# linux压缩软件介绍

在Linux系统中，提供了许多压缩工具，虽然方便，但也难免造成一些混乱。而实际上我们并不需要这么多的工具，其实，你只要掌握几种主要的压缩工具就可以了。我们经常能看到的压缩文件，实际上也就那么几种，像gz、bz2、Z、tar.gz、tar.bz2、tar.xz、tar.Z，至于像zip、rar这样的格式，其实不推荐在linux使用。

通常，linux下的压缩软件不能直接压缩目录，所以像后边的带tar实际上就是首先要对压缩的目录进行归档，其实就是把目录打包，而tar实际上也就是一个打包命令，而不是压缩软件，这一点一定要清楚，如果你只是压缩单个大文件，直接用压缩软件就好了，如果需要压缩目录，那就必须先用tar打包

gzip/gunzip .gz

compress/uncompress .Z

zip/unzip .zip

bzip2/bunzip2 .bz2

xz .xz

而通常，我们要做的都是对一批文件进行处理，所以tar就变成了一个功能很强的软件，它可以通过参数直接调用压缩软件，虽然会损失一些效率，但非常方便，下面就这些软件的使用方法，做一个总结，首先是单个文件的压缩、解压缩，然后重点介绍tar的使用。

## 1、gzip gunzip

这两个软件都可以压缩和解压缩.gz -gz .z -z .Z 和 –Z文件，它只能对单个文件进行压缩

压缩格式：

gzip filename

gzip -9 filename

注意他会删除filename，生成filename.gz文件，他有一个-r的参数，可以递归压缩目录，但你不要高兴的太早，你会发现，他只是可以用目录来做参数，批量压缩文件，就是说，分别把所有文件都压缩了，而不是生成一个压缩文件。gzip也有一些比较有意思的用法，这些man中都有，这里不再研究了。

解压缩格式：

gunzip filename.gz

gzip -d filename.gz

这里还是那句话，gzip没有提供保留源文件的参数，如果你要保留源文件，请事先做好备份。

使用上面的命令后，filename.gz会被解压缩并去掉后缀名.gz变成filename。gunzip 实际上是gzip程序的一部分，它跟gzip –d 这个命令有等同效果。但是我们使用的时候，经常使用gunzip, 因为这个用法似乎更酷些。

## 2、bzip2

bzip2是Linux里面压缩程序的另一个选择。它的压缩算法不同于gzip。与gzip 相比有其优点，也有一些缺点。bzip2的主要优点在于它压缩后文件的尺寸，对于相同文件，bzip2 压缩后的尺寸几乎总是小于gzip的压缩结果。有些时候，这个差距会相当大。这个优点对于带宽小的用户来说表现得尤为突出。需要提醒的是，当在一些公共ftp 服务器上下载文件时，在.gz和.bz2文件中尽量选择.bz2文件是一种基本的网络礼节，因为这样可以减少服务器的负担以给更多人服务。

bzip2的缺点在于，它占用的CPU的使用率会比gzip更多。这就是说，压缩同一个文件，bzip2要比gzip使用更长的时间，占用更多的CPU资源。所以，当你在两者之中考虑选用哪种压缩程序的时候，需要衡量的是速度和压缩后的文件尺寸哪个对你来说更重要些。bzip2的使用方法和gzip是一样的，所以我们不再花时间讨论了。像gunzip一样，bunzip2的效果等同于bzip2 –d。与gzip最大的不同是bzip2使用后缀名.bz2.

bzip2 filename

bunzip2 filename.bz2

bzip2 -d filename.bz2

## 3、zip

处理zip文件的两个程序。zip文件在Windows世界里面是极为常用的，所以Linux也有相应的程序来处理它们。zip文件的压缩程序叫做zip,解压缩程序叫做unzip.

$ zip foo \*

这行命令会建立一个包涵当前目录所有文件的的zip文件 foo.zip。zip会自动将.zip后缀名加上，所以我们不需要在命令中加上。你也可以加上一个参数-r使zip将当前目录下的所有文件夹目录也加在zip文件中：

$ zip -r foo \*

解压缩文件很简单，像这样：

$ unzip foo.zip

这样会解压缩foo.zip里面的所有文件和文件夹

zip程序还有一些高级应用参数，包括建立自解压包，压缩是保留源文件，调节压缩文件的尺寸等等，在这里就不一一赘述了。如果想了解更多，请用man查看帮助。

## 4、xz

这是一个非常厉害的压缩软件，有压缩之王的称号，就是说他比bzip2压缩比还要高。xz这个压缩软件可能很多人都很陌生，不过您可知道xz是绝大数linux默认就带的一个压缩工具。但它的知名度视乎不是很高，也和它的压缩特效有关，就是虽然压缩比很高，但压缩时间也是最长的。

之前xz使用一直很少，所以几乎没有什么人提起。最新一段时间会经常听到xz被采用的声音，像是最新的archlinux某些东西就使用xz压缩。不过xz也有一个坏处就是压缩时间比较长，比7z压缩时间还长一些。不过压缩是一次性的，所以可以忽略。

### xz压缩文件方法或命令

xz -z 要压缩的文件

如果要保留被压缩的文件加上参数 -k ，如果要设置压缩率加入参数 -0 到 -9调节压缩率。如果不设置，默认压缩等级是6.

### xz解压文件方法或命令

xz -d 要解压的文件

同样使用 -k 参数来保留被解压缩的文件。

创建或解压tar.xz文件的方法

习惯了 tar czvf 或 tar xzvf 的人可能碰到 tar.xz也会想用单一命令搞定解压或压缩。比如 z是针对 gzip，j是针对 bzip2，而J是针对xz的。

tar -Jcvf和tar -Jxvf就是压缩和解压缩的使用方法，当然，你也可以像下面的方法，分步骤来压缩打包，但实际上效果是略有不同，虽然两者都是调用xz来压缩的，但执行上可能略有不同，但这种差距通常可以忽略不计，我们也都是使用J参数，注意这个J是大写的J，不是j，因为它们对应的是不同的压缩方式。

分步骤创建tar.xz文件：

必须先打包，然后才能用xz来压缩，因为它不能处理多文件压缩。

tar cvf xxx.tar xxx/

先将目录xxx打包成xxx.tar文件，然后使用：

xz -z -k xxx.tar

来将 xxx.tar压缩成为 xxx.tar.xz

解压tar.xz文件：先 xz -d xxx.tar.xz 将 xxx.tar.xz解压成 xxx.tar 然后，再用 tar xvf xxx.tar来解包。

压缩

xz -z -k -9 filename

解压缩

xz -d -k filename.xz

## 5、tar

从上的介绍中，大家应该可以看到，实际上除了zip外，他们都不可以对目录压缩，而我们通常选择压缩软件时，一个是考虑速度，一个是时间，还有一种选择就是均衡性，所以，其实大多数的软件其实我们没有什么用，比如你要压缩比高的，那么其实除了gzip bzip2 xz，其他的就不要选了，如果你要求速度，xzip bzip2就不要选了，所以gun把gzip定位为最主要的解压缩程序。

但gzip最大的问题也是不能对目录操作，所以就有了tar这样的打包软件，注意tar其实不是一个压缩软件，他只是一个打包软件，但他可以通过参数调用其他压缩软件，这样一来，他就几乎变成了一个万能的压缩软件了。

我们看一下压缩，其实对于tar来说，就是打包，他用的参数是-c，我们通常用的参数是zcvf，这四个参数的含义是这样的，z表示用gzip压缩，c表示打包，v表示显示过程，f表示指定文件名，特别说一下这个f必须放在最后，看下边这个例子：

tar -cvf /mnt/jpg.tar \*.jpg //将目录里所有jpg文件打包成jpg.tar

tar -zcvf /mnt/jpg.tar.gz \*.jpg //将目录里所有jpg文件打包成jpg.tar后，并且将其用gzip压缩，生成一个自命名文件jpg.tar.gz

tar -jcvf /mnt/jpg.tar.bz2 \*.jpg //将目录里所有jpg文件打包成jpg.tar后，并且将其用bzip2压缩，生成一个自命名文件jpg.tar.bz2

tar -Zcvf /mnt/jpg.tar.Z \*.jpg //将目录里所有jpg文件打包成jpg.tar后，并且将其用compress压缩，生成一个自命名文件jpg.tar.Z

tar -Jcvf /mnt/jpg.tar.xz \*.jpg //将目录里所有jpg文件打包成jpg.tar后，并且将其用xz压缩，生成一个自命名文件jpg.tar.xz

rar a jpg.rar \*.jpg //rar格式的压缩，需要先下载rar for linux

zip jpg.zip \*.jpg //zip格式的压缩，需要先下载zip for linux

这里边的参数，就是我们平时用的最多的，注意大小写，和路径。这里其实只有-v参数没有什么实际意义，可有可无，如果先乱，就不要加了。

下面来说一下解压缩，其实就是把c参数，换成x参数就可以了，c表示打包，x就表示解包了，其他一样，当然可以加C参数，指定解压路径，这个很重要。

tar -xvf /mnt/jpg.tar -C /tmp //将目录里jpg.tar文件，解包到/tmp目录下

tar -zxvf /mnt/jpg.tar.gz -C /tmp //将目录里jpg.tar.gz文件,用gzip解压到/tmp目录下，然后再解包

tar -jxvf /mnt/jpg.tar.bz2 -C /tmp //将目录里jpg.tar.bz2文件,用bzip2解压缩到/tmp目录下，然后再解包

tar -Zxvf /mnt/jpg.tar.Z -C /tmp //将目录里jpg.tar.Z文件,用compress解压缩到/tmp目录下，然后再解包

tar -Jxvf /mnt/jpg.tar.xz -C /tmp //将目录里jpg.tar.xz文件，用xz解压缩到/tmp目录下，然后再解包

unrar e jpg.rar //解压rar

unzip jpg.zip //解压zip

其实，你学会这些，基本上就可以正常工作了，下面把tar的一些常用参数解释列了出来

-c: 建立压缩档案

-x：解压

-t：查看内容

-r：向压缩归档文件末尾追加文件

-u：更新原压缩包中的文件

这五个是独立的命令，压缩解压都要用到其中一个，可以和别的命令连用但只能用其中一个。下面的参数是根据需要在压缩或解压档案时可选的。

-z：有gzip属性的

-Z：有compress属性的

-j：有bz2属性的

-J：有xz属性的

-v：显示所有过程

-C：指定解压缩路径

-O：将文件解开到标准输出

下面的参数-f是必须的

-f: 使用档案名字，切记，这个参数是最后一个参数，后面只能接档案名。

# tar -cf all.tar \*.jpg

这条命令是将所有.jpg的文件打成一个名为all.tar的包。-c是表示产生新的包，-f指定包的文件名。

# tar -rf all.tar \*.gif

这条命令是将所有.gif的文件增加到all.tar的包里面去。-r是表示增加文件的意思。

# tar -uf all.tar logo.gif

这条命令是更新原来tar包all.tar中logo.gif文件，-u是表示更新文件的意思。

# tar -tf all.tar

这条命令是列出all.tar包中所有文件，-t是列出文件的意思

# tar -xf all.tar

这条命令是解出all.tar包中所有文件，-t是解开的意思

最后，总结一下吧，关于linux下的压缩软件，基本上你只要掌握上边这些，其实就差不多了，当然，还有很多格式和软件，对于一些不常用的，其实用的时候现学一下也就可以了。

# cp直接覆盖不提示的方法

新做了服务器，cp覆盖时，无论加什么参数-f之类的还是提示是否覆盖，这在大量cp覆盖操作的时候是不能忍受的。

把a目录下的文件复制到b目录，以下是代码片段：

　　cp -r a/\* b

　　执行上面的命令时，b存在的每个文件都会提示是否覆盖;

　　而执行以下是代码片段：

cp -r -f a/\* b

cp -rf a/\* b

　　执行上面的命令时，b存在的每个文件都不再会提示;

这是我们希望的理想状态，但是有时加了-f了，怎么还会有提示呢？

原来CentOS服务器会默认增加别名 alias cp='cp -i'，当你执行cp时，其实执行的是cp -i。在终端执行alias就可以看出来了。以下是代码片段：

　　[root@devdb ~]# alias

　　alias cp='cp -i'

　　通过alias命令可以这样解决，但这只是临时设置，要想永久设置，需要更改bash环境变量。以下是代码片段：

　　[root@devdb ~]# vi ~/.bashrc

　　在alias cp='cp -i'前加上"#"注释掉这行，：wq!保存推出，然后重新登陆就可以了。

# scp 使用方法

scp就是secure copy，一个在linux下用来进行远程拷贝文件的命令。有时我们需要获得远程服务器上的某个文件，该服务器既没有配置ftp服务器，也没有做共享，无法通过常规途径获得文件时，只需要通过简单的scp命令便可达到目的。通常scp只是用来临时解决拷贝问题。

## 一、将本机文件复制到远程服务器上

#scp /home/administrator/news.txt root@192.168.6.129:/etc/squid

/home/administrator/ 本地文件的绝对路径

news.txt 要复制到服务器上的本地文件

root 通过root用户登录到远程服务器（也可以使用其他拥有同等权限的用户）

192.168.6.129 远程服务器的ip地址（也可以使用域名或机器名）

/etc/squid 将本地文件复制到位于远程服务器上的路径

## 二、将远程服务器上的文件复制到本机

#scp remote@www.abc.com:/usr/local/sin.sh /home/administrator

remote 通过remote用户登录到远程服务器（也可以使用其他拥有同等权限的用户）

www.abc.com 远程服务器的域名（当然也可以使用该服务器ip地址）

/usr/local/sin.sh 欲复制到本机的位于远程服务器上的文件

/home/administrator 将远程文件复制到本地的绝对路径

## 三、注意两点：

1.如果远程服务器防火墙有特殊限制，scp便要走特殊端口，具体用什么端口视情况而定，命令格式如下：

#scp -p 4588 remote@www.abc.com:/usr/local/sin.sh /home/administrator

2.使用scp要注意所使用的用户是否具有可读取远程服务器相应文件的权限。

如：

# scp /var/log/sql-slow-queries.log root@192.168.1.50:/root

把本地/var/log/sql-slow-queries.log 这个文件copy到1.50上面/root 目录下面

# man命令的原理和用法

很多初始linux的人，都对man命令比较陌生，多数只是知道这个命令，却很少有人能真正用好它，原因，我想是大部分看不懂这个帮助，或者英语实在是太烂了。

其实man作为linux的帮助命令，其作用是十分重要的，这里再给大家详细说一下man的工作机制和原理以及用法。

linux man中的man就是manual的缩写，中文说法就是手册。在linux中，man手册就是用来查看系统中自带的各种命令、库函数等的帮助信息。

使用man命令其实是很容易的，这里先介绍一下man命令的语法格式：

man [SECTION NUMBER] MAN PAGE NAME

其中，SECTION NUMBER 表示man手册页的章节号

MAN PAGE NAME 表示查询页名称，通常就是命令、文件、库函数的名称

通常，man手册章节号是根据Unix GUN计划的规范定义的，主要分为以下几个部分：

第1章节：查询普通命令帮助，一般都在这一节中。

第2章节：查询系统调用帮助，就是查询由内核提供的函数帮助。

第3章节：查询c库函数帮助，包括头文件定义，行为和基本c库函数调用目的。

第4章节：查询特殊文件帮助，通常就是保存在/dev/目录下得各种设备文件。

第5章节：查询文件格式帮助，例如，密码文件passwd格式，手册中会详细介绍该文件的各字段含义。

第6章节：游戏和屏幕保护信息，这个对于服务器，就没啥用了

第7章节：杂集，什么叫各种命令和信息，应该是前面没有的吧。

第8章节：系统管理员命令和守护进程，该章节中的信息都是系统管理员才能使用的命令和进程。

以上八个章节是man手册的标准章节，而每一个章节中，帮助文档也都是标准布局的，这个标准布局主要包括如下部分：

NAME：表示命令、函数、系统调用或文件格式的名称

SYNOPSIS：表示命令、函数、系统调用、文件格式等语法格式

DESCRIPTION：对命令功能的描述

EXAMPLES：对命令使用给出示例

SEE ALSO：表示参考文档、web页面及其它相关程序

示例：

man 1 passwd

man 5 passwd

这两个命令显示的结果显然是不同了，

man 1 passwd查看的是passwd命令的帮助文档，显示的自然是passwd这个命令的帮助，

而man 5 passwd则查看的时passwd这个文件的格式，包括密码文件的所有字段的解释。

也就是说同样是passwd，一个是命令帮助，一个是文件格式帮助，一个位于第1章节，一个位于第5章节，这样解释，我想大家就应该很清楚了。

再看一个，

man 4 pts

man 3 stdin

man 3 stderr

这里大家应该发现还有一个比较奇怪的事就是，很多时候，我们用man命令时并没有加上章节，比如man passwd

这是因为，man命令是默认从手册的数字较小的章节中开始查找的。也就是说如果你要查看passwd命令，你只要写man passwd就可以了，而你要查看passwd文件格式，必须使用man 5 passwd

把 “man page” 转成文本文件，如：man ls | col -b > ~/Desktop/man\_ls.txt

另外，对于man命令的其他一些功能，比如，书签一类的，其实就比较鸡肋了，就是：状态下输入m键，然后输入书签名字，比如a，这时a就是一个书签，我们可以通过单引号或者双引号，来调用a返回书签位置。

man中还可以运行命令，就是先输入！，后面接着输入命令就好，其实这些东西，用处也不是很大，大家了解一些吧。。。。

# linux中命令后面带参数“-”与“--”区别

linux程序的命令行选项有三种风格：

Unix 风格、GNU 风格和 X toolkit 风格。

## Unix 风格

首先说的 -a 这种。在选项需要加参数的时候，紧跟在选项后面即可（或者加空格）。比如登录 mysql server 的时候：

$ mysql -u root -p

或者

$ mysql -uroot -p

均可。这时，root 就是 u 的参数，表示使用 root 用户登录。另外加不加空格看程序怎么才处理了，没有明确的规定。

## GNU 风格

即题主说的 --help 这种，使用两个连字符加上关键词（而不是单个字符）。这种风格的出现是因为有一些复杂的 GNU 程序，仅仅 26 个字母（或者算上大小写 52 个）不够使用而发展出来的。另外一个有点是容易理解，因为出现的不再是缩写的字母。选项参数可以使用空格分割也可以使用"="来分割。如：

