- и время, в течение которого сервер будет считаться неработающим.

По умолчанию время равно 10 секундам.

- backup помечает сервер как запасной сервер. На него будут передаваться запросы в случае, если не работают основные сервера.

• down помечает сервер как постоянно неработающий, используется совместно с директивой ip_hash.

Пример конфигурации:

```
upstream  backend  {
   server   backend1.example.com       weight=5;
   server   127.0.0.1:8080             max_fails=3  fail_timeout=30s;
   server   unix:/tmp/backend3;

   server   backup1.example.com:8080   backup;
}
```

---

syntax: upstream название { ... }
default: нет
context: http

Директива описывает группу серверов. Сервера могут слушать на разных портах, кроме того, можно одновременно использовать сервера, слушающие на TCP и unix сокетах.

Пример конфигурации:

```
upstream  backend  {
   server   backend1.example.com    weight=5;
   server   127.0.0.1:8080          max_fails=3  fail_timeout=30s;
   server   unix:/tmp/backend3;
}
```

Запросы распределяются по серверам в режиме round-robin с учётом весов серверов. В вышеприведённом примере каждые 7 семь запросов будут распределены так: 5 запросов на backend1.example.com и по одному запросу на второй и третий сервера. Если при попытке работы с сервером произошла ошибка, то запрос будет передан следующему серверу и так до тех пор, пока не будут опробованы все работающие сервера. Если не удастся получить успешный ответ от всех серверов, то клиенту будет возвращён результат работы с последним сервером.

Встроенные переменные

Модуль ngx_http_upstream поддерживает следующие встроенные переменные:

• $upstream_addr в переменной хранятся ip-адрес и порт сервера или путь к unix-сокету. Если при обработке запроса были сделаны обращения к нескольким серверам, то их адреса разделяются запятой, например,

"192.168.1.1:80, 192.168.1.2:80, unix:/tmp/sock". Если произошёл внутренний редирект от одной группы серверов на другую с помощью "X-Accel-Redirect" или error_page, то эти группы серверов разделяются двоеточием, например, "192.168.1.1:80, 192.168.1.2:80, unix:/tmp/sock : 192.168.10.1:80, 192.168.10.2:80".

- $upstream_response_time в переменной хранятся времена ответов серверов в секундах с точностью до миллисекунд. Несколько ответов также разделяются запятыми и двоеточиями.
- $upstream_status в переменной хранятся коды ответов серверов. Несколько ответов также разделяются запятыми и двоеточиями.
- $upstream_http_... в переменных хранятся строки заголовков ответов серверов, например, строка заголовка ответа "Server" доступна в переменной $upstream_http_server. Необходимо иметь ввиду, что запоминаются только строки последнего сервера.

Директивы модуля ngx_http_access_module

07.04.2006

Модуль ngx_http_access_module позволяет закрыть доступ для определённых IP-адресов клиентов.

Содержание

Пример конфигурации

```
location  /  {
    deny    192.168.1.1;
    allow   192.168.1.0/24;
    allow   10.1.1.0/16;
    deny    all;
}
```

Правила проверяются в порядке их записи до первого соответствия. В данном примере доступ разрешён только для сетей 10.1.1.0/16 и 192.168.1.0/24, кроме адреса 192.168.1.1. Если правил много, то лучше воспользоваться переменными модуля ngx_http_geo_module.

Директивы

---

syntax: allow [адрес|CIDR|all]
default: нет
context: http, server, location, limit_except

Директива разрешает доступ для указанной сети или адреса.

---

syntax: deny [адрес|CIDR|all]
default: нет
context: http, server, location, limit_except

Директива запрещает доступ для указанной сети или адреса.

Директивы модуля ngx_http_addition_module

12.08.2008

Модуль ngx_http_addition_module это фильтр, добавляющий текст до и после ответа. По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_addition_module`.

Содержание

Пример конфигурации

```
location / {
    add_before_body   /before_action;
    add_after_body    /after_action;
}
```

Директивы

---

syntax: add_before_body uri
default: нет
context: location

Директива добавляет перед телом ответа текст, выдаваемый в результате работы заданного подзапроса.

---

syntax: add_after_body uri
default: нет
context: location

Директива добавляет после тела ответа текст, выдаваемый в результате работы заданного подзапроса.

---

syntax: addition_types mime-тип [mime-тип ...]
default: addition_types text/html

context: http, server, location

Директива (0.7.9) разрешает добавлять текст в ответах с указанными MIME-типами в дополнение к "text/html".

Директивы модуля ngx_http_auth_basic_module

07.04.2006

Модуль ngx_http_auth_basic_module позволяет закрыть доступ с проверкой имени и пароля по протоколу HTTP Basic Authentication.

Содержание

Пример конфигурации

```
location  /  {
    auth_basic            "closed site";
    auth_basic_user_file  conf/htpasswd;
}
```

Директивы

---

syntax: auth_basic [строка|off]
default: auth_basic off
context: http, server, location, limit_except

Директива включает проверку имени и пароля по протоколу HTTP Basic Authentication. Заданный параметр используется в качестве realm. Параметр "off" позволяет отменить действие унаследованной с нижележащего уровня директивы.

---

syntax: auth_basic_user_file файл
default: нет
context: http, server, location, limit_except

Директива задаёт файл, в котором хранятся имена и пароли пользователей. Формат файла следующий:

```
# комментарий
имя1:пароль1
имя2:пароль2:комментарий
имя3:пароль3
```

Пароли должны быть зашифрованы функцией crypt(3). Для создания файла с паролями можно воспользоваться программой `htpasswd` из дистрибутива Apache.

Директивы модуля ngx_http_autoindex_module

19.07.2005

Модуль ngx_http_autoindex_module выдаёт листинг каталога. Обычно запрос попадает к модулю ngx_http_autoindex_module, когда модуль ngx_http_index_module не нашёл индексный файл.

Содержание

Пример конфигурации

```
location / {
    autoindex on;
}
```

Директивы

---

syntax: autoindex [on|off]
default: autoindex off
context: http, server, location

Директива разрешает или запрещает вывод листинга каталога.

---

syntax: autoindex_exact_size [on|off]
default: autoindex_exact_size on
context: http, server, location

Директива определяет, как выводить размеры файлов в листинге каталога точно, или округляя до килобайт, мегабайт и гигабайт.

---

syntax: autoindex_localtime [on|off]
default: autoindex_localtime off
context: http, server, location

Директива определяет, в какой временной зоне выводить время в листинге каталога в локальной или в GMT.

Директивы модуля ngx_http_browser_module

26.09.2006

Модуль ngx_http_browser_module создаёт переменные, значение которых зависят от строки "User-Agent" в заголовке запроса:

- $modern_browser — равна значению, заданному директивой modern_browser_value, если браузер опознан как современный;
- $ancient_browser — равна значению, заданному директивой ancient_browser_value, если браузер опознан как устаревший;
- $msie — равна "1", если браузер опознан как MSIE любой версии;

Содержание

Примеры конфигурации

Выбор индексного файла:

```
 modern_browser_value  "modern.";

modern_browser   msie       5.5;
modern_browser   gecko      1.0.0;
modern_browser   opera      9.0;
modern_browser   safari     413;
modern_browser   konqueror  3.0;

index  index.${modern_browser}html index.html;
```

Редирект для старых браузеров:

```
 modern_browser   msie       5.0;
modern_browser   gecko      0.9.1;
modern_browser   opera      8.0;
modern_browser   safari     413;
modern_browser   konqueror  3.0;

modern_browser   unlisted;

ancient_browser  Links  Lynx  netscape4;
```

```
if ($ancient_browser) {
    rewrite  ^  /ancient.html;
}
```

Директивы

---

syntax: ancient_browser строка [строка ...]
default: нет
context: http, server, location

Директива задаёт подстроки, при нахождении которых в строке "User-Agent", браузер считается устаревшим. Специальная строка "netscape4" соответствует регулярному выражению "^Mozilla/[1-4]".

---

syntax: ancient_browser_value строка
default: ancient_browser_value 1
context: http, server, location

Директива задаёт значение для переменных $ancient_browser.

---

syntax: modern_browser браузер версия|unlisted
default: нет
context: http, server, location

Директива задаёт версию браузера, начиная с которой он считается современным. В качестве браузера можно задать msie, gecko (браузеры, созданные на основе Mozilla) opera, safari, konqueror.

Версии можно задать в формате X, X.X, X.X.X, или X.X.X.X. Максимальные значения для каждого их форматов соответственно 4000, 4000.99, 4000.99.99, и 4000.99.99.99.

Специальное значение "unlisted" указывает считать современным браузер, не описанный директивами modern_browser и ancient_browser. В противном случае неперечисленный браузер будет считаться устаревшим. Если в запросе нет строки "User-Agent", то браузер считается неперечисленным.

---

syntax: modern_browser_value строка
default: modern_browser_value 1

context: http, server, location

Директива задаёт значение для переменных $modern_browser.

Директивы модуля ngx_http_charset_module

18.08.2008

Модуль ngx_http_charset_module добавляет указанную кодировку в строку "Content-Type" в заголовок ответа. Кроме того, модуль может перекодировать данные из одной кодировки в другую с некоторыми ограничениями:

- перекодирование осуществляется только в одну сторону от сервера к клиенту,
- перекодироваться могут только однобайтные кодировки
- или однобайтные кодировки в UTF-8 и обратно.

Содержание

Пример конфигурации

```
 include         conf/koi-win;

charset         windows-1251;
source_charset  koi8-r;
```

Директивы

---

syntax: charset кодировка|off
default: charset off
context: http, server, location, if в location

Директива charset добавляет в строку "Content-Type" в заголовке ответа указанную кодировку. Если эта кодировка отличается от указанной в директиве source_charset, то выполняется перекодирование.

Параметр "off" отменяет добавление кодировки в строку "Content-Type" в заголовке ответа.

Кодировка может быть задана переменной:

```
charset     $charset;
```

В этом случае необходимо, чтобы все возможные значения переменной присутствовали хотя бы один раз в любом месте конфигурации в виде директив charset_map, charset, или source_charset. Для кодировок utf-8, windows-1251 и koi8-r для этого достаточно включить в конфигурацию файлы conf/koi-win, conf/koi-utf и conf/win-utf. Для других кодировок можно просто сделать фиктивную таблцу перекодировки, например:

```
charset_map  iso-8859-5  _ { }
```

---

syntax: charset_map кодировка1 кодировка2 { ... }
default: нет
context: http

Директива charset_map описывает таблицу перекодирования из одной кодировки в другую. Таблица для обратного перекодирования строится на основании тех же данных. Коды символов задаются в шестнадцатеричном виде. Неописанные символы в переделах 80-FF заменяются на "?". При перекодировании из UTF-8 символы, отсутствующие в однобайтной кодировке, заменяются на "&#XXXX;".

Пример использования:

```
charset_map  koi8-r  windows-1251 {
   C0  FE ; # small yu
   C1  E0 ; # small a
   C2  E1 ; # small b
   C3  F6 ; # small ts
   ...
}
```

При описании таблицы перекодирования в UTF-8, значения этой кодировки должны описываться во второй колонке, например:

```
charset_map  koi8-r  utf-8 {
   C0  D18E ; # small yu
   C1  D0B0 ; # small a
   C2  D0B1 ; # small b
   C3  D186 ; # small ts
   ...
}
```

Полные таблицы преобразования из koi8-r в windows-1251 и из koi8-r и windows-1251 в utf-8 входят в дистрибутив и находятся в файлах conf/koi-win,

conf/koi-utf и conf/win-utf.

---

syntax: charset_types mime-тип [mime-тип ...]
default: charset_types text/html text/xml text/plain text/vnd.wap.wml application/x-javascript application/rss+xml.
context: http, server, location

Директива (0.7.9) разрешает работу модуля в ответах с указанными MIME-типами в дополнение к "text/html". По умолчанию используются

- text/html
- text/xml
- text/plain
- text/vnd.wap.wml
- application/x-javascript
- application/rss+xml

---

syntax: override_charset on|off
default: override_charset off
context: http, server, location, if в location

Директива определяет, выполнять ли перекодирование для ответов, полученных от проксированного сервера или от FastCGI-сервера, если в них уже есть charset в строке "Content-Type" в заголовке ответа. Если перекодирование разрешено, то в качестве исходной кодировки используется кодировка, указанная в полученном ответе.

Необходимо отметить, что если ответ был получен в подзапросе, то, независимо от директивы override_charset, всегда выполняется перекодирование из кодировки ответа в кодировку основного запроса.

---

syntax: source_charset кодировка
default: нет
context: http, server, location, if в location

Директива source_charset задают исходную кодировку ответа. Если эта кодировка отличается от указанной в директиве charset, то выполняется перекодирование.

