

第 2 章

Nginx 服务器的安装与配置

2.1 安装 Nginx 服务器所需要的系统资源

Nginx 是开源软件，您可以从其官方网站 (<http://www.nginx.net/>) 下载最新版本。Nginx 目前有 3 个版本：旧的稳定版 (0.6.x)、最新的稳定版 (0.7.x) 和开发版 (0.8.x)。0.8.x 开发版分支刚发布不久，Bug 会比较多，因此不建议用于生产环境。开发版一旦更新稳定下来，就会被加入稳定版分支。然而，新功能不一定会被加到旧的稳定版中去，所以，目前最合适使用的版本是 0.7.x。

从 0.7.52 版本开始，Nginx 官方网站开始提供 Windows 版本下载，Windows 版本的 Nginx 使用比较简单，只须下载完成后，将其解压缩到一个不包含空格的路径中，执行 `nginx.exe` 即可。但是，Windows 版本的 Nginx 性能要比 Linux/Unix 版本的 Nginx 差很多。本书重点介绍 Linux 环境下的 Nginx 编译安装。

一些 Linux 发行版和 BSD 的各个变种版本的安装包仓库中包含了编译后的二进制 Nginx 软件，很多预先编译好的安装包都比较陈旧，所以大多数情况下还是推荐直接从源码编译安装。

安装 Nginx 服务器之前，首先要安装一个 Linux/Unix 操作系统发行版，例如 Redhat、CentOS、Debian、Fedora Core、Gentoo、SUSE、Ubuntu、FreeBSD 等。

本书将以新浪、搜狐、网易、金山游戏官网等国内互联网公司最常用的 Linux 发行版——CentOS 为例，介绍 Nginx 的安装与使用。CentOS 是基于 RedHat Enterprise Linux 源代码重新编译、去除 RedHat 商标的产物，各种操作、使用和 RedHat 没有区别。CentOS 完全免费，修正了 RedHat 的很多 BUG，但 CentOS 不向用户提供技术支持，也不负任何商业责任。

编译 Nginx 的要求如下：

磁盘空间：需要保证有 10MB 以上的剩余磁盘空间。Nginx 安装完毕后会占据 4MB 左右的磁盘空间，实际的磁盘空间需求会因编译设置和是否安装第三方模块而有所不同。

GCC 编译器及相关工具：GCC 全称为 GNU Compiler Collection，是 GNU 社区推出的功能强大、性能优越的用于编程开发的自由编译器，是 GNU 的代表作品之一，目前可以编译的语言包括：C、C++、Objective-C、Fortran、Java 等。您必须确保您的操作系统安装有 GCC 编译器。另外，您还必须安装 Autoconf 和 Automake 工具，它们用于自动创建功能完善的 Makefile，当前大多数软件包都是用这一工具生成 Makefile 的，Nginx 也不例外。在 CentOS 系统下，您可以使用 yum 命令安装 GCC 编译器及相关工具：

```
yum -y install gcc gcc-c++ autoconf automake
```

模块依赖性：Nginx 的一些模块需要其他第三方库的支持，例如 gzip 模块需要 zlib 库，rewrite 模块需要 pcre 库，ssl 功能需要 openssl 库等。同样，如果是在 CentOS 系统下，我们可以使用 yum 命令安装或下载源码包编译安装这些模块依赖的库：

```
yum -y install zlib zlib-devel openssl openssl-devel pcre pcre-devel
```

2.2 Nginx 的下载

您可以访问 <http://www.nginx.net/> 网站，该网站中的以下三行内容分别提供了 Nginx 0.6.x、0.7.x、0.8.x 三个版本分支的源码包或 Windows 二进制文件下载。

The development stable versions are nginx-0.8.x, nginx/Windows-0.8.x, the change log

The latest stable versions are nginx-0.7.x, nginx/Windows-0.7.x, the change log

The latest legacy stable version is nginx-0.6.x, the change log

如果您希望在 Windows 上运行 Nginx，您可以点击链接“nginx/Windows-0.8.x”或“nginx/Windows-0.7.x”下载 Windows 下的二进制版 ZIP 压缩包。如果您希望在 Linux/Unix 环境下运行 Nginx，您可以点击链接“nginx-0.8.x”、“nginx-0.7.x”、“nginx-0.6.x”下载以 tar.gz 格式压缩源码包。