$ ls --human-readable --sort=time

如果使用 Unix 风格，那么上条命令则是

$ ls -ht

是不是更加易读呢？

## X toolkit 风格

这是一种比较不常见的风格，使用单个连字符加上关键词。只有 X 相关的程序才使用这种风格，一般不建议使用。

$ xeyes -display joesws:0 -geometry 1000x1000+0+0

看上去和 GNU 风格差不多，只是双连字符改成了单个连字符。

# make clean与make distclean的区别

make clean仅仅是清除之前编译的可执行文件及配置文件。

而make distclean要清除所有生成的文件。

Makefile

在符合GNU Makefile规范的Makefile文件中，包含了一些基本的预先定义的操作：

make

根据Makefile编译源代码，连接，生成目标文件，可执行文件。

make clean

清除上次的make命令所产生的object文件（后缀为“.o”的文件）及可执行文件。

make install

将编译成功的可执行文件安装到系统目录中，一般为/usr/local/bin目录。

make dist

产生发布软件包文件（即distribution package）。这个命令将会将可执行文件及相关文件打包成一个tar.gz压缩的文件用来作为发布软件的软件包。它会在当前目录下生成一个名字类似“PACKAGE-VERSION.tar.gz”的文件。PACKAGE和VERSION，是我们在configure.in中定义的AM\_INIT\_AUTOMAKE(PACKAGE, VERSION)。

make distcheck

生成发布软件包并对其进行测试检查，以确定发布包的正确性。这个操作将自动把压缩包文件解开，然后执行configure命令，并且执行make，来确认编译不出现错误，最后提示你软件包已经准备好，可以发布了。

make distclean

类似make clean，但同时也将configure生成的文件全部删除掉，包括Makefile

# 开源软件包安装之configure脚本详解

Linux环境下的软件安装，并不是一件容易的事情；如果通过源代码编译后在安装，当然事情就更为复杂一些；现在安装各种软件的教程都非常普遍，但万变不离其中，对基础知识的扎实掌握，安装各种软件的问题就迎刃而解了。Configure脚本配置工具就是基础之一，这些都是典型的使用GNU的AUTOCONF和AUTOMAKE产生的程序的安装步骤。

./configure

是用来检测你的安装平台的目标特征的。比如它会检测你是不是有CC或GCC，并不是需要CC或GCC，它是个shell脚本。通俗的讲就是检查make所需要的环境，并且生成Makefile文件，但并不执行Makefile中的具体指令。

make

是用来编译的，它从Makefile中读取指令，然后编译。

make install

是用来安装的，它也从Makefile中读取指令，安装到指定的位置。

AUTOMAKE和AUTOCONF是非常有用的用来发布C程序的东西。

1、configure

这一步一般用来生成 Makefile，为下一步的编译做准备，你可以通过在 configure 后加上参数来对安装进行控制，比如代码:

./configure --prefix=/usr

意思是将该软件安装在 /usr 下面，执行文件就会安装在 /usr/bin （而不是默认的/usr/local/bin),资源文件就会安装在 /usr/share（而不是默认的/usr/local/share）。同时一些软件的配置文件你可以通过指定 --sys-config= 参数进行设定。有一些软件还可以加上 --with、--enable、--without、--disable 等等参数对编译加以控制，

你可以通过 ./configure –help 察看详细的说明帮助。

2、make

这一步就是编译，大多数的源代码包都经过这一步进行编译（当然有些perl或python编写的软件需要调用perl或python来进行编译）。如果在 make 过程中出现 error ，你就要记下错误代码（注意不仅仅是最后一行），然后分析你的系统少哪些依赖库，这些需要自己仔细研究错误代码。通常都是这样，如果是软件本身的bug，通常如果需要解决，就只能向软件提供者寻求帮助，当然，如果你自己能够更改源代码，你也可以自己修改源代码。

3、make install

这条命令来进行安装（当然有些软件需要先运行 make check 或 make test 来进行一些测试），这一步一般需要你有 root 权限（因为要向系统写入文件）。所以通常得用 sudo make install

Linux的用户可能知道，在Linux下安装一个应用程序时，一般先运行脚本configure，然后用make来编译源程序，在运行make install，最后运行make clean删除一些临时文件。运行configure脚本，就可以生成Makefile文件，然后就可以运行make、make install和make clean。

configure是一个shell脚本，它可以自动设定源程序以符合各种不同平台上Unix系统的特性，并且根据系统叁数及环境产生合适的Makefile文件或是C的头文件(header file)，让源程序可以很方便地在这些不同的平台上被编译连接。

这时，运行configure脚本，就可产生出符合GNU规范的Makefile文件了：

$ ./configure

到此时，就可以运行make进行编译，在运行make install进行安装了，最后运行make clean删除临时文件。

$ make

$ make install (注：运行这个要有足够的权限)

$ make clean

#./configure && make && make install

利用configure所产生的Makefile文件有几个预设的目标可供使用，其中几个重要的简述如下：

make all：产生我们设定的目标，即此范例中的可执行文件。只打make也可以，此时会开始编译原始码，然后连结，并且产生可执行文件。

make clean：清除编译产生的可执行文件及目标文件(\*.o，object file)。

make distclean：除了清除可执行文件和目标文件外，把configure所产生的Makefile也清除掉。

make install：将程序安装至系统中。如果原始码编译无误，且执行结果正确，便可以把程序安装至系统预设的可执行文件存放路径。如果用bin\_PROGRAMS宏的话，程序会被安装至/usr/local/bin这个目录。

make dist：将程序和相关的档案包装成一个压缩文件以供发布。执行完在目录下会产生一个以PACKAGE-VERSION.tar.gz为名称的文件。 PACKAGE和VERSION这两个变数是根据configure.in文件中AM\_INIT\_AUTOMAKE(PACKAGE，VERSION)的定义。在此范例中会产生test-1.0.tar.gz的档案。

make distcheck：和make dist类似，但是加入检查包装后的压缩文件是否正常。这个目标除了把程序和相关文件包装成tar.gz文件外，还会自动把这个压缩文件解开，执行 configure，并且进行make all 的动作，确认编译无误后，会显示这个tar.gz文件可供发布了。

这个检查非常有用，检查过关的包，基本上可以给任何一个具备GNU开发环境-的人去重新编译。

关于makefile，请参考，和我一起写makefile，这里不再赘述。这里要重点说一下configure这个脚本的重要参数，下面对每一个选项进行简略的介绍：

--cache-file=FILE

'configure' 会在你的系统上测试存在的特性(或者bug!)。为了加速随后进行的配置，测试的结果会存储在一个cache file里。当configure每一个子树里都有'configure'脚本的复杂的源码树时，一个很好的cache file的存在会对编译有很大帮助。

--no-create

'configure'中的一个主要函数会制作输出文件。此选项阻止'configure'生成这个文件。你可以认为这是一种演习(dry run)，尽管缓存(cache)仍然被改写了。

--quiet

--silent

当 'configure'进行他的测试时，会输出简要的信息来告诉用户正在作什么。这样作是因为'configure'可能会比较慢，没有这种输出的话用户将会被扔在一旁疑惑正在发生什么，使用这两个选项中的任何一个都会把你扔到一旁。(译注：这两句话比较有意思，原文是这样的：If there was no such output, the user would be left wondering what is happening. By using this option, you too can be left wondering!)

--prefix=PEWFIX

'--prefix'是最常用的选项。制作出的'Makefile'会查看随此选项传递的参数，当一个包在安装时可以彻底的重新安置他的结构独立部分。

举一个例子，当安装一个包，例如说Emacs，下面的命令将会使Emacs Lisp file被安装到"/opt/gnu/share"：

$ ./configure --prefix=/opt/gnu

很多的选项，个人认为，你可以忽略其他的一切，但请把—prefix加上。这里以安装supersparrow-0.0.0为例，我们打算把它安装到目录 /usr/local/supersparrow,于是在supersparrow-0.0.0目录执行带选项的脚本./configure --prefix=/usr/local/supersparrow,执行成功后再编译、安装（make，make install）；安装完成将自动生成目录supersparrow,而且该软件所有的文件都被复制到这个目录。为什么要指定这个安装目录？是为了以后的维护方便，如果没有用这个选项，安装过程结束后，该软件所需的软件被复制到不同的系统目录下，很难弄清楚到底复制了那些文件、都复制到哪里去了—基本上是一塌糊涂。

用了—prefix选项的另一个好处是卸载软件或移植软件。当某个安装的软件不再需要时，只须简单的删除该安装目录，就可以把软件卸载得干干净净；移植软件只需拷贝整个目录到另外一个机器即可（相同的操作系统）。

一个小选项有这么方便的作用，建议在实际工作中多多使用。

不指定prefix，则

可执行文件默认放在/usr/local/bin，

库文件默认放/usr/local/lib，

配置文件默认放在 /usr/local/etc。

其它的资源文件放在/usr/local/share。

你要卸载这个程序，要么在原来的make目录下用一次make uninstall（前提是make文件指定过uninstall）,要么去上述目录里面把相关的文件一个个手工删掉。

指定prefix，直接删掉一个文件夹就够了。

--exec-prefix=EPREFIX

与'--prefix'选项类似，但是他是用来设置结构依赖的文件的安装位置，编译好的'emacs'二进制文件就是这样一个文件。如果没有设置这个选项的话，默认使用的选项值将被设为和'--prefix'选项值一样。

--bindir=DIR

指定二进制文件的安装位置，这里的二进制文件定义为可以被用户直接执行的程序。

--sbindir=DIR

指定超级二进制文件的安装位置。这是一些通常只能由超级用户执行的程序。

--libexecdir=DIR

指定可执行支持文件的安装位置。与二进制文件相反，这些文件从来不直接由用户执行，但是可以被上面提到的二进制文件所执行。

--datadir=DIR

指定通用数据文件的安装位置。

--sysconfdir=DIR

指定在单个机器上使用的只读数据的安装位置。

--sharedstatedir=DIR

指定可以在多个机器上共享的可写数据的安装位置。

--localstatedir=DIR

指定只能单机使用的可写数据的安装位置。

--libdir=DIR

指定库文件的安装位置。

--includedir=DIR

指定C头文件的安装位置。其他语言如C++的头文件也可以使用此选项。

--oldincludedir=DIR

指定为除GCC外编译器安装的C头文件的安装位置。

--infodir=DIR

指定Info格式文档的安装位置.Info是被GNU工程所使用的文档格式。

--mandir=DIR

指定手册页的安装位置。

--srcdir=DIR

这个选项对安装没有作用，他会告诉'configure'源码的位置。一般来说不用指定此选项，因为'configure'脚本一般和源码文件在同一个目录下。

--program-prefix=PREFIX

指定将被加到所安装程序的名字上的前缀。例如，使用'--program-prefix=g'来configure一个名为'tar'的程序将会使安装的程序被命名为'gtar'。当和其他的安装选项一起使用时，这个选项只有当他被`Makefile.in'文件使用时才会工作。

--program-suffix=SUFFIX

指定将被加到所安装程序的名字上的后缀。

--program-transform-name=PROGRAM

这里的PROGRAM是一个sed脚本。当一个程序被安装时，他的名字将经过`sed -e PROGRAM'来产生安装的名字。

--build=BUILD

指定软件包安装的系统平台。如果没有指定，默认值将是'--host'选项的值。

--host=HOST

指定软件运行的系统平台。如果没有指定。将会运行`config.guess'来检测。

--target=GARGET

指定软件面向(target to)的系统平台。这主要在程序语言工具如编译器和汇编器上下文中起作用。

如果没有指定，默认将使用'--host'选项的值。

--disable-FEATURE

一些软件包可以选择这个选项来提供为大型选项的编译时配置，例如使用Kerberos认证系统或者一个实验性的编译器最优配置。

如果默认是提供这些特性，可以使用'--disable-FEATURE'来禁用它，这里'FEATURE'是特性的名字，例如：

$ ./configure --disable-gui

--enable-FEATURE[=ARG]

相反的，一些软件包可能提供了一些默认被禁止的特性,可以使用'--enable-FEATURE'来起用它。

这里'FEATURE'是特性的名字。一个特性可能会接受一个可选的参数。例如：

$ ./configure --enable-buffers=128

`--enable-FEATURE=no'与上面提到的'--disable-FEATURE'是同义的。

--with-PACKAGE[=ARG]

在 自由软件社区里，有使用已有软件包和库的优秀传统。当用'configure'来配置一个源码树时，可以提供其他已经安装的软件包的信息。例如，倚赖于 Tcl和Tk的BLT器件工具包。要配置BLT，可能需要给'configure'提供一些关于我们把Tcl和Tk装的何处的信息：

$ ./configure --with-tcl=/usr/local --with-tk=/usr/local

'--with-PACKAGE=no'与下面将提到的'--without-PACKAGE'是同义的。

--without-PACKAGE

有时候你可能不想让你的软件包与系统已有的软件包交互。例如，你可能不想让你的新编译器使用GNU ld。通过使用这个选项可以做到这一点：

$ ./configure --without-gnu-ld

--x-includes=DIR

这个选项是'--with-PACKAGE'选项的一个特例。在Autoconf最初被开发出来时，流行使用'configure'来作为 Imake的一个变通方法来制作运行于X的软件。'--x-includes'选项提供了向'configure'脚本指明包含X11头文件的目录的方法。

--x-libraries=DIR

类似的，'--x-libraries'选项提供了向'configure'脚本指明包含X11库的目录的方法。

在源码树中运行'configure'是不必要的同时也是不好的。一个由'configure'产生的良好的'Makefile'可以构筑源码属于另一棵树的软件包。在一个独立于源码的树中构筑派生的文件的好处是很明显的：派生的文件，如目标文件，会凌乱的散布于源码树。这也使在另一个不同的系统或用不同的配置选项构筑同样的目标文件非常困难。建议使用三棵树：一棵源码树(source tree)，一棵构筑树(build tree)，一棵安装树(install tree)。这里有一个很接近的例子，是使用这种方法来构筑GNU malloc包：

具体编译如下：

./configure --prefix=/usr/local/apache2 --enable-rewrite --enable-so --enable-headers --enable-expires --with-mpm=worker --enable-modules=most --enable-deflate --prefix=/usr/local/apache2表示指定apache的安装路径，默认安装路径为/usr/local/apache2

--enable-rewrite提供URL规则的重写，即根据已知的URL地址，转换为其它想要的URL地址

--enable-so激活apache服务的DSO（Dynamic Shared Objects动态共享目标），即在以后可以以DSO的方式编译安装共享模块，这个模块本身不能以DSO方式编译。

--enable-headers提供允许对HTTP请求头的控制。

--enable-expires激活通过配置文件控制HTTP的“Expires:”和“Cache-Control:”头内容，即对网站图片、js、css等内容，提供客户端浏览器缓存的设置。这个是apache调优的一个重要选项之一。

--with-mpm=worker选择apache mpm的模式为worker模式。因为worker模式是更多的使用线程来处理请求，所以可以处理更多的并发请求。而系统资源的开销小玉基于进程的MPM prefork。如果不指定此参数，默认的模式是prefork进程模式。这个是apache调优的一个重要选项之一。

--enable-deflate提供对内容的压缩传输编码支持，一般是html、js、css等内容的站点。使用此参数会大大提高传输速度，提升访问者访问的体验。在生产环境中，这是apache调优的一个重要选项之一

# RPM包安装和删除

-ivh 获得一个详细的安装进程

--nodeps 忽略依赖关系

--force 强制安装

-U 包名:升级安装

-F 更新不管是什么直接覆盖

--replacepkgs这样 RPM 将忽略该错误信息 -vih --replacepkgs

-q name :查询

-qa查询所有已安装的RPM

-qi name 查询这个软件详细信息

-ql 显示列表

-qf path :查询一个文件属于哪个包

-qp 包名 :查文件是什么

-qpl 包名 :针对一个没装过的包

rpm -qc 配置文件

效验 rpm -Va

rpm -Vf /bin/ls

rpm -Vp 包名

5 -- MD5 校验和

S -- 文件长度

L -- 符号链接

T -- 文件修改日期

D -- 设备

U -- 用户

G -- 用户组

M -- 模式 (包含许可和文件类型)

? -- 不可读文件

rpm -qpl xxxxxx.rpm

1.如何安装rpm软件包

rmp软件包的安装能够使用程式rpm来完成。执行下面的命令

rpm -i your-package.rpm

其中your-package.rpm是您要安装的rpm包的文档名，一般置于当前目录下。

安装过程中可能出现下面的警告或提示：

... conflict with ... 可能是要安装的包里有一些文档可能会覆盖现有

的文档，缺省时这样的情况下是无法正确安装的能够用

rpm --force -i 强制安装即可

... is needed by ...

... is not installed ... 此包需要的一些软件您没有安装能够用

rpm --nodeps -i 来忽略此信息

也就是说，rpm -i --force --nodeps 能够忽略任何依赖关系和文档问题，什么包

都能安装上，但这种强制安装的软件包不能确保完全发挥功能

2.如何安装.src.rpm软件包

有些软件包是以.src.rpm结尾的，这类软件包是包含了源代码的rpm包，在安装时

需要进行编译。这类软件包有两种安装方法，

方法一：

1.执行rpm -i your-package.src.rpm

2. cd /usr/src/redhat/SPECS

3. rpmbuild -bp your-package.specs 一个和您的软件包同名的specs文档

4. cd /usr/src/redhat/BUILD/your-package/ 一个和您的软件包同名的目录

5. ./configure 这一步和编译普通的源码软件相同，能够加上参数

6. make

7. make install

方法二:

1.执行rpm -i you-package.src.rpm

2. cd /usr/src/redhat/SPECS

前两步和方法一相同

3. rpmbuild -bb your-package.specs 一个和您的软件包同名的specs文档

这时，在/usr/src/redhat/RPM/i386/ （根据具体包的不同，也可能是i686,noarch等等)