Директивы модуля ngx_http_dav_module

17.08.2009

Модуль ngx_http_dav_module обрабатывает HTTP- и WebDAV-методы PUT, DELETE, MKCOL, COPY и MOVE. По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_dav_module`.

Содержание

Пример конфигурации

```
location / {
    root                   /data/www;

    client_body_temp_path  /data/client_temp;

    dav_methods  PUT DELETE MKCOL COPY MOVE;

    create_full_put_path   on;
    dav_access             group:rw  all:r;

    limit_except  GET {
        allow  192.168.1.0/32;
        deny   all;
    }
}
```

Директивы

---

syntax: dav_access пользователи:права [пользователи:права] ...
default: dav_access user:rw
context: http, server, location

Директива задаёт права доступа для создаваемых файлов и каталогов, например,

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

> Если заданы какие-либо права для groups или all, то права для user указывать необязательно:

```
dav_access  group:rw  all:r;
```

---

syntax: dav_methods [off|put|delete|mkcol|copy|move] ...
default: dav_methods off
context: http, server, location

Директива разрешает указанные HTTP- и WebDAV-методы. Параметр off запрещает все методы, обарабатываемые данным модулем, игнорируя остальные параметры;

Файл, загружаемый методом PUT, записывается во временный файл, а потом этот файл переименовывается. Начиная с версии 0.8.9, временный файл и его постоянное место хранения могут располагаться на разных файловых системах, но 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 client_body_temp_path для данного location.

При создании файла с помощью метода PUT можно задать дату модификации, передав её в строке заголовка "Date".

---

syntax: create_full_put_path on|off
default: create_full_put_path off
context: http, server, location

По спецификации WebDAV метод PUT может создавать файл только в уже существующем каталоге. Данная директива разрешает создавать все необходимые промежуточные каталоги.

---

syntax: min_delete_depth число
default: min_delete_depth 0
context: http, server, location

Данная директива разрешает методу DELETE удалять файлы при условии, что число элементов в пути запроса не меньше заданного. Например, директива

```
min_delete_depth  4;
```

> разрешает удалять файлы по запросам

```
/users/00/00/name
/users/00/00/name/pic.jpg
/users/00/00/page.html
```

> и запрещает удаление

```
/users/00/00
```

(C) Игорь Сысоев
http://sysoev.ru

Директивы модуля ngx_http_empty_gif_module

15.11.2005

Модуль ngx_http_empty_gif_module выдаёт одно-пиксельный прозрачный GIF.

Содержание

Пример конфигурации

```
location = /_.gif {
    empty_gif;
}
```

Директивы

---

syntax: empty_gif
default: нет
context: location

Директива разрешает выдавать одно-пиксельный прозрачный GIF.

Директивы модуля ngx_http_fastcgi_module

17.08.2009

Модуль ngx_http_fastcgi_module позволяет передавать запросы удалённому FastCGI-серверу.

Содержание

Параметры, передаваемые FastCGI-серверу

Встроенные переменные

Пример конфигурации

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

   fastcgi_param  SCRIPT_FILENAME  /home/www/scripts/php$fastcgi_script_name;
   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

---

syntax: fastcgi_buffer_size размер
default: fastcgi_buffer_size 4k/8k
context: http, server, location

Директива задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от FastCGI-сервера. В этой части ответа находится, как правило, небольшой заголовок ответа. По умолчанию размер буфера равен размеру одного буфера в директиве fastcgi_buffers, однако его можно сделать меньше.

---

syntax: fastcgi_buffers число размер
default: fastcgi_buffers 8 4k/8k
context: http, server, location

Директива 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.

---

syntax: fastcgi_cache [зона|off]
default: off
context: http, server, location

Директива задаёт зону для кэширования. Одна и та же зона может использоваться в нескольких местах. Параметр "off" запрещает кэширование, унаследованное с предыдущего уровня конфигурации.

---

syntax: fastcgi_cache_bypass строка [...]
default: off
context: http, server, location

Директива задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одной из строк переменных не пустое и не равно "0", то ответ не берётся из кэша:

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

Можно использовать совместно с директивой fastcgi_no_cache.

---

syntax: fastcgi_cache_key строка
default: нет
context: http, server, location

Директива задаёт ключ для кэширования, например,

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

---

syntax: fastcgi_cache_path путь [levels=уровни] keys_zone=название:размер [inactive=время] [max_size=размер]
default: нет
context: http

Директива 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, например, при использовании

```
fastcgi_cache_path  /data/nginx/cache  levels=1:2   keys_zone=one:10m;
```

> имена файлов в кэше будут такого вида:

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

Кэшируемый ответ 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, но 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, задаваемый директивой fastcgi_temp_path для данного location.

Кроме того, все активные ключи и информация о данных хранятся в разделяемой памяти зоне, имя и размер которой задаётся параметром keys_zone. Если к данным кэша не обращются 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, и при превышении его размеров удаляет самые невостребованные данные.

---

syntax: fastcgi_cache_min_uses число
default: fastcgi_cache_min_uses 1
context: http, server, location

Директива задаёт число запросов, после которого ответ будет закэширован.

---

syntax: fastcgi_cache_valid ответ [ответ ...] время
default: нет
context: http, server, location

Директива задаёт время кэширования для разных ответов. Например, директивы

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

> задают время кэширования 10 минут для ответов 200 и 302, и 1 минуту для ответов 404.

Если указано только время кэширования,

```
fastcgi_cache_valid  5m;
```

> то кэшируются только ответы 200, 301 и 302.

Кроме того, может кэшировать любые ответы с помощью параметра "any":

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

---

syntax: fastcgi_cache_use_stale [error | timeout | invalid_header | updating | http_500 | http_503 | http_404 | off] [...]
default: fastcgi_cache_use_stale off
context: http, server, location

Директива определяет, в каких случаях можно использовать устаревший закэшированный ответ, если при работе с проксированным сервером возникла ошибка. Параметры директивы совпадают с параметрами директивы fastcgi_next_upstream. И, кроме того, есть параметр updating, которой разрешает использовать устаревший закэшированный ответ, если на данный момент он уже обновляется.

---

syntax: fastcgi_connect_timeout время
default: fastcgi_connect_timeout 60
context: http, server, location

Директива задаёт таймаут для соединения с FastCGI-сервером. Необходимо иметь в виду, что этот таймаут не может быть больше 75 секунд.

---

syntax: fastcgi_index имя
default: нет
context: http, server, location

Директива 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$fastcgi_script_name;
```

> и запросе "/page.php" параметр SCRIPT_FILENAME будет равен "/home/www/scripts/php/page.php", а при запросе "/" □ "/home/www/scripts/php/index.php".

---

syntax: fastcgi_hide_header имя
context: http, server, location

nginx не передаёт клиенту строки заголовка "Status" и "X-Accel-..." из ответа FastCGI-сервера. Директива fastcgi_hide_header задаёт дополнительные строки. Если же строки нужно наоборот разрешить, то нужно воспользоваться директивой fastcgi_pass_header.

---

syntax: fastcgi_ignore_client_abort [on|off]
default: fastcgi_ignore_client_abort off
context: http, server, location

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

---

syntax: fastcgi_ignore_headers имя [имя ...]
context: http, server, location

Директива fastcgi_ignore_headers запрещает обработку некоторых строк заголовка из ответа FastCGI-сервера. В директиве можно указать строки "X-Accel-Redirect", "X-Accel-Expires", "Expires" и "Cache-Control".

---

syntax: fastcgi_intercept_errors on|off
default: fastcgi_intercept_errors off
context: http, server, location

Директива определяет, передавать ли клиенту ответы FastCGI-сервера с кодом больше или равные 400 или же перенаправлять их на обработку nginx'у с помощью директивы error_page.

---

syntax: fastcgi_no_cache строка [...]
default: нет
context: http, server, location

Директива задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одной из строк переменных не пустое 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.

---

syntax: fastcgi_next_upstream
[error|timeout|invalid_header|http_500|http_503|http_404|off]
default: fastcgi_next_upstream error timeout
context: http, server, location

Директива определяет, в каких случаях запрос будет передан следующему серверу:

- error — произшла 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;
- 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_404 — сервер вернул ответ с кодом 404;
- 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.

---

syntax: fastcgi_param параметр значение
default: нет
context: http, server, location

Директива задаёт параметр, который будут передаваться FastCGI-серверу. В качестве значения можно использовать текст, переменные и их комбинации. Директивы наследуются с предыдущего уровня 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$fastcgi_script_name;
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;
```

---

syntax: fastcgi_pass fastcgi-server
default: нет
context: location, if в location

Директива задаёт адрес FastCGI-сервера. Адрес может быть указан в виде доменного имени или адреса и порта, например,

```
fastcgi_pass   localhost:9000;
```

> или в виде пути unix сокета:

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

Если доменное имя резолвится в несколько адресов, то все они будут использоваться в режиме round-robin. И кроме того, адрес может быть группой серверов.

---

syntax: fastcgi_pass_header имя
context: http, server, location

Директива разрешает передавать клиенту запрещённые для передачи строки.

---

syntax: fastcgi_read_timeout время
default: fastcgi_read_timeout 60
context: http, server, location

Директива 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, то nginx закрывает соединение.

---

syntax: fastcgi_redirect_errors on|off

Директива переименована в fastcgi_intercept_errors.

---

syntax: fastcgi_send_timeout время
default: fastcgi_send_timeout 60
context: http, server, location

Директива 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, то nginx закрывает соединение.

---

syntax: fastcgi_split_path_info regex
default: нет
context: location

Директива задаёт регулярное выражение, выделяющее значение для переменной $fastcgi_path_info. Регулярное выражение должно иметь два выделения, из которых первое становиться 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_script_name;
    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".

---

syntax: fastcgi_store on | off | строка
default: fastcgi_store off
context: http, server, location

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

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

Время модификации файлов выставляется согласно полученной строке "Last-Modified" в заголовке ответа. Ответ 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, но 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:

```
location /images/ {
   root                    /data/www;
   open_file_cache_errors  off;
   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/;
}
```

---

syntax: fastcgi_store_access пользователи:права [пользователи:права] ...
default: fastcgi_store_access user:rw
context: http, server, location

Директива задаёт права доступа для создаваемых файлов и каталогов, например,

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

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

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

---

syntax: fastcgi_temp_path путь [уровень1 [уровень2 [уровень3]]]
default: fastcgi_temp_path fastcgi_temp
context: http, server, location

Директива задаёт имя каталога для хранения временных файлов полученных от другого сервера. В каталоге может использоваться иерархия подкаталогов до трёх уровней. Например, при такой конфигурации

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

> имя временного будет такого вида:

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

Параметры, передаваемые FastCGI-серверу

Строки 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, строка заголовка "User-Agent" передаётся как параметр HTTP_USER_AGENT. Кроме строк 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, эта переменная равна URI запроса или же, если URI заканчивается слэшом, то URI запроса плюс имя индексного файла, задаваемого директивой fastcgi_index. Эту переменную можно использовать для задания параметра 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$fastcgi_script_name;
```

> параметр SCRIPT_FILENAME будут 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, эта переменная равна значению второго выделения, задаваемого директивой fastcgi_split_path_info. Эту переменную можно использовать для задания параметра PATH_INFO.

Директивы модуля ngx_http_flv_module

11.10.2006

Модуль ngx_http_flv_module делает специальную обработку передаваемых файлов: при передаче файла, начиная со смещения, указаного в аргументах запроса "start=XXX", добавляет FLV-заголовок перед запрашиваемым файлом.

По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_flv_module`.

Содержание

Пример конфигурации

```
location ~ \.flv$ {
    flv;
}
```

Директивы

---

syntax: flv
default: нет
context: location

Директива разрешает специальную обработку передаваемых файлов.

Директивы модуля ngx_http_geo_module

27.10.2009

Модуль ngx_http_geo_module создаёт переменные, значения которых зависят от IP-адреса клиента.

Содержание

Пример конфигурации

```
geo  $geo  {
    default          0;
    127.0.0.1/32     2;
    192.168.1.0/24   1;
    10.1.0.0/16      1;
}
```

Директивы

---

syntax: geo [$адрес] $переменная { ... }
default: нет
context: http

Директива geo описывает для указанной переменной зависимость значения от IP-адреса клиента. По умолчанию адрес берётся из переменной $remote_addr, но его также (0.7.27) можно получить из другой переменной, например:

