

Nginx内置变量列表

在OpenResty或者Nginx中使用lua编程时，直接在ngx.var后面接上变量名即可。例如：ngx.var.bytes_sent。

\$arg_name

请求中的参数名，即“?”后面的arg_name=arg_value形式的arg_name

\$args

请求中的参数值

\$binary_remote_addr

客户端地址的二进制形式，固定长度为4个字节

\$body_bytes_sent

传输给客户端的字节数，响应头不计算在内；这个变量和Apache的mod_log_config模块中的“%B”参数保持兼容

\$bytes_sent

传输给客户端的字节数（1.3.8，1.2.5）

\$connection

TCP连接的序列号（1.3.8，1.2.5）

\$connection_requests

TCP连接当前的请求数量（1.3.8，1.2.5）

\$content_length

“Content-Length” 请求头字段

\$content_type

“Content-Type” 请求头字段

\$cookie_name

cookie名称

\$document_root

当前请求的文档根目录或别名

\$document_uri

同 \$uri

\$host

优先级如下：HTTP请求行的主机名>“HOST”请求头字段>符合请求的服务器名

\$hostname

主机名

\$http_name

匹配任意请求头字段； 变量名中的后半部分“name”可以替换成任意请求头字段，如在配置文件中需要获取http请求头：“Accept-Language”，那么将“-”替换为下划线，大写字母替换为小写，形如：\$http_accept_language即可。

\$https

如果开启了SSL安全模式，值为“on”，否则为空字符串。

\$is_args

如果请求中有参数，值为“?”，否则为空字符串。

\$limit_rate

用于设置响应的速度限制，详见 `limit_rate`。

\$msec

当前的Unix时间戳 (1.3.9, 1.2.6)

\$nginx_version

nginx版本

\$pid

工作进程的PID

\$pipe

如果请求来自管道通信，值为“p”，否则为“.” (1.3.12, 1.2.7)

\$proxy_protocol_addr

获取代理访问服务器的客户端地址，如果是直接访问，该值为空字符串。(1.5.12)

\$query_string

同 `$args`

\$realpath_root

当前请求的文档根目录或别名的真实路径，会将所有符号连接转换为真实路径。

\$remote_addr

客户端地址

\$remote_port

客户端端口

\$remote_user

用于HTTP基础认证服务的用户名

\$request

代表客户端的请求地址

\$request_body

客户端的请求主体

此变量可在location中使用，将请求主体通过`proxy_pass`，`fastcgi_pass`，`uwsgi_pass`，和 `scgi_pass`传递给下一级的代理服务器。

\$request_body_file

将客户端请求主体保存在临时文件中。文件处理结束后，此文件需删除。如果需要之一开启此功能，需要设置 `client_body_in_file_only`。如果将文件传递给后端的代理服务器，需要禁用request body，即设置 `proxy_pass_request_body off`，`fastcgi_pass_request_body off`，`uwsgi_pass_request_body off`，或 `scgi_pass_request_body off`。

\$request_completion

如果请求成功，值为“OK”，如果请求未完成或者请求不是一个范围请求的最后一部分，则为空。

\$request_filename

当前连接请求的文件路径，由`root`或`alias`指令与URI请求生成。

`$request_length`

请求的长度（包括请求的地址，http请求头和请求主体）（1.3.12, 1.2.7）

`$request_method`

HTTP请求方法，通常为“GET”或“POST”

`$request_time`

处理客户端请求使用的时间（1.3.9, 1.2.6）；从读取客户端的第一个字节开始计时。

`$request_uri`

这个变量等于包含一些客户端请求参数的原始URI，它无法修改，请查看`$uri`更改或重写URI，不包含主机名，例如：“/cnphp/test.php?arg=freemouse”。

`$scheme`

请求使用的Web协议，“http”或“https”

`$sent_http_name`

可以设置任意http响应头字段；变量名中的后半部分“name”可以替换成任意响应头字段，如需要设置响应头Content-length，那么将“-”替换为下划线，大写字母替换为小写，形如：`$sent_http_content_length 4096`即可。

`$server_addr`

服务器端地址，需要注意的是：为了避免访问linux系统内核，应将ip地址提前设置在配置文件中。

`$server_name`

服务器名，www.cnphp.info

`$server_port`

服务器端口

`$server_protocol`

服务器的HTTP版本，通常为“HTTP/1.0”或“HTTP/1.1”