在这个目录下，有一个新的rpm包，这个是编译好的二进制文档。

执行rpm -i new-package.rpm即可安装完成。

3.如何卸载rpm软件包

使用命令 rpm -e 包名，包名能够包含版本号等信息，但是不能够有后缀.rpm

比如卸载软件包proftpd-1.2.8-1，能够使用下列格式：

rpm -e proftpd-1.2.8-1

rpm -e proftpd-1.2.8

rpm -e proftpd-

rpm -e proftpd

不能够是下列格式：

rpm -e proftpd-1.2.8-1.i386.rpm

rpm -e proftpd-1.2.8-1.i386

rpm -e proftpd-1.2

rpm -e proftpd-1

有时会出现一些错误或警告：

... is needed by ... 这说明这个软件被其他软件需要，不能随便卸载

能够用rpm -e --nodeps强制卸载

4.如何不安装但是获取rpm包中的文档

使用工具rpm2cpio和cpio

rpm2cpio xxx.rpm | cpio -vi

rpm2cpio xxx.rpm | cpio -idmv

rpm2cpio xxx.rpm | cpio --extract --make-directories

参数i和extract相同，表示提取文档。v表示指示执行进程

d和make-directory相同，表示根据包中文档原来的路径建立目录

m表示保持文档的更新时间。

5.如何查看和rpm包相关的文档和其他信息

下面任何的例子都假设使用软件包mysql-3.23.54a-11

1.我的系统中安装了那些rpm软件包

rpm -qa 讲列出任何安装过的包

假如要查找任何安装过的包含某个字符串sql的软件包

rpm -qa |grep sql

2.如何获得某个软件包的文档全名

rpm -q mysql 能够获得系统中安装的mysql软件包全名，从中能够获得

当前软件包的版本等信息。这个例子中能够得到信息mysql-3.23.54a-11

3.一个rpm包中的文档安装到那里去了？

rpm -ql 包名

注意这里的是不包括.rpm后缀的软件包的名称

也就是说只能用mysql或mysql-3.23.54a-11而不是mysql-3.23.54a-11.rpm。

假如只是想知道可执行程式放到那里去了，也能够用which，比如

which mysql

4.一个rpm包中包含那些文档

一个没有安装过的软件包，使用rpm -qlp \*\*\*\*.rpm

一个已安装过的软件包，还能够使用rpm -ql \*\*\*\*.rpm

5.如何获取关于一个软件包的版本，用途等相关信息？

一个没有安装过的软件包，使用rpm -qip \*\*\*\*.rpm

一个已安装过的软件包，还能够使用rpm -qi \*\*\*\*.rpm

6.某个程式是哪个软件包安装的，或哪个软件包包含这个程式

rpm -qf `which 程式名` 返回软件包的全名

rpm -qif `which 程式名` 返回软件包的有关信息

rpm -qlf `which 程式名` 返回软件包的文档列表

注意，这里不是引号，而是`，就是键盘左上角的那个键。

也能够使用rpm -qilf，同时输出软件包信息和文档列表

7.某个文档是哪个软件包安装的，或哪个软件包包含这个文档

注意，前一个问题中的方法，只适用和可执行的程式，而下面的方法，不但能够

用于可执行程式，也能够用于普通的任何文档。前提是知道这个文档名。

首先获得这个程式的完整路径，能够用whereis或which，然后使用rpm -qf例如：

# whereis ftptop

ftptop: /usr/bin/ftptop /usr/share/man/man1/ftptop.1.gz

# rpm -qf /usr/bin/ftptop

proftpd-1.2.8-1

# rpm -qf /usr/share/doc/proftpd-1.2.8/rfc/rfc0959.txt

proftpd-1.2.8-1

总结：

获得软件包相关的信息用rpm -q，q表示查询query，后面能够跟其他选项，比如

i 表示info，获得软件包的信息；

l 表示list，获得文档列表；

a 表示all，在任何包中执行查询；

f 表示file，根据文档进行相关的查询；

p 表示package，根据软件包进行查询

需要的查询条件能够使用grep产生，或从"` `"中的命令行产生

6.关于rpm软件包的一些相关知识

1.什么是rpm

rpm 即RedHat Package Management，是RedHat的发明之一

2.为什么需要rpm

在一个操作系统下，需要安装实现各种功能的软件包。这些软件包一般都有各自的

程式，但是同时也有错综复杂的依赖关系。同时还需要解决软件包的版本，连同安装，

配置，卸载的自动化问题。为了解决这些问题，RedHat针对自己的系统提出了一个

较好的办法来管理成千上百的软件。这就是RPM管理系统。在系统中安装了rpm管理系统

以后，只要是符合rpm文档标准的打包的程式都能够方便的安装，升级，卸载

3.是不是任何的linux都使用rpm

任何系统都需要包管理系统，因此很多linux都使用rpm系统。但rpm系统是为RH专门

但是TL,Mandrake等系统也都使用rpm。由于rpm的源程式能够在别的系统上进行编译，

所以有可能在别的系统上也使用rpm

除了rpm，其他一些系统也有自己的软件包管理程式，例如debian的deb包，

slakware也都有自己的包管理系统

4.rpm包的文档名为什么那么长

rpm包的文档名中包含了这个软件包的版本信息，操作系统信息，硬件需要等等。

比如mypackage-1.1-2TL.i386.rpm，其中mypackage是在系统中登记的软件包的名字

1.1是软件的版本号，2是发行号，TL表示用于TL操作系统，还可能是RH等。i386表示

用于intel x86平台，还可能是sparc等。

5.软件包文档名中的i386,i686是什么意思

rpm软件包的文档名中，不但包含了软件名称，版本信息，还包括了适用的硬件架构

的信息。

i386指这个软件包适用于intel 80386以上的x86架构的电脑(AI32)

i686指这个软件包适用于intel 80686以上(奔腾pro以上)的x86架构的电脑(IA32)

noarch指这个软件包于硬件架构无关，能够通用。

i686软件包的程式通常针对CPU进行了优化，所以，向后兼容比较用以，i386的包在

x86机器上都能够用。向前一般不兼容。但是现在的电脑，奔腾pro以下的CPU已很少

用，通常配置的机器都能够使用i686软件包

6.不同操作系统发行的rpm包可否混用？

对于已编译成二进制的rpm包，由于操作系统环境不同，一般不能混用。

对于以src.rpm发行的软件包，由于需要安装时进行本地编译，所以通常能够在不同

系统下安装。

7.使用rpm时碰到的一些特别问题

Q 我用rpm -e \*\*.rpm无法删除rpm包

A 包名不要包括rpm，

rpm -e 包名，能够包含版本号等信息，但是不能够有后缀.rpm

Q 在MS的系统下有没有读RPM文档的工具？

A wincmd with rpm plugins.....

Q 是否能够通过ftp安装安装升级rpm包？

A 能够。rpm -ivh

ftp://xxxxxxxx/PATH2SomeRPM

Q rpm安装时已有的包版本过高怎么办？

A 有时由于安装的软件包太老，而系统中相关的软件包版本比较新，所以可能需要

安装的包依赖的一些文档会很难找到。这时有两种解决办法，

第一是在系统文档中找到和需要的文档功能相同或相似的文档，做一个符号链接到

需要的目录下。

# 系统服务之RPM安装的独立服务

centos系统服务和windows服务从系统的角度来说差不多，但从用户的角度来说，还是有一些区别。服务分为两大类：

RPM包安装的独立服务和xinetd服务，

源代码安装服务

首先，Windows不会区分服务源，也不会区分安装位置，因为都一样。

而linux却不一样，从安装源上来说，有RPM包和源码包之分，因为这俩种源会被系统安装到不同的位置，通常RPM包由软件设计者决定安装目录，通常都是一些约定俗成的目录，但也不是一定的，因为它是由软件设计者决定的。

例如：

ssh服务，他的配置文件会被安装在/etc/ssh/sshd\_config，而他的启动脚本被放在/etc/rc.d/init.d/,

而通常执行这个服务，有俩个方式，其实都是用绝对路径实现的，一个是service 另一个是直接敲绝对路径

service sshd start

/etc/rc.d/init.d/sshd start 或者 /etc/init.d/sshd start 后者是前者的一个软连接，就是说这两个目录的内容是一样的。

其实这三个命令行都是一个意思，都是去/etc/rc.d/init.d这个目录下执行sshd这个脚本服务

大家注意一下，像这种ssh服务，其实是RPM安装的独立服务，还有一种基于xinetd的服务也是通过RPM包安装的，这个先不介绍，知道一下就好。

其次，就是用源码包得方式安装的服务，这个就可以根据用户的需要安装到任意目录，通常的规则是安装到/usr/local目录下，当然你可以指定任意目录。这里也不介绍了。

这里重点说一下用RPM安装的独立服务，启动方法前面已经介绍了，这里重点分两种方式再详细说一下，一种是手动启动，一种是自启动。

手动启动就是前边说的那种。参数主要有 start stop status restart

自动启动，这个要重点说一下，不是每一种服务都要服务器自动启动，这个

要根据需要，这一点和windows比较类似

关于这个status 用法说一下，这是一查看服务状态的参数

service sshd status //查看指定服务运行状态

service --status-all //查看所有服务运行状态

查看系统安装了那些RPM包安装服务，用chkconfig --list

注意这个命令只能查看RPM包安装的独立服务和xinetd服务，不能查看源代码安装服务

既然能查看，那显然就可以设置这些服务的自启动

chkconfig --level 2345 sshd [on/off]

通过这样的方式就可以设置sshd的是否自动启动，on表示开机启动，off表示关闭，2345表示启动级别，其实你不加这个-level 2345 也是一样的，应为chkconfig默认参数就是这个

centos中还提供了一个图形设置服务命令，就是ntsysv，这个和setup差不多，就是利用图形界面来设置服务，这个不做重点介绍，因为这个命令不是所有linux系统中都有的。

最后说一个方式，也是所有linux都支持的服务的自启动方法，也是本文介绍的重点，就是直接修改系统自动启动脚本，这个文件位于/etc/rc.d/rc.local，他还有一个软连接/etc/rc.local，

这个和上边的独立服务脚本位置一样，两个目录或者文件内容一样，你修改那个都可以。

#!/bin/sh

#

# This script will be executed \*after\* all the other init scripts.

# You can put your own initialization stuff in here if you don't

# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local

/opt/lampp/lampp start

上边就是这个文件的内容，你可以把你要添加的内容写在文件最后边，就可以了，注意一下文件第6行，那个touch命令，很多人对这个命令不理解，难道，每次系统启动，都要创建一个文件不成，touch命令的英文意思是摸、触碰的意思，并不是创建，它是发现这个文件不存在，则创建该文件，如果该文件存在，它只是去修改它的时间戳而已。

第7行，就是我们自己加入的独立服务，希望系统重启后，启动lampp服务，来运行discuz，呵呵

# 系统服务之基于xinetd的服务

接着上一节，来看一下，RPM包安装的服务中得另一种基于xinetd的服务。

这种基于xinetd的服务，实际上随着linux系统的升级，出于安全方面的考虑，基本上应用已经非常少了，目前还能看到的有vsFTPd、telnet等，但这些东西的安全性都不好，所以都已经逐渐退出了历史舞台，比如作为远程连接的telnet，目前基本上就是用来测试一下23号端口，已经不再作为远程控制的工具，而由更安全的ssh来替代。

但作为linux系统中还存在的服务，有必要，介绍性的说明一下

就以telnet为例：

大家应该发现一个问题，就是我们用chkconfig --list查看RPM包安装的服务时，并没有出现基于xinetd的服务，那是因为，这货压根就没有被安装，而且telnet也没有安装

为了给大家演示这个服务，我们需要手动安装这两个RPM包

yum -y install xinetd

yum -y install telnet-server

安装后再运行chkconfig --list你就会发现服务列表的最后已经出现了基于xinetd的服务，并且还有telnet服务，并且是出于关闭状态的，

运行netstat -tlun 查看一下系统打开的端口，telnet的23号端口是被关闭的。

那么启动这个服务，和独立服务不同，你不能用service telnet start，因为这个服务只能通过xinetd来启动，也就是要求telnet服务的配置文件中是启动选项是开启状态，然后你重启xinetd这个服务，那么telnet就会启动了，

首先编辑telnet服务配置文件vi /etc/xinetd.d/telnet

# default: on

# description: The telnet server serves telnet sessions; it uses \

# unencrypted username/password pairs for authentication.

service telnet

{

flags = REUSE

socket\_type = stream

wait = no

user = root

server = /usr/sbin/in.telnetd

log\_on\_failure += USERID

disable = no

}

前边东西不解释了，就看最后一行的disable，yes就是关闭服务，no就是启动服务。

然后重启xinetd服务，service xinetd restart

这时你会发现，chkconfig --list 中telnet服务已经开启，netstat -tlun中23号端口也打开了，要想关闭这个服务就执行相反的操作就可以了。

那么如何让这个服务自启动呢，chkconfig telnet on 或者用ntsysv

这里边有一个非常有意思的问题就是，当你执行chkconfig telnet on后，你会发现你不用重启xinetd服务，telnet服务已经开启，而且23端口也被打开了，而你执行chkconfig telnet off后发现telnet服务被关闭，而且端口也关闭，

也就是说这货根本就没有什么启动和自动启动之分，都一样，这显然不符合现代管理服务器服务的需要，所以这货就要被淘汰了。

好了

基于xinetd的服务，就给大家介绍到这了。

# 系统服务之基于源代码的服务

前面介绍了，关于RPM包的服务，包括独立服务和基于xinetd的服务，那么下面介绍基于源码包的服务，

以apache为例，

通常apache的默认安装目录/usr/local/apache2，那么我们如何启动它呢，其实这个和rpm包并没有什么区别，都是直接找到启动脚本来启动，那么我们是如何知道这个启动脚本的名称呢，这个就直接看系统的安装介绍就可以了，就是install readme一类的文件，通过查询帮助文档，我们知道apache的启动脚本为

/usr/local/apache2/bin/apachectl start|stop

然后，自启动就和前边的东西差不多了，同样是把启动脚本添加到/etc/rc.d/rc.local或者/etc/rc.local文件中，就是上边的语句。

这里非常有必要强调的一点，就是，前边我们说的service方式和chkconfig，默认情况是不能用于基于源码包的服务的，但是了解了基于rpm包原理后，其实，要想让这两个命令实现基于源码包服务的管理，也是没有问题的，

service就是调用/etc/rc.d/init.d/中得启动脚本，那么你可以用创建软连接的方式，把软连接直接丢到/etc/rc.d/init.d或者/etc/init.d/中，就可以了

ln -s /usr/local/apache2/bin/apachectl /etc/rc.d/init.d/apache

这样你就可以直接使用service来直接启动apache了

service apache start/stop/restart

同样，chkconfig也可以实现，但显然chkconfig和ntsysv的原理与service不同，那我们看一下有哪些不同，要想使用chkconfig和ntsysv我们还需要改动刚才的apache软连接脚本，让chkconfig可以识别：

# chkconfig: 35 86 76

# description: source package apache 2.4.12

把它们添加到#!/bin/sh下面一行就可以了

注意一下，很多人认为，这不是两条注释语句吗，有什么用，这就是chkconfig的不同之处，你必须添加这两行语句

注意他们的格式和语法，

chkconfig: 后边的35 表示启动级别，你可以直接写2345，就是自启动了。

86表示脚本启动顺序号，76表示脚本关闭顺序号，这个只要不和已有脚本冲突就可以，

可以到相应的/etc/rc.d/rc3.d 和 /etc/rc.d/rc5.d中查询，看看有没有脚本已经占用了，查看对应的级别rc.d就可以了。

# description:这个是保留字，就是这个脚本的服务描述，后边的内容可以自定义 例如：source package apache 2.4.12

这样，chkeconfig就可以识别apache了，

但你现在还不能直接使用，你需要首先把apache加入到chkconfig列表中，

chkconfig --add apache

这时，运行chkconfig -- list |grep apache

就可以查到了，启动级别是35， 如果要自启动，可以运行

chkconfig --level 2345 apache on|off 来开启或者关闭自启动

前边说了添加显然你也可以把apache从chkconfig列表中删除，运行

chkconfig --del apache

就可以删除apache服务了，当然如果要彻底删除，还要把我们最前边添加的脚本内容也一起删除。

ntsysv就不介绍了，因为，他和chkconfig的工作模式是一样的，当你把apache服务添加到chkconfig列表时，你就可以在他的列表中看到apache了。

# Discuz无法安装的那些破事

虽然有教程，不过，由于好多同学对一些参数一知半解，盲目跟随教程，所以，出现了问题，却无法解决，后来就把一些教程重新研究了一下，发现了很多问题。

首先，centos，apache ，mysql，php，这些源码安装都没有问题，问题就出现在他们的配置文件上，centos这里不再赘述，这里主要说一下，

apache的httpd.conf，mysql的my.cnf，和php的php.ini

说句实话，网上的那些所谓教程和经验，真的不怎么样，好多东西，都有问题，比如一个mysql.sock就是这样，这个文件你就是手动建一个都行，只要给出路径就好了，但前提是你必须真的给出它的实际路径。

还有关于apache目录权限的问题，以及mysql和php的一些安全问题，后边再说，先来说一下，我们遇到的一个非常棘手的问题，就是php不能调用mysql数据库，但phpinfo()却可以正常显示，mysql的信息。而我们也可以正常登陆mysql数据库，那么最后问题就是被定为在2个方向上，一个是目录权限，另一个就是配置文件了，最让我恶心的就是还按照教程，把php错误提示给关闭了，一片空白呀，你让我如何处理呢，所以有些自以为是的教程，本身都是漏洞百出，这样操作，你就死定了。正确的顺序应该是打开这些提示功能还有函数，等系统稳定后，再根据需要进行选择。

[httpd.conf]

#服务器主目录

ServerRoot "/usr/local/apache2"

#侦听端口

Listen 80

#注意一下这三行，首先是支持静态读写，然后是支持php，后边是支持php文件类型

LoadModule rewrite\_module modules/mod\_rewrite.so

LoadModule php5\_module modules/libphp5.so

AddType application/x-httpd-php .php

<IfModule unixd\_module>

#apache用户和组，这个是系统默认的，通常不需要更改，

User daemon

Group daemon

</IfModule>

ServerAdmin you@example.com

#设置服务器域名和端口

ServerName 192.168.195.3:80

#设定目录访问对象

<Directory />

AllowOverride none

Require all denied

</Directory>

#设定apache的web目录

DocumentRoot "/usr/local/apache2/htdocs"

<Directory "/usr/local/apache2/htdocs">

#Options Indexes FollowSymLinks

#禁止显示目录结构

Options FollowSymLinks

AllowOverride All

Require all granted

</Directory>

#添加支持index.php

<IfModule dir\_module>

DirectoryIndex index.html index.php

</IfModule>

<Directory "/usr/local/apache2/cgi-bin">

AllowOverride All

Options None

Require all granted

</Directory>

[my.conf]

[mysqld]

datadir=/usr/local/mysql/data

default-storage-engine=MyISAM

log-error=/usr/local/mysql/data/error.log

pid-file=/usr/local/mysql/data/mysql.pid

user=mysql

#tmpdir=/tmp

# Remove leading # and set to the amount of RAM for the most important data

# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.

# innodb\_buffer\_pool\_size = 128M

# Remove leading # to turn on a very important data integrity option: logging

# changes to the binary log between backups.

# log\_bin

# These are commonly set, remove the # and set as required.

# basedir = .....

# datadir = .....

port = 3306

# server\_id = .....

socket = /usr/local/mysql/mysql.sock

# Remove leading # to set options mainly useful for reporting servers.

# The server defaults are faster for transactions and fast SELECTs.

# Adjust sizes as needed, experiment to find the optimal values.

# join\_buffer\_size = 128M

# sort\_buffer\_size = 2M

# read\_rnd\_buffer\_size = 2M

sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES

[php.ini]

;;;;;;;;;;;;;;;;;;;

; About php.ini ;

;;;;;;;;;;;;;;;;;;;

; PHP's initialization file, generally called php.ini, is responsible for

; configuring many of the aspects of PHP's behavior.