2.3 Nginx 的安装

Nginx 从 0.7.52 版本开始有了官方的 Windows 版本，所以这里分别介绍 Nginx 在 Windows 下和 Linux 下的安装步骤。

2.3.1 Nginx 在 Windows 环境下的安装

Nginx 在 Windows 下的安装比较简单，将下载下来的 `nginx-0.x.xx.zip` 文件解压缩到一个不包含空格的路径中，例如 `d:\nginx`，然后在“开始”→“运行”→“cmd”中执行以下 DOS 命令即可启动 Nginx：

```
d:
cd d:\nginx
start nginx
```

如果要对启动的 Nginx 进程进行控制，可以使用 DOS 命令：

```
nginx -s [ stop | quit | reopen | reload ]
```

2.3.2 Nginx 在 Linux 环境下的安装

Nginx 在 Linux 环境下可以通过编译源码的方式来安装，最简单的安装命令如下：

```
tar zxvf nginx-0.x.xx.tar.gz
cd nginx-0.x.xx
./configure
make
sudo make install
```

按照以上命令，Nginx 将被默认安装到 `/usr/local/nginx` 目录下。您可以通过 `./configure --help` 命令查看 Nginx 可选择的编译选项。

Nginx 的 `configure` 脚本支持以下选项：

`--prefix=<path>`——Nginx 安装路径。如果没有指定，默认为 `/usr/local/nginx`。

`--sbin-path=<path>`——Nginx 可执行文件安装路径。只能安装时指定，如果没有指定，默认为 `<prefix>/sbin/nginx`。

`--conf-path=<path>`——在没有给定 `-c` 选项下默认的 `nginx.conf` 的路径。如果没有指定，默认为 `<prefix>/conf/nginx.conf`。

`--pid-path=<path>`——在 `nginx.conf` 中没有指定 `pid` 指令的情况下，默认的 `Nginx.pid` 的路径。如果没有指定，默认为 `<prefix>/logs/nginx.pid`。

`--lock-path=<path>`——`nginx.lock` 文件的路径。

`--error-log-path=<path>`——在 `nginx.conf` 中没有指定 `error_log` 指令的情况下，默认的错误日志的路径。如果没有指定，默认为 `<prefix>/logs/error.log`。

`--http-log-path=<path>`——在 `nginx.conf` 中没有指定 `access_log` 指令的情况下，默认访问日

志的路径。如果没有指定，默认为 `<prefix>/logs/access.log`。

`--user=<user>`——在 `nginx.conf` 中没有指定 `user` 指令的情况下，默认的 Nginx 使用的用户。如果没有指定，默认为 `nobody`。

`--group=<group>`——在 `nginx.conf` 中没有指定 `user` 指令的情况下，默认的 Nginx 使用的组。如果没有指定，默认为 `nobody`。

`--builddir=DIR`——指定编译的目录。

`--with-rtsig_module`——启用 `rtsig` 模块。

`--with-select_module` (`--without-select_module`)——允许或不允许开启 `SELECT` 模式，如果 `configure` 没有找到更合适的模式，比如：`kqueue` (`sun os`)、`epoll` (`linux kernel 2.6+`)、`rtsig` (实时信号) 或 `/dev/poll` (一种类似 `select` 的模式，底层实现与 `SELECT` 基本相同，都是采用轮训方法)，`SELECT` 模式将是默认安装模式。

`--with-poll_module` (`--without-poll_module`)——允许或不允许开启 `POLL` 模式，如果没有更合适的模式，比如：`kqueue` (`sun os`)、`epoll` (`linux kernel 2.6+`)，则可以开启。

`--with-http_ssl_module`——开启 `HTTP SSL` 模块，使 Nginx 可以支持 `HTTPS` 请求。这个模块需要已经安装 `OPENSSL`，在 `DEBIAN` 上是 `libssl`。

