# Nginx安装

## 安装依赖

1. 安装编译工具，例如GNU的GCC。

yum install gcc

2．安装PCRE（Perl Compatible Regular Expression），因为Nginx的Rewrite模块和HTTP模块会使用到PCRE正则表达式语法。

yum install pcre pcre-devel

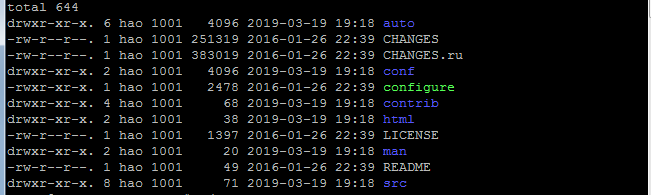
或者安装PCRE相关的所有安装包：yum install pcre\*

1. zlib库提供了压缩算法，Nginx的各个模块需要使用gzib压缩

yum install zlib zlib-devel

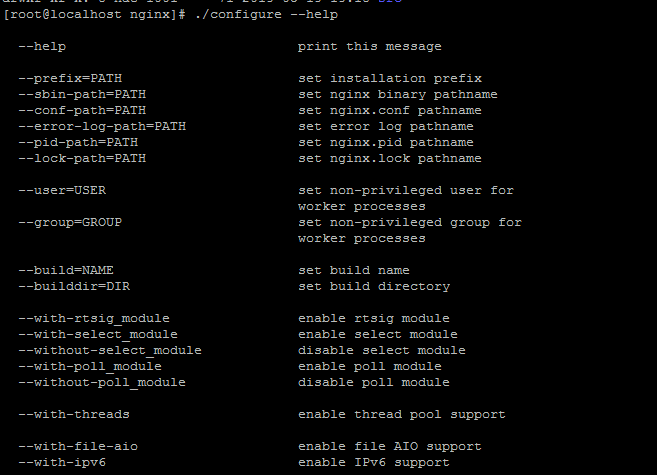
1. Nginx服务器提供安全网页时，则会使用OpenSSL

yum install openssl openssl-devel



## 配置选项

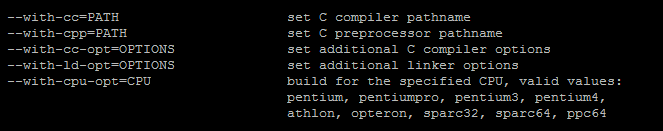
路径选择：在运行configure脚本命令的时候，可以开启一些开关选项。可以运行./configure –help命令列出有效的开关变量。



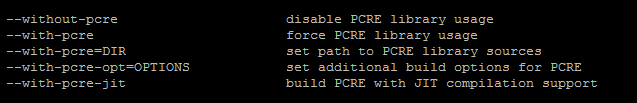
先决条件选项：先决条件的格式有库文件和二进制文件，即使它们已经安装到系统里，可能有时候配置脚本还是找不到它们的位置，例如，如果它们安装到非标准路径中。