打开短目录访问，就是<?php ?>可以被识别。

short\_open\_tag = On

就是这行代码，有问题，导致无法安装继续，其实就是尼玛少加了双引号

open\_basedir = ".:/tmp/"

禁用的一些函数可以写在下面，用以提高安全性，

disable\_functions =

显示PHP版本

expose\_php = On

打开错误提示

display\_errors = On

关于目录权限其实是一个很简单的问题，但开始时忽略了，想当然的认为，我是root用户还有什么不可以的吗，其实这就是头脑发热的结果，尼玛如果这些东西都是root的，你让其他用户情以何堪呀，显然是把选项给搞反了。而且，仔细看过上面apache的配置文件的同学其实早就发现了，它默认的用户和组，其实ceontos中早就已经创建了，而且这个用户和组，为了安全，千万不要改为root，这也是用户目录需要设置权限的原因。应为所有web用户都是通过系统伪用户daemon登陆的，所以这个根目录必须要给daemon绝对权限，如果像我原来，弄得是root的，daemon当然没有权限访问了，它要不空白就怪了。

chown daemon.daemon -R /usr/local/apache2/htdocs #更改目录所有者

chmod 700 /usr/local/apache2/htdocs -R #更改apache网站目录权限

大家发现了吗，如果你把权限设置成777，好多教程就是这么写的，那你就真的不用改所属权限了，但这样显然有悖于服务器的安全性。

所以上边两条语句是标准的权限设置，

基本上上就这点问题，大家一定注意一个问题，就是一定要把错误提示打开，否则，你真的不知道干什么。有了提示，其实，也就是分分钟就可以搞定的事了，哈哈哈哈。

# CentOS下源码编译安装apache

前面介绍服务种类的时候，我们介绍了通过源代码安装的服务，那么还没有真正介绍如何通过源代码安装，

那么，我们就以apache为例，给大家完整演示一下，基于源代码的服务安装，这部分内容，其实比服务本身的使用要难一些，下面就让我们一起来看看，

我们现在分以下步骤进行安装apache：

1、 安装编译环境

2、 卸载原有apache

3、 下载解压源码包

4、 安装apache

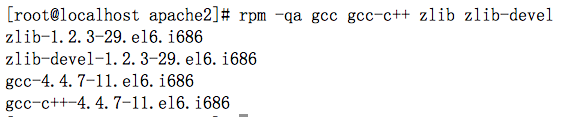
5、 测试apache

6、 查看apache安装生成的目录

7、 查看apache的配置文件

8、 apache加入系统服务

首先，我们是源代码安装，所以需要一个编译环境，基本上就是需要gcc gcc-c++ zlib zlib-devel这4个包，我们先查询一下



这个是我们已经安装好了的环境，他们当然都已经存在了，如果他们不存在，那么就需要安装它，执行yum -y install gcc gcc-c++ zlib zlib-devel

然后查询一下，如果都能查到，就表示安装好了。

第2步，我们要先卸载原来已经安装的apache在源码安装apache之前，我们要先卸载系统中，已经通过rpm包安装的apache。如下：

rpm -qa |grep httpd

rpm -e --nodeps httpd-2.2.15-29.el6.centos.x86\_64

rpm -e --nodeps httpd-tools-2.2.15-29.el6.centos.x86\_64

其中，-e参数是自动查找依赖关系，--nodeps是强制删除，不考虑其他软件包的依赖。

由于我们是新安装的centos，没有安装过apache，所以这个就没有了

第3步，我们就要获取apache的源码包

这个源码包通常我们直接到apache网站下载就可以了，http://httpd.apache.org/download.cgi

也可以用http://apache.fayea.com/httpd/httpd-2.4.12.tar.gz

如果网络允许，你也可以这样wget http://apache.fayea.com/httpd/httpd-2.4.12.tar.gz

我这里下载的是bz2版，因为它更小

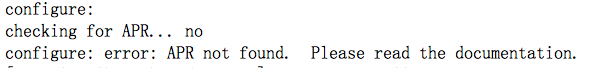
然后tar -jxvf httpd-2.4.12.tar.bz2 -C /tmp

第4步，安装apache

然后，进入cd /tmp/thhpd-2.4.12

执行./configure --prefix=/usr/local/apache2 && make && make install

这个时候通常会报错，提示你缺少apache实时运行环境apr



关于这个apr运行环境，就像你运行java是的java运行环境和dircetx运行实库差不多。

apr下载地址： http://apr.apache.org/download.cgi

pcer下载地址：http://jaist.dl.sourceforge.net/project/pcre/pcre/

大家注意一下上面的地址和下边的地址略有不同，只是地方不一样而已，但都是官方的源代码，到哪里下载都可以

wget http://apache.fayea.com/apr/apr-1.5.1.tar.bz2

wget http://apache.fayea.com/apr/apr-util-1.5.4.tar.bz2

wget http://jaist.dl.sourceforge.net/ ... 2/pcre-8.32.tar.bz2

取得源码包后，就可以安装这个apr运行环境了，

首先，解压源码包到指定目录tmp

tar -jxvf apr-1.5.1.tar.bz2 -C /tmp

tar -jxvf apr-util-1.5.4.tar.bz2 -C /tmp

tar -jxvf pcre-8.32.tar.bz2 -C /tmp

然后，清除原来安装的环境

yum remove apr-util-devel apr apr-util-mysql apr-docs apr-devel apr-util apr-util-docs

然后就可以到tmp目录下，各自的目录中分别安装了：

[root@localhost apr-1.5.1]# ./configure --prefix=/usr/local/apr && make && make install

[root@localhost apr-util-1.5.4]# ./configure --prefix=/usr/local/apr-util --with-apr=/usr/local/apr/bin/apr-1-config && make && make install

[root@localhost pcre-8.32]# ./configure --prefix=/usr/local/pcre &&make && make install

通常情况，如果你是按照我这个步骤来的，就不会有问题了，但很多人可能没有安装第1步的编译环境，那么你就需要安装gcc-c++这个编译器了。

这个时候，我们再执行apache的安装一般，就不会有问题了。

[root@localhost httpd-2.4.12]# ./configure --prefix=/usr/local/apache2 --with-apr=/usr/local/apr \

> --with-apr-util=/usr/local/apr-util/ \

> --with-pcre=/usr/local/pcre && make && make install

这样，我们自己官网上下载的apache源码包就被我们安装到了，我们自己指定的目录/usr/local/apache2这个目录中，其实你不指定这个目录，源码包也是默认安装到这个目录的，当然你也可以指定其他目录，而且你要删除apache时，只要把这个目录删除就可以了，不像RPM包那样，如果没有-e参数，删除一个RPM包服务，简直就是噩梦。

第5步，测试apach

这一步就其实很简单，本来没有什么可写的，就是直接在浏览器敲入你要测试服务器的地址就可以了，但实际情况可能不是这样，

当然首先你要启动apache服务，这个在INSTALL中已经说明了启动方法

/usr/local/apache2/bin/apachectl start|stop|restart

这时，大多的教程会告诉你在浏览器中键入127.0.0.1或者192.168.195.3（安装apache服务的服务器地址），就会显示

It works!

表示apache已经正常工作了，其实你一定会很失望的，就是通常，你都将的到一个无法连接的提示，这是为什么呢，

那是因为你的防火墙没有开80端口。

解答：vi /etc/sysconfig/iptables 在22端口那一行下面照抄一行，并将22改成80，保存。重启一下iptables就好了。

service iptables restart

关于第6步和第7步，查看apach安装目录和配置文件，这里不再赘述，

第8步 把apache加入系统服务

其实这一步，对于大多数用户来说，真的没有什么实际的意义，不过，为配合前面讲的系统服务的分类，这里给大家介绍一下，如何把我们自己安装的源码服务，加入到系统服务中，就是能通过service chkconfig ntsysv调用的服务，这部分内容大家可参考我关于系统服务的讲解既可以了，非常简单。

通常，我们都是把/usr/local/apache2/bin/apachectl start 这一行加入到/etc/rc.d/rc.local中，让系统自动启动，就可以了。

敬请期待，后面的mysql的源码安装，以及configure命令的参数详讲。

# CentOS下源码编译安装mysql

mysql5.6.16的安装和之前的5.5、5.1有些不同，编译的时候不再使用./configure来进行了，使用了cmake命令来进行编译项目。

从整个安装过程来看，前边的安装部分，和装apache都差不多，都是先准备好编译环境，然后就是直接安装，就可以了，但数据库的安装，因为其本身的一些特性，cmake参数比较复杂，另外，后边的配置，及使用，都需要你对linux有相当的了解，否则，这个源码的安装，它的复杂度，还是比较高的。

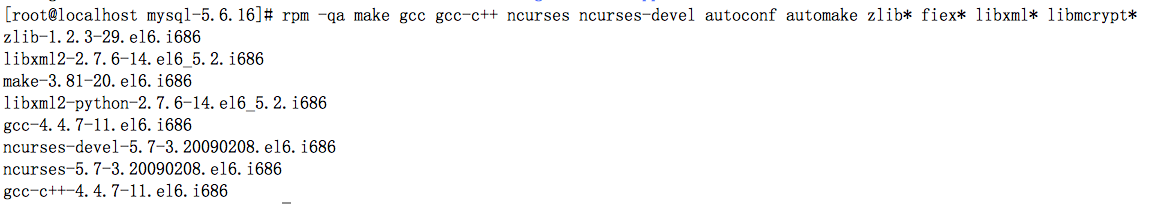
安装前准备：

1.安装所需的依赖包：make、gcc、gcc-c++、ncurses、ncurses-devel，这几个包是必须的，zlib、zlib-devel是否必须未测试，但我的环境由于先安装了apache，所以gcc、zlib、zlib-devel这些包都已经安装过了，

yum -y install make gcc gcc-c++ ncurses ncurses-devel bison bison-devel\

autoconf automake zlib\* fiex\* libxml\* libmcrypt\*

这里要说明一下，前边说的那几个必须的，你必须安装，后边的一些依赖包，是否必须，没有验证，我的系统目前只安装了下面的东西：



如果不放心就，全部安装吧，bison-devel居然没安装成功，去网上下载一个rpm包，要不报警告，看着不爽，就安装了。

rpm -ivh bison-devel

2.准备软件以下软件(本人存放于/mnt/share/tools/下)

cmake-3.2.2.tar.gz

mysql-5.6.24.tar.gz

关于cmake就没有啥好说的了，直接到官网下载，就好，倒是这个mysql源码，俺在mysql所谓的官网上找了1天都没有发现在哪里可以下载，而且还发现一个非常SB的问题，就是你下载的时候，居然会锁定你的操作系统，比如我在windows系统下要下载mac版的mysql居然切换不了，就是那个菜单，选完之后没有反应，因为是安装的centos最小版，没有地址无法下载，费了一番周折后，终于找到了他的下载地址，这里要批评一些mysql的网站，差评。

http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.16.tar.gz

就是这个地址，仔细看你会发现，中间多了一个get，呵呵，无语。。。。。。

准备工作结束，下面就开始安装：

第一步：mysql5.5以后都是通过cmake来编译的，所以要先安装cmake

shell# cd /mnt/share/tools/

shell# tar -zxf cmake-3.2.2.tar.gz -C /tmp #解压到指定目录/tmp/cmake-3.2.2下

shell# cd cmake-3.2.2

shell# ./configure && make && make install #这里好多教程都是用的./bootstrap; make; make install

来说一下，为什么不要用分号，分号只是顺序执行，而逻辑与是只有前边执行正确了，才执行后边的。

第二步：检查mysql环境

检查是否安装有mysql，CentOS6.6采用最小安装（minimal）的时候默认是会安装mysql的，

rpm -qa | grep -i mysql #i表示不区分大小写

如果发现系统已经安装了rpm版的mysql，我们首先要把它卸载了，如果是安装过源码版mysql，要升级，也需要把原来的卸载了，这个两种安装方式的mysql，其卸载方式是不一样的，这一点，大家要注意，这个很可能是比安装还要复杂的东西，希望大家注意。

这两种安装方式卸载mysql的共同点就是，首先，你要把mysql服务关闭，尤其对于源码安装的mysql，如果未关闭服务卸载，再次安装时，多数都会报错。

关闭mysql服务，本身很简单，你只要执行类似service mysql stop这样的命令就可以了

但是，我们并不建议这样关闭，因为这样的操作，很可能由于系统的原因，并没有关闭mysql服务，我们通常用杀死进程和关闭自启动的服务，然后重启系统，这样就可以彻底关闭mysql服务了。

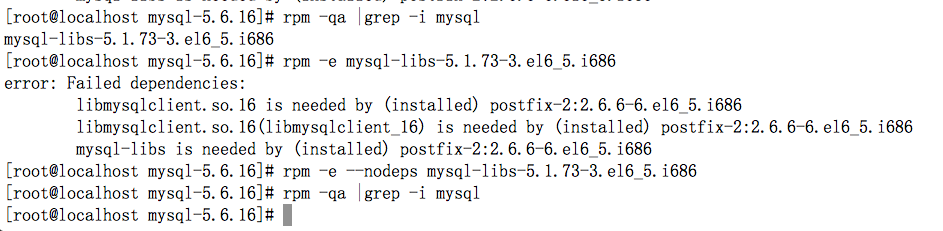
现在你可以安全的卸载mysql了，先说rpm包的卸载。

1、卸载mysql安装包

rpm -e mysql\_package\_name

rpm -e --nodeps mysql\_package\_name #不考虑依赖包，强制删除

rpm -e --allmatches mysql\_package\_name #删除所有跟mysql\_package\_name匹配的所有版本的包



用过yum安装的可以执行下面的命令

yum -y remove mysql\_package\_name

2、rm -rf /var/lib/mysql

如果这个目录不删除，再重新安装之后，密码还是之前的密码，不会重新初始化！

网上查了很久都没有文章提到这个，最后还是自己摸索找出来的。

3、rm /etc/my.cnf

删除mysql配置文件

4、find / -name mysql

再次查找关于mysql的目录，如果有，则全部删除

到此，用rpm安装的mysql就彻底删除了。

卸载完成！怎么确定是不是真的卸载干净了呢？

一是看安装输出：

如果没有卸载干净，安装server时输入只有两行：

二就是安装完成，启动服务之后

使用命令：“mysql -uroot -p”输入你之前安装后设置的密码，看能不能登录，如果还可以登录，说明没有卸载干净！

用源码安装的mysql删除就比较简单了，只要把对应目录删除就好了。

第三步：创建msyql安装目录和数据库存放目录

1.首先删除系统原来的mysql用户和组通过查询/etc/passwd /etc/group这两个文件，如果发现有mysql用户和mysql组，就执行下面的命令把它删掉

userdel mysql

groupdel mysql

2.创建mysql的用户和组

groupadd mysql

useradd -g mysql mysql -s /usr/sbin/nologin

3.创建安装目录和数据库目录(可根据自己实际情况选择目录)

mkdir -p /usr/local/mysql/ #-p表示可以递归创建多层目录

mkdir -p /usr/local/mysql/data

第三步：解压文件

shell# tar -zxf mysql-5.6.22.tar.gz

第四步：编译文件

shell# cd mysql-5.6.22

shell# cmake -DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql \

-DMYSQL\_UNIX\_ADDR=/usr/local/mysql/mysql.sock \

-DSYSCONFDIR=/usr/local/mysql \

-DDEFAULT\_CHARSET=utf8 \

-DDEFAULT\_COLLATION=utf8\_general\_ci \

-DWITH\_EXTRA\_CHARSETS:STRING=utf8,gbk,gb2312 \

-DWITH\_MYISAM\_STORAGE\_ENGINE=1 \

-DENABLED\_LOCAL\_INFILE=ON \

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \

-DWITH\_READLINE=1 \

-DENABLED\_LOCAL\_INFILE=1 \

-DMYSQL\_DATADIR=/usr/local/mysql/data

参数说明：

-DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql //安装目录

-DMYSQL\_UNIX\_ADDR=/usr/local/mysql/mysql.sock \ //设置监听套接字路径，这必须是一个绝对路径名。

#设置服务器的字符集。缺省情况下，MySQL使用latin1的（CP1252西欧）字符集。#cmake/character\_sets.cmake文件包含允许的字符集名称列表

-DDEFAULT\_CHARSET=utf8 　　　　 //使用utf8字符

#设置服务器的排序规则

-DDEFAULT\_COLLATION=utf8\_general\_ci //校验字符

存储引擎选项：

//MyISAM，MERGE，MEMORY，和CSV引擎是默认编译到服务器中，并不需要明确地安装。

//静态编译一个存储引擎到服务器，使用-DWITH\_engine\_STORAGE\_ENGINE= 1

//可用的存储引擎值有：ARCHIVE, BLACKHOLE, EXAMPLE, FEDERATED, INNOBASE (InnoDB), PARTITION (partitioning support), 和PERFSCHEMA (Performance Schema)

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1

-DWITH\_ARCHIVE\_STORAGE\_ENGINE=1

-DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1

-DWITH\_PERFSCHEMA\_STORAGE\_ENGINE=1

-DEXTRA\_CHARSETS=all 　　　　 //安装所有扩展字符集

-DENABLED\_LOCAL\_INFILE=1 　　 //允许从本地导入数据-DINSTALL\_DATADIR=/usr/local/mysql/data //数据库存放目录

第五步：安装

shell# make && make install

###如果需要更改配置

shell# make clean

shell# rm -rf CMakeCache.txt

第六步：复制配置文件

shell# cp support-files/my-default.cnf /usr/local/mysql/my.cnf

第七步：设置权限

shell# chmod +x /usr/local/mysql/

shell# chown -R mysql.mysql /usr/local/mysql

第八步：配置开机启动

shell# cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql

shell# chmod +x /etc/init.d/mysql

shell# chkconfig mysql on

shell# chkconfig --list | grep mysql //此句是查询开机启动是否添加成功

第九步：配置环境变量(修改/etc/profile文件)

shell# vi /etc/profile

将下面这句加入到配置文件里(加在文件的最下方的done前面就可以了)

export PATH="$PATH:/usr/local/mysql/bin"

shell# source /etc/profile //使配置文件profile马上生效

第十步：修改my.cnf配置(我的配置文件在/usr/local/mysql/my.cnf)

shell# vi /usr/local/mysql/my.cnf

在[mysqld]下面添加

datadir=/usr/local/mysql/data

default-storage-engine=MyISAM

以下的为可选：

log-error=/usr/local/mysql/data/error.log

pid-file=/usr/local/mysql/data/mysql.pid

user=mysql

tmpdir=/tmp

说明：1)tmpdir=/tmp //如果不指定，默认使用/tmp，注意下/tmp权限，要有其它用户的写权限

2)在msyql5.5以之前，在mysql解压目录(是刚解压的目录，而不是安装的目录)里(/usr/local/software/mysql/support-files)里有my-small-cnf,my-medium.cnf,my-large.cnf,my-huge.cnf四个文件

如果内存≤64M 复制my-small.cnf

如果内存=128M 复制my-medium.cnf

如果内存=512M 复制my-large.cnf

如果内存是1-2G 复制my-huge.cnf

不过在5.6以后，没有了这几个文件，只有一个my-default.cnf,直接复制这个就可以了

第11步：安装默认数据表

shell# /usr/local/mysql/scripts/mysql\_install\_db --user=mysql

--basedir=/usr/local/mysql

--datadir=/usr/local/mysql/data

--defaults-file=/usr/local/mysql/my.cnf

第12步：启动mysql

shell# /usr/local/mysql/bin/mysqld\_safe -defaults-file=/usr/local/mysql/my.cnf &