`--with-http_realip_module`——启用 `ngx_http_realip_module`。

`--with-http_addition_module`——启用 `ngx_http_addition_module`。

`--with-http_sub_module`——启用 `ngx_http_sub_module`。

`--with-http_dav_module`——启用 `ngx_http_dav_module`。

`--with-http_flv_module`——启用 `ngx_http_flv_module`。

`--with-http_stub_status_module`——启用“server status”页。

`--without-http_charset_module`——禁用 `ngx_http_charset_module`。

`--without-http_gzip_module`——禁用 `ngx_http_gzip_module`。如果启用，需要 `zlib`。

`--without-http_ssi_module`——禁用 `ngx_http_ssi_module`。

`--without-http_userid_module`——禁用 `ngx_http_userid_module`。

`--without-http_access_module`——禁用 `ngx_http_access_module`。

`--without-http_auth_basic_module`——禁用 `ngx_http_auth_basic_module`。

`--without-http_autoindex_module`——禁用 `ngx_http_autoindex_module`。

`--without-http_geo_module`——禁用 `ngx_http_geo_module`。

--without-http_map_module——禁用 ngx_http_map_module。

--without-http_referer_module——禁用 ngx_http_referer_module。

--without-http_rewrite_module——禁用 ngx_http_rewrite_module。如果启用，需要 PCRE。

--without-http_proxy_module——禁用 ngx_http_proxy_module。

--without-http_fastcgi_module——禁用 ngx_http_fastcgi_module。

--without-http_memcached_module——禁用 ngx_http_memcached_module。

--without-http_limit_zone_module——禁用 ngx_http_limit_zone_module。

--without-http_empty_gif_module——禁用 ngx_http_empty_gif_module。

--without-http_browser_module——禁用 ngx_http_browser_module。

--without-http_upstream_ip_hash_module——禁用 ngx_http_upstream_ip_hash_module。

--with-http_perl_module——启用 ngx_http_perl_module。

--with-perl_modules_path=PATH——指定 perl 模块的路径。

--with-perl=PATH——指定 perl 执行文件的路径。

--http-log-path=PATH——指定 http 默认访问日志的路径。

--http-client-body-temp-path=PATH——指定 http 客户端请求缓存文件存放目录的路径。

--http-proxy-temp-path=PATH——指定 http 反向代理缓存文件存放目录的路径。

--http-fastcgi-temp-path=PATH——指定 http FastCGI 缓存文件存放目录的路径。

--without-http——禁用 HTTP server。

--with-mail——启用 IMAP4/POP3/SMTP 代理模块。

--with-mail_ssl_module——启用 ngx_mail_ssl_module。

--with-cc=PATH——指定 C 编译器的路径。

--with-cpp=PATH——指定 C 预处理器的路径。

--with-cpu-opt=CPU——为特定的 CPU 编译，有效的值包括：pentium、pentiumpro、pentium3、pentium4、athlon、opteron、amd64、sparc32、sparc64、ppc64。

--without-pcre——禁止 PCRE 库的使用。同时也会禁止 HTTP rewrite 模块。在“location”配置指令中的正则表达式也需要 PCRE。

