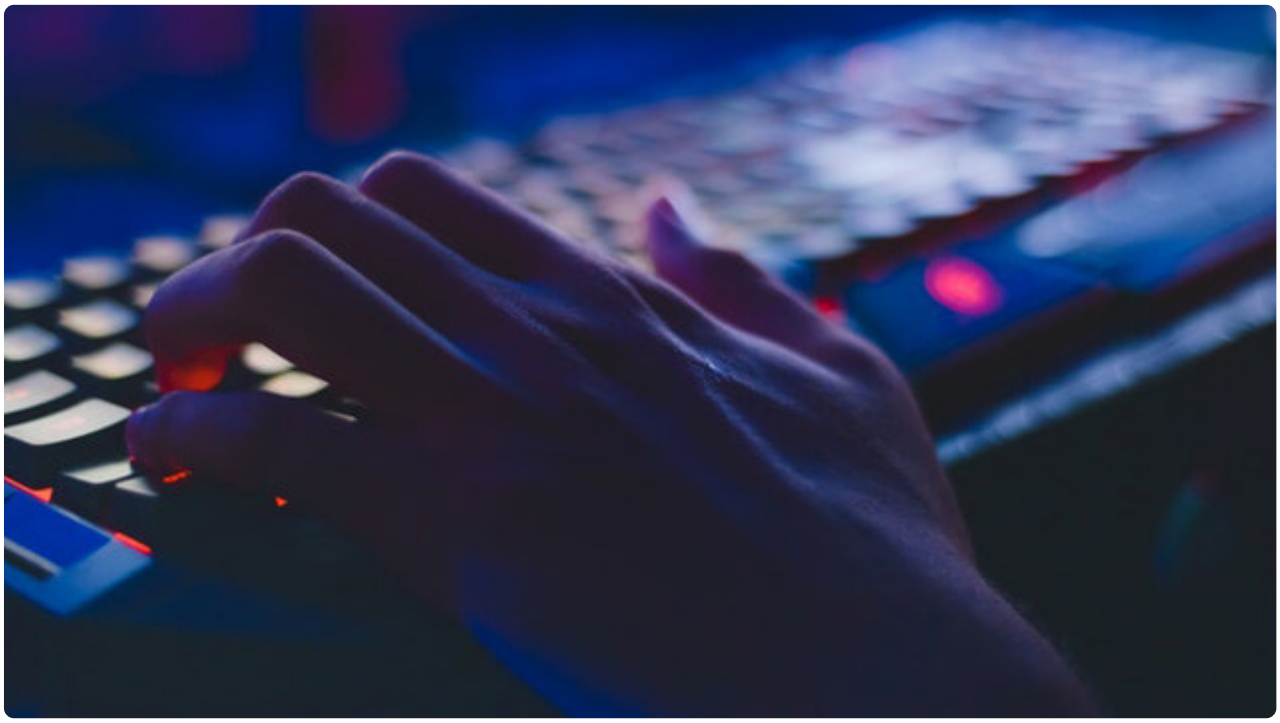
04 与新朋友 Nginx 友好交流

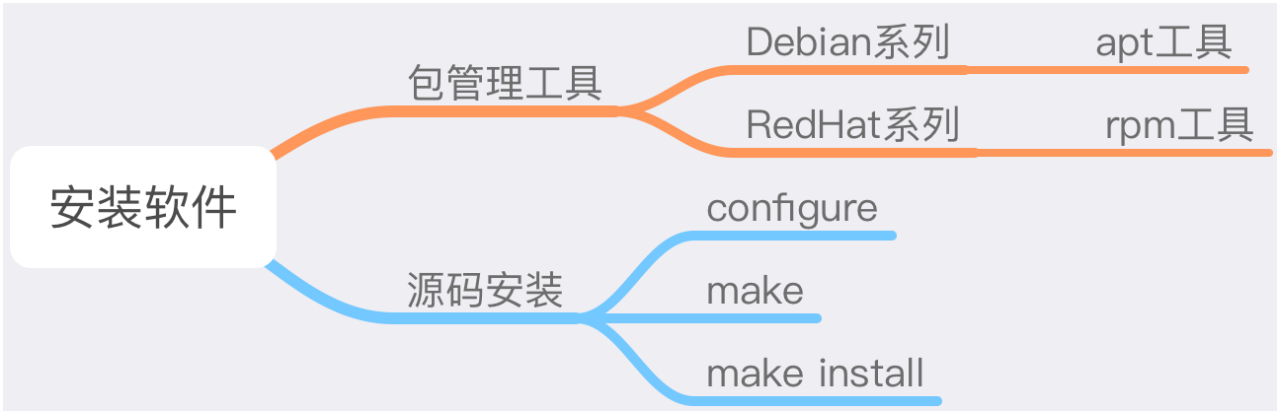
更新时间：2020-08-10 14:52:41



> 人的一生可能燃烧也可能腐朽，我不能腐朽，我愿意燃烧起来！——奥斯特洛夫斯基

必备知识

在本章内容开始之前，我们先说一下 Unix/Linux 环境下软件安装的两种方法。



包管理工具是 Linux 发行版提供的官方安装软件的方法，它可以自动帮我们解决各种依赖，让软件安装变得更简单。不过官方的安装源不会实时更新，可能安装的不是最新版本。作为一个专业的程序猿，我们当然要使用**逼格更高的源码安装**方法了。

> 之所以推荐使用源码安装，是因为这样会让我们对软件的一些特性有更大的灵活性控制。

源码安装简介

在 Unix/Linux 环境下，如果使用源码安装软件的话，一般会经过三个步骤，即 configure 、 make 、 make install 。

configure 的作用：

这一步一般是用来生成 Makefile 文件，为下面的 make 做准备。一般情况下， configure 后面会带一些参数，对编译和安装进行控制。比如说，一般会有一个 prefix 参数，用于控制程序的安装路径；

make 的作用：
这一步就是对程序中的源文件进行编译，生成可执行文件。这个命令其实就是执行第一步生成的 Makefile 文件，按照文件的规则自动的编译源文件；