// & 是代表后台运行的意思

或者使用：

shell# service msyql start/stop/restart来启动、关闭、重启msyql服务

或者是：/etc/init.d/mysql start/stop/restart

到此mysql就安装完成了！！！

-DMYSQL\_UNIX\_ADDR=/usr/local/mysql/mysql.sock \ //设置监听套接字路径，这必须是一个绝对路径名。

大家一定要注意这个监听套接字，他是mysql和其它服务器的通讯接口，尤其在安装apache nigix时，我们有可能会指定这个套接字，这个mysql套接字，由mysql生成，updatedb后，可通过locate mysq.sock来查询他的绝对路径，当然我们这里指定了他的位置，所以你可以找到/usr/local/mysql/mysql.sock

# CentOS下源码编译安装php5.6.8

编译安装php5.6.8是头一次接触过了，这里介绍的是在CentOS6.5(64位)系统编译安装php5.6.8了，安装成功之后还进行了一些基础的配置，具体过程看下文。这是LAMP最后一个环境的安装，一个比一个难是吧，其实也没啥。

一、下载相关安装包

下载php

http://cn2.php.net/distributions/php-5.6.8.tar.gz

下载libmcrypt（php扩展）

http://nchc.dl.sourceforge.net/p ... mcrypt-2.5.8.tar.gz

下载yasm（php扩展）

http://www.tortall.net/projects/yasm/releases/yasm-1.3.0.tar.gz

下载t1lib（php扩展）

ftp://sunsite.unc.edu/pub/Linux/libs/graphics/t1lib-5.1.2.tar.gz

下载gd库安装包

https://bitbucket.org/libgd/gd-libgd/downloads/libgd-2.1.0.tar.gz

下载libvpx（gd库需要）

http://anduin.linuxfromscratch.o ... ibvpx-v1.3.0.tar.xz

或

https://webm.googlecode.com/files/libvpx-v1.3.0.tar.bz2

下载tiff（gd库需要）

http://download.osgeo.org/libtiff/tiff-4.0.3.tar.gz

下载libpng（gd库需要）

http://downloads.sourceforge.net/libpng/libpng-1.6.17.tar.xz

下载freetype（gd库需要）

http://ring.u-toyama.ac.jp/archi ... eetype-2.5.4.tar.gz

下载jpegsrc（gd库需要）

http://www.ijg.org/files/jpegsrc.v9a.tar.gz

二、安装编译工具及库文件（使用yum命令安装）

这个地方也许是这个教程中，最需要大家注意的地方，通常由于网络的原因，我们都需要架设本地yum源，当然如果你是用的网络源，或者用的但dvd版，通常都不会有什么问题，但问题一般都处在，架设本地yum源时，用的双dvd版，这时你的dvd2光盘，也要拷贝到本地源的Packages中，说白了，就是你在本地架设的yum源的Packages中就是一个rpm库，你必须把所有的rpm都弄到里边去，你在本地安装的时候，才不会出现，无法下载的错误，好多人，都是只拷贝了dvd1，所以像下边的很多rpm包都在dvd2中，而你的本地yum源中根本就没有，当然就会出现下载错误了。

通常建议把下边的所有包都安装了。

yum install -y zlib-devel fontconfig fontconfig-devel libXpm\* libtool\* libxml2 libxml2-devel curl curl-devel

建议安装个全的

yum install -y apr\* autoconf automake bison bzip2 bzip2\* cloog-ppl compat\* cpp

yum install -y curl curl-devel fontconfig fontconfig-devel freetype freetype\*

yum install -y freetype-devel gcc gcc-c++ gtk+-devel gd gettext gettext-devel

yum install -y glibc kernel kernel-headers keyutils keyutils-libs-devel krb5-devel

yum install -y libcom\_err-devel libpng libpng\* libpng-devel libjpeg\* libsepol-devel

yum install -y libselinux-devel libstdc++-devel libtool\* libgomp libxml2 libxml2-devel libXpm\* libX\*

yum install -y libtiff libtiff\* make mpfr ncurses\* ntp openssl nasm nasm\*

yum install -y openssl-devel patch pcre-devel perl php-common php-gd

yum install -y policycoreutils ppl telnet t1lib t1lib\* wget zlib-devel

三、开始安装

1、安装yasm

cd /home

tar zxvf yasm-1.3.0.tar.gz

cd yasm-1.3.0

./configure

make && make install

2、安装libmcrypt

cd /home

tar zxvf libmcrypt-2.5.8.tar.gz

cd libmcrypt-2.5.8

./configure

make && make install

3、安装libvpx

cd /home

xz -d libvpx-v1.3.0.tar.xz #注意如果要保留源文件，要加上-k参数，这里没有加，解压完，源文件就没有了。

tar -xvf libvpx-v1.3.0.tar

cd libvpx-v1.3.0

./configure --prefix=/usr/local/libvpx --enable-shared --enable-vp9

make && make install

4、安装tiff

cd /home

tar zxvf tiff-4.0.3.tar.gz

cd tiff-4.0.3

./configure --prefix=/usr/local/tiff --enable-shared

make && make install

5、安装libpng

cd /home

xz -d libpng-1.6.16.tar.xz

tar -xvf libpng-1.6.16.tar

cd libpng-1.6.16

./configure --prefix=/usr/local/libpng --enable-shared

make && make install

6、安装freetype

cd /home

tar zxvf freetype-2.5.4.tar.gz

cd freetype-2.5.4

./configure --prefix=/usr/local/freetype --enable-shared --without-png

make && make install

7、安装jpeg

cd /home

tar zxvf jpegsrc.v9a.tar.gz

cd jpeg-9a

./configure --prefix=/usr/local/jpeg --enable-shared

make && make install

8、安装libgd

cd /home

tar zxvf libgd-2.1.0.tar.gz

cd libgd-2.1.0

./configure \

--prefix=/usr/local/libgd \

--enable-shared \

--with-jpeg=/usr/local/jpeg \

--with-png=/usr/local/libpng \

--with-freetype=/usr/local/freetype \

--with-fontconfig=/usr/local/freetype \

--with-xpm=/usr/ \

--with-tiff=/usr/local/tiff \

--with-vpx=/usr/local/libvpx

make && make install

9、安装t1lib

cd /home

tar zxvf t1lib-5.1.2.tar.gz

cd t1lib-5.1.2

./configure --prefix=/usr/local/t1lib --enable-shared

make without\_doc

make install

10、安装php

cp -frp /usr/lib64/libltdl.so\* /usr/lib/

cp -frp /usr/lib64/libXpm.so\* /usr/lib/

cd /home

tar -zvxf php-5.6.8.tar.gz

cd php-5.6.8

export LD\_LIBRARY\_PATH=/usr/local/libgd/lib

下面部分分为2各部分，一个是安装支持apache的php服务，另一个是安装支持nginx的php服务

# 核对每一项安装的路径配置 本人这里的–with-mysql-sock=/tmp/mysqld.sock 有的为 –with-mysql-sock=/tmp/mysql.sock这个一定要注意，如果你在安装mysql时指定了mysql套接字，那就不是这个路径，

我自己的路径就是：/usr/local/mysql/mysql.sock

10.1 支持apache的php服务编译方法

./configure \

--prefix=/usr/local/php \

--with-config-file-path=/usr/local/php/etc \

--with-apxs2=/usr/local/apache2/bin/apxs \

--with-mysql=/usr/local/mysql \

--with-mysqli=/usr/local/mysql/bin/mysql\_config

--with-mysql-sock=/tmp/mysql.sock \

--with-gd \

--with-iconv

--with-freetype \

--with-jpeg \

--with-png \

--with-zlib \

--with-libxml

--enable-xml \

--enable-discard-path \

--enable-magic-quotes

--enable-safe-mode \

--enable-bcmath \

--enable-shmop \

--enable-sysvsem

--enable-inline-optimization \

--with-curlwrappers \

--enable-mbregex

--enable-fastcgi \

--enable-force-cgi-redirect \

--enable-mbstring

--enable-ftp \

--enable-gd-native-ttf \

--with-openssl \

--enable-pcntl

--enable-sockets \

--with-xmlrpc \

--enable-zip \

--enable-soap \

--without-pear

--with-gettext \

--with-mime-magic \

--enable-suhosin \

--enable-session

--with-mcrypt

10.2 支持nginx的php服务编译方法

./configure \

--prefix=/usr/local/php \

--with-config-file-path=/usr/local/php/etc \

--with-mysql=/usr/local/mysql \

--with-mysqli=/usr/local/mysql/bin/mysql\_config \

--with-mysql-sock=/usr/local/mysql/mysql.sock \

--with-pdo-mysql=/usr/local/mysql \

--with-gd --with-png-dir=/usr/local/libpng \

--with-jpeg-dir=/usr/local/jpeg \

--with-freetype-dir=/usr/local/freetype \

--with-xpm-dir=/usr/ \

--with-vpx-dir=/usr/local/libvpx/ \

--with-zlib-dir=/usr/local/zlib \

--with-t1lib=/usr/local/t1lib \

--with-iconv \

--enable-libxml \

--enable-xml \

--enable-bcmath \

--enable-shmop \

--enable-sysvsem \

--enable-inline-optimization \

--enable-opcache \

--enable-mbregex \

--enable-fpm \

--enable-mbstring \

--enable-ftp \

--enable-gd-native-ttf \

--with-openssl \

--enable-pcntl \

--enable-sockets \

--with-xmlrpc \

--enable-zip \

--enable-soap \

--without-pear \

--with-gettext \

--enable-session \

--with-mcrypt \

--with-curl \

--enable-ctype

make && make install

cp php.ini-production /usr/local/php/etc/php.ini #复制php配置文件到安装目录

rm -rf /etc/php.ini #删除系统自带配置文件

ln -s /usr/local/php/etc/php.ini /etc/php.ini #添加软链接到 /etc目录

cp /usr/local/php/etc/php-fpm.conf.default /usr/local/php/etc/php-fpm.conf #拷贝模板文件为php-fpm配置文件

ln -s /usr/local/php/etc/php-fpm.conf /etc/php-fpm.conf #添加软连接到 /etc目录

vi /usr/local/php/etc/php-fpm.conf #编辑#大家这里注意一下，下面的用户名和组的设置，都是www，其实这个设置是根据你的nginx设置的，因为你像安装apache一样，都是要先安装nginx，并且我们配置nginx时，设置了用户和组都是www，所以这里为了让php正常启动，你必须指定用户和组，这个组就是nginx的用户和组。

user = www #设置php-fpm运行账号为www

group = www #设置php-fpm运行组为www

pid = run/php-fpm.pid #取消前面的分号

:wq! #保存退出

设置 php-fpm开机启动

cp /home/php-5.6.5/sapi/fpm/init.d.php-fpm /etc/rc.d/init.d/php-fpm #拷贝php-fpm到启动目录

chmod +x /etc/rc.d/init.d/php-fpm #添加执行权限

chkconfig php-fpm on #设置开机启动

#编辑配置文件

vi /usr/local/php/etc/php.ini

找到：disable\_functions =

修改为：

disable\_functions = passthru,exec,system,chroot,scandir,chgrp,chown,shell\_exec,proc\_open,proc\_get\_status,ini\_alter,ini\_alter,ini\_restore,dl,openlog,syslog,readlink,symlink,popepassthru,stream\_socket\_server,escapeshellcmd,dll,popen,disk\_free\_space,checkdnsrr,checkdnsrr,getservbyname,getservbyport,disk\_total\_space,posix\_ctermid,posix\_get\_last\_error,posix\_getcwd, posix\_getegid,posix\_geteuid,posix\_getgid, posix\_getgrgid,posix\_getgrnam,posix\_getgroups,posix\_getlogin,posix\_getpgid,posix\_getpgrp,posix\_getpid, posix\_getppid,posix\_getpwnam,posix\_getpwuid, posix\_getrlimit, posix\_getsid,posix\_getuid,posix\_isatty, posix\_kill,posix\_mkfifo,posix\_setegid,posix\_seteuid,posix\_setgid, posix\_setpgid,posix\_setsid,posix\_setuid,posix\_strerror,posix\_times,posix\_ttyname,posix\_uname

#列出PHP可以禁用的函数，如果某些程序需要用到这个函数，可以删除，取消禁用。

找到：;date.timezone =

修改为：date.timezone = PRC #设置时区

找到：expose\_php = On

修改为：expose\_php = Off #禁止显示php版本的信息

找到：short\_open\_tag = Off

修改为：short\_open\_tag = ON #支持php短标签

注意：short\_open\_tag 必须打开.开启以后可以使用PHP的段标签：(<? ?>)。同时，只有开启这个才可以使用 <?= 以代替 <? echo 。有一些开源系统里会使用短标签，如果不打开的话会报错。

找到opcache.enable=0

修改为opcache.enable=1 #php支持opcode缓存

找到：;opcache.enable\_cli=1 #php支持opcode缓存

修改为：opcache.enable\_cli=0

在最后一行添加：zend\_extension=opcache.so #开启opcode缓存功能

:wq! #保存退出

11、配置nginx支持php

vi /usr/local/nginx/conf/nginx.conf

修改/usr/local/nginx/conf/nginx.conf 配置文件,需做如下修改

user www www; #首行user去掉注释,修改Nginx运行组为www www；必须与/usr/local/php/etc/php-fpm.conf中的user,group配置相同，否则php运行出错

index index.html index.htm index.php; #添加index.php

Shell

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

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

include fastcgi\_params;

}

#取消FastCGI server部分location的注释,注意fastcgi\_param行的参数,改为$document\_root$fastcgi\_script\_name,或者使用绝对路径

/etc/init.d/nginx restart #重启nginx

service php-fpm start #启动php-fpm

老大的教程有两个问题，一个是PHP没有安装对apache的支持，另一个是关于MySQL套接字，，这个文件的绝对路径有问题

还有关于freetype的安装有问题，看出错信息可解决，再有就是关于php的启动进程php-fpm用www是无法启动的，系统中压根就没有www 组，需要手动创建

# CentOS 6.2编译安装Apache2.4.1+MySQL5.5.21+PHP5.3.10配置

http://www.osyunwei.com/archives/2696.html

说明：

操作系统：CentOS 6.2 32位

系统安装教程：CentOS 6.2安装(超级详细图解教程)

http://www.osyunwei.com/archives/1537.html

准备篇：

一、配置好IP、DNS 、网关，确保使用远程连接工具能够连接服务器

CentOS 设置IP地址、网关、DNS教程：

http://www.osyunwei.com/archives/423.html

二、配置防火墙，开启80端口、3306端口

vi /etc/sysconfig/iptables

-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT（允许80端口通过防火墙）

-A INPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT（允许3306端口通过防火墙）

特别提示：很多网友把这两条规则添加到防火墙配置的最后一行，导致防火墙启动失败，正确的应该是添加到默认的22端口这条规则的下面

添加好之后防火墙规则如下所示：

系统运维 www.osyunwei.com 温馨提醒：qihang01原创内容版权所有,转载请注明出处及原文链接

#########################################################

# Firewall configuration written by system-config-firewall

# Manual customization of this file is not recommended.

\*filter

:INPUT ACCEPT [0:0]

:FORWARD ACCEPT [0:0]

:OUTPUT ACCEPT [0:0]

-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

-A INPUT -p icmp -j ACCEPT

-A INPUT -i lo -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT

-A INPUT -j REJECT --reject-with icmp-host-prohibited

-A FORWARD -j REJECT --reject-with icmp-host-prohibited

COMMIT

#########################################################

/etc/init.d/iptables restart #最后重启防火墙使配置生效

三、关闭SELINUX

vi /etc/selinux/config

#SELINUX=enforcing #注释掉

#SELINUXTYPE=targeted #注释掉

SELINUX=disabled #增加

:wq 保存，关闭

shutdown -r now #重启系统

四 、系统约定

软件源代码包存放位置：/usr/local/src

源码包编译安装位置：/usr/local/软件名字

五、下载软件包

1、下载apache

http://www.apache.org/dist/httpd/httpd-2.4.1.tar.gz

2、下载MySQL

http://mysql.mirror.kangaroot.net/Downloads/MySQL-5.5/mysql-5.5.21.tar.gz

3、下载php

http://cn.php.net/distributions/php-5.3.10.tar.gz

4、下载cmake（MySQL编译工具）

http://www.cmake.org/files/v2.8/cmake-2.8.7.tar.gz

5、下载libmcrypt（PHPlibmcrypt模块）

ftp://mcrypt.hellug.gr/pub/crypto/mcrypt/libmcrypt/libmcrypt-2.5.7.tar.gz

6、下载apr（Apache库文件）

http://mirror.bit.edu.cn/apache/apr/apr-1.4.6.tar.gz

7、下载apr-util（Apache库文件）

http://mirror.bit.edu.cn/apache/apr/apr-util-1.4.1.tar.gz

用WinSCP远程连接到服务器，把下载好的软件上传到服务器/usr/local/src下面

六、安装编译工具及库文件（使用CentOS yum命令安装）

系统运维 www.osyunwei.com 温馨提醒：qihang01原创内容版权所有,转载请注明出处及原文链接

yum install make apr\* autoconf automake gcc gcc-c++ zlib-devel openssl openssl-devel pcre-devel gd kernel keyutils patch perl kernel-headers compat\* mpfr cpp glibc libgomp libstdc++-devel ppl cloog-ppl keyutils-libs-devel libcom\_err-devel libsepol-devel libselinux-devel krb5-devel zlib-devel libXpm\* freetype libjpeg\* libpng\* php-common php-gd ncurses\* libtool\* libxml2 libxml2-devel patch