Nginx为我们提供了一些开关选项：

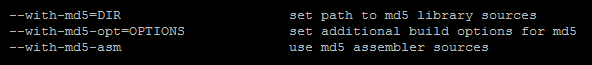
1. 编译选项



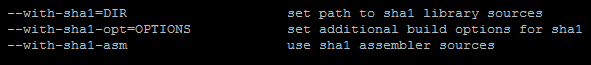
1. PCRE选项



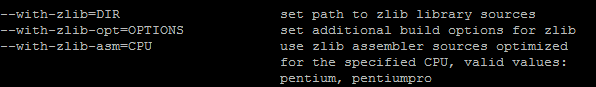
1. MD5选项



1. SHA1选项



1. zlib选项



1. OpenSSL选项



### 模块选择

* 默认开启的模块

--without-select\_module

--without-poll\_module disable poll module

--without-http\_charset\_module 该模块可以将服务器发出的HTTP响应重编码

--without-http\_gzip\_module disable ngx\_http\_gzip\_module

--without-http\_ssi\_module disable ngx\_http\_ssi\_module

--without-http\_userid\_module 该模块可以通过HTTP请求头

--without-http\_access\_module disable ngx\_http\_access\_module

--without-http\_auth\_basic\_module disable ngx\_http\_auth\_basic\_module

--without-http\_autoindex\_module disable ngx\_http\_autoindex\_module

--without-http\_geo\_module disable ngx\_http\_geo\_module

--without-http\_map\_module disable ngx\_http\_map\_module

--without-http\_split\_clients\_module disable ngx\_http\_split\_clients\_module

--without-http\_referer\_module disable ngx\_http\_referer\_module

--without-http\_rewrite\_module disable ngx\_http\_rewrite\_module

--without-http\_proxy\_module disable ngx\_http\_proxy\_module

--without-http\_fastcgi\_module disable ngx\_http\_fastcgi\_module

--without-http\_uwsgi\_module disable ngx\_http\_uwsgi\_module

--without-http\_scgi\_module disable ngx\_http\_scgi\_module

--without-http\_memcached\_module disable ngx\_http\_memcached\_module

--without-http\_limit\_conn\_module disable ngx\_http\_limit\_conn\_module

--without-http\_limit\_req\_module disable ngx\_http\_limit\_req\_module

--without-http\_empty\_gif\_module disable ngx\_http\_empty\_gif\_module

--without-http\_browser\_module disable ngx\_http\_browser\_module

--without-http\_upstream\_hash\_module

--without-http\_upstream\_ip\_hash\_module

--without-http\_upstream\_least\_conn\_module

--without-http\_upstream\_keepalive\_module

--without-http disable HTTP server

--without-http-cache disable HTTP cache

--without-pcre disable PCRE library usage

* 默认禁用的模块

--with-http\_ssl\_module 开启SSL模块，支持使用HTTPS协议

--with-http\_spdy\_module enable ngx\_http\_spdy\_module

--with-http\_realip\_module 开启Peal IP的支持，该模块用于从客户端请求头数据中读取real IP地址

--with-http\_addition\_module 开启Addition模块，该模块允许你追加或前置数据到响应的主题部分。

--with-http\_xslt\_module enable ngx\_http\_xslt\_module

--with-http\_image\_filter\_module 开启Image Filter模块，该模块是让你修改图像，注意：如果想编译该模块，需要在系统中安装libgd库

--with-http\_geoip\_module enable ngx\_http\_geoip\_module

--with-http\_sub\_module 开启Substitution模块，该模块用于在网页中替换文本

--with-http\_dav\_module 开启WebDAV模块

--with-http\_flv\_module 开启FLV模块，该模块用于处理.flv（flash视频）文件

--with-http\_mp4\_module enable ngx\_http\_mp4\_module

--with-http\_gunzip\_module enable ngx\_http\_gunzip\_module

--with-http\_gzip\_static\_module 开启Gzip静态模块，该模块用于发送预压缩的文件。

--with-http\_auth\_request\_module enable ngx\_http\_auth\_request\_module

--with-http\_random\_index\_module 开启Random Index模块，该模块用于挑选一个随机的文件作为该目录的index

--with-http\_secure\_link\_module 开启Secure Link模块，该模块用于在URL中检测关键字的存在。

--with-http\_degradation\_module enable ngx\_http\_degradation\_module

--with-http\_stub\_status\_module 开启Stub Status模块，该模块会产生一个服务器状态和信息页。

* 杂项

--with-mail enable POP3/IMAP4/SMTP proxy module

--with-mail\_ssl\_module enable ngx\_mail\_ssl\_module

--without-mail\_pop3\_module disable ngx\_mail\_pop3\_module

--without-mail\_imap\_module disable ngx\_mail\_imap\_module

--without-mail\_smtp\_module disable ngx\_mail\_smtp\_module

* 事件管理，允许你为Nginx定时器选择事件通知系统，仅适用于高级用户
* 用户和组选项

--user 指定nginx进程的默认用户，这个设置仅用于在配置文件中省略user指令来指定用户的情况

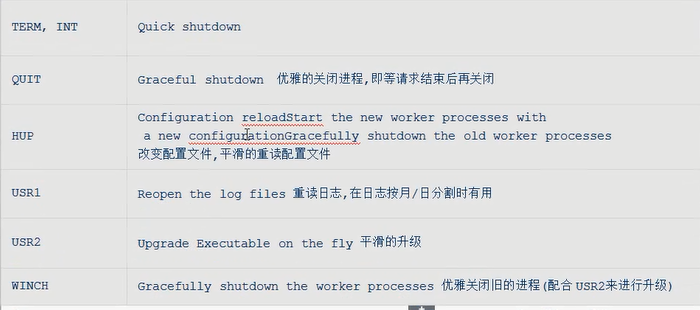
--group 指定nginx进程默认的组，这个设置仅用于在配置文件中省略使用group指令指定用户的情况。

--without-mail\_pop3\_module disable ngx\_mail\_pop3\_module

--without-mail\_imap\_module disable ngx\_mail\_imap\_module

--without-mail\_smtp\_module disable ngx\_mail\_smtp\_module

# Nginx信号控制



具体使用：Kill –信号选项 nginx端口号

Kill –INT port:迅速关闭nginx服务（不建议在生产环境建议使用）。

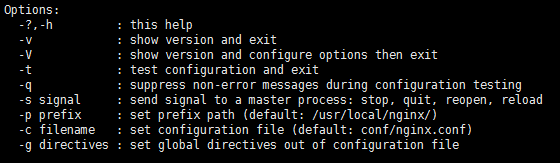
Kill –QUIT port:在请求结束后，关闭nginx服务。

Kill –HUP port：在改变配置文件后(nginx.conf),在不停止nginx服务下重新加载配置文件。

Kill –USR1 port :重新阅读日志文件(/usr/local/nginx/logs/access.log。

Kill –USR2 port:平滑的升级nginx

Kill –WINCH post: 优雅的关闭旧的进程。



Nginx –s stop：立即停止守护进程

Nginx –s quit：温和地停止守护进程

Nginx –s reopen：重新打开日志文件

Nginx –x reload：重新载入配置文件

Nginx –t ：测试配置文件

Nginx –t –c filepath：测试指定的配置文件

Nginx –V ：显示当前Nginx的版本号和configure脚本的各种开关选项

## 添加系统服务

在/etc/init.d目录下添加nginx文件，输入如下代码：

set -e

PATH=/usr/local/sbin:/usr/local/bin:/sbin/:/bin:/usr/sbin:/usr/bin

DESC="nginx daemon"

NAME=nginx

DAEMON=/usr/local/nginx/sbin/$NAME

SCRIPTNAME=/etc/init.d/$NAME

test -x $DAEMON || exit 0

d\_start(){

$DAEMON || echo -n "already running"

}

d\_stop(){

$DAEMON -s quit || echo -n "not running"

}

d\_reload(){

$DAEMON -s reload || echo -n "could not reload"

}

case "$1" in

start)

echo -n "Starting $DESC :$NAME"

d\_start

echo "."