`$status`

HTTP响应代码（1.3.2, 1.2.2）

`$tcpinfo_rtt`, `$tcpinfo_rttvar`, `$tcpinfo_snd_cwnd`, `$tcpinfo_rcv_space`

客户端TCP连接的具体信息

`$time_iso8601`

服务器时间的ISO 8610格式（1.3.12, 1.2.7）

`$time_local`

服务器时间（LOG Format 格式）（1.3.12, 1.2.7）

`$uri`

请求中的当前URI(不带请求参数，参数位于`$args`)，可以不同于浏览器传递的`$request_uri`的值，它可以通过内部重定向，或者使用index指令进行修改，`$uri`不包含主机名，如“/foo/bar.html”。

指令名称	说明
lua_use_default_type	是否使用default_type指令定义的Content-Type默认值
lua_code_cache	*_by_lua_file文件是否cache
lua_regex_cache_max_entries	
lua_regex_match_limit	
lua_package_path	用Lua写的lua外部库路径（.lua文件）
lua_package_cpath	用C写的lua外部库路径（.so文件）
init_by_lua	master进程启动时挂载的lua代码
init_by_lua_file	
init_worker_by_lua	worker进程启动时挂载的lua代码，常用来执行一些定时器任务
init_worker_by_lua_file	
set_by_lua	设置变量
set_by_lua_file	
content_by_lua	handler模块
content_by_lua_file	
rewrite_by_lua	
rewrite_by_lua_file	
access_by_lua	
access_by_lua_file	
header_filter_by_lua	header filter模块
header_filter_by_lua_file	
body_filter_by_lua	body filter模块，ngx.arg[1]代表输入的chunk，ngx.arg[2]代表当前chunk是否为last
body_filter_by_lua_file	
log_by_lua	
log_by_lua_file	
lua_need_request_body	是否读请求体，跟ngx.req.read_body()函数作用类似

指令名称	说明
lua_shared_dict	创建全局共享的table（多个worker进程共享）
lua_socket_connect_timeout	TCP/unix 域socket对象connect方法的超时时间
lua_socket_send_timeout	TCP/unix 域socket对象send方法的超时时间
lua_socket_send_lowat	设置cosocket send buffer的低water值
lua_socket_read_timeout	TCP/unix 域socket对象receive方法的超时时间
lua_socket_buffer_size	cosocket读buffer大小
lua_socket_pool_size	cosocket连接池大小
lua_socket_keepalive_timeout	cosocket长连接超时时间
lua_socket_log_errors	是否打开cosocket错误日志
lua_ssl_ciphers	
lua_ssl_crl	
lua_ssl_protocols	
lua_ssl_trusted_certificate	
lua_ssl_verify_depth	
lua_http10_buffering	
rewrite_by_lua_no_postpone	
lua_transform_underscores_in_response_headers	
lua_check_client_abort	是否监视client提前关闭请求的事件，如果打开监视，会调用ngx.on_abort()注册的回调
lua_max_pending_timers	
lua_max_running_timers	

table	说明
ngx.arg	指令参数，如跟在content_by_lua_file后面的参数
ngx.var	变量，ngx.var.VARIABLE引用某个变量
ngx.ctx	请求的lua上下文
ngx.header	响应头，ngx.header.HEADER引用某个头
ngx.status	响应码
API	说明
ngx.log	输出到error.log
print	等价于 ngx.log(ngx.NOTICE, ...)
ngx.send_headers	发送响应头
ngx.headers_sent	响应头是否已发送
ngx.resp.get_headers	获取响应头
ngx.timer.at	注册定时器事件
ngx.is_subrequest	当前请求是否是子请求
ngx.location.capture	发布一个子请求
ngx.location.capture_multi	发布多个子请求
ngx.exec	
ngx.redirect	
ngx.print	输出响应
ngx.say	输出响应，自动添加'\n'
ngx.flush	刷新响应
ngx.exit	结束请求
ngx.eof	
ngx.sleep	无阻塞的休眠（使用定时器实现）
ngx.get_phase	
ngx.on_abort	注册client断开请求时的回调函数
ndk.set_var.DIRECTIVE	
ngx.req.start_time	请求的开始时间

table	说明
ngx.req.http_version	请求的HTTP版本号
ngx.req.raw_header	请求头（包括请求行）
ngx.req.get_method	请求方法
ngx.req.set_method	请求方法重载
ngx.req.set_uri	请求URL重写
ngx.req.set_uri_args	
ngx.req.get_uri_args	获取请求参数
ngx.req.get_post_args	获取请求表单
ngx.req.get_headers	获取请求头
ngx.req.set_header	
ngx.req.clear_header	
ngx.req.read_body	读取请求体
ngx.req.discard_body	扔掉请求体
ngx.req.get_body_data	
ngx.req.get_body_file	
ngx.req.set_body_data	
ngx.req.set_body_file	
ngx.req.init_body	
ngx.req.append_body	
ngx.req.finish_body	
ngx.req.socket	
ngx.escape_uri	字符串的url编码
ngx.unescape_uri	字符串url解码
ngx.encode_args	将table编码为一个参数字符串
ngx.decode_args	将参数字符串编码为一个table
ngx.encode_base64	字符串的base64编码
ngx.decode_base64	字符串的base64解码
ngx.crc32_short	字符串的crs32_short哈希