```
geo  $arg_remote_addr  $geo {
    ...;
}
```

> Если значение переменной не представляет из себя правильный IP-адрес, то используется адрес "255.255.255.255".

Адреса задаются в виде CIDR или в виде диапазонов (0.7.23). Кроме того, есть четыре специальных параметра:

- delete — удаляет описанную сеть (0.7.23).
- default — значение переменной, если адрес клиента не соответствует ни одному заданному адресу. При использовании CIDR вместо default можно также

написать 0.0.0.0/0.

- include включает файл с адресами и значениями. Включений может быть несколько.
- proxy задаёт адреса проски-серверов (0.8.7, 0.7.63), при запросе с которых будет использоваться адрес в переданной в строке заголовка запроса "X-Forwarded-For". В отличие от обычных адресов, адреса прокси-серверов проверяются последовательно.
- ranges указывает, что адреса задаются в виде диапазонов (0.7.23). Эта директива должны быть первой. Для ускорения загрузки гео-базы нужно располагать адреса в порядке возрастания.

Пример описания:

```
    geo  $country  {
        default          ZZ;
        include          conf/geo.conf;
        delete           127.0.0.0/16;
        proxy            192.168.100.0/24;

        127.0.0.0/24     US;
        127.0.0.1/32     RU;
        10.1.0.0/16      RU;
        192.168.1.0/24   UK;
    }
```

В файле conf/geo.conf могут быть такие строки:

```
10.2.0.0/16      RU;
192.168.2.0/24   RU;
```

В качестве значения выбирается максимальное совпадение, например, для адреса 127.0.0.1 будет выбрано значение "RU", а не "US".

Пример описания диапазонов:

```
    geo  $country  {
        ranges;
        default                    ZZ;
        127.0.0.0-127.0.0.0        US;
        127.0.0.1-127.0.0.1        RU;
        127.0.0.1-127.0.0.255      US;
        10.1.0.0-10.1.255.255      RU;
        192.168.1.0-192.168.1.255  UK;
    }
```

Директивы модуля ngx_http_geoip_module

20.07.2009

Модуль ngx_http_geoip_module создаёт переменные, значения которых зависят от IP-адреса клиента, используя готовые базы MaxMind (0.8.6+). По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_geoip_module`. Для сборки и работы этого модуля нужна библиотека MaxMind GeoIP.

Содержание

Пример конфигурации

```
http {
   geoip_country  GeoIP.dat;
   geoip_city     GeoLiteCity.dat;
   ...
```

Директивы

---

syntax: geoip_country база
default: нет
context: http

Директива geoip_country указывает базу для определения страны в зависимости от значения IP-адреса клиента. При использовании этой базы доступны следующие переменные:

- $geoip_country_code; — двухбуквенный код страны, например, "RU", "US".
- $geoip_country_code3; — трёхбуквенный код страны, например, "RUS", "USA".
- $geoip_country_name; — название страны, например, "Russian Federation", "United States".

---

syntax: geoip_city база
default: нет
context: http

Директива geoip_city указывает базу для определения страны, региона и города в зависимости от значения IP-адреса клиента. При использовании этой базы доступны следующие переменные:

- $geoip_city_country_code; — двухбуквенный код страны, например, "RU", "US".
- $geoip_city_country_code3; — трёхбуквенный код страны, например, "RUS", "USA".
- $geoip_city_country_name; — название страны, например, "Russian Federation", "United States".
- $geoip_region; — название региона страны (область, край, штат, провинция, федеральная земля и тому подобное), например, "Moscow City", "DC".
- $geoip_city; — название города, например, "Moscow", "Washington".
- $geoip_postal_code; — почтовый индекс.

Директивы модуля ngx_http_gzip_module

28.08.2009

Модуль ngx_http_gzip_module - это фильтр, сжимающий ответ методом gzip, что позволяет уменьшить размер передаваемых данных в 2 и более раз.

Содержание

Пример конфигурации

```
 gzip             on;
gzip_min_length  1000;
gzip_proxied     expired no-cache no-store private auth;
gzip_types       text/plain application/xml;
```

Для записи в лог степени сжатия можно использовать переменную $gzip_ratio.

Директивы

---

syntax: gzip on|off
default: gzip off
context: http, server, location, if в location

Разрешает или запрещает сжатие ответа методом gzip.

---

syntax: gzip_buffers число размер
default: gzip_buffers 32 4k/16 8k
context: http, server, location

Директива задаёт число и размер буферов, в которые будет сжиматься ответ. По умолчанию размер одного буфера равен размеру страницы, в зависимости от платформы это или 4K, или 8K. До версии 0.7.28 по умолчанию использовалось 4 буфера размером 4K или 8K.

---

syntax: gzip_comp_level 1..9
default: gzip_comp_level 1
context: http, server, location

Устанавливает уровень сжатия ответа методом gzip.

---

syntax: gzip_disable regex [regex ...]
default: нет
context: http, server, location

Директива (0.6.23) запрещает сжатие ответа методом gzip для запросов со строками "User-Agent", совпадающими с заданными регулярными выражениями.

Специальная маска "msie6" (0.7.12) соответствует регулярному выражению "MSIE [4-6]\.", но работает быстрее. Начиная с версии 0.8.11, из этой маски исключается "MSIE 6.0; ... SV1".

---

syntax: gzip_min_length длина
default: gzip_min_length 20
context: http, server, location

Устанавливает минимальную длину ответа, для которых будет выполняться сжатие ответа методом gzip. Длина определяется только из строки "Content-Length" заголовка ответа.

---

syntax: gzip_http_version 1.0|1.1
default: gzip_http_version 1.1
context: http, server, location

Устанавливает минимальную версию HTTP запроса для сжатия ответа.

---

syntax: gzip_proxied
[off|expired|no-cache|no-store|private|no_last_modified|no_etag|auth|any] ...
default: gzip_proxied off
context: http, server, location

Разрешает или запрещает сжатие ответа методом gzip для проксированных запросов в зависимости от запроса и ответа. То что, запрос проксированный, определяется на основании строки "Via" в заголовке запроса. В директиве можно указать одновременно несколько паарметров:

- off запрещает сжатие для всех проксированных запросов, игнорируя остальные параметры;
- expired разрешить сжатие, если в ответе есть строка "Expires" со значением, запрещающим кэширование;
- no-cache разрешить сжатие, если в ответе есть строка "Cache-Control" с параметром "no-cache";
- no-store разрешить сжатие, если в ответе есть строка "Cache-Control" с параметром "no-store";
- private разрешить сжатие, если в ответе есть строка "Cache-Control" с параметром "private";
- no_last_modified разрешить сжатие, если в ответе нет строки "Last-Modified";
- no_etag разрешить сжатие, если в ответе нет строки "ETag";
- auth разрешить сжатие, если в запросе есть строка "Authorization";
- any разрешить сжатие для всех проксированных запросов;

---

syntax: gzip_types mime-тип [mime-тип ...]
default: gzip_types text/html
context: http, server, location

Разрешает сжатие ответа методом gzip для указанных MIME-типов в дополнение к "text/html". "text/html" сжимается всегда.

---

syntax: gzip_vary on|off
default: gzip_vary off
context: http, server, location

Разрешает или запрещает выдавать в ответе строку заголовка "Vary: Accept-Encoding", если директивы gzip или gzip_static активны.

Директивы модуля ngx_http_gzip_static_module

27.12.2007

Модуль ngx_http_gzip_static_module позволяет отдавать вместо обычного файла предварительно сжатый файл с таким же именем и с суффиксом ".gz".

По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_gzip_static_module`.

Содержание

Пример конфигурации

```
gzip_static      on;
gzip_proxied     expired no-cache no-store private auth;
```

Директивы

---

syntax: gzip_static on|off
default: gzip_static off
context: http, server, location

Разрешает или запрещает проверку готового сжатого файла. При использовании также учитываются директивы gzip_http_version, gzip_proxied, gzip_disable и gzip_vary.

Сжимать файлы можно с помощью программы gzip или совместимой с ней. Желательно, чтобы время модификации исходного и сжатого файлов совпадали.

Директивы модуля ngx_http_headers_module

12.08.2008

Модуль ngx_http_headers_module позволяет выдавать строки "Expires" и "Cache-Control" и добавлять произвольные строки в заголовке ответа.

Содержание

Примеры конфигурации

```
 expires     24h;
expires     modified +24h;
expires     @24h;
expires     0;
expires     -1;
expires     epoch;
add_header  Cache-Control  private;
```

Директивы

---

syntax: add_header название значение
default: нет
context: http, server, location

Директива добавляет строку в заголовке ответа при условии, что код ответа равен 200, 204, 301, 302 или 304. В значении можно использовать переменные.

---

syntax: expires [modified][время|epoch|max|off]
default: expires off
context: http, server, location

Разрешает или запрещает добавлять или менять строки "Expires" и "Cache-Control" в заголовке ответа. В качестве параметра можно задать положительное или отрицательное время.

Время в строке "Expires" получается как сумма текущего времени и времени, заданного в директиве. Если используется параметр "modified" (0.7.0, 0.6.32), то время получается как сумма времени модификации файла и времени, заданного в директиве.

Кроме того, с помощью префикса "@" можно задать время суток (0.7.9, 0.6.34):

```
expires   @15h30m;
```

Параметр "epoch" означает абсолютное время 1 января 1970 года 00:00:01 GMT. Содержимое строки "Cache-Control" зависит от знака заданного времени:

- отрицательное время "Cache-Control: no-cache".
- положительное время или равное нулю "Cache-Control: max-age=#", где "#" - это время в секундах, заданное в директиве.

Параметр "max" задаёт время 31 декабря 2037 23:55:55 GMT для строки "Expires" и 10 лет для строки "Cache-Control".

Параметр "off" запрещает добавлять или менять строки "Expires" и "Cache-Control" в заголовке ответа.

Директивы модуля ngx_http_image_filter_module

25.09.2009

Модуль ngx_http_image_filter_module — это фильтр для преобразования изображений в форматах JPEG, GIF и PNG (0.7.54+). По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_image_filter_module`. Для сборки и работы этого модуля нужна библиотека libgd. Рекомендуется использовать самую последнюю версию библиотеки, на текущий момент (май 2009) это версия 2.0.35.

Содержание

Пример конфигурации

```
location /img/ {
   proxy_pass      http://backend;
   image_filter    resize  150 100;
   error_page      415   = /empty;
}

location = /empty {
   empty_gif;
}
```

Директивы

---

syntax: image_filter (test|size|resize ширина высота|crop ширина высота)
default: нет
context: location

Директива задаёт тип преобразования изображения:

- test — проверка того, что ответ действительно является изображением в формате JPEG, GIF или PNG. В противном случае выдаётся ошибка 415.
- size — выдаёт информацию об изображении в формате JSON, например:

```
{ "img" : { "width": 100, "height": 100, "type": "gif" } }
```

> В случае ошибки выдаётся
>
> {}

- resize – пропорционально уменьшает изображение до указанных размеров. Если нужно уменьшить только по одному измерению, то в качестве второго можно указать "-". В случае ошибки выдаётся код 415.
- crop – пропорционально уменьшает изображение до размера большой стороны и обрезает лишние края по другой стороне. Если нужно уменьшить только по одному измерению, то в качестве второго можно указать "-". В случае ошибки выдаётся код 415.

---

syntax: image_filter_buffer размер
default: image_filter_buffer 1M
context: http, server, location

Директива задаёт максимальный размер буфера для чтения изображения.

---

syntax: image_filter_jpeg_quality [0..100]
default: image_filter_jpeg_quality 75
context: http, server, location

Директива задаёт коэффициент потери информации при обработке изображений в формате JPEG. Максимальное рекомендуемое значение – 95.

---

syntax: image_filter_transparency [on|off]
default: image_filter_transparency on
context: http, server, location

Директива определяет, сохранять ли прозрачность при обработке изображений в формате PNG с цветами, заданными палитрой, и формате GIF. Потеря прозрачности позволяет получить более качественное изображение. Прозрачность альфа-канала в формате PNG сохраняется всегда.

Директивы модуля ngx_http_index_module

12.08.2008

Модуль ngx_http_index_module обслуживает запросы, оканчивающиеся слэшом.

Содержание

Пример конфигурации

```
location / {
    index index.$geo.html index.html;
}
```

Директивы

---

syntax: index файл [файл ...]
default: index index.html
context: http, server, location

Директива определяет файлы, которые будут использоваться в качестве индекса. В имени файла можно использовать переменные. Наличие файлов проверяется в порядке их перечисления. В конце списка может стоять файл с абсолютным путём. Пример использования:

```
index index.$geo.html index.0.html /index.html;
```

Необходимо иметь ввиду, что при использовании индексного файла делается внутренний редирект и запрос может быть обработан уже в другом location'е. Например, запрос "/" будет фактически обработан во втором location'е как "/index.html" в такой конфигурации:

```
location = / {
    index index.html;
}

location / {
    ...
}
```

Директивы модуля ngx_http_limit_zone_module

06.10.2009

Модуль ngx_http_limit_zone_module позволяет ограничить число одновременных соединений для заданной сессии или, как частный случай, с одного адреса.

Содержание

Пример конфигурации

```
http {
   limit_zone   one  $binary_remote_addr  10m;

   ...

   server {

       ...