安装篇

以下是用putty工具远程登录到服务器，在命令行下面操作的

1、安装libmcrypt

cd /usr/local/src

tar zxvf libmcrypt-2.5.7.tar.gz #解压

cd libmcrypt-2.5.7 #进入目录

./configure #配置

make #编译

make install #安装

2、安装cmake

cd /usr/local/src

tar zxvf cmake-2.8.7.tar.gz

cd cmake-2.8.7

./configure

make #编译

make install #安装

3、安装apr

cd /usr/local/src

tar zxvf apr-1.4.6.tar.gz

cd apr-1.4.6

./configure --prefix=/usr/local/apr

make

make install

4、安装apr-util

cd /usr/local/src

tar zxvf apr-util-1.4.1.tar.gz

cd apr-util-1.4.1

./configure --prefix=/usr/local/apr-util --with-apr=/usr/local/apr/bin/apr-1-config

make

make install

5、安装mysql

groupadd mysql #添加mysql组

useradd -g mysql mysql -s /bin/false #创建用户mysql并加入到mysql组，不允许mysql用户直接登录系统

mkdir -p /data/mysql #创建MySQL数据库存放目录

chown -R mysql:mysql /data/mysql #设置MySQL数据库目录权限

mkdir -p /usr/local/mysql #创建MySQL安装目录

cd /usr/local/src

tar zxvf mysql-5.5.21.tar.gz #解压

cd mysql-5.5.21

cmake . -DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql -DMYSQL\_DATADIR=/data/mysql -DSYSCONFDIR=/etc #配置

make #编译

make install #安装

cd /usr/local/mysql

cp ./support-files/my-huge.cnf /etc/my.cnf #拷贝配置文件（注意：/etc目录下面默认有一个my.cnf，直接覆盖即可）

vi /etc/my.cnf #编辑配置文件,在 [mysqld] 部分增加下面一行

datadir = /data/mysql #添加MySQL数据库路径

:wq! #保存退出

./scripts/mysql\_install\_db --user=mysql #生成mysql系统数据库

cp ./support-files/mysql.server /etc/rc.d/init.d/mysqld #把Mysql加入系统启动

chmod 755 /etc/init.d/mysqld #增加执行权限

chkconfig mysqld on #加入开机启动

vi /etc/rc.d/init.d/mysqld #编辑

basedir = /usr/local/mysql #MySQL程序安装路径

datadir = /data/mysql #MySQl数据库存放目录

service mysqld start #启动

vi /etc/profile #把mysql服务加入系统环境变量：在最后添加下面这一行

export PATH=$PATH:/usr/local/mysql/bin

下面这两行把myslq的库文件链接到系统默认的位置，这样你在编译类似PHP等软件时可以不用指定mysql的库文件地址。

ln -s /usr/local/mysql/lib/mysql /usr/lib/mysql

ln -s /usr/local/mysql/include/mysql /usr/include/mysql

shutdown -r now #需要重启系统，等待系统重新启动之后继续在终端命令行下面操作

mysql\_secure\_installation #设置Mysql密码

根据提示按Y 回车输入2次密码

或者直接修改密码/usr/local/mysql/bin/mysqladmin -u root -p password "123456" #修改密码

service mysqld restart #重启

到此，mysql安装完成！

6、安装apache2

cd /usr/local/src

tar -zvxf httpd-2.4.1.tar.gz

cd httpd-2.4.1

mkdir -p /usr/local/apache2 #创建安装目录

./configure --prefix=/usr/local/apache2 --with-apr=/usr/local/apr --with-apr-util=/usr/local/apr-util --with-ssl --enable-ssl --enable-module=so --enable-rewrite --enable-cgid --enable-cgi #配置

make #编译

make install #安装

/usr/local/apache2/bin/apachectl -k start #启动

vi /usr/local/apache2/conf/httpd.conf #编辑配置文件

找到：#ServerName www.example.com:80

修改为：ServerName www.osyunwei.com:80

找到：DirectoryIndex index.html

修改为：DirectoryIndex index.html index.php

找到：Options Indexes FollowSymLinks

修改为：Options FollowSymLinks #不显示目录结构

找到AllowOverride None

修改为：AllowOverride All #开启apache支持伪静态，有三处都做修改

LoadModule rewrite\_module modules/mod\_rewrite.so #取消前面的注释，开启apache支持伪静态

vi /etc/profile #添加apache服务系统环境变量

在最后添加下面这一行

export PATH=$PATH:/usr/local/apache2/bin

cp /usr/local/apache2/bin/apachectl /etc/rc.d/init.d/httpd #把apache加入到系统启动

vi /etc/init.d/httpd #编辑文件

在#!/bin/sh下面添加以下两行

#chkconfig:2345 10 90

#description:Activates/Deactivates Apache Web Server

chown daemon.daemon -R /usr/local/apache2/htdocs #更改目录所有者

chmod 700 /usr/local/apache2/htdocs -R #更改apache网站目录权限

chkconfig httpd on #设置开机启动

/etc/init.d/httpd start

service httpd restart

7、安装php

cd /usr/local/src

tar -zvxf php-5.3.10.tar.gz

cd php-5.3.10

mkdir -p /usr/local/php5 #建立php安装目录

./configure --prefix=/usr/local/php5 --with-config-file-path=/usr/local/php5/etc --with-apxs2=/usr/local/apache2/bin/apxs --with-mysql=/usr/local/mysql --with-mysqli=/usr/local/mysql/bin/mysql\_config --with-mysql-sock=/tmp/mysql.sock --with-gd --with-iconv --with-freetype --with-jpeg --with-png --with-zlib --with-libxml --enable-xml --enable-discard-path --enable-magic-quotes --enable-safe-mode --enable-bcmath --enable-shmop --enable-sysvsem --enable-inline-optimization --with-curlwrappers --enable-mbregex --enable-fastcgi --enable-force-cgi-redirect --enable-mbstring --enable-ftp --enable-gd-native-ttf --with-openssl --enable-pcntl --enable-sockets --with-xmlrpc --enable-zip --enable-soap --without-pear --with-gettext --with-mime-magic --enable-suhosin --enable-session --with-mcrypt #配置

make #编译

make install #安装

mkdir /usr/local/php5/etc

cp php.ini-production /usr/local/php5/etc/php.ini #复制php配置文件到安装目录

rm -rf /etc/php.ini #删除系统自带的配置文件

ln -s /usr/local/php5/etc/php.ini /etc/php.ini #创建配置文件软链接

vi /usr/local/php5/etc/php.ini #编辑

找到：;open\_basedir =

修改为：open\_basedir = .:/tmp/ #防止php木马跨站，重要！！

找到：disable\_functions =

修改为：

disable\_functions = passthru,exec,system,chroot,scandir,chgrp,chown,shell\_exec,proc\_open,proc\_get\_status,ini\_alter,ini\_alter,ini\_restore,dl,openlog,syslog,readlink,symlink,popepassthru,stream\_socket\_server,escapeshellcmd,dll,popen,disk\_free\_space,checkdnsrr,checkdnsrr,getservbyname,getservbyport,disk\_total\_space,posix\_ctermid,posix\_get\_last\_error,posix\_getcwd, posix\_getegid,posix\_geteuid,posix\_getgid, posix\_getgrgid,posix\_getgrnam,posix\_getgroups,posix\_getlogin,posix\_getpgid,posix\_getpgrp,posix\_getpid, posix\_getppid,posix\_getpwnam,posix\_getpwuid, posix\_getrlimit, posix\_getsid,posix\_getuid,posix\_isatty, posix\_kill,posix\_mkfifo,posix\_setegid,posix\_seteuid,posix\_setgid, posix\_setpgid,posix\_setsid,posix\_setuid,posix\_strerror,posix\_times,posix\_ttyname,posix\_uname

#列出PHP可以禁用的函数，如果某些程序需要用到这个函数，可以删除，取消禁用。

找到：;date.timezone =

修改为：date.timezone = PRC

找到：expose\_php = On

修改为：expose\_php = OFF #禁止显示php版本的信息

找到：display\_errors = On

修改为：display\_errors = OFF #关闭错误提示

系统运维 www.osyunwei.com 温馨提醒：qihang01原创内容版权所有,转载请注明出处及原文链接

8、配置apache支持php

vi /usr/local/apache2/conf/httpd.conf #编辑apache配置文件

在LoadModule php5\_module modules/libphp5.so这一行下面添加、

AddType application/x-httpd-php .php （注意：php .php这个点前面有一个空格）

service httpd restart #重启apache

service mysqld restart #重启mysql

测试篇

cd /usr/local/apache2/htdocsvi index.php #输入下面内容

<?php

phpinfo();

?>

:wq! #保存

在客户端浏览器输入服务器IP地址，可以看到相关的配置信息！

网站程序上传到/usr/local/apache2/htdocs目录里面，如果安装有问题，请检查目录权限

确保改目录为以下权限

chown daemon.daemon -R /usr/local/apache2/htdocs

chmod -R 700 /usr/local/apache2/htdocs

至此，CentOS 6.2编译安装Apache2.4.1+MySQL5.5.21+PHP5.3.10配置完成。

# 编译安装 freetype 报错 make: [install] 错误 1 (忽略) 及解决办法

在安装LNMP... centos 其中

freetype

./configure

make

当 make install 时总是:

rm -f /usr/local/modules/freetype/include/freetype2/freetype/cache/\*

rmdir /usr/local/modules/freetype/include/freetype2/freetype/cache

rm -f /usr/local/modules/freetype/include/freetype2/freetype/internal/\*

rmdir /usr/local/modules/freetype/include/freetype2/freetype/internal

rmdir: ‘/usr/local/modules/freetype/include/freetype2/freetype/internal’: 没有那个文件或目录

make: [install] 错误 1 (忽略)

/usr/bin/install -c -m 644 ./builds/unix/ft2unix.h \

/usr/local/modules/freetype/include/ft2build.h

/usr/bin/install -c -m 644 ./builds/unix/ftconfig.h \

/usr/local/modules/freetype/include/freetype2/freetype/config/ftconfig.h

/usr/bin/install -c -m 644 /opt/src/lamp/freetype-2.2.1/objs/ftmodule.h \

/usr/local/modules/freetype/include/freetype2/freetype/config/ftmodule.h

/usr/bin/install -c -m 755 ./builds/unix/freetype-config \

/usr/local/modules/freetype/bin/freetype-config

/usr/bin/install -c -m 644 ./builds/unix/freetype2.m4 \

/usr/local/modules/freetype/share/aclocal/freetype2.m4

/usr/bin/install -c -m 644 ./builds/unix/freetype2.pc \

/usr/local/modules/freetype/lib/pkgconfig/freetype2.pc

(报错)

解决办法：

mkdir usr/local/modules/freetype/include/freetype2/freetype/internal

需要在freetype的安装目录下创建internal 这个目录然后，

make install

# apache2 httpd.conf不负责翻译

最近在centos上，配置源码版apache，遇到了一些问题，就把这个主配置文件详细的研究了一下，这些服务器的配置文件，都是看上去非常复杂，其实需要你实际操作一下，才能真正了解和掌握。所以就有了这一篇，篇幅比较大的帖子。

Apache2 httpd.conf配置文件中文版详解

#

#

#这是Apache服务器主要配置文件。

#它包含给服务器指示的配置指令。

#参见<URL:http://httpd.ache.org/doc-2.0/>以取得关于这些指令的详细信息

#

#不要只是简单的阅读这些指令信息而不去理解它。

#这里只是做了简单的说明，如果你没有参考在线文件，你就会被警告。

#

#这些配置指令被分为下面三个部分：

#1. 控制整个Apache服务器行为的部分（即全局环境变量）

#2. 定义主要或者默认服务参数的指令，也为所有虚拟主机提供默认的设置参数

#3. 虚拟主机的设置参数

#

#配置和日志文件名：如果你指定的文件名以“/”开始（win32下以“dirver:/”），

#服务器将使用绝对路径，如果文件名不是以“/”开始的，那么它将把ServerRoot

#的值附加在文件名的前面，例如，对“logs/foo.log"，如果ServerRoot的值

#为“/usr/local/apache2”，则该文件应为“/usr/local/apache2/logs/foo.log”

#

##第一区：全局环境参数

#

#这里设置的参数将影响整个Apache服务器的行为；

#例如Apache能够处理的并发请求的数量等。

#

#ServerRoot:指出服务器保存其配置、出错和日志文件等的根目录。

#

#注意！如果你想要将它指定为NFS或其它网络上的位置，

#请一定要去阅读与LockFile有关的文档（可能在

#<URL:http://httpd.apache.org/docs-2.0/mod/mpm\_common.html#lockfile>)。

#这将会使你自己也能解决很多问题。

#

#路径的结尾不要添加斜线。

#

ServerRoot "/usr/loacl/apache2"

#

#串行访问的锁文件必须保存在本地磁盘上

#

<IfModule !mpm\_winnt.c>

<IfModule !mpm\_neware.c>

#LockFile logs/accept.lock

</IfModule>

</IfModule>

#ScoreBoardFile:用来保存内部服务进程信息的文件。

#如果未指明（默认），记分板（scoreboard）将被保存在一个匿名的共享内存段中，

#并且它不能被第三方软件所使用。

#如果指定了，要确保不能使用两个Apache使用同一个记分板文件，

#这个记分板文件必须保存在本地磁盘上。

#

<IfModule !mpm\_netware.c>

<IfModule !perchild.c>

#ScoreBoardFile logs/apache\_runtime\_status

<IfModule>

<IfModule>

#

#PidFile:记录服务器启动进程号的文件。

#

<IfModule !mpm\_neware.c>

PidFile logs/httpd.pid

</IfModule>

#

#Timeout:接收和发送前超时秒数

#

Timeout 300

#

#KeepAlive:是否允许稳固的连接（每个连接有多个请求），

#设为"Off"则停用。

#

KeepAlive On

#

#MaxKeepAliveRequests:在稳固连接期间允许的最大请求数，

#设为0表示无限制接入。

#我们推荐你将其设为一个较大的值，以便提高性能

MaxKeepAliveRequests 100

#

#KeepAliveTimeout:在同一个连接上从同一台客户上接收请求的秒数

#

KeepAliveTimeout 15

##

##Server-Pool大小设定（针对MPM的）

##

# prefork MPM

# StartServers:启动时服务器启动的进程数

# MinSpareServers:保有的备用进程的最小数目

# MaxSpareServers:保有的备用进程的最大数目

# MaxClients:服务器允许启动的最大进程数

# MaxRequestsPerChild:一个服务进程允许的最大请求数

<IfModule prefork.c>

StartServers 5

MinSpareServers 5

MaxSpareServers 10

MaxClients 150

MaxRequestPerChild 0

</IfModule>

# worker MPM

# StartServers:服务器启动时的服务进程数目

# MaxClients:允许同时连接的最大用户数目

# MinSpareThreads:保有的最小工作线程数目

# MaxSpareThreads:允许保有的最大工作线程数目

# ThreadsPerChild:每个服务进程中的工作线程常数

# MaxRequestsPerChild:服务进程中允许的最大请求数目

<IfModule worker.c>

StartServers 2

MaxClients 150

MinSpareThreads 25

MaxSpareThreads 75

ThreadsPerChild 25

MaxRequestsPerChild 0

</IfModule>

# perchild MPM

# NumServers:服务进程数量

# StartThreads:每个服务进程中的起始线程数量

# MinSpareThreads:保有的最小线程数量

# MaxSpareThreads:保有的最大线程数量

# MaxThreadsPerChild:每个服务进程允许的最大线程数

# MaxRequestsPerChild:每个服务进程允许连接的最大数量

<IfModule perchild.c>

NumServers 5

StartThreads 5

MinSpareThreads 5

MaxSpareThreads 10

MaxThreadsPerChild 20

MaxRequestsPerChild 0

</IfModule>

# WinNT MPM

# ThreadsPerChild:服务进程中工作线程常数

# MaxRequestsPerChild:服务进程允许的最大请求数

<IfModule mpm\_winnt.c>

ThreadsPerChild 250

MaxRequestsPerChild 0

</IfModule>

# BeOS MPM

# StartThreads:服务器启动时启动的线程数

# MaxClients:可以启动的最大线程数（一个线程等于一个用户）

# MaxRequestsPerThread:每个线程允许的最大请求数

<IfModule beos.c>

StartThreads 10

MaxClients 50

MaxRequestsPerThread 10000

</IfModule>

# NetWare MPM

# ThreadStachSize:为每个工作线程分配的堆栈尺寸

# StartThreads:服务器启动时启动的线程数

# MinSpareThreads:用于处理实发请求的空闲线程数

# MaxSpareThreads:空闲线程的最大数量

# MaxThreads:在同一时间活动的最大线程数

# MaxRequestPerChild:一个线程服务请求的最大数量，

# 推荐将其设置为0，以实现无限制的接入

<IfModule mpm\_netware.c>

ThreadStackSize 65536

StartThreads 250

MinSpareThreads 25

MaxSpareThreads 250

MaxThreads 1000

MaxRequestPerChild 0

</IfModule>

# OS/2 MPM

# StartServers:启动的服务进程数量

# MinSpareThreads:每个进程允许的最小空闲线程

# MaxSpareThreads:每个进程允许的最大空闲线程

# MaxRequestsPerChild:每个服务进程允许的最大连接数

<IfModule mpmt\_os2.c>

StartServers 2

MinSpareThreads 5

MaxSpareThreads 10

MaxRequestsPerChild 0

</IfModule>

#

# Listen:允许你绑定Apache服务到指定的IP地址和端口上，以取代默认值

# 参见<VirtualHost>指令

# 使用如下命令使Apache只在指定的IP地址上监听，

# 以防止它在IP地址0.0.0.0上监听

#

# Listen 12.34.56.78:80

Listen 80

#

# 动态共享支持（DSO）

#

# 为了能够使用那些以DSO模式编译的模块中的函数，你必须有相应的“LoadModule”行，

# 因此，在这里包含了这些指令，以便能在使用它之前激活。

# 那些静态编译的模块不需要在这里列出 (即以“httpd -l”列出的模块)

#

# 示例：

# LoadModule foo\_module modules/mod\_foo.so

#

#

# ExtendedStatus：当调用“server-status”时，控制Apache是产生“全”状态

# 信息（ExtendedStatus On），还是产生基本信息（ExtendedStatus Off）。

# 默认为off

#

# ExtendedStatus On

### 第二区：“主”服务配置

#

# 这一区建立被 “主” 服务器用的指令值,以回应那些不被 <VirtualHost>

# 定义处理的任何请求。

# 这些数值也提供默认值给后面定义的<VirtualHost>容器。

# 如果<VirtualHost>中有定义，那么这里定义的指令值将被

# <VirtualHost>中的定义所覆盖。

#

<IfModule !mpm\_winnt.c>

<IfModule !mpm\_neware.c>

#

# 如果你想使httpd以另外的用户或组来运行，你必须在开始时以root方式启动

# 然后再将它切换为你想要使用的用户或组。

#

# User/Group:运行httpd的用户和组

# 在SCO (ODT3)上使用“User nouser”和“Group nogroup”

# 在HPUX上，你可能不能以nobody身份使用共享内存，建议创建一个www用户。

# 注意一些核心（kernel ） 在组ID 大于60000 时拒绝setgid(Group)或semctl

(IPC\_SET)，

#节在这些系统上不要使用“Group #-1”。

#

User nobody

Group #-1

</IfModule>

</IfModule>

#

# ServerAdmin:你的邮件地址，当发生问题时Apache将向你发出邮件。

# 作为一个出错文档，这个地址显示在server-generated页上，

# 例如：admin@your-domain.com

#

ServerAdmin kreny@sina.com

#

# ServerName指定Apache用于识别自身的名字和端口号。

# 通常这个值是自动指定的，但是我们推荐你显式的指定它以防止启动时出错

#

# 如果你为你的主机指定了一个无效的DNS名，server-generated重定向将不能工作。

# 参见UseCanonicalName指令

#

# 如果你的主机没有注册DNS名，在这里键入它的IP地址

# 无论如何，你必须使用它的IP地址来提供服务，

# 这里使用一种容易理解的方式重定向服务

ServerName www.dalouis.com:80

#

# UseCanonicalName: 决定Apache 如何构造URLS 和 SERVER\_NAME 和

SERVER\_PORT 的指令。

# 当设置为 “Off”时,Apache会使用用户端提供的主机名和端口号。

# 当设置为“On”,Apache会使用ServerName指令的值。

#

UseCanonicalName Off

#

# DocumentRoot:你的文档的根目录。默认情况下，所有的请求从这个目录进行应答。

# 但是可以使用符号链接和别名来指向到其他的位置。

#

DocumentRoot "/home/redhat/public\_html"