make install 的作用：

这个命令是执行 Makefile 文件中的 install 标签内容。用来安装上一步生成的可执行文件。

> Makefile 文件包含了很多规则，作为 Unix/Linux 开发，我们应该对这些内容进行简单的了解，至少应该能够知道这些东西是干什么的。

安装步骤

在安装之前，我们要进行一些准备工作，包括操作系统、 nginx 源代码等。

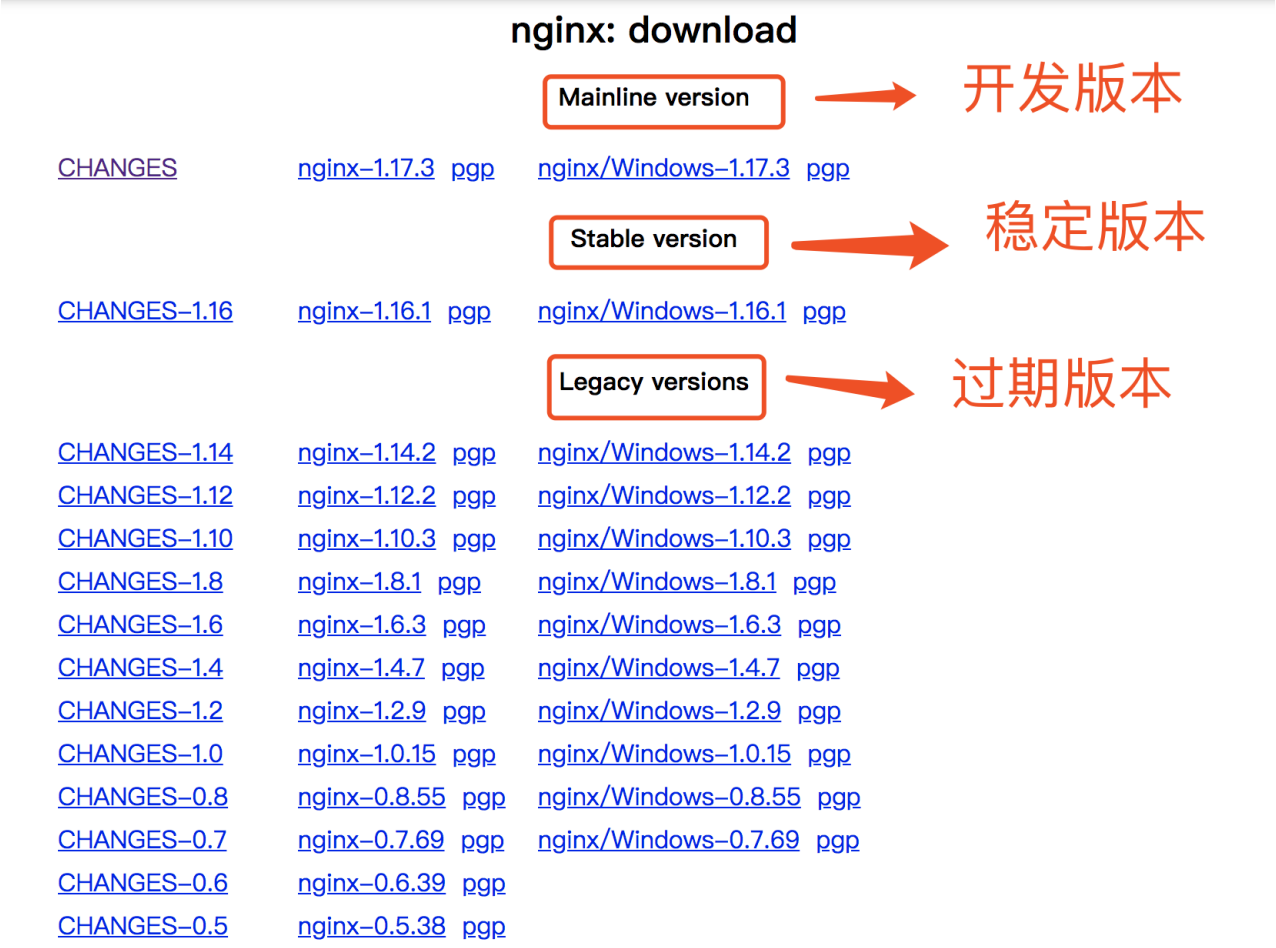
操作系统

对于绝大多数互联网公司来说，服务器操作系统应该都是 Linux 系列，我们本文使用的操作系统是 Centos。

> 如果自己没有 Linux 操作系统的话，我们也可以安装一个虚拟机，或者使用大名鼎鼎的 Docker ，下载一个 Centos 镜像即可。

获取 nginx源代码

打开nginx官网，点击右侧列表的 download 进入源码下载页面。我们可以看到 Nginx 提供了三种版本的下载，分别是 开发版本、稳定版本、过期版本。



我们选择当前最新的稳定版本 nginx-1.16.1。

nginx-1.16.1 是 linux 版本的源码，nginx/Windows-1.16.1 是 windows 版本的源码。两个 pgp 分别是对应平台版本源码经过 PGP 加密之后的签名，我们可以通过这个签名验证下载的内容是否正确。

右键点击 nginx-1.16.1，选择复制链接地址，然后通过 wget 下载源码。

```
wget http://nginx.org/download/nginx-1.16.1.tar.gz
```

```
[root@1cedd1e3eb14 nginx-source-code]# wget http://nginx.org/download/nginx-1.16.1.tar.gz
--2019-09-09 00:27:23--  http://nginx.org/download/nginx-1.16.1.tar.gz
Resolving nginx.org (nginx.org)... 95.211.80.227, 62.210.92.35, 2001:1af8:4060:a004:21::e3
Connecting to nginx.org (nginx.org)|95.211.80.227|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1032630 (1008K) [application/octet-stream]
Saving to: 'nginx-1.16.1.tar.gz'

100%[=======================================================================================>] 1,032,630   21.1KB/s   in 40s

2019-09-09 00:28:03 (25.4 KB/s) - 'nginx-1.16.1.tar.gz' saved [1032630/1032630]
```

