**1.Nginx安装**

我使用的环境是64位 Ubuntu 14.04。nginx依赖以下模块：

  gzip模块需要 zlib 库

  rewrite模块需要 pcre 库

  ssl 功能需要openssl库

* 1. **安装pcre**

**Wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.38.tar.gz**

1.         获取pcre编译安装包，在<http://www.pcre.org/>上可以获取当前最新的版本

2.         解压缩pcre-xx.tar.gz包。

3.         进入解压缩目录，执行./configure。

4.         make & make install

**1.2.安装openssl**

**Wget http://www.openssl.org/source/openssl-0.9.8zh.tar.gz**

1.         获取openssl编译安装包，在<http://www.openssl.org/source/>上可以获取当前最新的版本。

2.         解压缩openssl-xx.tar.gz包。

3.         进入解压缩目录，执行./config。

4.         make & make install

**1.3.安装zlib**

**Wget http://www.zlib.net/zlib-1.2.8.tar.gz**

1.         获取zlib编译安装包，在<http://www.zlib.net/>上可以获取当前最新的版本。

2.         解压缩openssl-xx.tar.gz包。

3.         进入解压缩目录，执行./configure。

4.         make & make install

**1.4.安装nginx**

**wget nginx.org/download/nginx-1.9.7.tar.gz**

1.         获取nginx，在http://nginx.org/en/download.html上可以获取当前最新的版本。

2.         解压缩nginx-xx.tar.gz包。

3.         进入解压缩目录，执行./configure

4.         make & make install

若安装时找不到上述依赖模块，使用--with-openssl=<openssl\_dir>、--with-pcre=<pcre\_dir>、--with-zlib=<zlib\_dir>指定依赖的模块目录。如已安装过，此处的路径为安装目录；若未安装，则此路径为编译安装包路径，nginx将执行模块的默认编译安装。

启动nginx之后，浏览器中输入[http://localhost](http://localhost/)可以验证是否安装启动成功。

[](http://images.cnitblog.com/blog/92071/201412/051025494514816.jpg)

**2.Nginx配置**

安装完成之后，配置目录conf下有以下配置文件，过滤掉了xx.default配置：

|  |
| --- |
| tyler@ubuntu:/opt/nginx-1.7.7/conf$ tree |grep -v default  .  ├── fastcgi.conf  ├── fastcgi\_params  ├── koi-utf  ├── koi-win  ├── mime.types  ├── nginx.conf  ├── scgi\_params  ├── uwsgi\_params  └── win-utf |

**除了nginx.conf，其余配置文件，一般只需要使用默认提供即可**。

**2.1.nginx.conf**

nginx.conf是主配置文件，默认配置去掉注释之后的内容如下图所示：

  worker\_process表示工作进程的数量，一般设置为cpu的核数

  worker\_connections表示每个工作进程的最大连接数

  server{}块定义了虚拟主机

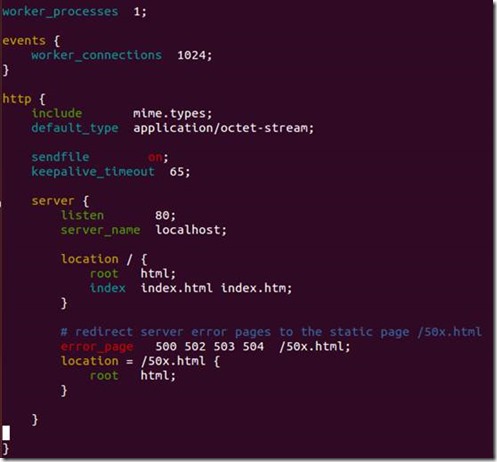
  listener监听端口

  server\_name监听域名

  location{}是用来为匹配的 URI 进行配置，URI 即语法中的“/uri/”。location  / { }匹配任何查询，因为所有请求都以 / 开头。

  root指定对应uri的资源查找路径，这里html为相对路径，完整路径为/opt/ opt/nginx-1.7.7/html/

  index指定首页index文件的名称，可以配置多个，以空格分开。如有多个，按配置顺序查找。

[](http://images.cnitblog.com/blog/92071/201412/051025518118428.jpg)

从配置可以看出，nginx监听了80端口、域名为localhost、跟路径为html文件夹（我的安装路径为/opt/nginx-1.7.7，所以/opt/nginx-1.7.7/html）、默认index文件为index.html， index.htm、服务器错误重定向到50x.html页面。

可以看到/opt/nginx-1.7.7/html/有以下文件：

|  |
| --- |
| tyler@ubuntu:/opt/nginx-1.7.7/html$ ls  50x.html  index.html |

这也是上面在浏览器中输入[http://localhost](http://localhost/)，能够显示欢迎页面的原因。实际上访问的是/opt/nginx-1.7.7/html/index.html文件。