;;

stop)

echo -n "Stopping $DESC:$NAME"

d\_stop

echo "."

;;

reload)

echo -n "Reload $DESC configuration..."

d\_reload

echo "reloaded."

;;

restart)

echo -n "Restarting $DESC:$NAME"

d\_stop

sleep 2

d\_start

echo "."

;;

\*)

echo "Usage:$SCRIPTNAME {start|stop|restart|reload}" >&2

exit 3

;;

esac

exit 0

# Nginx配置文件

Nginx的配置文件为nginx.conf文件，如果想使用新的配置文件启动nginx可以使用命令：/usr/local/nginx/sbin/nginx –c newConfigurationFileName

worker\_processes nums：全局配置，指有nums个工作的子进程，可以修改，但是太大没有意义，一般设置CPU个数\*内核数。

如：worker\_processes 1 ：标志Nginx作为单个工作进程

Include fileName：执行一个对特定文件的包含，换句话说，就是在该配置文件的内容中，将被插入。

如：include mime.types

|  |  |
| --- | --- |
| 标准名称 | 描述 |
| nginx.conf | 应用程序的基本配置文件 |
| mime.conf | 一个文件扩展列表文件，它们与MIME类型关联 |
| fastcgi.conf | 与FastCGI相关的配置文件 |
| proxy.conf | 与Proxy相关的配置文件 |
|  |  |

log\_format:设置日志格式

events { worker\_connections 1024; }：一般配置nginx连接属性，这里值一个子进程最多允许的连接，可以修改。

http{server{}}:配置http服务器的主要段，server指的是虚拟的主机，可以配置多个

1. server{

listen post –监听的端口号

server\_name serverName –serverName可以为域名或IP

location{

root path –可以是相对路径或绝对路径，指定目录是上级目录，并且该上级目录含有location指定名称的同名目录。

alias –指定的目录是准确的，给location指定一个目录

index index.html index.htm --首页

access\_log logs/host.access.log main;--虚拟主机的日志文件路径和使用格式

proxy\_pass –代

}

}

注意事项：

1.使用alias时，目录名后面一定要加”/“。

2.使用alias标签的目录块中不能使用rewrite的break。

3.alias在使用正则匹配时，必须捕捉要匹配的内容并在指定的内容处使用。

4.alias只能位于location块中

## 指令值的单位

k或K：千字节 m或M：兆字节 ms：Milliseconds（毫秒）

client\_max\_body\_size 2M client\_max\_body\_size 2045k

s：Second m：Minutes h：Hours

d：Days w：Weeks M：Months （30天） y：Years（365天）年

client\_body\_timeout 3m client\_body\_timeout 180s

## 变量

Nginx模块提供了各种变量，如：Nginx核心模块提供了$nginx\_version变量。在设置log\_format指令时。

'$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

字符串值：1.可以没有引号 root /home/www.haodashen/www;

2.使用特殊字符，如：空格符” ”，分号;，花括号{}，就需要使用单引号或双引号将其包括。Root ‘/home/www.haodashen/www web page’

## 基本模块

基本模块提供的指令允许你定义的Nginx基本功能的变量，在编译时它们不能被禁用。

### 核心模块（Core）

基本特征和指令，例如进程管理和安全。

核心模块指令：

**daemon：**启用或禁用守护进程，默认值为on

**debug\_points：**激活调试点，当一个调试点为了附加一个调试器来改变方向而使用stop来打断应用程序，使用abort来放弃一个调试点并且建立一个内核转储文件。

值为：stop或abort

默认值为：None

**env：**允许定义环境变量

语法：env MY\_VARIABLE

Env MY\_VARIABLE=my\_value

**error\_log：**使用环境：main ,http ,server和location

语法：error\_log/file/path level



这里的级别有：debug , info , notice ,warn ,error和crit（详细程度由低到高：debug提供了全部日志记录，crit仅报告关键错误）

能够提供不同的错误日志级别：应用程序，HTTP服务，虚拟主机和虚拟主机目

录。

通过重定向，日志记录可以重定向到/dev/null，在配置文件的根部使用下列指令

禁用错误日志记录：error\_log /dev/null crit ;

**lock\_file**：使用lock文件是为了互拆现象，默认是不使用，除非在编译时已启用

文件路径：lock\_file logs/nginx.lock

使用lock文件是为了互拆现象。默认是不使用的，除非在编译时已启用。

**log\_not\_found：**使用环境：main，http，server和location

开启或禁用404错误，如果日志中填满了无法访问favicon.ico或robots.txt文件而产生的404错误，可以考虑将其关闭，默认值为on

log\_not\_found on

**master\_process：**如果设置为on，Nginx将开启多个进程，一个主进程和worker进程；如果禁用，Nginx会以独一无二的进程来运行。该指令仅被用于测试，作为一个master进程，默认值为on

master\_process on

**pid：**

**ssl\_engine：**

**thread\_stack\_size**：定义线程堆栈的大小。

thread\_stack\_size 1m;