       location /download/ {
           limit_conn   one  1;
       }
```

Директивы

---

syntax: limit_zone название $переменная размер
default: нет
context: http

Директива описывает зону, в которой хранятся состояния сессий. Значения сессий определяется заданной переменной. Пример использования:

```
limit_zone   one  $binary_remote_addr  10m;
```

В качестве сессии используется адрес клиента. Обратите внимание, что вместо переменной $remote_addr используется переменная $binary_remote_addr. Длина значений переменной $remote_addr может быть от 7 до 15 байт, поэтому размер состоянии равен 32 или 64 байтам. Длина всех значений переменной $binary_remote_addr всегда 4 байта и размер состояния всегда 32 байта. В зоне

размером 1 мегабайт может разместиться около 32000 состояний размером 32 байта.

---

syntax: limit_conn зона число
default: нет
context: http, server, location

Директива задаёт максимальное число одновременных соединений для одной сессии. При превышении этого числа запрос завершается кодом "Service unavailable" (503). Например, директивы

```
limit_zone   one  $binary_remote_addr  10m;

  server {
      location /download/ {
          limit_conn   one  1;
      }
```

> позволяют не более одного одновременного соединения с одного адреса.

---

syntax: limit_conn_log_level [info|notice|warn|error]
default: limit_conn_log_level error
context: http, server, location

Директива (0.8.18) задаёт уровень логирования случаев ограничения числа соединений.

Директивы модуля ngx_http_limit_req_module

06.10.2009

Модуль ngx_http_limit_req_module (0.7.21) позволяет ограничить число запросов для заданной сессии или, как частный случай, с одного адреса. Ограничение делается с помощью метода leaky bucket.

Содержание

Пример конфигурации

```
http {
   limit_req_zone  $binary_remote_addr  zone=one:10m   rate=1r/s;

   ...

   server {

       ...

       location /search/ {
           limit_req   zone=one  burst=5;
       }
```

Директивы

---

syntax: limit_req_zone $переменная zone=название:размер rate=скорость
default: нет
context: http

Директива описывает зону, в которой хранятся состояния сессий. Значения сессий определяется заданной переменной. Пример использования:

```
limit_req_zone  $binary_remote_addr  zone=one:10m   rate=1r/s;
```

В данном случае состояния сессий хранятся в зоне "one" размером 10 мегабайт и средняя скорость запросов для этой зоны не может более 1 запроса в секунду.

В качестве сессии используется адрес клиента. Обратите внимание, что вместо переменной $remote_addr используется переменная $binary_remote_addr, позволяющая уменьшить размер состояния до 64 байт. В зоне размером 1 мегабайт может разместиться около 16000 состояний размером 64 байта.

Скорость задаётся в запросах в секунду. Если же нужна скорость меньше одного запроса в секунду, то она задаётся в запросах в минуту, например, ползапроса в секунду это 30r/m.

---

syntax: limit_req zone=название burst=число [nodelay]
default: нет
context: http, server, location

Директива задаёт зону (zone) и максимально возможные всплески запросов (burst). Если скорость запросов превышает описанную в зоне, то их обработка запроса задерживается так, чтобы запросы обрабывались с заданной скоростью. Избыточные запросы задерживаются до тех пор, пока их число не превысит заданное число всплесков. В этом случае запрос завершается кодом "Service unavailable" (503). По умолчанию число всплесков равно нулю. Например, директивы

```
limit_req_zone  $binary_remote_addr  zone=one:10m   rate=1r/s;

  server {
      location /search/ {
          limit_req   zone=one  burst=5;
      }
```

> позволяют в среднем не более 1 запроса в секунду со всплесками не более 5 запросов.

Если же избыточные запросы в пределах лимита всплесков задерживать не надо, то нужно использовать параметр nodelay:

```
          limit_req   zone=one  burst=5  nodelay;
```

---

syntax: limit_req_log_level [info|notice|warn|error]
default: limit_req_log_level error
context: http, server, location

Директива (0.8.18) задаёт уровень логирования случаев ограничения числа запросов и задержек при обработке запроса. Задержки логируются на один

уровень ниже, чем ограничения, например, если задан "limit_req_log_level notice", то задержки будут логироваться на уровне info.

Директивы модуля ngx_http_log_module

07.07.2008

Модуль ngx_http_log_module записывает логи запросов в указанном формате.

Содержание

Пример конфигурации

```
log_format  gzip  '$remote_addr - $remote_user [$time_local] '
                  '"$request" $status $bytes_sent '
                  '"$http_referer" "$http_user_agent" "$gzip_ratio"';

access_log  /spool/logs/nginx-access.log  gzip  buffer=32k;
```

Директивы

---

syntax: access_log путь [формат [buffer=размер]]|off
default: access_log log/access.log combined
context: http, server, location, if в location, limit_except

Директива access_log задаёт путь, формат и размер буфера для буферизированной записи в лог. На одном уровне может использоваться несколько логов. Параметр "off" отменяет все директивы access_log для текущего уровня. Если формат не указан, то используется предопределённый формат "combined".

Размер буфера должен быть не больше размера атомарной записи в дисковый файл. Для FreeBSD 3.0-6.0 этот размер неограничен.

В пути файла можно использовать переменные (0.7.6+), но такие логи имеют некоторые ограничения:

- пользователь, с правами которого работают рабочие процессы, должен иметь права на создание файлов в каталоге с такими логами;
- не работает буферизация;

• файл открывается для каждой записи в лог и сразу же после записи закрывается. Но дескрипторы часто используемых файлов могут храниться в open_log_file_cache. При вращении логов нужно иметь в виду, что в течение времени, заданного параметром valid директивы open_log_file_cache, запись может продолжаться в старый файл.
при каждой записи в лог проверяется существование каталога root'a для запроса если этот каталог не существует, то лог не создаётся. Поэтому root и access_log нужно описывать на одном уровне:

```
server {
   root        /spool/vhost/data/$host;
   access_log  /spool/vhost/logs/$host;
   ...
```

---

syntax: log_format название строка [строка ...]
default: log_format combined "..."
context: http

Директива log_format описывает формат лога.

Кроме общих переменных в формате можно использовать переменные, существующие только на момент записи в лог:

• $body_bytes_sent, число байт, переданное клиенту за вычетом заголовка ответа, переменная совместима с параметром %B модуля Apache mod_log_config;
• $bytes_sent, число байт, переданное клиенту;
• $connection, номер соединения;
• $msec, время в секундах с точностью до миллисекунд на момент записи в лог;
• $pipe, "p" если запрос был pipelined;
• $request_length, длина тела запроса;
• $request_time, время обработки запроса в секундах с точностью до миллисекунд;
• $status, статус ответа;
• $time_local, локальное время в common log format.

Строки заголовка, переданные клиенту, начинаются с префикса "sent_http_", например, $sent_http_content_range.

В конфигурации всегда существует предопределённый формат "combined":

```
log_format  combined  '$remote_addr - $remote_user [$time_local] '
                      '"$request" $status $body_bytes_sent '
                      '"$http_referer" "$http_user_agent"';
```

---

syntax: open_log_file_cache max=N [inactive=время]|off
default: open_log_file_cache off
context: http, server, location

Директива задаёт кэш, в котором хранятся дескрипторы файлов часто используемых логов, имена которых заданы переменными.

Параметры директивы:

- max задаёт маскимальное число дескрипторов в кэше; при переполнении кэша наиболее давно не используемые дескрипторы закрываются (LRU);
- inactive задаёт время, после которого дескриптор кэша закрывается, если к нему не было обращений в течение этого времени; по умолчанию 10 секунд;
- min_uses задаёт минимальное число использований файла в течение времени, заданного параметром inactive в директиве open_log_file_cache, после которого дескриптор файла будет оставаться окрытым в кэше; по умолчанию 1 использование;
- valid задаёт, через какое время нужно проверять, что файл ещё существует под тем же именем; по умолчанию 60 секунд;
- off запрещает кэш.

Пример использования:

```
open_log_file_cache      max=1000  inactive=20s  valid=1m  min_uses=2;
```

Директивы модуля ngx_http_map_module

17.06.2007

Модуль ngx_http_map_module создаёт переменные, значение которых зависят от значения других переменных.

Содержание

Пример конфигурации

```
map  $http_host  $name  {
    hostnames;

    default          0;

    example.com      1;
    *.example.com    1;
    test.com         2;
    *.test.com       2;
    .site.com        3;
    wap.*            4;
}
```

Директивы

---

syntax: map $переменная1 $переменная2 { ... }
default: нет
context: http

Директива создаёт переменную, значение которой зависит от значения исходной переменной. Директива поддерживает три специальных параметра:

- default задаёт значение для второй переменной, если значение первой переменной не найдено.
- hostnames указывает, что в качестве исходных значений можно использовать маску для первой или последней части имени хоста, например,

```
*.example.com  1;
example.*      1;
```

> Вместо двух записей

```
example.com    1;
*.example.com  1;
```

> можно использовать одну

```
.example.com   1;
```

- include включает файл со значениями. Включений может быть несколько.

---

syntax: map_hash_max_size число
default: map_hash_max_size 2048
context: http

Директива задаёт максимальный размер хэш-таблиц для переменных map.
Подробнее смотри в описании настройки хэшей.

---

syntax: map_hash_bucket_size число
default: map_hash_bucket_size 32/64/128
context: http

Директива задаёт размер корзины в хэш-таблицах для переменных map.
Значение по умолчанию зависит от размера строки кэша процессора.
Подробнее смотри в описании настройки хэшей.

Директивы модуля ngx_http_memcached_module

27.12.2007

Модуль ngx_http_memcached_module позволяет получать ответ из сервера memcached. Ключ задаётся в переменной $memcached_key. Ответ в memcached должен быть предварительно помещён внешним по отношению к nginx'у способом.

Содержание

Пример конфигурации

```
server {
    location / {
        set             $memcached_key   "$uri?$args";
        memcached_pass  host:11211;
        error_page      404 502 504 = @fallback;
    }

    location @fallback {
        proxy_pass      http://backend;
    }
}
```

Директивы

---

syntax: memcached_buffer_size размер
default: memcached_buffer_size 4k/8k
context: http, server, location

Директива задаёт размер буфера, в который будет читаться ответ, получаемый от сервера memcached. Ответ синхронно передаётся клиенту сразу же по мере его поступления.

---

syntax: memcached_connect_timeout время
default: memcached_connect_timeout 60
context: http, server, location

Директива задаёт таймаут для соединения с сервером memcached. Необходимо иметь в виду, что этот таймаут не может быть больше 75 секунд.

---

syntax: memcached_next_upstream [error|timeout|invalid_response|not_found|off]
default: memcached_next_upstream error timeout
context: http, server, location

Директива определяет, в каких случаях запрос будет передан следующему серверу memcached:

- error — произшла 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;
- invalid_response — сервер вернул пустой или неверный ответ;
- not_found — сервер не нашёл ответ;
- 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.

---

syntax: memcached_pass URL
default: нет
context: location, if в location, limit_except

Директива задаёт адрес сервера memcached. Адрес может быть указан в виде доменного имени или адреса и порта:

```
memcached_pass   localhost:11211;
```

Если доменное имя резолвится в несколько адресов, то все они будут использоваться в режиме round-robin. И кроме того, адрес может быть группой серверов.

---

syntax: memcached_read_timeout время
default: memcached_read_timeout 60
context: http, server, location

Директива задаёт таймаут при чтении ответа сервера memcached. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени сервер ничего не передаст, то nginx закрывает соединение.

---

syntax: memcached_send_timeout время
default: memcached_send_timeout 60
context: http, server, location

Директива задаёт таймаут при передаче запроса серверу memcached. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени сервер не примет новых данных, то nginx закрывает соединение.

Директивы модуля ngx_http_perl_module

19.12.2007

Модуль ngx_http_perl_module позволяет работать со встроенным в nginx perl'ом: делать обработчики location и переменной и вставлять вызовы perl'а в SSI. По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_perl_module`. Для сборки необходим perl версии 5.6.1 и выше, и компилятор C, совместимый с тем, которым был собран perl.

Содержание

Известные проблемы

Модуль экспериментальный, поэтому возможно всё.

Для того, чтобы во время переконфигурации perl перекомпилировал изменённые модули, его нужно собрать с параметрами -Dusemultiplicity=yes или -Dusethreads=yes. Кроме того, чтобы во время работы perl меньше терял память, его нужно собрать с параметром -Dusemymalloc=no. Узнать значения этих параметров у уже собранного perl'а можно так (в примерах приведены желаемые значения параметров):

```
$perl -V:usemultiplicity
usemultiplicity='define';