--with-pcre=DIR——指定 PCRE 库的源代码的路径。

--with-pcre-opt=OPTIONS——设置 PCRE 的额外编译选项。

--with-md5=DIR——设置 MD5 库的源代码路径。

--with-md5-opt=OPTIONS——设置 MD5 库的额外编译选项。

--with-md5-asm——使用 MD5 汇编源码。

--with-sha1=DIR——设置 sha1 库的源代码路径。

--with-sha1-opt=OPTIONS——设置 sha1 库的额外编译选项。

--with-sha1-asm——使用 sha1 汇编源码。

--with-zlib=DIR——设置 zlib 库的源代码路径。

--with-zlib-opt=OPTIONS——设置 zlib 库的额外编译选项。

--with-zlib-asm=CPU——zlib 针对 CPU 的优化，合法的值是：pentium、pentiumpro。

--with-openssl=DIR——设置 OpenSSL 库的源代码路径。

--with-openssl-opt=OPTIONS——设置 OpenSSL 库的额外编译选项。

--with-debug——启用调试日志。

--add-module=PATH——添加一个在指定路径中能够找到的第三方模块。

在不同版本间，选项可能会有些许变化，请总是使用 `./configure --help` 命令来检查当前的选项列表。

一个自定义编译选项的示例如代码 2-1 所示：

代码 2-1

```
./configure
--prefix=/usr \
--sbin-path=/usr/sbin/nginx \
--conf-path=/etc/nginx/nginx.conf \
--error-log-path=/var/log/nginx/error.log \
--pid-path=/var/run/nginx/nginx.pid \
--lock-path=/var/lock/nginx.lock \
--user=nginx \
--group=nginx \
--with-http_ssl_module \
--with-http_flv_module \
--with-http_gzip_static_module \
--http-log-path=/var/log/nginx/access.log \
--http-client-body-temp-path=/var/tmp/nginx/client/ \
--http-proxy-temp-path=/var/tmp/nginx/proxy/ \
--http-fastcgi-temp-path=/var/tmp/nginx/fcgi/
```

2.4 Nginx 的启动、停止、平滑重启

在 Linux 下, Nginx 服务主要的操作就是启动、停止和平滑重启。

2.4.1 Nginx 的启动

启动 Nginx, 可以执行以下命令。假设 Nginx 安装在/usr/local/nginx/目录中, 那么启动 Nginx 的命令就是:

```
/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf
```

参数“-c”指定了配置文件的路径, 如果不加“-c”参数, Nginx 会默认加载其安装目录的 conf 子目录中的 nginx.conf 文件, 在本例中即: /usr/local/nginx/sbin/nginx/conf/nginx.conf。

2.4.2 Nginx 的停止

Nginx 的停止方法有很多种, 一般通过发送系统信号给 Nginx 主进程的方式来停止 Nginx。

我们可以通过 ps 命令来查找 Nginx 的主进程号:

```
ps -ef | grep nginx
```

这时候屏幕会显示如图 2-1 所示的信息。

root	28881	1	0	01:32	?	00:00:00	nginx: master process /usr/local/webserver/nginx/sbin/nginx
www	28882	28881	0	01:32	?	00:00:00	nginx: worker process
www	28883	28881	0	01:32	?	00:00:00	nginx: worker process
www	28884	28881	0	01:32	?	00:00:00	nginx: worker process
www	28885	28881	0	01:32	?	00:00:00	nginx: worker process
root	28892	28848	0	01:34	pts/0	00:00:00	grep nginx

图 2-1 查看 Nginx 进程 ID

从图 2-1 中可以看到, 1 个 Nginx 进程的备注信息为“master process”, 表示它为主进程, 另外的 4 个进程备注信息为“worker process”, 表示它们为子进程。28881 为主进程号。

如果在 nginx.conf 配置文件中指定了 pid 文件存放的路径(例如: pid /usr/local/webserver/nginx/logs/nginx.pid;), 该文件中存放的就是 Nginx 当前的主进程号。如果没有指定 pid 文件存放的路径, nginx.pid 文件默认存放在 Nginx 安装目录的 logs 目录下。所以, 我们可以直接通过以下命令来完成平滑重启, 省下寻找 Nginx 主进程号的步骤:

```
kill - 信号类型 `usr/local/webserver/nginx/logs/nginx.pid`
```

(1) 从容停止 Nginx。

```
kill - QUIT Nginx 主进程号
```


或

```
kill - QUIT `/usr/local/webserver/nginx/logs/nginx.pid`
```

(2) 快速停止 Nginx。

```
kill - TERM Nginx 主进程号
```

```
kill - TERM `/usr/local/webserver/nginx/logs/nginx.pid`
```

或

```
kill - INT Nginx 主进程号
```

```
kill - INT `/usr/local/webserver/nginx/logs/nginx.pid`
```

(3) 强制停止所有 Nginx 进程。

```
pkill -9 nginx
```

2.5 Nginx 的平滑重启

如果改变了 Nginx 的配置文件 (nginx.conf)，想重启 Nginx，同样可以通过发送系统信号给 Nginx 主进程的方式来进行。不过，重启之前，要确认 Nginx 配置文件 (nginx.conf) 的语法是正确的，否则 Nginx 将不会加载新的配置文件。通过以下命令可以判断 Nginx 配置文件是否正确：

```
/usr/local/webserver/nginx/sbin/nginx -t -c  
/usr/local/webserver/nginx/conf/nginx.conf
```