table	说明
ngx.crc32_long	字符串的crc32_long哈希
ngx.hmac_sha1	字符串的hmac_sha1哈希
ngx.md5	返回16进制MD5
ngx.md5_bin	返回2进制MD5
ngx.sha1_bin	返回2进制sha1哈希值
ngx.quote_sql_str	SQL语句转义
ngx.today	返回当前日期
ngx.time	返回UNIX时间戳
ngx.now	返回当前时间
ngx.update_time	刷新时间后再返回
ngx.localtime	
ngx.utctime	
ngx.cookie_time	返回的时间可用于cookie值
ngx.http_time	返回的时间可用于HTTP头
ngx.parse_http_time	解析HTTP头的时间
ngx.re.match	
ngx.re.find	
ngx.re.gmatch	
ngx.re.sub	
ngx.re.gsub	
ngx.shared.DICT	
ngx.shared.DICT.get	
ngx.shared.DICT.get_stale	
ngx.shared.DICT.set	
ngx.shared.DICT.safe_set	
ngx.shared.DICT.add	
ngx.shared.DICT.safe_add	
ngx.shared.DICT.replace	

table	说明
ngx.shared.DICT.delete	
ngx.shared.DICT.incr	
ngx.shared.DICT.flush_all	
ngx.shared.DICT.flush_expired	
ngx.shared.DICT.get_keys	
ngx.socket.udp	
udpsock:setpeername	
udpsock:send	
udpsock:receive	
udpsock:close	
udpsock:settimeout	
ngx.socket.tcp	
tcpsock:connect	
tcpsock:sslhandshake	
tcpsock:send	
tcpsock:receive	
tcpsock:receiveuntil	
tcpsock:close	
tcpsock:settimeout	
tcpsock:setoption	
tcpsock:setkeepalive	
tcpsock:getreusedtimes	
ngx.socket.connect	
ngx.thread.spawn	
ngx.thread.wait	
ngx.thread.kill	
coroutine.create	
coroutine.resume	

table	说明
coroutine.yield	
coroutine.wrap	
coroutine.running	
coroutine.status	
ngx.config.debug	编译时是否有 --with-debug选项
ngx.config.prefix	编译时的 --prefix选项
ngx.config.nginx_version	返回nginx版本号
ngx.config.nginx_configure	返回编译时 ./configure的命令行选项
ngx.config.ngx_lua_version	返回ngx_lua模块版本号
ngx.worker.exiting	当前worker进程是否正在关闭（如reload、shutdown期间）
ngx.worker.pid	返回当前worker进程的pid
常量	说明
Core constants	ngx.OK (0) ngx.ERROR (-1) ngx.AGAIN (-2) ngx.DONE (-4) ngx.DECLINED (-5) ngx.nil
HTTP method constants	ngx.HTTP_GET ngx.HTTP_HEAD ngx.HTTP_PUT ngx.HTTP_POST ngx.HTTP_DELETE ngx.HTTP_OPTIONS ngx.HTTP_MKCOL ngx.HTTP_COPY ngx.HTTP_MOVE ngx.HTTP_PROPFIND ngx.HTTP_PROPPATCH ngx.HTTP_LOCK ngx.HTTP_UNLOCK ngx.HTTP_PATCH ngx.HTTP_TRACE
HTTP status constants	ngx.HTTP_OK (200) ngx.HTTP_CREATED (201) ngx.HTTP_SPECIAL_RESPONSE (300) ngx.HTTP_MOVED_PERMANENTLY (301) ngx.HTTP_MOVED_TEMPORARILY (302) ngx.HTTP_SEE_OTHER (303) ngx.HTTP_NOT_MODIFIED (304) ngx.HTTP_BAD_REQUEST (400) ngx.HTTP_UNAUTHORIZED (401) ngx.HTTP_FORBIDDEN (403) ngx.HTTP_NOT_FOUND (404) ngx.HTTP_NOT_ALLOWED (405) ngx.HTTP_GONE (410) ngx.HTTP_INTERNAL_SERVER_ERROR (500) ngx.HTTP_METHOD_NOT_IMPLEMENTED (501) ngx.HTTP_SERVICE_UNAVAILABLE (503) ngx.HTTP_GATEWAY_TIMEOUT (504)
Nginx log level constants	ngx.STDERR ngx.EMERG ngx.ALERT ngx.CRIT ngx.ERR ngx.WARN ngx.NOTICE ngx.INFO ngx.DEBUG