解压源码：

```
tar -zxvf nginx-1.16.1.tar.gz
```

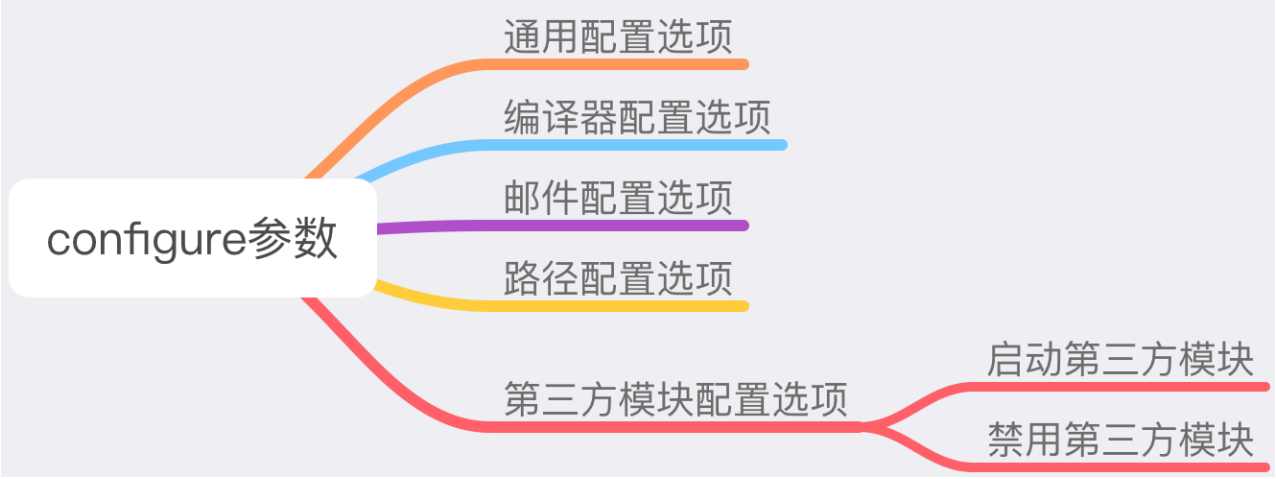
```
[root@1cedd1e3eb14 nginx-source-code]# tar -zxvf nginx-1.16.1.tar.gz
nginx-1.16.1/
nginx-1.16.1/auto/
nginx-1.16.1/conf/
nginx-1.16.1/contrib/
nginx-1.16.1/src/
nginx-1.16.1/configure
nginx-1.16.1/LICENSE
nginx-1.16.1/README
nginx-1.16.1/html/
nginx-1.16.1/man/
nginx-1.16.1/CHANGES.ru
nginx-1.16.1/CHANGES
nginx-1.16.1/man/nginx.8
nginx-1.16.1/html/50x.html
nginx-1.16.1/html/index.html
nginx-1.16.1/src/core/
nginx-1.16.1/src/event/
nginx-1.16.1/src/http/
nginx-1.16.1/src/mail/
nginx-1.16.1/src/misc/
nginx-1.16.1/src/os/
nginx-1.16.1/src/stream/
nginx-1.16.1/src/stream/ngx_stream.c
nginx-1.16.1/src/stream/ngx_stream.h
```

编译源代码

下面进入到刚才解压的 nginx 源码目录中开始对源码进行编译。在文章的开头我们说过，一般情况下，Linux 的软件在编译的时候要经过三个步骤，configure/make以及make install，而 nginx 也不例外。

configure步骤

在前面介绍过，`configure`脚本的作用就是生成`Makefile`文件。这个脚本可以带一些参数，用于控制程序的编译行为。我简单的总结了一下`configure`执行的命令参数，如下所示：



我们通过执行`configure --help`来查看完整的命令参数。我们截取一部分，如下图所示：

```
[root@1cedd1e3eb14 nginx-1.16.1]# ./configure --help

  --help                             print this message

  --prefix=PATH                      set installation prefix
  --sbin-path=PATH                   set nginx binary pathname
  --modules-path=PATH                set modules path
  --conf-path=PATH                   set nginx.conf pathname
  --error-log-path=PATH              set error log pathname
  --pid-path=PATH                    set nginx.pid pathname
  --lock-path=PATH                   set nginx.lock pathname

  --user=USER                        set non-privileged user for
                                     worker processes
  --group=GROUP                      set non-privileged group for
                                     worker processes

  --build=NAME                       set build name
  --builddir=DIR                     set build directory

  --with-select_module               enable select module
  --without-select_module            disable select module
  --with-poll_module                 enable poll module
  --without-poll_module              disable poll module

  --with-threads                     enable thread pool support

  --with-file-aio                    enable file AIO support
```

我们列举几个比较重要的命令参数：

1) 通用配置选项

选项	功能	备注
--prefix=<PATH>	nginx 的前缀路径	其他路径都依赖于该路径，默认为 /usr/local/nginx
--sbin-path=<PATH>	生成的可执行程序保存路径	默认为 prefix + /sbin/nginx
--conf-path=<PATH>	配置文件路径	默认为 prefix + /conf/nginx.conf
--error-log-path=<PATH>	错误日志路径	默认为 prefix + /logs/error.log
--pid-path=<PATH>=<PATH>	pid 文件的保存路径	默认为 prefix + /logs/nginx.pid

2) 第三方模块

第三方模块分为两种，一种是默认自动编译到 nginx 可执行文件中的模块，一种是没有自动编译到 nginx 可执行文件中。

对于前者，我们可以使用 --without-XXX_module 的方式来取消自动编译。比如 --without-http_gzip_module 就是不再将 gzip 压缩模块编译到 nginx 中。

对于后者，我们可以使用 --with--XX_module 的方式将模块编译到可执行程序中。比如我们可以通过 --with-http_geoip_module 命令将地理位置的 geoip 模块编译到最终的 nginx 可执行程序中。