**timer\_resolution：**控制系统调用gettimeofday()时间间隔，它会通过调用噶函数来与内部时钟同步。如果这个值没有设置，在每一次内核事件通知后时钟都会被刷新。

timer\_resolution 100ms ;

user：在编译时指定，如果没有定义，则使用Nginx的master进程的用户和用户组。

**worker\_threads：**为每一个worker进程定义一定数量的线程。 默认值：无。

worker\_threads 8 ;

**worker\_cpu\_affinity：**该指令和worker\_processes协同工作，它可以让你的worker进程影响CPU内核，数字序列和worker进程一样多。如果你配置Nginx使用3个worker进程，那么就会有三个数字组合，对于双核CPU，每一块有两位数字：worker\_cpu\_affinity 01 01 10 ;

第一个01指示第一个worker进程会受第二个内核的影响。

第二个01指示第二个worker进程会受第二个内核的影响

第三个10指示第三个worker进程会受第一个内核的影响。

注意：该选项只适合linux，该选项允许你将一个worker进程绑定到一个CPU上。：

如：worker\_processes 4 ;

worker\_cpu\_affinity 0001 0010 0100 1000 ；将每一个进程绑定到一个CPU上。

Worker\_processes 2 ;

Worker\_cpu\_affinity 0101 1010;

将第一个worker进程绑定到CPU0和CPU2

第二个worker进程绑定到CPU1和CPU3

**worker-priority：**定义worker进程的优先级，从-20（最高级）到19（最低级），默认值为0。

**worker\_processes：**定义worker进程的数量，Nginx可将请求的处理分到多个worker进程。默认值为：1。

**worker\_rlimit\_core：**定义每个worker进程的内核文件大小。

默认值：无。

worker\_rlimit\_core 100m;

**worker\_rlimit\_nofile：**定义一个worker线程能够同时处理文件数量。

默认值：无。

Worker\_rlimit\_nofile 10000;

**worker\_rlimit\_sigpending：**定义每个用户能够被排入队列的信号数量。如果队列满，会由这个限制而导致信息被忽略。

worker\_rlimit\_sinpending 10000 ;

**working\_directory：**worker进程工作的目录，仅用于定义内核文件的位置。对于该目录，worker进程用户必须有写的权限，用于能够写入内核。在配置文件编译时通过prefix开关选项指定。

working\_directory /usr/local/nginx/ ;

### 事件模块（Events）

该模块提供的指令可以用来配置网络机制，指令的一些参数会对Nginx应用程序的性能产生重要的影响。

下面给出的指令必须方法events区段中:

**accept\_mutex：**启用或禁用使用一个接受互斥锁来打开套接字监听。

默认值：on

accept\_mutex on ;

**accept\_mutex\_delay：**定义一个worker进程在尝试再次获取资源之前应该等待的时间。如果指令accept\_mutex设置为off，则不使用该值。

accept\_mutex\_delay 500ms ;

默认值：500 milliseconds

**debug\_connection：**对于匹配客户端的IP端或IP区段，要记录详细的日志信息，这些信息放在error\_log指令指定的文件中，开启debug级别的详细日志。

debug\_connection 192.168.253.120

注意：为了使用该项功能，在配置编译Nginx时使用开关变量debug。

**multi\_accept：**定义Nginx是否立即接受从所有监听队列进入的连接。

默认值：off

**use：**值：/dev/poll ，epoll，eventport，kqueue，rtsig或select

默认值：在编译时指定。

在有效的模型中选择event的模型，但Nginx会自动选择最合适的一个。

select：默认的标准模块，如果OS不支持更有效的模型，则使用它。

poll：在自动选择上它优先于select，但在所有的系统上都无效。

**worker\_connection：**定义一个worker进程能够同时连接的数量

worker\_connection 1024

### 配置模块（Configuration）

Nginx的Configuration模块是一个简单的模块，它提供的include指令能够将其他文件包含再Nginx配置文件，在配置文件中的任何地方均可以插入该指令，并且在它的后面只有一个参数---文件路径

如：include sites/\*.conf

include /file/path.conf

# HTTP配置

HTTP核心模块包含HTTP服务器所有基本的区段（block），指令和变量。如果在配置编译时包括了它，那么它默认是启用的。

## 区段结构

http：该区段嵌入配置文件的根部，在这个区段中允许定义指令和嵌入HTTP相关模块的区段，该模块可以插入多次，但如果发生这种事，则后面的区段中的指令值会取代前面指令的值。

server：这个区段允许你声明一个站点。

location：定义一组设置，应用于网站的一个特定的位置，该区段能够用于server区段，也能嵌套在其他location中。

## 模块指令（ngx\_http\_core\_module）

### 套接字和主机配置

**listen：**指定用于提供Web服务站点监听的套接字所使用的IP地址/或端口号，网站一般都在80端口提供服务。

语法：listen [address][port][其他选项]

其他选项：default：指定该server区段用于默认的网站，在该IP地址和端口上接受任何客户端的请求。

ssl：指定的网站提供SSL服务。

**server\_name：**在server区段中指定一个或多个主机名，Nginx收到Http请求时，它会与所有的server区段相比较，然后找到与客户端请求的header中的Host相匹配的server区段。

语法：server\_name hostname [hostname2……]