**2.2.mime.types**

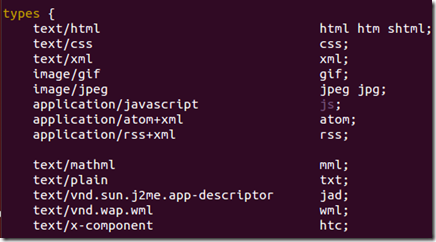
**文件扩展名与文件类型映射表，nginx根据映射关系，设置http请求响应头的Content-Type值**。当在映射表找不到时，使用nginx.conf中default-type指定的默认值。例如，默认配置中的指定的default-type为application/octet-stream。

    include       mime.types;

    default\_type  application/octet-stream;

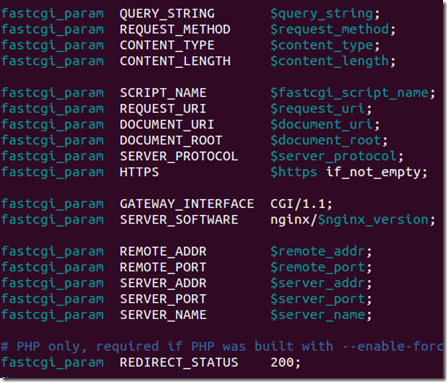
默认

下面截一段mime.types定义的文件扩展名与文件类型映射关系，完整的请自行查看：

[](http://images.cnitblog.com/blog/92071/201412/051025529831971.png)

**2.3.fastcgi\_params**

nginx配置Fastcgi解析时会**调用fastcgi\_params配置文件来传递服务器变量**，这样CGI中可以获取到这些变量的值。默认传递以下变量：

[](http://images.cnitblog.com/blog/92071/201412/051025550617368.png)

这些变量的作用从其命名可以看出。

**2.4.fastcgi.conf**

对比下fastcgi.conf与fastcgi\_params文件，可以看出只有以下差异：

|  |
| --- |
| tyler@ubuntu:/opt/nginx-1.7.7/conf$ diff fastcgi.conf fastcgi\_params  2d1  < fastcgi\_param  SCRIPT\_FILENAME    $document\_root$fastcgi\_script\_name; |

即fastcgi.conf只比fastcgi\_params多了一行“fastcgi\_param  SCRIPT\_FILENAME    **$document\_root**$fastcgi\_script\_name;”

原本只有fastcgi\_params文件，fastcgi.conf是nginx 0.8.30 (released: 15th of December 2009)才引入的。主要为是解决以下问题（参考：<http://www.dwz.cn/x3GIJ>）：

原本Nginx只有fastcgi\_params，后来发现很多人在定义SCRIPT\_FILENAME时使用了硬编码的方式。例如，fastcgi\_param SCRIPT\_FILENAME**/var/www/foo**$fastcgi\_script\_name。于是为了规范用法便引入了fastcgi.conf。

不过这样的话就产生一个疑问：为什么一定要引入一个新的配置文件，而不是修改旧的配置文件？这是因为fastcgi\_param指令是数组型的，和普通指令相同的是：内层替换外层；和普通指令不同的是：当在同级多次使用的时候，是新增而不是替换。换句话说，如果在同级定义两次SCRIPT\_FILENAME，那么它们都会被发送到后端，这可能会导致一些潜在的问题，为了避免此类情况，便引入了一个新的配置文件。

因此不再建议大家使用以下方式（搜了一下，网上大量的文章，并且nginx.conf的默认配置也是使用这种方式）：

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

而使用最新的方式：

include fastcgi.conf;

**2.5.uwsgi\_params**

与fastcgi\_params一样，**传递哪些服务器变量**，只有前缀不一样，以**uwsgi\_param**开始而非**fastcgi\_param**。

**2.6.scgi\_params**

与fastcgi\_params一样，**传递哪些服务器变量**，只有前缀不一样，以**uwsgi\_param**开始而非**fastcgi\_param**。

**2.7.koi-utf、koi-win、win-utf**

这三个文件都是与编码转换映射文件，用于在输出内容到客户端时，将一种编码转换到另一种编码。

koi-win： charset\_map  koi8-r < -- > windows-1251

koi-utf： charset\_map  koi8-r < -- > utf-8

win-utf： charset\_map  windows-1251 < -- > utf-8

koi8-r是[斯拉夫文字](http://zh.wikipedia.org/wiki/%E6%96%AF%E6%8B%89%E5%A4%AB%E8%AF%AD%E6%97%8F)8位元编码，供[俄语](http://zh.wikipedia.org/wiki/%E4%BF%84%E8%AF%AD)及保加利亚语使用。在[Unicode](http://zh.wikipedia.org/wiki/Unicode)未流行之前，KOI8-R 是最为广泛使用的俄语编码，使用率甚至起[ISO/IEC 8859-5](http://zh.wikipedia.org/wiki/ISO/IEC_8859-5)还高。这3个文件存在是因为作者是俄国人的原因。