了解了上面的一些基本内容之后，我们就可以进行 configure 过程了。执行下面的命令：

```
./configure --prefix=/usr/local/nginx
```

我们可以在屏幕上看到输出一大堆的信息，这些带 checking 字样的是 configure 脚本自动判断当前操作系统的一些特性，比如获取当前操作系统的内核版本号，是否支持 epoll，判断 int 类型的长度等等。

```
[root@1cedd1e3eb14 nginx-1.16.1]# ./configure --prefix=/usr/local/nginx
checking for OS
 + Linux 4.9.184-linuxkit x86_64
checking for C compiler ... found
 + using GNU C compiler
 + gcc version: 4.8.5 20150623 (Red Hat 4.8.5-36) (GCC)
checking for gcc -pipe switch ... found
checking for -Wl,-E switch ... found
checking for gcc builtin atomic operations ... found
checking for C99 variadic macros ... found
checking for gcc variadic macros ... found
checking for gcc builtin 64 bit byteswap ... found
checking for unistd.h ... found
checking for inttypes.h ... found
checking for limits.h ... found
checking for sys/filio.h ... not found
checking for sys/param.h ... found
checking for sys/mount.h ... found
checking for sys/statvfs.h ... found
checking for crypt.h ... found
checking for Linux specific features
checking for epoll ... found
checking for EPOLLRDHUP ... found
```

突然，在最下面出现了一个触目惊心的 error，这就尴尬了：

```
checking for PCRE library ... not found
checking for PCRE library in /usr/local/ ... not found
checking for PCRE library in /usr/include/pcre/ ... not found
checking for PCRE library in /usr/pkg/ ... not found
checking for PCRE library in /opt/local/ ... not found

./configure: error: the HTTP rewrite module requires the PCRE library.
You can either disable the module by using --without-http_rewrite_module
option, or install the PCRE library into the system, or build the PCRE library
statically from the source with nginx by using --with-pcre=<path> option.
```

从上面错误信息里面，我们可以看出来，nginx先后从 /usr/local/，/usr/include/pcre/，/usr/pkg/，/opt/local/四个位置中找 PCRE 模块，但是都没有找到，所以就报错了。那么什么是 PCRE 模块呢？

> Q1: PCRE是什么？
> A1: PCRE的全称是Perl Compatible Regular Expressions，是一个兼容perl的正则表达式库，使用c语言实现，性能非常的高。nginx使用PCRE实现了http rewrite功能。
>
> Q2: http rewrite是啥？
> A2: 不要着急，我们后面的文章会告诉你~~~
>
> Q3:为什么要从四个位置查找呢？
> A3:： 因为不同的操作系统，软件安装的默认目录是不同的，nginx为了兼容不同的平台，所以要从不同的位置查找

报错信息里面也给了三种解决办法：
① 禁用rewrite模块，即执行configure的时候，指定--without-http_rewrite_module；
② 将PCRE模块安装到默认的系统目录中，这样nginx就可以自动的从默认位置找到PCRE模块；
③ 使用源码编译PCRE，将编译后的文件放到自定义的目录中，在configure的时候通过--with-pcre=<path>的方式。

> 同志们，不要害怕error，优秀软件的error会告诉你很多信息~~~

知道了错误的原因，我们就可以很轻松地搞定它了，我们使用上面的第②种解决办法，将PCRE库安装到系统默认的位置。

```
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]# yum install -y pcre pcre-devel
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: mirrors.huaweicloud.com
 * extras: mirrors.huaweicloud.com
 * updates: mirrors.huaweicloud.com
Package pcre-8.32-17.el7.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
---> Package pcre-devel.x86_64 0:8.32-17.el7 will be installed
--> Finished Dependency Resolution
```

安装成功之后，我们重新执行上面的 ./configure --prefix=/usr/local/nginx 命令，可以发现，能够找到 PCRE 库了。

```
checking for struct dirent.d_type ... found
checking for sysconf(_SC_NPROCESSORS_ONLN) ... found
checking for sysconf(_SC_LEVEL1_DCACHE_LINESIZE) ... found
checking for openat(), fstatat() ... found
checking for getaddrinfo() ... found
checking for PCRE library ... found
checking for PCRE JIT support ... found
checking for zlib library ... not found
```