**server\_name\_in\_redirect：**用于内部重定向，如果设置为开启，Nginx将使用server\_name指令中指定的第一个主机名来进行重定向；如果设置为关闭，Nginx将使用客户端HTTP请求头中的Host的值进行重定向。

默认值为：on

**server\_names\_hash\_max\_size：**Nginx使用hash表来进行变量数据收集，意在加速请求进程。该指令用于定义存储服务器名称数量哈希表的最大值。如果服务器使用的主机名总数超过512个，则需要增加该值。默认值：512

**server\_names\_hash\_bucket\_size：**定义在服务器名称哈希表中一个条目的最大长度。如果服务器名称长度大于32个字符，必须增加该值。

**port\_in\_redirect：**就重定向来说，该指令定义Nginx是否将端口重定向后的端口添加在URL中。

**tcp\_nodelay：**开启或关系使用TCP\_NODELAY套接字选项，仅用于keep-alive连接。

**tcp\_nopush：**开启或禁用TCP\_NOPUSH或TCP\_CORK的socker选项。注意，该选项如果tcp\_nopush设置为on，那么Nginx将尝试在单个TCP数据包中发送整个HTTP响应头，默认值：off。

**sendfile：**如果这条指令被启用，Nginx将使用sendfile内核来调用处理文件传递。如

果禁用该指令，那么Nginx将自己处理文件传递。该选项可以会影响到服务器的性能，具体取决于被传输文件的物理地址。默认值为off。

**sendfile\_max\_chunk：**用于定义每次sendfile每一次调用数据的最大值。

**send\_lowat：**允许你在TCP套接字利用SO\_SNDLOWAT标志，这个选项仅在FrssBSD操作系统下可用。它的功能在于定义了用做输出操作的缓冲区大小。

**reset\_timeout\_connection：**当一个客户端连接超时时，其相关的信息可能保留在内存，正是依赖于这种状态，相关的信息才得以保留。然而启用该指令后，如果连接超时，将清除所有与内存的关联。

### 路径和文档

root：定义文档的根目录，该目录包含你希望为访问者提供的内容。

alias：只能放在location区段中，它为Nginx指定的文件路径提供别名，使Nginx为指定的请求找到访问的文件。

error\_page：允许你修改访问URI所产生的HTTP响应状态码并选择性地将它替换为其他代码。

if\_modified\_since：定义Nginx如何处理HTTP头中的If-Modified-Since，这种头多数是搜索引擎蜘蛛的请求，会指出上一次的时间和日期，如果被请求的文件从上次爬行后没有修改过，那么服务器简单地返回一个“304 Not Modified response”代码，而不包括任何body。

该指令接受三个值：off：忽略If-Modified-Since头。

exact：如果在HTTP头中指定的时间和日期与实际请求的文件的修改日期准确匹配，则返回”304 Not Modified”；如果文件修改日期在指定之前或有意隐瞒，文件通常会正常送达（200 Ok响应）

before：如果在HTTP头中指定的时间和日期在被请求文件修改日期之前或一样，则返回“304 Not Modified”

index：定义一个默认的页面，如果在请求中没有指定文件名，Nginx就会使用该页面提供服务。可以指定多个文件名，但使用的是第一个找到的文件，如果没有找到指定的文件，Nginx要么尝试产生一个自动的文件索引，要么返回403错误页面。也可以插入一个绝对路径名。

recursive\_error\_pages：有的时候，通过指令error\_page提供的错误页面本身也发生了错误，在这种情况下，指令error\_page会被再次使用，该指令开启或禁用递归错误页面。

try\_files：视图找到指定的文件，如果指定的这些文件存在的文件都不是，就跑到命名

location区段。

### MIME类型

## 文件缓存和处理

**direction：**如果启用该指令，需要指定一个值，大于该值的文件将通过Direct I/O系统机制读取。这便允许Nginx从存储驱动器上读取数据并且直接放入内存中，而不会经过复杂的中间缓冲处理。启用该指令后,sendfile指令将自动禁用，它们不能一起使用。

**open\_file\_cache：**该指令允许启用缓存，该缓存用于存储被打开文件的相关信息，通俗说，实际上不是存储文件的内容本身，而是与文件相关的信息，例如：文件描述符；文件和目录存在的信息；文件错误等。

该指令结束两个参数：

max=X，这里的X是能够缓存的最大条目。如果达到这个数量，老的条目将被删除，从而给新的条目留下存储空间。

inactive=Y，这里的Y是设定每一个被缓冲的条目可存储的时间长度。默认情况下，Nginx将等待60s，然后条目被清除。如果访问的是缓存条目，则重置其定时器，如果一个条目被访问的次数多余指令open\_file\_cache\_min\_user指定的值，缓存的条目则不会被清理。

**open\_file\_cache\_errors** ：开启或禁用缓存文件错误，由指令open\_file\_cache开启对文件的缓存

**open\_file\_cache\_min\_uses：**为了保护条目，该指令定义条目总访问次数，默认情况下，在open\_file\_cache中的条目会在闲置一段时间后被清除，但如果该条目是活动的，则可以阻止Nginx移除该缓存条目。