如果配置文件不正确，屏幕将会提示配置文件的第几行出错：

```
[emerg]: unknown directive "abc" in /usr/local/webserver/nginx/conf/nginx.conf:55
```

```
configuration file /usr/local/webserver/nginx/conf/nginx.conf test failed
```

如果配置文件正确，屏幕将提示以下两行信息：

```
the configuration file /usr/local/webserver/nginx/conf/nginx.conf syntax is ok
```

```
configuration file /usr/local/webserver/nginx/conf/nginx.conf test is successful
```

这时候，就可以平滑重启 Nginx 了。

```
kill -HUP Nginx 主进程号
```

```
kill -HUP `/usr/local/webserver/nginx/logs/nginx.pid`
```

当 Nginx 接收到 HUP 信号时，它会尝试先解析配置文件（如果指定配置文件，就使用指定的，否则使用默认的），如果成功，就应用新的配置文件（例如，重新打开日志文件或监听的套

接字)。之后, Nginx 运行新的工作进程并从容关闭旧的工作进程。通知工作进程关闭监听套接字, 但是继续为当前连接的客户提供服务。所有客户端的服务完成后, 旧的工作进程被关闭。如果新的配置文件应用失败, Nginx 将继续使用旧的配置进行工作。

2.6 Nginx 的信号控制

在上一节中, 我们使用了信号来控制 Nginx 停止、平滑重启, Nginx 支持以下几种信号:

- TERM, INT 快速关闭;
- QUIT 从容关闭;
- HUP 平滑重启, 重新加载配置文件;
- USR1 重新打开日志文件, 在切割日志时用途较大;
- USR2 平滑升级可执行程序;
- WINCH 从容关闭工作进程。

2.7 Nginx 的平滑升级

当需要将正在运行中的 Nginx 升级、添加/删除服务器模块时, 可以在不中断服务的情况下, 使用新版本、重编译的 Nginx 可执行程序替换旧版本的可执行程序。步骤如下:

(1) 使用新的可执行程序替换旧的可执行程序, 对于编译安装的 Nginx, 可以将新版本编译安装到旧版本的 Nginx 安装路径中。替换之前, 您最好备份一下旧的可执行文件。

(2) 发送以下指令:

```
kill -USR2 旧版本的 Nginx 主进程号
```

(3) 旧版本 Nginx 的主进程将重命名它的 .pid 文件为 .oldbin (例如: /usr/local/webserver/nginx/logs/nginx.pid.oldbin), 然后执行新版本的 Nginx 可执行程序, 依次启动新的主进程和新的工作进程。

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

(4) 此时, 新、旧版本的 Nginx 实例会同时运行, 共同处理输入的请求。要逐步停止旧版本的 Nginx 实例, 你必须发送 WINCH 信号给旧的主进程, 然后, 它的工作进程就将开始从容关闭:

kill -WINCH 旧版本的 Nginx 主进程号

(5) 一段时间后, 旧的工作进程 (worker process) 处理了所有已连接请求后退出, 仅由新的工作进程来处理输入的请求了:

PID	PPID	USER	%CPU	VSZ	WCHAN	COMMAND
33126	1	root	0.0	1164	pause	nginx: master process /usr/local/nginx/sbin/nginx
36264	33126	root	0.0	1148	pause	nginx: master process /usr/local/nginx/sbin/nginx
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)

(6) 这时候, 我们可以决定是使用新版本, 还是恢复到旧版本:

kill -HUP 旧的主进程号: Nginx 将在不重载配置文件的情况下启动它的工作进程;

kill -QUIT 新的主进程号: 从容关闭其工作进程 (worker process);

kill -TERM 新的主进程号: 强制退出;

kill 新的主进程号或旧的主进程号: 如果因为某些原因新的工作进程不能退出, 则向其发送 kill 信号。

新的主进程退出后, 旧的主进程会移除 .oldbin 前缀, 恢复为它的 .pid 文件, 这样, 一切都恢复到升级之前了。如果尝试升级成功, 而你也希望保留新的服务器时, 可发送 QUIT 信号给旧的主进程, 使其退出而只留下新的服务器运行:

PID	PPID	USER	%CPU	VSZ	WCHAN	COMMAND
36264	1	root	0.0	1148	pause	nginx: master process /usr/local/nginx/sbin/nginx
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)