但是尴尬的是， configure 又双叒叕报错了~~

```
checking for zlib library ... not found

./configure: error: the HTTP gzip module requires the zlib library.
You can either disable the module by using --without-http_gzip_module
option, or install the zlib library into the system, or build the zlib library
statically from the source with nginx by using --with-zlib=<path> option.
```

看错误信息，这次是因为 zlib 库没有找到。和 pcre 错误信息相同，这次也给出了三种解决办法，我们同样适用第②种解决办法安装 zlib 库。

```
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]# yum install -y zlib zlib-devel
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: mirrors.huaweicloud.com
 * extras: mirrors.huaweicloud.com
 * updates: mirrors.huaweicloud.com
Package zlib-1.2.7-18.el7.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
---> Package zlib-devel.x86_64 0:1.2.7-18.el7 will be installed
--> Finished Dependency Resolution
```

再次执行 configure 命令，这个问题就解决了。并且成功的创建了 Makefile 文件。

```
checking for zlib library ... found
creating objs/Makefile

Configuration summary
  + using system PCRE library
  + OpenSSL library is not used
  + using system zlib library

  nginx path prefix: "/usr/local/nginx"
  nginx binary file: "/usr/local/nginx/sbin/nginx"
  nginx modules path: "/usr/local/nginx/modules"
  nginx configuration prefix: "/usr/local/nginx/conf"
  nginx configuration file: "/usr/local/nginx/conf/nginx.conf"
  nginx pid file: "/usr/local/nginx/logs/nginx.pid"
  nginx error log file: "/usr/local/nginx/logs/error.log"
  nginx http access log file: "/usr/local/nginx/logs/access.log"
  nginx http client request body temporary files: "client_body_temp"
  nginx http proxy temporary files: "proxy_temp"
  nginx http fastcgi temporary files: "fastcgi_temp"
  nginx http uwsgi temporary files: "uwsgi_temp"
  nginx http scgi temporary files: "scgi_temp"
```

汇总信息

make步骤

经过上一步的configure，已经生成了Makefile文件，我们就可以通过执行make命令对nginx进行编译，如下：

```
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]# make
make -f objs/Makefile
make[1]: Entering directory `/nginx-source-code/nginx-1.16.1
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-paramet
s -I src/os/unix -I objs \
        -o objs/src/core/nginx.o \
        src/core/nginx.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-paramet
s -I src/os/unix -I objs \
        -o objs/src/core/ngx_log.o \
        src/core/ngx_log.c
```