#

# Apache可以存取的每个目录都可以配置存取权限（包括它的子目录）。

#

# 首先，我们配置一个高限制的特征。

# 这将禁止访问文件系统所在的目录，并添加你希望允许访问的目录块。

# 如下所示

<Directory />

Order Deny,Allow

Deny from all

</Directory>

#

# 注意从这里开始你一定要明确地允许哪些特别的特征能够被使用。

# - 所以，如果Apache没有象你所期待的那样工作的话,

# 请检查你是否在下面明确的指定它可用。

#

#

# 这将改变到你设置的DocumentRoot

#

<Directory "/home/redhat/public\_html">

#

# Options：这个指令的值可以是“None”，“All”，或者下列选项的任意组合：

# Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews

#

# 注意，“MultiViews”必须被显式的指定，“Options All”不能为你提供这个特性。

#

# 这个指令既复杂又重要，请参见

#“http://httpd.apache.org/docs-2.0/mod/core.html#optioins”以取得更多的信息。

#

Options FollowSymLinks

#

# AllowOverride控制那些被放置在.htaccess文件中的指令。

# 它可以是“All”，“None”，或者下列指令的组合：

# Options FileInfo AuthConfig Limit

#

AllowOverride None

#

# 控制谁可以获得服务。

#

Order allow,deny

Allow from all

</Directory>

#

# UserDir:指定在得到一个~user请求时将会添加到用户home目录后的目录名。

#

UserDir public\_html

# 为防止在UserDir指令上的漏洞，对root用户设置

# 象“./”这样的UserDir是非常有用的。

# 如果你使用Apache 1.3或以上版本，我们强烈建议你

# 在你的服务器配置文件中包含下面的行

UserDir disabled root

#

# 下面是一个使用UserDir指令使一个站点的目录具有只读属性的示例：

#

# <Directory /home/\*/public\_html>

# AllowOverride FileInfo AuthConfig Limit Indexes

# Options MultiViews Indexes SymLinksIfOwnerMatch IncludeNoExec

# <Limit GET POST OPTIONS PROPFIND>

# Order allow,deny

# Allow from all

# </Limit>

# <LimitExcept GET POST OPTIONS PROPFIND>

# Order deny,allow

# Deny from all

# </LimitExcept>

# </Directory>

#

# DirectoryIndex:定义请求是一个目录时，Apache向用户提供服务的文件名

#

# index.html.var文件（一个类型映象文件）用于提供一个文档处理列表，

# 出于同样的目的，也可以使用MultiViews选项，但是它会非常慢。

#

DirectoryIndex index.php index.html index.html.var

#

# AccessFileName:在每个目录中查询为目录提供附加配置指令的文件的文件名。

# 参见AllowOverride指令。

#

AccessFileName .htaccess

#

# 下面的行防止.htaccess和.htpasswd文件被Web客户查看。

#

<Files ~ "^\.ht">

Order allow,deny

Deny from all

</Files>

#

# Typeconfig:定义在哪里查询mime.types文件。

#

TypeConfig conf/mime.types

#

# DefaultType:定义当不能确定MIME类型时服务器提供的默认MIME类型。

# 如果你的服务主要包含text或HTML文档，“text/plain”是一个好的选择；

# 如果大多是二进制文档，诸如软件或图像，你应使用

# “application/octer-stream”来防止浏览器象显示文本那样显示二进制文件。

#

DefaultType text/plain

#

# mod\_mime\_magic 允许服务器从自己定义自己类型的文件中使用不同的线索

（hints），

# 这个MIMEMagicFile指令定义hints定义所在的文件。

#

<IfModule mod\_mime\_magic.c>

MIMEMagicFile conf/magic

</IfModule>

#

# HostnameLookups：指定记录用户端的名字还是IP地址，例如，本指令为on时

# 记录主机名，如www.apache.org；为off时记录IP地址，204.62.129.132。

# 默认值为off，这要比设为on好得多，因为如果设为on则每个用户端请求都将会

# 至少造成对 nameserver 进行一次查询。

#

HostnameLookups Off

#

# EnableMMAP:控制是否进行内存转储（如果操作系统支持的话）。

# 默认为on，如果你的服务器安装在网络文件系统上（NFS），请关闭它。

# 在一些系统上，关闭它会提升系统性能（与文件系统类型无关）；

# 具体情况请参阅http://httpd.apache.org/docs-2.0/mod/core.html#enablemmap

#

# EnableMMAP off

#

# EnableSendfile:控制是否使用sendfile kernel支持发送文件

# （如果操作系统支持的话）。默认为on，如果你的服务器安装在网络文件系统

# （NFS）上，请你关闭它。

# 参见http://httpd.apache.org/docs-2.0/mod/core.html#enablesendfile

#

# EnableSendfile off

#

# ErrorLog:错误日志文件定位。

# 如果你没有在<VirtualHost>内定义ErrorLog指令，这个虚拟主机的错误信息

# 将记录在这里。如果你在那儿定义了ErrorLog，这些错误信息将记录在你所

# 定义的文件里，而不是这儿定义的文件。

#

ErrorLog logs/error\_log

#

# LogLevel:控制记录在错误日志文件中的日志信息数量。

# 可能的值包括：debug，info，notice，warn，error，crit，alert，emerg。

#

LogLevel warn

#

# 下面的指令为CustomLog指令定义格式别名。

#

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

LogFormat "%h %l %u %t \"%r\" %>s %b" common

LogFormat "%{Referer}i -> %U" referer

LogFormat "%{User-agent}i" agent

# 你需要安装了mod\_logio.c模块才能使用%I和%O。

# LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O"

combinedio

#

# 指定接入日志文件的定位和格式（一般日志格式）。

# 如果你没有在<VirtualHost>内定义这个指令，传输信息将记录在这里，

# 如果你定义了这个指令，则记录在你指定的位置，而不是这儿定义的位置。

#

CustomLog logs/access\_log common

#

# 如果你想要记录agent和referer信息，可以使用下面的指令

#

# CustomLog logs/referer\_log referer

# CustomLog logs/agent\_log agent

#

# 如果你想要使用一个文件记录access，agent和referer信息，

# 你可以如下定义这个指令：

#

# CustomLog logs/access\_log combined

#

# ServerTokens

# 这个指令定义包含在HTTP回应头中的信息类型。默认为“Full”，

# 这表示在回应头中将包含模块中的操作系统类型和编译信息。

# 可以设为列各值中的一个：

# Full | OS | Minor | Minimal | Major | Prod

# Full传达的信息最多，而Prod最少。

#

ServerTokens Full

#

# 随意的添加包含服务器版本和虚拟主机名字一行信息到server-generated输出页中

# （内部错误文档，FTP目录列表，mod\_status和mod\_info输出等等，除了CGI错误

# 或自定义的错误文档以外）。

# 设为“EMail”将包含一个指向ServerAdmin的mailto:连接。

# 可以为如下值：On | Off | EMail

#

ServerSignature On

#

# Aliases:在这时添加你需要的别名，格式如下：

# Alias 别名 真实名

#

# 注意，如果你在别名的未尾包含了“/”，那么在URL中也需要包含“/”。

# 因此，“/icons”不是这个示例中的别名。

# 如果别名中以“/”结尾，那么真实名也必须以“/”结尾，

# 如果别名中省略了结尾的“/”，那么真实名也必须省略。

#

# 我们使用别名“/icons/”来表示FancyIndexed目录列表，如果你不使用、

# FancyIndexing，你可以注释掉它。

#

# Alias /icons/ "/usr/local/apache2/icons/"

# <Directory "/usr/local/apache2/icons">

# Options Indexes MultiViews

# AllowOverride None

# Order allow,deny

## Allow from all

# </Directory>

#

# 这将改变ServerRoot/manual。这个别名提供了手册页所在的位置，

# 即使你改变了你的DocumentRoot。如果你对有无手册页并不在意的话，

# 你可以注释掉它。

#

Alias /manual "/usr/loacl/apache2/manual"

<Directory "/usr/local/apache2/manual">

Options Indexes FollowSymLinks MultiViews IncludesNoExec

AddOutputFilter Includes html

Order allow,deny

Allow from all

</Directory>

#

# ScriptAlias:指定包含服务脚本的目录。

# ScriptAliases 本质上与Aliases一样，除了这里的文档在请求时做为程序处理处理以外。

# 尾部的“/”规则与Alias一样

#

ScriptAlias /cgi-bin/ "/usr/loacl/apache2/cgi-bin/"

# 这里是添加php 4支持的指令

AddType application/x-httpd-php .php

LoadModule php4\_module modules/libphp4.so

<IfModule mod\_cgid.c>

#

# 添加mod\_cgid.c设置，mod\_cgid提供使用cgid进行通讯的UNIX套接字的

# 脚本接口路径。

#

# Scriptsock logs/cgisock

</IfModule>

#

# 将"/usr/local/apache2/cgi-bin"改为你的ScriptAliased指定的CGI目录，

# 如果你配置了的话。

#

<Directory "/usr/local/apache2/cgi-bin">

AllowOverride None

Options None

Order allow,deny

Allow from all

</Directory>

#

# Redirect允许你告诉客户端使用存在于服务器名字空间中的文档，

# 而不是现在的，这帮助客户定位那些改变了位置的文档。

# 例如：

# Redirect permanent /foo http://www.example.com/bar

#

# 控制server-generated目录列表显示的指令

#

#

# IndexOptions:控制server-generated目录列表显示特征。

#

IndexOptions FancyIndexing VersionSort

#

# AddIcon\* 指令告诉服务器不同扩展名的图象文件如何显示，

# 只适用于FancyIndexed指令

#

AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/\*

AddIconByType (IMG,/icons/image2.gif) image/\*

AddIconByType (SND,/icons/sound2.gif) audio/\*

AddIconByType (VID,/icons/movie.gif) video/\*

AddIcon /icons/binary.gif .bin .exe

AddIcon /icons/binhex.gif .hqx

AddIcon /icons/tar.gif .tar

AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv

AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip

AddIcon /icons/a.gif .ps .ai .eps

AddIcon /icons/layout.gif .html .shtml .htm .pdf

AddIcon /icons/text.gif .txt

AddIcon /icons/c.gif .c

AddIcon /icons/p.gif .pl .py

AddIcon /icons/f.gif .for

AddIcon /icons/dvi.gif .dvi

AddIcon /icons/uuencoded.gif .uu

AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl

AddIcon /icons/tex.gif .tex

AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..