**open\_file\_cache\_valid：**启用文件缓存机制很重要，但是缓存的信息很快就变得过时，尤其对于移动迅速的文件系统，从这个角度讲，过了一段时间后，信息需要重新校验，该指令指定的缓存重新有效之前Nginx将等待的时间。

## 模块变量

HTTP核心模块引入了大量的变量

### 请求头

|  |  |
| --- | --- |
| 变量名称 | 描述 |
| $http\_host | HTTP头中Host的值，这个值是一个字符串，它指出客户端设法要到达主机的主机名 |
| $http\_user\_agent | HTTP中User-Agent的值，该字符串指出客户端使用的WEB浏览器 |
| $http\_referer | HTTP头中Referer的值，该字符串告诉服务器从那个页面链接而来 |
| $http\_via | HTTP头中Via的值，该值会告知我们客户端可能使用的代理 |
| $http\_x\_forward\_for | HTTP头中 X-Forward-For的值，如果客户端在代理之后，通过该变量能够取得真实的客户端IP地址 |
| $http\_cookie | HTTP头中cookie的值，该变量中包含客户端发送的cookie数据 |
| $http…. | 获取其他通过客户端发送的头信息可以使用$http\_再跟一个头名称，名称要小写，将头名称中的短横想（-）换成下划线（\_） |

### 响应头

|  |  |
| --- | --- |
| $sent\_http\_content\_type | HTTP头中Content-Type的值，指出被传递资源的MIME类型 |
| $sent\_http\_content\_length | HTTP头中Content-Length的值，它会告知响应体的长度 |
| $sent\_http\_location | HTTP头中Location的值，它指出想要访问的location资源与原始请求中指定location的不同 |
| $sent\_http\_last\_modified | HTTP头中Last\_Modified的值，相当于修改请求资源的日期 |
| $sent\_http\_connection | HTTP头中Connection的值，它定义了连接是否持续有效或者已关闭 |
| $sent\_http\_keep\_alive | HTTP头中keep-Alive的值，定义了提供持续连接的时间长度 |
| $sent\_http\_tranfer\_encoding | HTTP头中Transfer-Encoding的值，该变量指出有关响应体所使用编码方式的相关信息 |
| $sent\_http\_cache\_control | HTTP头中Cache-Control的值，该变量会告诉客户端浏览器是否对资源进行缓存 |
| $sent\_http\_... | 获取其他发送到客户端的头信息，可以使用$sent\_http\_再跟一个头名称，名称要小写，将头名称中的横线（-）换成下划线（\_） |

### Nginx产生的变量

|  |  |
| --- | --- |
| $arg\_XXX | 允许你访问查询字符串（GET参数），这里的XXX替换为具体的参数 |
| $agrs | 所有结合在一起的字符串查询参数 |
| $binary\_remote\_addr | 作为二进制数据的客户端IP地址（4个字节） |
| $binary\_bytes\_sent | 在响应body中发送的字节数 |
| $content\_length | 相当于HTTP头中的Content-Length |
| $content\_type | 相当于HTTP头中的Content-Type |
| $cookie\_XXX | 允许你访问cookie数据，这里的XXX替换具体的参数 |
| $document\_root | 对当前的请求返回root指令指定的值。 |
| $document\_uri | 返回当前的uri，如果内部重定向被执行，那么它将不同于原始的URI请求，它与变量$uri相同。 |
| $host | 该变量相当于HTTP请求头中的Host，Nginx自身会给这个变量赋值，这种情况用于原始请求头中没有提供Host |
| $hostname | 返回服务器的系统名称 |
| $is\_args | 如果定义$agrs变量，$is\_args |
| $limit\_rate | 返回每一个连接的速率，这个速率是通过limit\_rate指令来定义的，可以通过设置这个指令来编辑这个变量 |
| $nginx\_version | 返回正在运行的Nginx的版本号 |
| $pid | 返回Nginx的进程标识符 |
| $query\_string | 如同$args |
| $remote\_addr | 返回客户端的IP地址 |
| $remote\_port | 返回客户端的套接字的端口 |
| $remote\_user | 如果使用的认证，则返回客户名称 |
| $realpath\_root | 在客户端请求中返回文档的根目录，在处理符号链接时，将链接到实际的路径 |
| $request\_body | 返回客户端的请求体，或body为空 |
| $request\_body\_file | 如果请求体被保存，则该变量会指出临时文件的路径 |
| $request\_completion | 如果请求完成将返回OK，否则为空字符串 |
| $request\_filename | 返回在当前请求中提供的全文件名 |
| $request\_method | 指出在请求中使用的HTTP方法 |
| $request\_uri | 相当于原始的URI请求，在整个处理过程中保持不变 |
| $scheme | 返回http或https，取决于客户端的请求。 |
| $server\_addr | 返回服务器的IP地址。每一个变量都需要系统调用。注意：在高流量的设置中，可能会影响系统的总体性能。 |
| $server\_name | 表示指令server\_name的值，在处理请求时使用 |
| $server\_port | 表示服务器接收请求数据的套接字端口 |
| $server\_protocol | 返回协议和版本号，通常是HTTP/1.0或HTTP/1.1 |
| $uri | 等同于$document\_uri |

# 日志

## 日志格式

1. main格式

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

Mian格式使我们定义好的日志格式，并起了一个名字方便我们调用。

2.自定义格式

$remote\_user用于记录远程客户端用户名称；

$time\_local用于记录访问时间与时区；

$request用于记录请求URL与HTTP协议；

$status用于记录请求状态，例如成功时状态为200，页面找不到时状态为404；

$body\_bytes\_sent用于记录发送客户端的文件主体内容大小；

$http\_referer用于记录是从哪个页面链接访问过来的；

$http\_user\_agent用于记录客户浏览器的相关信息。

$http\_x\_forwarded\_for  当前端有代理服务器时，设置web节点记录客户端地址的配置，此参数生效的前提是代理服务器也要进行相关的x\_forwarded\_for设置