make install步骤

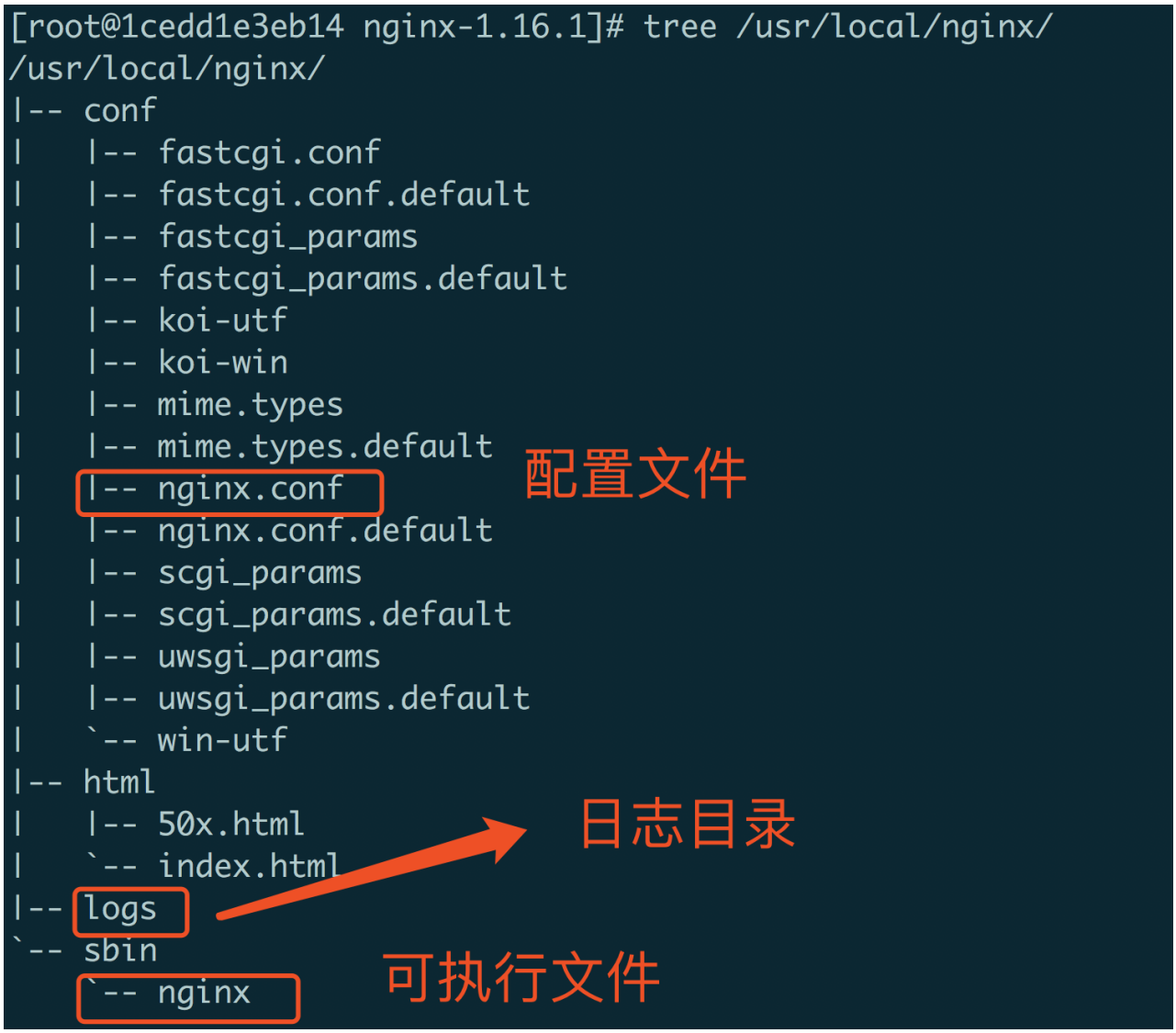
编译成功之后，就剩下最后一步安装了。执行make install命令就行了：

> 这一步可能需要root权限