$perl -V:usemymalloc
usemymalloc='n';
```

Необходимо учитывать, что после пересборки perl'а с новыми параметрами -Dusemultiplicity=yes или -Dusethreads=yes придётся также переустановить и все бинарные perl'овые модули — они просто перестанут работать с новым perl'ом.

Возможно, основной процесс, а вслед за ним и рабочие процессы, будет увеличиваться в размерах при каждой переконфигурации. Когда основной процесс вырастет до неприемлемых размеров, можно воспользоваться процедурой обновления сервера на лету, не меняя при этом сам исполняемый файл.

Если perl'овый модуль выполняет длительную операцию, например, определяет адрес по имени, соединяется с другим сервером, делает запрос к базе данных, то на это время все остальные запросы данного рабочего процесса не будут обрабатываться. Поэтому рекомендуется ограничиться операциями, время исполнения которых короткое и предсказуемое, например, обращение к локальной файловой системе.

> Нижеописанные проблемы относятся только к версиям nignx'а до 0.6.22.

> Данные, возвращаемые методами объекта запроса $r, имеют только текстовое значение, причём само значение хранится в памяти, выделяемой не perl'ом, а nginx'ом из собственных пулов. Это позволяет уменьшить число операций копирования в большинстве случаев, однако в некоторых ситуациях это приводит к ошибке, например, при попытке использования таких значений в численном контексте рабочий процесс выходит с ошибкой (FreeBSD):

```
nginx in realloc(): warning: pointer to wrong page
Out of memory!
Callback called exit.
```

> или (Linux):

```
*** glibc detected *** realloc(): invalid pointer: ... ***
Out of memory!
Callback called exit.
```

> Обход такой ситуации простой — нужно присвоить значение метода переменной, например, такой код

```
my $i = $r->variable('counter') + 1;
```

> нужно заменить на

```
my $i = $r->variable('counter');
$i++;
```

> Так как строки внутри nginx'а в большинстве случаев хранятся без завершающего нуля, то они в таком же виде возвращаются методами объекта запроса $r (исключения составляют методы $r->filename и $r->request_body_file). Поэтому такие значения нельзя использовать в качестве имени файла и тому подобном. Обход такой же, как и предыдущей ситуации присвоение значения переменной (при этом происходит копирование данных и добавление необходимого нуля) или же использование в выражении, например:

```
open FILE, '/path/' . $r->variable('name');
```

Пример конфигурации

```
http {

   perl_modules  perl/lib;
   perl_require  hello.pm;

   perl_set  $msie6  '

       sub {
           my $r = shift;
           my $ua = $r->header_in("User-Agent");

           return "" if $ua =~ /Opera/;
           return "1" if $ua =~ / MSIE [6-9]\.\d+/;
           return "";
       }

   ';

   server {
       location / {
           perl  hello::handler;
       }
   }
```

модуль perl/lib/hello.pm:

```
package hello;

use nginx;

sub handler {
   my $r = shift;

   $r->send_http_header("text/html");
   return OK if $r->header_only;

   $r->print("hello!\n<br/>");

   if (-f $r->filename or -d _) {
       $r->print($r->uri, " exists!\n");
   }

   return OK;
```

```
}

1;
__END__
```

Директивы

---

syntax: perl модуль::функция|'sub { ... }'
default: нет
context: location, limit_except

Директива устанавливает обработчик для данного location.

---

syntax: perl_modules путь
default: нет
context: http

Директива задаёт дополнительный путь для perl'овых модулей.

---

syntax: perl_require модуль
default: нет
context: http

Директива задаёт имя модуля, который будет подгружаться при каждой переконфигурации. Директив может быть несколько.

---

syntax: perl_set $переменная модуль::функция|'sub { ... }'
default: нет
context: http

Директива устанавливает обработчик переменной.

Вызов perl'а из SSI

Формат команды следующий
<!--# perl sub="модуль::функция" arg="параметр1" arg="параметр2" ... -->

- $r->args метод возвращает аргументы запроса.
- $r->filename метод возвращает имя файла, соответствующее URI запроса.

$r->has_request_body(обработчик) метод возвращает 0, если в запросе нет тела. Если же тело запроса есть, то устанавливается указанный обработчик и возвращается 1. По окончании приёма тела nginx вызовет установленный обработчик. Обратите внимание, что нужно передавать ссылку на функцию обработчика. Пример использования:

```
package hello;

use nginx;

sub handler {
   my $r = shift;

   if ($r->request_method ne "POST") {
       return DECLINED;
   }

   if ($r->has_request_body(\&post)) {
       return OK;
   }

   return HTTP_BAD_REQUEST;
}

sub post {
   my $r = shift;

   $r->send_http_header;

   $r->print("request_body: \"", $r->request_body, "\"<br/>");
   $r->print("request_body_file: \"", $r->request_body_file, "\"<br/>\n");

   return OK;
}

1;

__END__
```

- $r->allow_ranges метод разрешает использовать byte ranges при передаче ответа.
- $r->discard_request_body метод указывает nginx'у игнорировать тело запроса.
- $r->header_in(строка) метод возвращает значение заданной строки в заголовке запроса клиента.
- $r->header_only метод определяет, нужно ли передавать клиенту только заголовок ответа или весь ответ.
- $r->header_out(строка, значение) метод устанавливает значение для заданной строки в заголовке ответа.

- $r->internal_redirect(uri) — метод делает внутренний редирект на указанный uri. Редирект происходит уже после завершения perl'ового обработчика.
- $r->print(текст, ...) — метод передаёт клиенту данные.
- $r->request_body — метод возвращает тело запроса клиента при условии, что тело не записано во временный файл. Для того, чтобы тело запроса клиента гарантировано находилось в памяти, нужно ограничить его размер с помощью client_max_body_size и задать достаточной размер для буфера client_body_buffer_size.
- $r->request_body_file — метод возвращает имя файла, в котором хранится тело запроса клиента. По завершению работы файл необходимо удалить. Для того, чтобы тело запроса клиента всегда записывалось в файл, нужно указать client_body_in_file_only on.
- $r->request_method — метод возвращает HTTP метод запроса клиента.
- $r->remote_addr — метод возвращает IP-адрес клиента.
- $r->flush — метод немедленно передаёт данные клиенту.
- $r->sendfile(имя [, смещение [, длина]]) — метод передаёт клиенту содержимое указанного файла. Необязательные параметры указывают начальное смещение и длину передаваемых данных. Собственно передача данных происходит уже после завершения perl'ового обработчика. Необходимо учитывать, что при использовании этого метода в подзапросе и директиве sendfile on содержимое файла не будет проходить через gzip, SSI и charset фильтры.
- $r->send_http_header(тип) — метод передаёт клиенту заголовок ответа. Необязательный параметр "тип" устанавливает значение строки "Content-Type" в заголовке ответа. Пустая строка в качестве типа запрещает строку "Content-Type".
- $r->status(код) — метод устанавливает код ответа.

$r->sleep(миллисекунды, обработчик) — метод устанавливает указанный обработчик и останавливает обработку запроса на заданное время. nginx в это время продолжает обрабатывать другие запросы. По истечении указанного времени nginx вызовет установленный обработчик. Обратите внимание, что нужно передавать ссылку на функцию обработчика. Для передачи данных между обработчиками следует использовать $r->variable(). Пример использования:

```
package hello;

use nginx;

sub handler {
   my $r = shift;

   $r->discard_request_body;
   $r->variable("var", "OK");
```

```
    $r->sleep(1000, \&next);

    return OK;
}

sub next {
    my $r = shift;

    $r->send_http_header;
    $r->print($r->variable("var"));

    return OK;
}

1;

__END__
```

- $r->unescape(текст) — метод декодирует текст, заданный в виде %XX.
- $r->uri — метод возвращает URI запроса.
- $r->variable(имя [, значение]) — метод возвращает или устанавливает значение указанной переменной. Переменные локальны для каждого запроса.

(C) Игорь Сысоев
http://sysoev.ru

Директивы модуля ngx_http_proxy_module

17.08.2009

Модуль ngx_http_proxy_module позволяет передавать запросы другому серверу.

Содержание

Встроенные переменные

Пример конфигурации

```
location / {
   proxy_pass        http://localhost:8000;
   proxy_set_header  Host       $host;
   proxy_set_header  X-Real-IP  $remote_addr;
}
```

Директивы

---

syntax: proxy_buffer_size размер
default: proxy_buffer_size 4k/8k
context: http, server, location

Директива задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от проксируемого сервера. В этой части ответа находится, как правило, небольшой заголовок ответа. По умолчанию размер буфера равен размеру одного буфера в директиве proxy_buffers, однако его можно сделать меньше.

---

syntax: proxy_buffering on|off
default: proxy_buffering on
context: http, server, location

Директива разрешает использовать буферизацию ответа проксируемого сервера. Если буферизация включена, то nginx принимает ответ проксируемого сервера как можно быстрее, сохраняя его в буфера, заданные директивами proxy_buffer_size и proxy_buffers. Если ответ не помещается полностью в память, то его часть записывается на диск.

Если буферизация выключена, то ответ синхронно передаётся клиенту сразу же по мере его поступления. nginx не пытается считать весь ответ проксируемого сервера, максимальный размер данных, который nginx может принять от сервера задаётся директивой proxy_buffer_size.

---

syntax: proxy_buffers число размер
default: proxy_buffers 8 4k/8k
context: http, server, location

Директива задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от проксируемого сервера. По умолчанию размер одного буфера равен размеру страницы, в зависимости от платформы это или 4K, или 8K.

---

syntax: proxy_cache [зона|off]
default: off
context: http, server, location

Директива задаёт зону для кэширования. Одна и та же зона может использоваться в нескольких местах. Параметр "off" запрещает кэширование, унаследованное с предыдущего уровня конфигурации.

---

syntax: proxy_cache_bypass строка [...]
default: off
context: http, server, location

Директива задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одной из строк переменных не пустое и не равно "0", то ответ не берётся из кэша:

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

Можно использовать совместно с директивой proxy_no_cache.

---

syntax: proxy_cache_key строка
default: $scheme$proxy_host$request_uri
context: http, server, location

Директива задаёт ключ для кэширования, например,

```
proxy_cache_key  "$host$request_uri $cookie_user";
```

> По умолчанию значение директивы близко к строке

```
proxy_cache_key  $scheme$proxy_host$uri$is_args$args;
```

---

syntax: proxy_cache_path путь [levels=уровни] keys_zone=название:размер [inactive=время] [max_size=размер]
default: нет
context: http

Директива 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, например, при использовании

```
proxy_cache_path  /data/nginx/cache  levels=1:2   keys_zone=one:10m;
```

> имена файлов в кэше будут такого вида:

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

Кэшируемый ответ 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, но 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, задаваемый директивой proxy_temp_path для данного location.

Кроме того, все активные ключи и информация о данных хранятся в разделяемой памяти зоне, имя и размер которой задаётся параметром keys_zone. Если к данным кэша не обращются 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, и при превышении его размеров удаляет самые невостребованные данные.

---

syntax: proxy_cache_min_uses число
default: proxy_cache_min_uses 1
context: http, server, location

Директива задаёт число запросов, после которого ответ будет закэширован.

---

syntax: proxy_cache_valid ответ [ответ ...] время
default: нет
context: http, server, location

Директива задаёт время кэширования для разных ответов. Например, директивы

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

> задают время кэширования 10 минут для ответов 200 и 302, и 1 минуту для ответов 404.

Если указано только время кэширования,