$remote\_addr             客户端地址

$remote\_user             客户端用户名称                                --

$time\_local              访问时间和时区

$request                 请求的URI和HTTP协议

$http\_host               请求地址，即浏览器中你输入的地址（IP或域名）

$status                  HTTP请求状态

$upstream\_status         upstream状态

$body\_bytes\_sent         发送给客户端文件内容大小

$http\_referer            url跳转来源

$http\_user\_agent         用户终端浏览器等信息

$ssl\_protocol            SSL协议版本

$ssl\_cipher              交换数据中的算法

$upstream\_addr           后台upstream的地址，即真正提供服务的主机地址

$request\_time            整个请求的总时间

$upstream\_response\_time  请求过程中，upstream响应时间

# Location

Location有”定位“的意思，在nginx配置中比不可少，location可以把网站的不同部分，定位到不同的处理方法上。

语法：location[=|~|~\*|^~] patt{},中括号可以不写任何参数，作为一般匹配。

**=：**URI的定位必须与指定的模式精确匹配，该模式在这里限定为一个简单的文本字符串，不能使用正则表达式。

server{

server\_name [www.haodashen.com](http://www.haodashen.com);

location = /abcd {……}

}

对location区段配置的访问：

可用<http://www.haodashen.com/abcd>

可用<http://www.haodashen.com/ABCD>（）

可用<http://www.haodashen.com/abcd&param1?param2>（不管查询参数）

不可用<http://www.haodashen.com/abcd/>（结尾斜杠）

不可用<http://www.haodashen.com/abcde>（在指定的模式下添加而额外的字符）

**（无）：**URI必须以指定的模式开始，不可以使用正则表达式。

server{

server\_name [www.haodashen.com](http://www.haodashen.com);

location /abcd {……}

}

对location区段配置的访问：

可用<http://www.haodashen.com/abcd>（严格匹配）

可用<http://www.haodashen.com/abcd&param1?param2>（不管查询参数）

可用<http://www.haodashen.com/abcd/>（结尾斜杠）

可用<http://www.haodashen.com/abcde>（在指定的模式下添加而额外的字符）

**~：**客户端请求的URI与指定的正则表达式匹配必须区分大小写。

server{

server\_name [www.haodashen.com](http://www.haodashen.com);

location ~ ^/abcd$ {……}

}

对location区段配置的访问：

可用<http://www.haodashen.com/abcd>（严格匹配）

不可用<http://www.haodashen.com/ABCD>（区分大小写）

可用<http://www.haodashen.com/abcd&param1?param2>（不管查询参数）

不可用<http://www.haodashen.com/abcd/>（结尾斜杠）

不可用<http://www.haodashen.com/abcde>（在指定的模式下添加而额外的字符）

**~\*：**对客户端请求的URI与指定的正则表达式匹配，必须不区分大小写

server{

server\_name [www.haodashen.com](http://www.haodashen.com);

location ~\* ^/abcd$ {……}

}

对location区段配置的访问：

可用<http://www.haodashen.com/abcd>（严格匹配）

可用<http://www.haodashen.com/ABCD>（不区分大小写）

可用<http://www.haodashen.com/abcd&param1?param2>（不管查询参数）

不可用<http://www.haodashen.com/abcd/>（结尾斜杠）

不可用<http://www.haodashen.com/abcde>（在指定的模式下添加而额外的字符）

**^~：**类似于无标志行为，URI的定位必须以指定模式开始，不同的是，如果模式匹配，那么nginx就会停止搜索其他模式。

大致可以分为三类：

1. location =patt{ }[精准匹配]
2. location patt {}[一般匹配]
3. location ~patt{}[正则匹配]

## location匹配顺序

1．"="前缀指令匹配，如果匹配成功，则停止其他匹配。

2．没有修饰符的location区段，如果指定字符串请求的URI精确匹配，则Nginx使用该location的设置。

3．带有^~修饰符的location区段，如果指定的字符串与请求的URL开始匹配，则Nginx使用该location的设置。

4．带有~或~\*修饰符的location区段，如果指定的字符串与请求的URL开始匹配，则Nginx使用该location的设置。

5．没有修饰符的location区段，如果指定字符串与URI请求开始匹配，则Nginx使用该location的设置。

、

------------------------------------------------------------------

2.普通字符串指令匹配，顺序是从长到短，匹配成功的location如果使用^~，则停止其他匹配（正则匹配）

3.正则表达式指令匹配，按照配置文件里的顺序，成功就停止其他匹配

4.如果第三步中有匹配成功，则使用该结果，否则使用第二步结果

--------------------------------------------------------------------

Ps: （1）匹配的顺序是先匹配普通字符串，然后再匹配正则表达式。另外普通字符串匹配顺序是根据配置中字符长度从长到短，也就是说使用普通字符串配置的location顺序是无关紧要的，反正最后nginx会根据配置的长短来进行匹配，但是需要注意的是正则表达式按照配置文件里的顺序测试。找到第一个比配的正则表达式将停止搜索。