测试

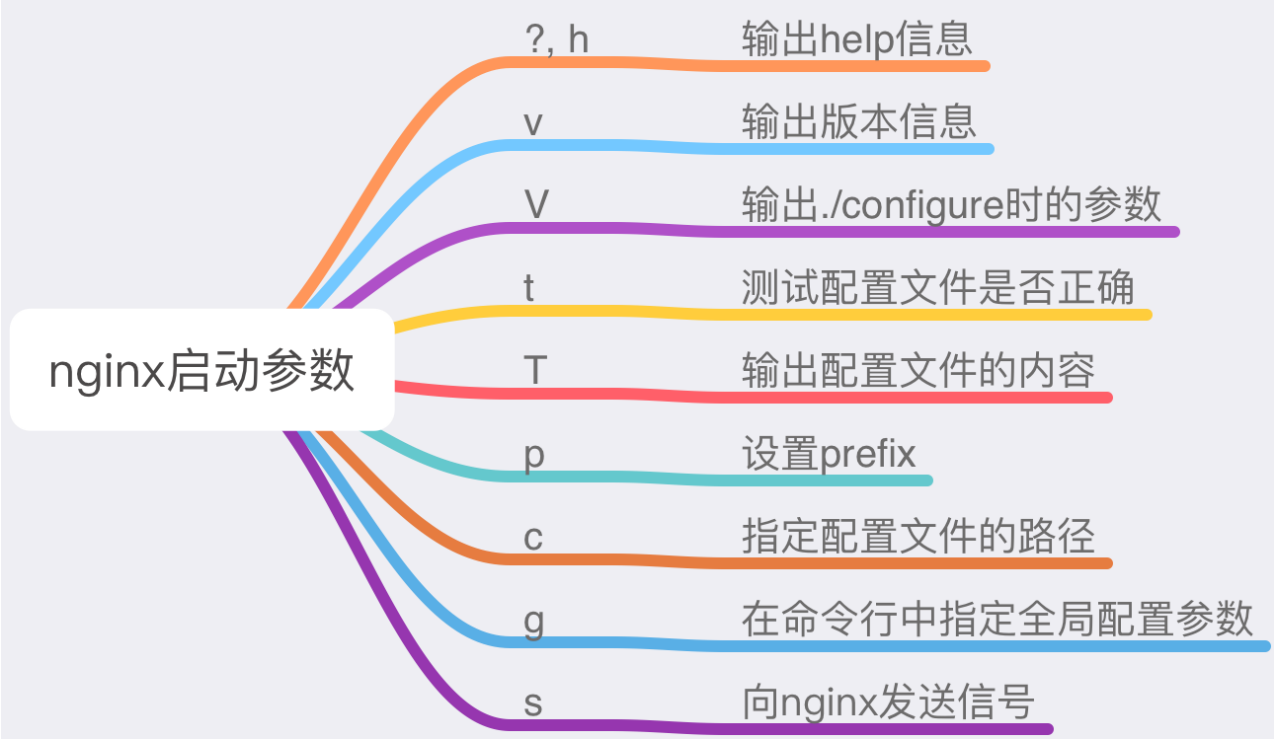
经过上面的重重考验，我们终于自己编译了一个nginx源码，生成了一个可执行文件，下面我们就测试一下我们的劳动成果吧。