```
proxy_cache_valid  5m;
```

> то кэшируются только ответы 200, 301 и 302.

Кроме того, может кэшировать любые ответы с помощью параметра "any":

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

---

syntax: proxy_cache_use_stale [error | timeout | invalid_header | updating | http_500 | http_502 | http_503 | http_504 | http_404 | off] [...]
default: proxy_cache_use_stale off
context: http, server, location

Директива определяет, в каких случаях можно использовать устаревший закэшированный ответ, если при работе с проксированным сервером возникла ошибка. Параметры директивы совпадают с параметрами директивы proxy_next_upstream. И, кроме того, есть параметр updating, которой разрешает использовать устаревший закэшированный ответ, если на данный момент он уже обновляется.

---

syntax: proxy_connect_timeout время
default: proxy_connect_timeout 60
context: http, server, location

Директива задаёт таймаут для соединения с проксированным сервером. Необходимо иметь в виду, что этот таймаут не может быть больше 75 секунд.

---

syntax: proxy_hide_header имя
context: http, server, location

nginx не передаёт клиенту строки заголовка "Date", "Server", "X-Pad" и "X-Accel-..." из ответа проксированного сервера. Директива proxy_hide_header задаёт дополнительные строки. Если же строки нужно наоборот разрешить, то нужно воспользоваться директивой proxy_pass_header.

---

syntax: proxy_ignore_client_abort [on|off]
default: proxy_ignore_client_abort off
context: http, server, location

Директива определяет, закрывать ли соединение с проксированным сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

---

syntax: proxy_ignore_headers имя [имя ...]
context: http, server, location

Директива proxy_ignore_headers запрещает обработку некоторых строк заголовка из ответа проксированного сервера. В директиве можно указать строки "X-Accel-Redirect", "X-Accel-Expires", "Expires" и "Cache-Control".

---

syntax: proxy_intercept_errors [on|off]
default: proxy_intercept_errors off
context: http, server, location

Директива определяет, передавать ли клиенту проксированные ответы с кодом больше или равные 400 или же перенаправлять их на обработку nginx'у с помощью директивы error_page.

---

syntax: proxy_next_upstream [error | timeout | invalid_header | http_500 | http_502 | http_503 | http_504 | http_404 | off] [...]
default: proxy_next_upstream error timeout

context: http, server, location

Директива определяет, в каких случаях запрос будет передан следующему серверу:

- error — произшла 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;
- invalid_header — сервер вернул пустой или неверный ответ;
- http_500 — сервер вернул ответ с кодом 500;
- http_502 — сервер вернул ответ с кодом 502;
- http_503 — сервер вернул ответ с кодом 503;
- http_504 — сервер вернул ответ с кодом 504;
- http_404 — сервер вернул ответ с кодом 404;
- 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.

---

syntax: proxy_no_cache строка [...]
default: нет
context: http, server, location

Директива задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одной из строк переменных не пустое и не равно "0", то ответ не будет сохранён:

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

Можно использовать совместно с директивой proxy_cache_bypass.

---

syntax: proxy_pass URL
default: нет
context: location, if в location, limit_except

Директива задаёт адрес проксируемоего сервера и URI, на который будет отображаться location. Адрес может быть указан в виде доменного имени или адреса и порта, например,

```
    proxy_pass   http://localhost:8000/uri/;
```

> или в виде пути unix сокета:

```
    proxy_pass   http://unix:/tmp/backend.socket:/uri/;
```

> путь указан после слова unix и заключён между двумя двоеточиями.

Если доменное имя резолвится в несколько адресов, то все они будут использоваться в режиме round-robin. И кроме того, адрес можно задать группой серверов.

При передаче запроса серверу часть URI, соответствующая location, заменяется на URI, указанный в директиве proxy_pass. Но из этого правила есть два исключения, в которых нельзя определить заменяемый location:

- если location задан регулярным выражением;
- если внутри проксируемого location с помощью директивы rewrite изменяется URI и именно с этой конфигурацией будет обрабатываться запрос (break):

```
location  /name/ {
   rewrite      /name/([^/]+)  /users?name=$1  break;
   proxy_pass   http://127.0.0.1;
}
```

Для этих случаев URI передаётся без отображения.

Кроме того, можно указать, чтобы URI запроса передавалось в том же виде, как его прислал клиент, а не в в обработанном виде. Во время обработки

- два и более слэшей преобразуются в один слэш: "//" □ "/";
- убираются ссылки на текущий каталог: "/./" □ "/";
- убираются ссылки на предыдущий каталог: "/dir/../" □ "/".

Если на сервер нужно передать URI в необработанном виде, то для этого в директиве proxy_pass нужно указать URL сервера без URI:

```
location  /some/path/ {
   proxy_pass   http://127.0.0.1;
}
```

Имя сервера, его порт и передаваемый URI можно также полностью задать в помощью переменных:

```
    proxy_pass   http://$host$uri;
```

> или так:

```
proxy_pass   $request;
```

В этом случае имя сервера ищется среди описанных групп серверов и если не найдено, то определяется с помощью resolver'a.

---

syntax: proxy_pass_header имя
context: http, server, location

Директива разрешает передавать от проксируемого сервера клиенту запрещённые для передачи строки.

---

syntax: proxy_redirect [default|off|редирект замена]
default: proxy_redirect default
context: http, server, location

Директива задаёт текст, который нужно изменить в строках заголовка "Location" и "Refresh" в ответе проксируемого сервера. Предположим, проксируемый сервер вернул строку "Location: http://localhost:8000/two/some/uri/". Директива

```
proxy_redirect   http://localhost:8000/two/   http://frontend/one/;
```

> перепишет эту строку в виде "Location: http://frontend/one/some/uri/".

В заменяемой строке можно не указывать имя сервера:

```
proxy_redirect   http://localhost:8000/two/   /;
```

> тогда будет поставлено основное имя сервера и порт, если он отличен от 80.

Изменение по умолчанию, задаваемое параметром "default", использует параметры директив location и proxy_pass. Поэтому две нижеприведённые конфигурации одинаковы:

```
location /one/ {
   proxy_pass       http://upstream:port/two/;
   proxy_redirect   default;
location /one/ {
   proxy_pass       http://upstream:port/two/;
   proxy_redirect   http://upstream:port/two/   /one/;
```

В заменяемой строке можно использовать переменные:

```
    proxy_redirect    http://localhost:8000/    http://$host:$server_port/;
```

Директив может быть несколько:

```
    proxy_redirect    default;
  proxy_redirect    http://localhost:8000/    /;
  proxy_redirect    http://www.example.com/   /;
```

Параметр "off" запрещает все директивы proxy_redirect на данном уровне:

```
    proxy_redirect    off;
  proxy_redirect    default;
  proxy_redirect    http://localhost:8000/    /;
  proxy_redirect    http://www.example.com/   /;
```

С помощью этой директивы можно также добавлять имя хоста к относительным редиректам, выдаваемым проксируемым сервером:

```
    proxy_redirect    /    /;
```

---

syntax: proxy_read_timeout время
default: proxy_read_timeout 60
context: http, server, location

Директива задаёт таймаут при чтении ответа проксированного сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени проксируемый сервер ничего не передаст, то nginx закрывает соединение.

---

syntax: proxy_redirect_errors [on|off]

Директива переименована в proxy_intercept_errors.

---

syntax: proxy_send_timeout время
default: proxy_send_timeout 60
context: http, server, location

Директива задаёт таймаут при передаче запроса проксированному серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени проксируемый сервер не

примет новых данных, то nginx закрывает соединение.

---

syntax: proxy_set_header заголовок значение
default: Host и Connection
context: http, server, location

Директива позволяет переопределять или добавлять строки заголовка запроса, передаваемые проксируемому серверу. В качестве значения можно использовать текст, переменные и их комбинации. Директивы наследуются с предыдущего уровня при условии, что на данном уровне не описаны свои директивы proxy_set_header. По умолчанию переопределяются только две строки:

```
proxy_set_header  Host        $proxy_host;
proxy_set_header  Connection  close;
```

Неизменённую строку заголовка запроса "Host" можно передать так:

```
proxy_set_header  Host        $http_host;
```

Однако, если эта строка отсутствует в запросе клиента, то ничего передаваться не будет. В этом случае лучше воспользоваться переменной $host, её значение равно имени сервера в строке заголовка запроса "Host" или же основному имени сервера, если строки нет:

```
proxy_set_header  Host        $host;
```

Кроме того, можно передать имя сервера вместе с портом проксируемого сервера:

```
proxy_set_header  Host        $host:$proxy_port;
```

Если значение строки заголовка пустая строка, то строка вообще не будет передаваться проксируемому серверу:

```
proxy_set_header  Accept-Encoding  "";
```

---

syntax: proxy_ssl_session_reuse [on|off]
default: proxy_ssl_session_reuse on
context: http, server, location

Директива определяет, использовать ли повторно SSL-сессии при работе с проксированным сервером. Если в логах появляются ошибки

"SSL3_GET_FINISHED:digest check failed", то можно попробовать выключить повторное использование сессий.

---

syntax: proxy_store on | off | строка
default: proxy_store off
context: http, server, location

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

```
proxy_store   /data/www$original_uri;
```

Время модификации файлов выставляется согласно полученной строке "Last-Modified" в заголовке ответа. Ответ 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, но 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 proxy_temp_path для данного location.

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

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

location /fetch/ {
    internal;

    proxy_pass              http://backend/;
    proxy_store             on;
    proxy_store_access      user:rw  group:rw  all:r;
    proxy_temp_path         /data/temp;

    alias                   /data/www/;
}
```

или так:

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

location @fetch {
   internal;

   proxy_pass          http://backend;
   proxy_store         on;
   proxy_store_access  user:rw  group:rw  all:r;
   proxy_temp_path     /data/temp;

   root                /data/www;
}
```

---

syntax: proxy_store_access пользователи:права [пользователи:права] ...
default: proxy_store_access user:rw
context: http, server, location

Директива задаёт права доступа для создаваемых файлов и каталогов, например,

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

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

```
proxy_store_access  group:rw  all:r;
```

---

syntax: proxy_temp_path путь [уровень1 [уровень2 [уровень3]]]
default: proxy_temp_path proxy_temp
context: http, server, location

Директива задаёт имя каталога для хранения временных файлов полученных от другого сервера. В каталоге может использоваться иерархия подкаталогов до трёх уровней. Например, при такой конфигурации

```
proxy_temp_path  /spool/nginx/proxy_temp 1 2;
```

> имя временного будет такого вида:

```
/spool/nginx/proxy_temp/7/45/00000123457
```

Встроенные переменные

В модуле ngx_http_proxy_module есть встроенные переменные, которые можно использовать для формирования заголовков с помощью директивы proxy_set_header:

- $proxy_host, эта переменная равна имени проксируемого хоста и порта;
- $proxy_port, эта переменная равна порту проксируемого хоста;
- $proxy_add_x_forwarded_for, эта переменная равна строке заголовка запроса клиента "X-Forwarded-For" и добавленной к ней через запятую переменной $remote_addr. Если же строки "X-Forwarded-For" в запросе клиента нет, то переменная $proxy_add_x_forwarded_for равна переменной $remote_addr.

Директивы модуля ngx_http_random_index_module

09.08.2008

Модуль ngx_http_random_index_module выдаёт случайный файл в качестве индексного файла каталога. Модуль работает до модуля ngx_http_index_module. По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_random_index_module`.

Содержание

Пример конфигурации

```
location / {
    random_index on;
}
```

Директивы

---

syntax: random_index [on|off]
default: random_index off
context: location

Директива разрешает или запрещает работу модуля.

Директивы модуля ngx_http_realip_module

13.10.2006

Модуль ngx_http_realip_module позволяет менять адрес клиента на переданный в указанной строке заголовка. По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_realip_module`.

Содержание

Пример конфигурации

```
 set_real_ip_from   192.168.1.0/24;
set_real_ip_from   192.168.2.1;
real_ip_header     X-Real-IP;
```

Директивы

---

syntax: set_real_ip_from [адрес|CIDR]
default: нет
context: http, server, location

Директива описывает доверенные адреса, которые передают верный адрес для замены.

---

syntax: real_ip_header имя строки[X-Real-IP|X-Forwarded-For]
default: real_ip_header X-Real-IP
context: http, server, location

Директива указывает название строки в заголовке запроса, в котором передаётся адрес для замены. В случае строки "X-Forwarded-For" используется последний адрес в значении строки. Для остальных строк используется всё значение.

Директивы модуля ngx_http_referer_module

15.08.2007

Модуль ngx_http_referer_module позволяет блокировать доступ к сайту с неверными значениями строки "Referer" в заголовке запроса. Следует иметь в виду, что подделать запрос с нужной строкой "Referer" не составляет большого труда, поэтому цель использования данного модуля заключается не в стопроцентном блокировании подобных запросов, а в блокировании массового потока запросов, сделанных обычными браузерами. Нужно также учитывать, что обычные браузеры могут не передавать строку "Referer" даже для верных запросов.

Содержание

Пример конфигурации

```
 valid_referers   none  blocked  server_names
                 *.example.com  www.example.info/galleries/
                 ~\.google\. ;

if ($invalid_referer) {
    return   403;
}
```

Директивы

---

syntax: valid_referers [none|blocked|server_names|строка] ...
default: нет
context: server, location

Директива задаёт значения строки "Referer" в заголовке запроса, при которых встроенная переменная $invalid_referer будет иметь значение 0.

Параметры могут быть следующие:

- none строка "Referer" в заголовке запроса отсутствует;
- blocked строка "Referer" в заголовке запроса присутствует, но её значение удалено файрволлом или прокси-сервером; к таким строкам относятся строки, неначинающиеся на "http://";

- server_names в строке "Referer" в заголовке запроса указано одно из имён сервера;
- произвольная строка задаёт имя сервера и необязательное начало URI. В начале или конце имени сервера может быть "*". При проверке порт сервера в строке "Referer" игнорируется.
- регулярное выражение в начале должен быть символ "~". Необходимо учитывать, что на совпадение с выражением будет проверяться текст, начинающийся после "http://".

Пример использования:

```
valid_referers   none  blocked  server_names
                *.example.com  example.*  www.example.info/galleries/
                ~\.google\. ;
```

Директивы модуля ngx_http_rewrite_module

09.01.2007

Модуль ngx_http_rewrite_module позволяет изменять URI с помощью регулярных выражений, делать редиректы и выбирать конфигурацию в зависимости от переменных. Если директивы этого модуля описаны на уровне сервера, то они выполняются до того, как определяется location для запроса. Если в выбранном location тоже есть директивы модуля ngx_http_rewrite_module, то они также выполняются. Если URI изменился в результате исполнения директив внутри location, то снова определяется location для уже нового URI. Этот цикл может повторяться до 10 раз, после чего nginx возвращает ошибку "Server Internal Error" (500).

Содержание

Директивы

---

syntax: break
default: нет
context: server, location, if

Директива завершает обработку текущего набора директив ngx_http_rewrite_module.

Пример использования:

```
if ($slow) {
   limit_rate  10k;
   break;
```

```
}
```

---

syntax: if (условие) { ... }
default: нет
context: server, location

Директива if проверяет истинность условия, если оно истинно, то выполняется указанный в фигурных скобках код и запрос обрабатывается в соответствии с заданной там же конфигурацией. Конфигурация внутри директивы if наследуется из предыдущего уровня.

В качестве условия могут быть заданы:

- имя переменной; ложными значениями переменной являются пустая строка "" или любая строка, начинающиеся на "0";
- сравнение переменной со строкой с помощью операторов "=" и "!=";
- проверка переменной с помощью регулярного выражения без учёта регистра символов "~*" и с учётом "~". В регулярных выражениях можно использовать выделения, которые затем доступны в виде переменных $1 $9. Также можно использовать отрицательные операторы "!~" и "!~*". Если в регулярном выражении встречаются символы "}" или ";", то всё выражение нужно заключить в одинарные или двойные кавычки.
- проверка существования файла с помощью операторов "-f" и "!-f";
- проверка существования каталога с помощью операторов "-d" и "!-d";
- проверка существования файла, каталога или символической ссылки с помощью операторов "-e" и "!-e";
- проверка исполняемости файла с помощью операторов "-x" и "!-x".

Примеры использования:

```
 if ($http_user_agent ~ MSIE) {
    rewrite  ^(.*)$  /msie/$1  break;
}

if ($http_cookie ~* "id=([^;]+)(?:;|$)" ) {
    set  $id  $1;
}

if ($request_method = POST ) {
    return 405;
}


if ($slow) {
    limit_rate  10k;
}
```

```
if ($invalid_referer) {
    return   403;
}
```

Значение встроенной переменной $invalid_referer задаётся директивой valid_referers.

---

syntax: return код
default: нет
context: server, location, if

Директива return завершает исполнение кода и возвращает клиенту указанный код. Можно использовать следующие значения: 204, 400, 402 — 406, 408, 410, 411, 413, 416 и 500 — 504. Кроме того, нестандартный код 444 закрывает соединение без передачи заголовка ответа.

---

syntax: rewrite regex замена флаг
default: нет
context: server, location, if

Директива rewrite изменяет URI в соответствии с регулярным выражением и строкой замены. Директивы выполняются в порядке их следования в конфигурационном файле. С помощью флагов можно досрочно прекратить исполнение директив. Если строка замены начинается с "http://", то клиенту будет возвращён редирект и обработка директив также завершается.

Флаги могут быть следующими:

- last — завершает обработку текущего набора директив ngx_http_rewrite_module, после чего ищется соответствие URI и location;
- break — завершает обработку текущего набора директив ngx_http_rewrite_module;
- redirect — возвращает временный редирект с кодом 302; используется, если заменяющая строка не начинается с "http://";
- permanent — возвращает постоянный редирект с кодом 301.

Пример использования:

```
 rewrite  ^(/download/.*)/media/(.*)\..*$  $1/mp3/$2.mp3  last;
rewrite  ^(/download/.*)/audio/(.*)\..*$  $1/mp3/$2.ra   last;
return   403;
```

Если же эти директивы поместить в location /download/, то нужно заменить флаг last на break, иначе nginx сделает 10 циклов и вернёт ошибку 500:

```
location /download/ {
   rewrite  ^(/download/.*)/media/(.*)\..*$  $1/mp3/$2.mp3  break;
   rewrite  ^(/download/.*)/audio/(.*)\..*$  $1/mp3/$2.ra   break;
   return   403;
}
```

Если в строке замены указаны аргументы, то предыдущие аргументы запроса добавляются после них. Можно отказаться от этого добавления, указав в конце строки замены знак вопроса:

```
rewrite  ^/users/(.*)$  /show?user=$1?  last;
```

Если в регулярном выражении встречаются символы "}" или ";", то всё выражение нужно заключить в одинарные или двойные кавычки.

---

syntax: set переменная значение
default: нет
context: server, location, if

Директива устанавливает значение для указанной переменной. В качестве значения можно использовать текст, переменные и их комбинации.

---

syntax: uninitialized_variable_warn on|off
default: uninitialized_variable_warn on
context: http, server, location, if

Директива определяет, нужно ли писать в лог предупреждение о неинициализированной переменной.

Внутреннее устройство

Директивы модуля ngx_http_rewrite_module компилируется на стадии конфигурирования во внутренние коды, исполняемые во время запроса интерпретатором. Интерпретатор представляет из себя простую стековую виртуальную машину.

Например, директивы

```
 location /download/ {
    if ($forbidden) {
        return   403;
    }

    if ($slow) {
        limit_rate  10k;
    }

    rewrite  ^/(download/.*)/media/(.*)\..*$  /$1/mp3/$2.mp3  break;
}
```

будет скомпилированы в такие коды:

```
 переменная  $forbidden
проверка на ноль
    возврат  403
    завершение всего кода
переменная  $slow
проверка на ноль
проверка регулярного выражения
копирование  "/"
копирование  $1
копирование  "/mp3/"
копирование  $2
копирование  ".mp3"
завершение регулярного выражения
завершение всего кода
```

Обратите внимание, что кода для директивы limit_rate нет, поскольку она не имеет отношения к модулю ngx_http_rewrite_module. Для блока if создаётся такая же конфигурация, как и для блока location. Если условие истинно, то запрос получает конфигурацию, соответствующую блоку if, и в этой конфигурации limit_rate равен 10k.

Директиву

```
 rewrite  ^/(download/.*)/media/(.*)\..*$  /$1/mp3/$2.mp3  break;
```

можно сделать на один код меньше, если в регулярном выражении включить первый слэш в скобки:

```
 rewrite  ^(/download/.*)/media/(.*)\..*$  $1/mp3/$2.mp3  break;
```

тогда её коды будут выглядеть так:

```
 проверка регулярного выражения
копирование  $1
копирование  "/mp3/"
копирование  $2
копирование  ".mp3"
завершение регулярного выражения
завершение всего кода
```

Директивы модуля ngx_http_secure_link_module

13.10.2008

Модуль ngx_http_secure_link_module — это модуль проверяющий правильность запрашиваемой ссылки (0.7.18+). По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_secure_link_module`.

Содержание

Пример конфигурации

```
location /p/ {
   secure_link_secret   some_secret_word;

   if ($secure_link = "") {
       return 403;
   }
}
```

Директивы

---

syntax: secure_link_secret слово
default: нет
context: location

Директива задаёт секретное слово для проверки правильности ссылки. Полный URL защищённой ссылки вглядит так:

```
/prefix/hash/ссылка
```

> где hash считается как

```
md5(ссылка, секретное_слово);
```

Префикс — произвольная строка, не включающая слэш.

- $secure_link, эта переменная равна ссылке выделенной из полного URL'а. Если хэш неверный, то переменная равна пустой строке.

Директивы модуля ngx_http_split_clients_module

24.05.2010

Модуль ngx_http_split_clients_module создаёт переменные для A/B split-тестирования.

Содержание

Пример конфигурации

```
http {
   split_clients "${remote_addr}AAA"  $variant {
                  0.5%                .one;
                  2.0%                .two;
                  *                   "";
   }

   server {
       location / {
           index  index${variant}.html;
```

Директивы

---

syntax: split_clients $переменная1 $переменная2 { ... }
default: нет
context: http

Директива создаёт переменную для A/B split-тестирования, например:

```
split_clients "${remote_addr}AAA"  $variant {
              0.5%                .one;
              2.0%                .two;
              *                   "";
}
```

> Значение исходной строки переменных хэшируется с помощью CRC32. В приведённом примере при значениях хэша от 0 до 21474836 (0.5%) переменная $variant получит значение ".one". При значениях хэша от 21474837 до 107374182 (2%) □ ".two". И при значениях хэша от 107374183 до 4294967297 □ "".

Директивы модуля ngx_http_ssi_module

15.10.2007

Модуль ngx_http_ssi_module фильтр, обрабатывающий команды SSI (Server Side Includes) в проходящих через него ответах. На данный момент список поддерживаемых команд SSI неполон.

Содержание

Пример конфигурации

```
location / {
    ssi  on;
    ...
}
```

Директивы

---

syntax: ssi [on|off]
default: ssi off
context: http, server, location, if в location

Директива разрешает обработку команд SSI в ответах.

---

syntax: ssi_silent_errors [on|off]
default: ssi_silent_errors off
context: http, server, location

Директива разрешает не выводить строку "[an error occurred while processing the directive]", если во время обработки SSI произошла ошибка.

syntax: ssi_types mime-тип [mime-тип ...]
default: ssi_types text/html
context: http, server, location

Разрешает обработку команд SSI в ответах с указанными MIME-типами в дополнение к "text/html".

Команды SSI

Формат команды следующий

```
<!--# команда параметр1=значение параметр2=значение ... -->
```

Ниже перечислены поддерживаемые команды:

block — команда описывает блок, который можно использовать как заглушку в команде include. Внутри блока могут быть команды SSI.
name — имя блока.
Пример использования:

```
<!--# block name="one" -->
заглушка
<!--# endblock -->
```

config — команда задаёт некоторые параметры при обработке SSI.

- errmsg — строка, выводящаяся при ошибке во время обработки SSI. По умолчанию используется такая строка:

"[an error occurred while processing the directive]"

timefmt — строка, используемая функцией strftime(3) для вывода дат и времени. По умолчанию используется такой формат:

```
"%A, %d-%b-%Y %H:%M:%S %Z"
```

- Для вывода времени в секундах подходит формат "%s".

echo — команда выводит значение переменной.

- var — имя переменной.
- encoding — способ кодирования. Возможны три значения — none, url и entity. По умолчанию используется entity.

default — нестандартный параметр, задающий строку, которая выводится, если переменная не опеределена. По умолчанию выводится строка "none". Команда

```
<!--# echo var="name" default="нет" -->
```

> заменяет такую последовательность команд

```
<!--# if expr="$name" --><!--# echo var="name" --><!--#
      else -->нет<!--# endif -->
```

if — команда выполняет условное включение. Поддерживаются следующие команды:

```
<!--# if expr="..." -->
...
<!--# elif expr="..." -->
...
<!--# else -->
...
<!--# endif -->
```

> На данный момент поддерживаются только один уровень вложенности.

expr — выражение. В выражении может быть проверка существования переменной:

```
<!--# if expr="$name" -->
```

> сравнение переменной с текстом:

```
<!--# if expr="$name = text" -->
<!--# if expr="$name != text" -->
```

> или с регулярным выражением:

```
<!--# if expr="$name = /text/" -->
<!--# if expr="$name != /text/" -->
```

- Если в text встречаются переменные, то производится подстановка их значений.

include — команда включает в ответ результат другого запроса.

file — задаёт включаемый файл, например:

```
<!--# include file="footer.html" -->
```

virtual — задаёт включаемый запрос, например:

```
<!--# include virtual="/remote/body.php?argument=value" -->
```

- Несколько запросов на одной странице, обрабатываемые через прокси или FastCGI, работают параллельно. Если нужно последовательная обработка, то нужно воспользоваться параметром wait.

stub — нестандартный параметр, задающий имя блока, содержимое которого будет выведено, если тело ответа на включаемый запрос пустое или при исполнении запроса произошла ошибка, например:

```
<!--# block name="one" -->&amp;nbsp;<!--# endblock -->
<!--# include virtual="/remote/body.php?argument=value" stub="one" -->
```

- при этом содержимое замещающего блока обрабатывается в контексте включаемого запроса.

wait — нестандартный параметр, указывающий, нужно ли ждать полного исполнения данного запроса, прежде чем продолжать выполнение SSI,

например:

```
<!--# include virtual="/remote/body.php?argument=value" wait="yes" -->
```

set нестандартный параметр, указывающий, что удачный результат выполнения запроса нужно записать в заданную переменную, например:

```
<!--# include virtual="/remote/body.php?argument=value" set="one" -->
```

- Необходимо учитывать, что в переменные можно записать только результаты ответов, полученные через модули ngx_http_proxy_module и ngx_http_memcached_module.

set команда присваивает значение переменной.

- var имя переменной.
- value значение переменной. Если в присваиваемом значении есть переменные, то производится подстановка их значений.

Встроенные переменные

Модуль ngx_http_ssi_module поддерживает две встроенные переменные:

- $date_local, эта переменная равна текущему времени в локальной временной зоне. Формат даты задаётся командой config с параметром timefmt.
- $date_gmt, эта переменная равна текущему времени в GMT. Формат даты задаётся командой config с параметром timefmt.

Директивы модуля ngx_http_ssl_module

14.10.2009

Модуль ngx_http_ssl_module обеспечивает работу по протоколу HTTPS. Поддерживается проверка сертификатов клиентов с ограничением если в файле, заданном директивой ssl_certificate, указана цепочка сертификатов, то при проверке клиентских сертификатов nginx также будет использовать и сертификаты этих промежуточных CA.

По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_ssl_module`. Для сборки и работы этого модуля нужна библиотека OpenSSL.

Содержание

Пример конфигурации

Для уменьшения загрузки процессора рекомендуется

- установить число рабочих процессов равным числу процессоров,
- разрешить keep-alive соединения,
- включить разделяемый кэш сессий,
- выключить встроенный кэш сессий
- и, возможно, увеличить время жизни сессии (по умолчанию 5 минут):

```
worker_processes  2;

http {

   ...

   server {
       listen               443;
       keepalive_timeout    70;

       ssl                  on;
       ssl_protocols        SSLv3 TLSv1;
       ssl_ciphers          AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
       ssl_certificate      /usr/local/nginx/conf/cert.pem;
       ssl_certificate_key  /usr/local/nginx/conf/cert.key;
       ssl_session_cache    shared:SSL:10m;
       ssl_session_timeout  10m;

       ...
   }
```

Директивы

---

syntax: ssl [on|off]
default: ssl off
context: http, server

Директива разрешает протокол HTTPS для данного виртуального сервера.

---

syntax: ssl_certificate файл
default: нет
context: http, server

Директива указывает файл с сертификатом в формате PEM для данного виртуального сервера. Если вместе с основным сертификатом нужно указать промежуточные, то они должны находиться в этом же файле в следующем порядке сначала основной сертификат, а затем промежуточные. В этом же файле может находиться секретный ключ в формате PEM.

Нужно иметь ввиду, что из-за ограничения протокола HTTPS виртуальные сервера должны слушать на разных IP-адресах:

```
 server {
    listen           192.168.1.1:443;
    server_name      one.example.com;
    ssl_certificate  /usr/local/nginx/conf/one.example.com.cert;
    ...
}

server {
    listen           192.168.1.2:443;
    server_name      two.example.com;
    ssl_certificate  /usr/local/nginx/conf/two.example.com.cert;
    ...
}
```

иначе для второго сайта будет выдаваться сертификат первого сервера.

---

syntax: ssl_certificate_key файл
default: нет
context: http, server

Директива указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

---

syntax: ssl_client_certificate файл
default: нет
context: http, server

Директива указывает файл с сертификатами CA в формате PEM, используемыми для для проверки клиентских сертификатов.

---

syntax: ssl_ciphers шифры
default: ssl_ciphers HIGH:!ADH:!MD5
context: http, server

Директива описывает разрешённые шифры. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL, например:

```
ssl_ciphers  ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

Полный список можно посмотреть с помощью команды `openssl ciphers`.

---

syntax: ssl_crl файл
default: нет
context: http, server

Директива (0.8.7) указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми для для проверки клиентских сертификатов.

---

syntax: ssl_dhparam файл
default: нет
context: http, server

Директива (0.7.2) указывает файл с параметрами для шифров с обменом EDH-ключами.

---

syntax: ssl_prefer_server_ciphers [on|off]
default: ssl_prefer_server_ciphers off
context: http, server

Директива указывает, чтобы при использовании протоколов SSLv3 и TLSv1 серверные шифры были более приоритетны, чем клиентские.

---

syntax: ssl_protocols [SSLv2] [SSLv3] [TLSv1]
default: ssl_protocols SSLv3 TLSv1
context: http, server

Директива разрешает указанные протоколы.

---

syntax: ssl_verify_client on|off|optional
default: ssl_verify_client off
context: http, server

Директива разрешает проверку клиентских сертификатов. Параметр optional (0.8.7+) запрашивает сертификат клиента и проверяет его, если он предоставлен. Результат проверки можно узнать в переменной $ssl_client_verify.

---

syntax: ssl_verify_depth число
default: ssl_verify_depth 1
context: http, server

Директива устанавливает глубину проверку в цепочке клиентских сертификатов.

---

syntax: ssl_session_cache off|none|[builtin[:размер]] [shared:название:размер]
default: ssl_session_cache none
context: http, server

Директива задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

- off жёсткое запрещение использования кэша сессий: nginx явно говорит клиенту, что сессии не могут использоваться повторно.
- none мягкое запрещение использования кэша сессий: nginx говорит клиенту, что сессии могут использоваться повторно, но на самом деле не используются.
- builtin встроенный в OpenSSL кэш, используется в рамках только одного рабочего процесса. Размер кэша задаётся в сессиях. Если размер не задан, то он равен 20480 сессиям. Использование встроенного кэша может вести к фрагментации памяти.
- shared разделяемый между всеми рабочими процессами. Размер кэша задаётся в байтах, в 1 мегабайт может поместиться около 4000 сессий. У каждого разделяемого кэша должно быть произвольное название. Кэш с одинаковым названием может использоваться в нескольких виртуальных серверах.

Можно использовать одновременно оба типа кэша, например:

```
ssl_session_cache  builtin:1000  shared:SSL:10m;
```

> однако использование только разделяемого кэша без встроенного должно быть более эффективным.

---

syntax: ssl_session_timeout время
default: ssl_session_timeout 5m
context: http, server

Директива задаёт время, в течение которого клиент может повторно использовать параметры сессии, хранящейся в кэше.

Обработка ошибок

Модуль ngx_http_ssl_module поддерживает несколько нестандартных кодов ошибок, которые можно использовать для перенаправления с помощью директивы error_page:

- 495 — при проверке клиентского сертификата произошла ошибка;
- 496 — клиент не предоставил требуемый сертификат;
- 497 — обычный запрос был послан на порт HTTPS.

Перенаправление делается после того, как запрос полностью разобран и доступны такие переменные, как $request_uri, $uri, $arg и прочие.

Встроенные переменные

Модуль ngx_http_ssl_module поддерживает несколько встроенных переменных:

- $ssl_cipher возвращает строку используемых шифров для установленного SSL-соединения;
- $ssl_client_cert возвращает клиентский сертификат для установленного SSL-соединения в формате PEM перед каждой строкой которого, кроме первой, вставляется символ табуляции; предназачен для использования в директиве proxy_set_header.
- $ssl_client_raw_cert возвращает клиентский сертификат для установленного SSL-соединения в формате PEM;
- $ssl_client_serial возвращает серийный номер клиентского сертификата для установленного SSL-соединения;
- $ssl_client_s_dn возвращает строку subject DN клиентского сертификата для установленного SSL-соединения;
- $ssl_client_i_dn возвращает строку issuer DN клиентского сертификата для установленного SSL-соединения.
- $ssl_client_verify возвращает результат проверки клиентского сертификата: "SUCCESS", "FAILED" и, если серфтикат не был предоставлен - "NONE".
- $ssl_protocol возвращает протокол установленного SSL-соединения;
- $ssl_session_id возвращает идентификатор сессии установленного SSL-соединения;

Директивы модуля ngx_http_sub_module

19.04.2007

Модуль ngx_http_sub_module — это фильтр, изменяющий в ответе одну заданную строку на другую. По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_sub_module`.

Содержание

Пример конфигурации

```
location / {
    sub_filter        </head>
           '</head><script language="javascript" src="$script"></script>';
    sub_filter_once   on;
}
```

Директивы

---

syntax: sub_filter строка замена
default: нет
context: http, server, location

Директива задаёт строку, которую нужно заменить, и строку замены. Заменяемая строка проверяется без учёта регистра. В строке замены можно использовать переменные.

---

syntax: sub_filter_once on|off
default: sub_filter_once on
context: http, server, location

Директива определяет, сколько раз нужно искать заменяемую строку — один раз или несколько.

---

syntax: sub_filter_types mime-тип [mime-тип ...]
default: sub_filter_types text/html
context: http, server, location

Директива разрешает замену строк в ответах с указанными MIME-типами в дополнение к "text/html".

Директивы модуля ngx_http_userid_module

09.11.2005

Модуль ngx_http_userid_module выдаёт cookies для идентификации клиентов. Для записи в лог можно использовать переменные $uid_got и $uid_set. Модуль совместим с модулем mod_uid для Apache.

Содержание

Пример конфигурации

```
 userid          on;
userid_name     uid;
userid_domain   example.com;
userid_path     /;
userid_expires  365d;
userid_p3p      'policyref="/w3c/p3p.xml", CP="CUR ADM OUR NOR STA NID"';
```

Директивы

---

syntax: userid [on|v1|log||off]
default: userid off
context: http, server, location

Разрешает или запрещает выдавать cookie и записывать приходящие cookie в лог:

- on разрешает выдавать cookie версии 2 и записывать приходящие cookie в лог;
- v1 разрешает выдавать cookie версии 1 и записывать приходящие cookie в лог;

• log запрещает выдавать cookie, но разрешает записывать приходящие cookie в лог;
• off запрещает выдавать cookie и записывать приходящие cookie в лог;

---

syntax: userid_domain [имя|none]
default: userid_domain none
context: http, server, location

Директива задаёт домен, для которого устанавливается cookie. Параметр "none" запрещает выдавать домен для cookie.

---

syntax: userid_expires [время|max]
default: нет
context: http, server, location

Директива задаёт время, в течение которого браузер должен хранить cookie. Параметр "max" задаёт время 31 декабря 2037 года 23:55:55 GMT. Это максимальное время, которое понимают старые браузеры.

---

syntax: userid_name имя
default: userid_name uid
context: http, server, location

Директива задаёт имя cookie.

---

syntax: userid_p3p строка
default: нет
context: http, server, location

Директива задаёт значение для строки заголовка P3P, который будет выдваться вместе с cookie.

---

syntax: userid_path путь
default: userid_path /
context: http, server, location

Директива задаёт путь, для которого устанавливается cookie.

---

syntax: userid_service число
default: userid_service IP-адрес сервера
context: http, server, location

Директива задаёт номер сервиса, выдавшего cookie. По умолчанию для cookie первой версии используется ноль, а для второй IP-адрес сервера.

Директивы модуля ngx_http_xslt_module

04.08.2008

Модуль ngx_http_xslt_module это фильтр, преобразующий XML-ответ с помощью одного или нескольких XSLT-шаблонов (0.7.8+). По умолчанию модуль не собирается, нужно разрешить его сборку при конфигурировании параметром `--with-http_xslt_module`. Для сборки и работы этого модуля нужны библиотеки libxml2 и libxslt.

Содержание

Пример конфигурации

```
location / {
    xml_entities       /site/dtd/entities.dtd;
    xslt_stylesheet    /site/xslt/one.xslt   param=value;
    xslt_stylesheet    /site/xslt/two.xslt;
}
```

Директивы

---

syntax: xml_entities путь
default: нет
context: http, server, location

Директива задаёт файл DTD, в котором описаны символьные сущности. Этот файл компилируется на стадии конфигурации. По техническим причинам модуль не имеет возможности использовать внешнее подмножество, заданное в обрабатываемом XML, поэтому оно игнорируется, а вместо него используется специально заданный файл. В этом файле не нужно описывать структуру XML, достаточно только объявления необходимых символьных сущностей, например:

```
<!ENTITY nbsp " ">
```

---

syntax: xslt_stylesheet шаблон [параметр ...]
default: нет

context: location

Директива задаёт XSLT-шаблон и параметры для этого шаблона. Шаблон компилируется на стадии конфигурации. Параметры задаются в формате

```
param=value
```

Их можно задавать как по отдельности, так и группировать в одной строке, разделяя символом ":". Если же в самих параметрах встречается символ ":", то его нужно экранировать в виде "%3A". Кроме того, необходимо помнить о требовании libxslt, чтобы параметры, содержащие не только алфавитно-цифровые символы, были заключены в одинарные или двойные кавычки, например:

```
param1='http%3A//www.example.com':param2=value2
```

В описании параметров можно использовать переменные, например, целая строка параметров может быть взята из одной переменной:

```
location / {
   xslt_stylesheet    /site/xslt/one.xslt
                      $arg_xslt_params
                      param1='$value1':param2=value2
                      param3=value3;
}
```

Можно указать несколько шаблонов в этом случае они будут п