1. 一般情况下，匹配成功了普通字符串location后还会进行正则表达location匹配。有两种方法改变这种行为，其一就是使用“=”前缀，这时执行的是严格匹配，并且匹配成功后立即停止其他匹配，同时处理这个请求；另外一种就是使用“^~”前缀，如果把这个前缀用于一个常规字符串那么告诉nginx 如果路径匹配那么不测试正则表达式。

# Nginx重写（rewrite）

Nginx中的rewrite的主要功能是实现URL地址的重定向，

重写中使用的指令：（1）if 空格(条件) {}：设置条件在进行重写。

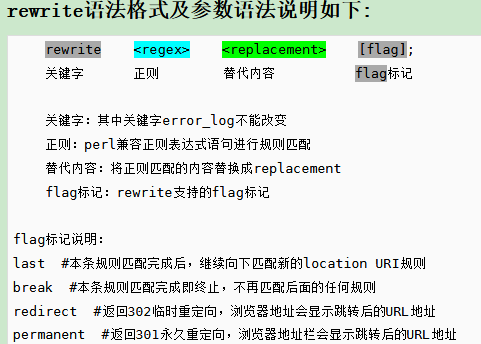
条件的三种写法：1.“=”来判断是否相等，区分大小写。

2.“~”使用正则匹配，区分大小写，“~\*”不区分大小写的正则。

3.-f -d -e判断是否为文件，为目录，是否存在。

（2）return:返回状态码

1. set:设置变量。
2. break:跳出rewrite
3. rewrite:重写。



# Nginx Module

## ngx\_http\_gzip\_module

gzip常用配置参数：

gzip on|off：是否开启gzip

gzip\_buffers 32 4K| 16 8K ：缓冲(压缩在内存中缓冲几块? 每块多大?)

gzip\_comp\_level [1-9] ：推荐6 压缩级别(级别越高,压的越小,越浪费CPU计算资源)

gzip\_disable ：正则匹配UA 什么样的Uri不进行gzip

gzip\_min\_length 200 ：开始压缩的最小长度(再小就不要压缩了,意义不在)

gzip\_http\_version 1.0|1.1 ：开始压缩的http协议版本(可以不设置,目前几乎全是1.1协议)

gzip\_proxied：设置请求者代理服务器,该如何缓存内容

gzip\_types text/plain application/xml ：对哪些类型的文件用压缩 如txt,xml,html ,css

gzip\_vary on|off ： 是否传输gzip压缩标志

## ngx\_http\_upstream\_module

### 调度算法

#### rr轮询（默认）

按客户端请求顺序把客户端的请求逐一的分配到不同的后端的服务器，如果后端服务器宕机，宕机服务器会自动剔除，是用户访问不受影响，将请求分配给正常的服务器。，

#### weight（权重）

在轮询算法的基础上加上权重（默认为rr+weight），权重轮询和访问成，权重越大，转发的请求越多，可以根据服务器的配置和性能分配权重大小，可以有效的解决新旧服务器性能不均进行请求分配问题。

#### ip\_hash

每一个请求按访问的ip的hash结果进行分配，当新的请求到达时，先将其客户端的ip通过哈希算法哈希出一个值，在随后请求客户端，ip的哈希值只要相同，就会分配到同一台机器，该调度算法可以有效的解决动态网页Session共享问题，但有时会导致请求分配不均，即无法保证1：1的负载均衡

#### fair（第三方，NO）动态算法

按照后端服务器RS的响应时间来分配请求，响应时间短有限分配，此种算法可以依据网页大小和加载时间长短智能地进行负载均衡，也就是根据后端服务器的响应时间来分配请求，响应时间短的优先分配，，Nginx本身不支持fair的，如果需要使用这种调度算法，必须瞎子啊Nginx的upstram\_fair模板

#### url\_hash（第三方，NO）

按访问url的hash结果来分配请求，让每个url定向同一个服务器，后端服务器为缓存服务器时效果明显，在upstream加入hash语句，server语句不能写入weight等其他参数，hash\_method是使用的hash算法。

url\_hash按照访问的url的hash结果分配请求，使每一个url定向到同一台后端服务器，可以进一步提高后端缓存服务器的效率命中率，Nginx本身不支持url\_hash,如果需要使用该调度算法，必须下载Nginx的hash软件包。

url\_hash(Web缓存节点)和ip\_hash(会话保持)类似。

#### least\_conn

最少连接数，哪一个机器的连接数少就分配。

#### 一致性hash

http://tengine.taobao.org/

## ngx\_http\_proxy\_module

#### http proxy模块参数

Nginx代理功能是使用http proxy模块实现的，默认在安装nginx时已经安装了http proxy模块，因此可以直接使用。

Proxy\_set\_header:设置后端的服务器获取用户的主机名或者真是ip值，以及代理 者的真是IP地址。

Client\_body\_buffer\_size:用于指定客户端请求主体缓冲区大小，可以理解为先保存到本地再传给用户。

Proxy\_connect\_timeout:表示与后端服务器连接超时的时间，即发起握手等待响应的超时时间。

Proxy\_send\_timeout:表示后端服务器的数据回传时间，即在规定的时间内，后端服务器必须传完所需的数据，否则，Nginx将断开这个连接。

Proxy\_read\_timeout:设置Nginx从代理后端服务器获取信息的时间，表示连接建立成功后，Nginx等待后端服务器响应的时间。