我们在configure的时候指定了--prefix=/usr/local/nginx，那么默认情况下最终生成的可执行文件，配置文件，错误日志文件等都会在这个目录下，我们看一下这个目录的结构：



启动nginx

nginx启动的时候可以带很多参数，如下图：



其中常用的就是 -t，-c，-p，-g，-s 这几个。

-c：指定 nginx 启动时使用的配置文件，默认为 /usr/local/nginx/conf/nginx.conf；

-t：测试配置文件的语法是否正确；

-p：指定 nginx 服务器使用的文件的路径前缀，默认为 /usr/local/nginx；

-g：通过命令行指定一些全局配置选项；

-s：向 nginx 进程发送信号。

我们可以通过直接执行 nginx 可执行文件(不带任何参数)来启动 nginx服务 。

```
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]#/usr/local/nginx/sbin/nginx
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]# ps -ef | grep nginx
root      7281     1  0 00:26 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx
nobody    7282  7281  0 00:26 ?        00:00:00 nginx: worker process
root      7284     1  0 00:26 pts/0    00:00:00 grep --color=auto nginx
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]#
```


> 这种情况下使用的是默认的配置文件，即 /usr/local/nginx/conf/nginx.conf 。

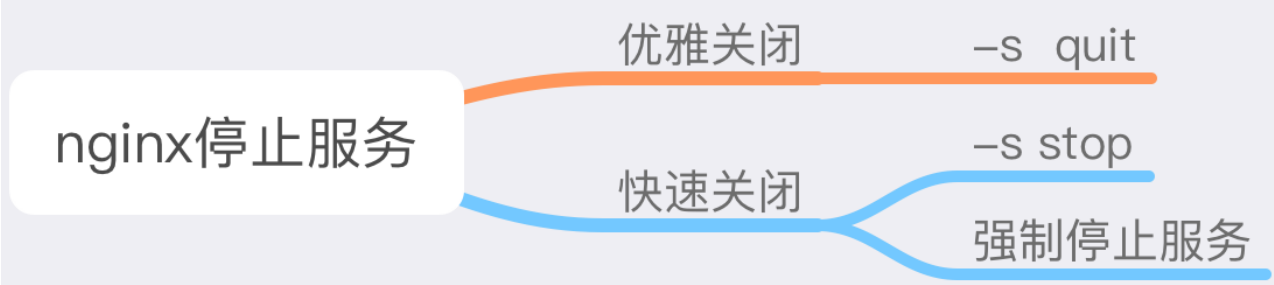
我们也可以通过curl命令来查看nginx是否启动成功。

```
[root@1cedd1e3eb14 nginx-1.16.1]#
[root@1cedd1e3eb14 nginx-1.16.1]# curl http://localhost/
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

nginx欢迎页面内容

停止nginx

nginx提供了两种方法来停止服务：优雅关闭和快速关闭。



快速关闭: nginx强制停止服务，master和worker进程收到信号之后，会立即结束运行。

优雅关闭: nginx会按照下面的步骤停止服务：

1). 关闭监听端口，停止接收新的连接；

2). Nginx处理完当前的所有。剩余请求；

3). 停止Nginx服务

其实这两种方式都是对kill命令的一个封装，我们也可以直接使用kill命令完成相同的功能。只不过kill命令要知道nginx的master进程的pid，我们可以结合ps命令来查找进程号。

其实还有一种方法来获取master进程的pid，那就是nginx.pid文件，这个文件的内容就是master进程的pid，大家知道这种方法就行了，尽量使用nginx提供的-s命令。

重新加载配置文件

当配置文件发生改变之后，牛逼的 nginx 可以不用断开服务，直接重新加载配置就行了。

```
nginx -s reload
```

总结

本文介绍了一下 nginx 安装过程，以及一些 nginx 常用的命令。希望大家学习愉快！

}

03 HTTP 请求与响应

05 Linux的 IO 原理