AddIcon /icons/hand.right.gif README

AddIcon /icons/folder.gif ^^DIRECTORY^^

AddIcon /icons/blank.gif ^^BLANKICON^^

#

# DefaultIcon 为那些没有显式定义图标的文件提供处理

#

DefaultIcon /icons/unknown.gif

#

# AddDescription允许你在server-generated索引后放置一个简短的说明。

# 只对FancyIndexed指令有效。

# 格式：AddDescription "说明" 文件名

#

# AddDescription "GZIP compressed document" .gz

# AddDescription "tar archive" .tar

# AddDescription "GZIP compressed tar archive" .tgz

#

# ReadmeName指定服务器默认查找的README文件的名字，并添加到目录列表中

#

# HeaderName指定目录列表前缀文件的文件名

ReadmeName README.html

HeaderName HEADER.html

#

# IndexIgnore指定目录索引忽略并且不包含在列表中的文件名集合，

# 支持shell类型的通配符。

#

IndexIgnore .??\* \*~ \*# HEADER\* README\* RCS CVS \*,v \*,t

#

# AddEncoding允许你在信息传送中使用(Mosaic/X 2.1+)解压缩信息，

# 注意：不是所有的浏览器都支持这个选项。

# 尽管名字相似，但是下列的指令与上面的FancyIndexing定制指令不同。

#

AddEncoding x-compress Z

AddEncoding x-gzip gz tgz

#

# DefaultLanguage和AddLanguage允许你指定文档的语言。

# 这使你可以让用户用容易理解的语言浏览文档。

#

# 指定默认的语言，这意味着所有没有指定语言的包都将使用该语言。

# 多数情况下，你也许并不想设置它，除非你确信这样做是正确的。

# 通常，不使用确定的语言比使用错误的语言要好。

#

# DefaultLanguage nl

#

# 注意1：作为语言关键字的词缀毫无疑问是不能一样的--采用波兰

# 文的文档（网络标准语言代码是pl）将希望使用“AddLanguage pl .po”

# 来避免与perl脚本的一般词缀产生二义性。

#

# 注意2: 下面的例子举例说明在一些范例中语言的二字符缩写与它的国家

# 的二字符缩写不相同,例如 “Danmark/dk” 和 “Danmark/da” 的比较.

#

# 注意3: 在 “ltz” 的情况下我们使用三字符词缀，违犯了 RFC的规定，

# 运行中将修复它并使用RFC1766标准取得参考数据。

#

# Danish (da) - Dutch (nl) - English (en) - Estonian (et)

# French (fr) - German (de) - Greek-Modern (el)

# Italian (it) - Norwegian (no) - Norwegian Nynorsk (nn) - Korean (ko)

# Portugese (pt) - Luxembourgeois\* (ltz)

# Spanish (es) - Swedish (sv) - Catalan (ca) - Czech(cz)

# Polish (pl) - Brazilian Portuguese (pt-br) - Japanese (ja)

# Russian (ru) - Croatian (hr)

#

AddLanguage da .dk

AddLanguage nl .nl

AddLanguage en .en

AddLanguage et .et

AddLanguage fr .fr

AddLanguage de .de

AddLanguage he .he

AddLanguage el .el

AddLanguage it .it

AddLanguage ja .ja

AddLanguage pl .po

AddLanguage ko .ko

AddLanguage pt .pt

AddLanguage nn .nn

AddLanguage no .no

AddLanguage pt-br .pt-br

AddLanguage ltz .ltz

AddLanguage ca .ca

AddLanguage es .es

AddLanguage sv .sv

AddLanguage cz .cz

AddLanguage ru .ru

AddLanguage tw .tw

AddLanguage zh-tw .tw

AddLanguage hr .hr

# LanguagePriority允许你在会话过程中优先使用一些语言。

#

# 以优先次序递减的方式列出它们。我们或多或少地采用按字母排列顺序的方式

# 排列它们。也许你想要改变这个顺序。

LanguagePriority en da nl et fr de el it ja ko no pl pt pt-br ltz ca es sv tw

#

# ForceLanguagePriority 允许你为MULTIPLE CHOICES(Prefer)[在通讯的情况下]

# 或NOT ACCEPTABLE(Fallback)[没有可接受的语言匹配的情况]提供一个结果页。

#

ForceLanguagePriority Prefer Fallback

#

# 为发送出的所有页指定默认的字符集，这总是一个好主意，并且为你的

# web站点的国际化打开了大门，这不正是你曾经想要的吗。同样地，指定

# 默认字符集有一些小的损害，如一个使用iso-8859-1(latin1)标准命令

# 的页面，除非以别的方式指定例如你仅仅以显式方式声明它。

# 也有一些与那些总是鼓励你使用默认字符集的javascropt和URL语法有关

# 的浏览器安全原因。

#

#AddDefaultCharset ISO-8859-1

AddDefaultCharse GB2312

#

# 一般以文件扩展名的方式使用字符集。也许你想要避免与语言扩展发生

# 碰撞，除非你在每次改变后都做了很好的测试。

# 参见http://www.iana.org/assignments/character-sets以取得字符集

# 的名字列表和它们各自的RFCs。

#

AddCharset ISO-8859-1 .iso8859-1 .latin1

AddCharset ISO-8859-2 .iso8859-2 .latin2 .cen

AddCharset ISO-8859-3 .iso8859-3 .latin3

AddCharset ISO-8859-4 .iso8859-4 .latin4

AddCharset ISO-8859-5 .iso8859-5 .latin5 .cyr .iso-ru

AddCharset ISO-8859-6 .iso8859-6 .latin6 .arb

AddCharset ISO-8859-7 .iso8859-7 .latin7 .grk

AddCharset ISO-8859-8 .iso8859-8 .latin8 .heb

AddCharset ISO-8859-9 .iso8859-9 .latin9 .trk

AddCharset ISO-2022-JP .iso2022-jp .jis

AddCharset ISO-2022-KR .iso2022-kr .kis

AddCharset ISO-2022-CN .iso2022-cn .cis

AddCharset Big5 .Big5 .big5

# 对于俄语，使用了多个字符集（如何使用主要依靠客户端）：

AddCharset WINDOWS-1251 .cp-1251 .win-1251

AddCharset CP866 .cp866

AddCharset KOI8-r .koi8-r .koi8-ru

AddCharset KOI8-ru .koi8-uk .ua

AddCharset ISO-10646-UCS-2 .ucs2

AddCharset ISO-10646-UCS-4 .ucs4

AddCharset UTF-8 .utf8

# 下面的字符集没有映射到一个特定的标准(iso)上，但是它们在浏览器

# 中被广泛的支持。注意那些大写字母。

# (它不应该,但是它是为兼容一些浏览器而做)

#

# 参见http://www.iana.org/assianments/character-sets以取得

# 它们的列表。但是浏览器支持较少。

#

AddCharset GB2312 .gb2312 .gb

AddCharset utf-7 .utf7

AddCharset utf-8 .utf8

AddCharset big5 .big5 .b5

AddCharset EUC-TW .euc-tw

AddCharset EUC-JP .euc-jp

AddCharset EUC-KR .euc-kr

AddCharset shift\_jis .sjis

#

# AddType允许你为指定的文件类型添加或覆盖mime.types文件中配置的MIME

#

AddType application/x-tar .tgz

AddType image/x-icon .ico

#

# AddHandler允许你映射确定的文件扩展名到“handlers”:

# 与文件类型无关的行为。这既能编译到服务器中也可以添加到Action指令

# 中（看下面）。

# 为了在ScriptAliased指令指定的以外使用CGI脚本：

#（要使它可用，你还需要在Options中添加“ExecCGI”。

#

# AddHandler cgi-script .cgi

#

# 对于那些包含他们自己的HTTP头的文件

#

# AddHandler send-as-is asis

#

# 对于server-parsed imagemap文件：

#

# AddHandler imap-file map

#

# agemap 文件:

#

#AddHandler imap- 文件映像

#

# 对于类型映像转移资源)

#(这是默认的设定以允许Apache的“It Worked”页能多种语言分发)。

#

AddHandler type-map var

#

# 过滤器允许你在将它发送到客户端前进行处理。

#

# 为了在服务器端分析包含(SSI)的.shtml文档：

# （要执行这个指令，你还需要在Options指令中添加“Includes”。）

#

# AddType text/html .shtml

# AddOutputFilter INCLUDES .shtml

#

# Action让你定义当调用匹配的媒体文件时将要执行的脚本。这将减少

# 那些经常使用的CGI脚本的URL路径名的重复输入。

# 格式：Action media/type /cgi-script/location

# 格式：Action handler-name /cgi-script/location

#

#

# 可配置的错误应答有三种风格：

# 1）plain text 2)local redirects 3) external redirects

#

# 一些示例：

# ErrorDocument 500 "The server made a boo boo."

# ErrorDocument 404 /missing.html

# ErrorDocument 404 "/cgi-bin/missing\_handler.pl"

# ErrorDocument 402 http://www.example.com/subscription\_info.html

#

#

# 综合应用这些指令，我们可以创建一个国际化的出错应答。

#

# 我们使用Alias来重定向任意/error/HTTP\_<error>.html.var应答到

# 我们的多语言错误消息集合。使用正确的文本替代它。

#

# 通过加入下面的行，你就能够改变这些消息的显示，而不必改变

# HTTP\_<error>.html.var文件。

#

# Alias /error/include/ "/your/include/path/"

#

# 以将/usr/local/apache2/error/include/下的文件拷贝到/your/inclue/path/下

# 开始，你可以创建你自己的文件集合，甚至是其于每个虚拟主机的。

# 不管你的ServerSignature如何设置，默认的包含文件将显示你的

# Aapche版本号和你的ServerAdmin邮件地址

#

# 国际化的错误文档需要mod\_alias，mod\_include和mod\_negotiation三个

# 模块。要激活它们，取消下面30行的注释符号

# Alias /error/ "/usr/local/apache2/error/"

#

# <Directory "/usr/local/apache2/error">

# AllowOverride None

# Options IncludesNoExec

# AddOutputFilter Includes html

# AddHandler type-map var

# Order allow,deny

# Allow from all

# LanguagePriority en de es fr it nl sv

# ForceLanguagePriority Prefer Fallback

# </Directory>

#

# ErrorDocument 400 /error/HTTP\_BAD\_REQUEST.html.var

# ErrorDocument 401 /error/HTTP\_UNAUTHORIZED.html.var

ErrorDocument 403 /error.php

# ErrorDocument 404 /error/HTTP\_NOT\_FOUND.html.var

# ErrorDocument 405 /error/HTTP\_METHOD\_NOT\_ALLOWED.html.var

# ErrorDocument 408 /error/HTTP\_REQUEST\_TIME\_OUT.html.var

# ErrorDocument 410 /error/HTTP\_GONE.html.var

# ErrorDocument 411 /error/HTTP\_LENGTH\_REQUIRED.html.var

# ErrorDocument 412 /error/HTTP\_PRECONDITION\_FAILED.html.var

# ErrorDocument 413 /error/HTTP\_REQUEST\_ENTITY\_TOO\_LARGE.html.var

# ErrorDocument 414 /error/HTTP\_REQUEST\_URI\_TOO\_LARGE.html.var

# ErrorDocument 415 /error/HTTP\_SERVICE\_UNAVAILABLE.html.var

# ErrorDocument 500 /error/HTTP\_INTERNAL\_SERVER\_ERROR.html.var

# ErrorDocument 501 /error/HTTP\_NOT\_IMPLEMENTED.html.var

# ErrorDocument 502 /error/HTTP\_BAD\_GATEWAY.html.var

# ErrorDocument 503 /error/HTTP\_SERVICE\_UNAVAILABLE.html.var

# ErrorDocument 506 /error/HTTP\_VARIANT\_ALSO\_VARIES.html.var

#

# 下面的命令更改标准的HTTP应答行为以处理己知的浏览器问题。

#

BrowserMatch "Mozilla/2" nokeepalive

BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

BrowserMatch "RealPlayer 4\.0" force-response-1.0

BrowserMatch "Java/1\.0" force-response-1.0

BrowserMatch "JDK/1\.0" force-response-1.0

#

# 下面命令关闭对那些没有尾部“/”的目录的非GET请求的重定向，

# 这些命令修复了微软的采用DAV方法不能正确处理重定向的WEB文件夹的问题。

# Apple下的DAV文件系统和Gnome下的VFS对DAV的支持也是采用这样的方法

# 进行处理的。

#

BrowserMatch "Microsoft Data Access Internet Publishing Provider" redirect-carefully

BrowserMatch "^WebDrive" redirect-carefully

BrowserMatch "^WebDAVFS/1.[012]" redirect-carefully

BrowserMatch "^gnome-vfs" redirect-carefully

#

# 允许你使用URL：http://servername/server-status来通过mod\_status生

# 成并报告服务器状态信息。改变.example.com为你自己的域名。

#

# <Location /server-status>

# SetHandler server-status

# Order deny,allow

# Deny from all

# Allow from .example.com

# </Location>

#

# 允许使用URL:http://servername/server-info来远程报告服务器配置信息

# （需要mod\_info.c支持）。改变“.example.com”为你自己的域名。

#

# <Location /server-info>

# SetHandler server-info

# Order deny,allow

# Deny from all

# Allow from .example.com

# </Location>

#

# 代理服务器命令，去掉下面的行使代理服务可用。

#

# <IfModule mod\_proxy.c>

# ProxyRequests On

# <Proxy \*>

# Order deny,allow

# Deny from all

# Allow from .example.com

# </Proxy>

#

# 安装或关闭HTTP/1.1“通道”头处理。

# （“Full”添加服务器版本信息，“Block”移掉所有输出“通道”头信息。

# 可以设为下面各选项之一：Off | On | Full | Block

#

# ProxyVia On

# 最好为代理服务安装高速缓冲，去掉下面几行的注释符号：

# （没有CacheRoot则不缓冲）

#

# CacheRoot "/usr/local/apache2/proxy"

# CacheSize 5

# CacheGcInterval 4

# CacheMaxExpire 24

# CacheLastModifiedFactor 01

# CacheDefaultExpire 1

# NoCache a-domain.com another-domain.edu joes.garage-sale.com

# </IfModule>

# 代理命令结束。

#

# 附加的特定模块配置。

#

<IfModule mod\_ssl.c>

Include conf/ssl.conf

</IfModule>

## 第三区：虚拟主机

#

# VirtualHost:你可以通过设置虚拟主机容器以实现在你的主机上保有多个

# 域名/主机名。大多数配置信息只使用基于名字的虚拟主机，因此服务器

# 不必担心IP地址的问题，下面的命令以\*号代替虚拟主机名。

#

# 在你试着配置你的虚拟主机以前，请参见

# URL:http://httpd.apache.org/docs-2.0/vhosts/>以取得更多的信息。

#

# 你可以使用命令行选项“-S”来检验你的虚拟主机配置。

#

# 使用基于名字的虚拟主机。

#

# NameVirtualHost \*

#

# 虚拟主机示例：

# 几乎所有的Apache命令都可以在虚拟主机容器中使用。

# 第一个虚拟主机区是用于向服务名未知的请求进行应答的配置。

#

# <VirtualHost \*>

# ServerAdmin webmaster@dummy-host.example.com

# DocumentRoot /www/docs/dummy-host.example.com

# ServerName dummy-host.example.com

# ErrorLog logs/dummy-host.example.com-error\_log

# CustomLog logs/dummy-host.example.com-access\_log commom

# </virtualHost>

# Apache2 httpd.conf 配置详解(一)

常用配置指令说明

1. ServerRoot：服务器的基础目录，一般来说它将包含conf/和logs/子目录，其它配置文件的相对路径即基于此目录。默认为安装目录，不需更改。

语法：ServerRoot directory-path

如：　ServerRoot "/usr/local/apache-2.2.6"

注意，此指令中的路径最后不要加 / 。

2. Listen：指定服务器监听的IP和端口。默认情况下Apache会在所有IP地址上监听。Listen是Apache2.0以后版本必须设置的指令，如果在配置文件中找不到这个指令，服务器将

无法启动。

语法：Listen [IP-address:]portnumber [protocol]

Listen指令指定服务器在那个端口或地址和端口的组合上监听接入请求。如果只指定一个端口，服务器将在所有地址上监听该端口。如果指定了地址和端口的组合，服务器将在指

定地址的指定端口上监听。可选的protocol参数在大多数情况下并不需要，若未指定该参数，则将为443端口使用默认的https 协议，为其它端口使用http协议。

使用多个Listen指令可以指定多个不同的监听端口和/或地址端口组合。

默认为：Listen 80

如果让服务器接受80和8080端口上请求，可以这样设置：

Listen 80

Listen 8080

如果让服务器在两个确定的地址端口组合上接受请求，可以这样设置：

Listen 192.168.2.1:80

Listen 192.168.2.2:8080

如果使用IPV6地址，必须用方括号把IPV6地址括起来：

Listen [2001:db8::a00:20ff:fea7:ccea]:80

3. LoadModule：加载特定的DSO模块。Apache默认将已编译的DSO模块存放于4.1目录结构小节中所示的动态加载模块目录中。

语法：LoadModule module filename

如：LoadModule rewrite\_module modules/mod\_rewrite.so

如果filename使用相对路径，则路径是相对于ServerRoot所指示的相对路径。

Apache配置文件默认加载所有已编译的DSO模块，笔者建议只加载如下模块：authn\_file、authn\_default、 authz\_host、authz\_user、authz\_default、auth\_basic、dir、alias

、filter、speling、 log\_config、env、vhost\_alias、setenvif、mime、negotiation、rewrite、deflate、 expires、headers、cache、file-cache、disk-cache、mem-cache

。

4. User：设置实际提供服务的子进程的用户。为了使用这个指令，服务器必须以root身份启动和初始化。如果你以非root身份启动服务器，子进程将不能够切换至非特权用户，并

继续以启动服务器的原始用户身份运行。如果确实以root用户启动了服务器，那么父进程将仍然以root身份运行。

用于运行子进程的用户必须是一个没有特权的用户，这样才能保证子进程无权访问那些不想为外界所知的文件，同样的，该用户亦需没有执行那些不应当被外界执行的程序的权限

。强烈建议专门为Apache子进程建立一个单独的用户和组。一些管理员使用nobody用户，但是这并不能总是符合要求，因为可能有其他程序也在使用这个用户。

例：User daemon

5. Group：设置提供服务的Apache子进程运行时的用户组。为了使用这个指令，Apache必须以root初始化启动，否则在切换用户组时会失败，并继续以初始化启动时的用户组运行

。

例：Group daemon

6. ServerAdmin：设置在所有返回给客户端的错误信息中包含的管理员邮件地址。

语法：ServerAdmin email-address|URL

如果httpd不能将提供的参数识别为URL，它就会假定它是一个email-address ，并在超连接中用在mailto:后面。推荐使用一个Email地址，因为许多CGI脚本是这样认为的。如果你

确实想使用URL，一定要保证指向一个你能够控制的服务器，否则用户将无法确保一定可以和你取得联系。

7. ServerName：设置服务器用于辨识自己的主机名和端口号。

语法：ServerName [scheme://]fully-qualified-domain-name[:port]

可选的'scheme://'前缀仅在2.2.3以后的版本中可用，用于在代理之后或离线设备上也能正确的检测规范化的服务器URL。

当没有指定ServerName时，服务器会尝试对IP地址进行反向查询来推断主机名。如果在ServerName中没有指定端口号，服务器会使用接受请求的那个端口。

为了加强可靠性和可预测性，建议使用ServerName显式的指定一个主机名和端口号。

如果使用的是基于域名的虚拟主机，在<VirtualHost>段中的ServerName将是为了匹配这个虚拟主机，在"Host:"请求头中必须出现的主机名。

8. DocumentRoot：设置Web文档根目录。

语法：DocumentRoot directory-path

在没有使用类似Alias这样的指令的情况下，服务器会将请求中的URL附加到DocumentRoot后面以构成指向文档的路径。

如果directory-path不是绝对路径，则被假定为是相对于ServerRoot的路径。

指定DocumentRoot时不应包括最后的"/"。

9. <Directory>：<Directory>和</Directory>用于封装一组指令，使之仅对某个目录及其子目录生效。

语法：<Directory Directory-path> ... </Directory>

Directory-path可以是一个目录的完整路径，或是包含了Unix shell匹配语法的通配符字符串。在通配符字符串中，"?"匹配任何单个的字符，"\*"匹配任何字符序列。也可以使

用"[]"来确定字符范围。在"~" 字符之后也可以使用正则表达式。

如果有多个(非正则表达式)<Directory>配置段符合包含某文档的目录(或其父目录)，那么指令将以短目录优先的规则进行应用，并包含.htaccess文件中的指令。

正则表达式将在所有普通配置段之后予以考虑。所有的正则表达式将根据它们出现在配置文件中的顺序进行应用。

<Directory>指令不可被嵌套使用，也不能出现在<Limit>或<LimitExcept>配置段中。

10. <Files>：提供基于文件名的访问控制，类似于<Directory>和<Location>指令。

语法：<Files filename> ... </Files>

filename参数应当是一个文件名或是一个包含通配符的字符串，其中"?"匹配任何单个字符，"\*"匹配任何字符串序列。在"~"字符之后可以使用正则表达式。

在此配置段中定义的指令将作用于其基本名称(不是完整的路径)与指定的文件名相符的对象。<Files>段将根据它们在配置文件中出现的顺序被处理：在<Directory>段和.htaccess

文件被处理之后，但在<Location>段之前。<Files>能嵌入到<Directory>段中以限制它们作用的文件系统范围，也可用于.htaccess文件当中，以允许用户在文件层面上控制对它们

自己文件的访问。

11. <IfModule>：封装根据指定的模块是否启用而决定是否生效的指令。

语法：<IfModule [!]module-file|module-identifier> ... </IfModule>

module-file是指编译模块时的文件名，比如mod\_rewrite.c　。

module-identifier是指模块的标识符，比如mod\_rewrite　。

在<IfModule>配置段中的指令仅当测试结果为真的时候才进行处理，否则所有其间的指令都将被忽略。

12. Options：控制在特定目录中将使用哪些服务器特性

语法：Options [+|-]option [[+|-]option] ...

option可以为None，不启用任何额外特性，或者下面选项中的一个或多个：

All 　除MultiViews之外的所有特性，这是默认设置。

ExecCGI 　允许使用mod\_cgi执行CGI脚本。

FollowSymLinks 　服务器允许在此目录中使用符号连接，如果此配置位于<Location>配置段中，则会被忽略。

Includes 　允许使用mod\_include提供的服务器端包含。

IncludesNOEXEC 　允许服务器端包含，但禁用"#exec cmd"和"#exec cgi"，但仍可以从ScriptAlias目录使用"#include virtual"虚拟CGI脚本。

Indexes 　如果一个映射到目录的URL被请求，而此目录中又没有DirectoryIndex(例如：index.html)，那么服务器会返回由mod\_autoindex生成的一个格式化后的目录列表。

MultiViews 　允许使用mod\_negotiation提供内容协商的"多重视图"(MultiViews)。

SymLinksIfOwnerMatch 　服务器仅在符号连接与其目的目录或文件的拥有者具有相同的uid时才使用它。 如果此配置出现在<Location>配置段中，则将被忽略。

一般来说，如果一个目录被多次设置了Options ，则最特殊的一个会被完全接受(其它的被忽略)，而各个可选项的设定彼此并不融合。然而，如果所有作用于Options指令的可选项

前都加有"+" 或"-"符号，此可选项将被合并。所有前面加有"+"号的可选项将强制覆盖当前的可选项设置，而所有前面有"-"号的可选项将强制从当前可选项设置中去除。

13. AllowOverride：确定允许存在于.htaccess文件中的指令类型。

语法：AllowOverride All|None|directive-type [directive-type] ...

如果此指令被设置为None ，那么.htaccess文件将被完全忽略。事实上，服务器根本不会读取.htaccess文件。

当此指令设置为All时，所有具有".htaccess"作用域的指令都允许出现在.htaccess文件中。

directive-type可以是下列各组指令之一：

AuthConfig 　允许使用与认证授权相关的指令

FileInfo 　允许使用控制文档类型的指令、控制文档元数据的指令、mod\_rewrite中的指令、mod\_actions中的Action指令

Indexes 　允许使用控制目录索引的指令

Limit 　允许使用控制主机访问的指令

Options[=Option,...] 　允许使用控制指定目录功能的指令(Options和XBitHack)。可以在等号后面附加一个逗号分隔的(无空格的)Options选项列表，用来控制允许Options指令

使用哪些选项。

AllowOverride仅在不包含正则表达式的<Directory>配置段中才是有效的。在<Location>, <DirectoryMatch>, <Files>配置段中都是无效的。

Order：控制默认的访问状态与Allow和Deny指令生效的顺序。

Ordering取值范围是以下几种范例之一：

Deny,Allow 　Deny指令在Allow指令之前被评估。默认允许所有访问。任何不匹配Deny指令或者匹配Allow指令的客户都被允许访问。

Allow,Deny 　Allow指令在Deny指令之前被评估。默认拒绝所有访问。任何不匹配Allow指令或者匹配Deny指令的客户都将被禁止访问。

Mutual-failure 　只有出现在Allow列表并且不出现在Deny列表中的主机才被允许访问。这种顺序与"Order Allow,Deny"具有同样效果，不赞成使用。

关键字只能用逗号分隔，它们之间不能有空格，在所有情况下每个Allow和Deny指令语句都将被评估。

Allow：控制哪些主机可以访问服务器的该区域。可以根据主机名、IP地址、 IP地址范围或其他环境变量中捕获的客户端请求特性进行控制。

语法：Allow from all|host|env=env-variable [host|env=env-variable] ...

这个指令的第一个参数总是"from"，随后的参数可以有三种不同形式：如果指定"Allow from all"，则允许所有主机访问，按照下述Deny和Order指令的配置；若要只允许特定的主

机或主机群访问服务器，host可以用下面任何一种格式来指定：一个（部分）域名、完整的IP地址、部分IP地址、网络/掩码、网络/nnn无类别域间路由规格；第三种参数格式允许

对服务器的访问由环境变量的一个扩展指定，指定"Allow from env=env-variable"时，如果环境变量env-variable存在则访问被允许，使用由mod\_setenvif提供的指令，服务器用

一种基于客户端请求的弹性方式提供了设置环境变量的能力。因此，这条指令可以用于允许基于像User-Agent(浏览器类型)、Referer或其他 HTTP请求头字段的访问。

Deny：控制哪些主机被禁止访问服务器的该区域。可以根据主机名、IP地址、 IP地址范围或其他环境变量中捕获的客户端请求特性进行控制。

语法：Deny from all|host|env=env-variable [host|env=env-variable] ...

此指令的参数设置和Allow指令完全相同。

14. DirectoryIndex：当客户端请求一个目录时寻找的资源列表。

语法：DirectoryIndex Local-url [Local-url] ...

Local-url(%已解码的)是一个相对于被请求目录的文档的URL(通常是那个目录中的一个文件)。可以指定多个URL，服务器将返回最先找到的那一个，比如：

DirectoryIndex index.html index.php

15. ErrorLog：指定当服务器遇到错误时记录错误日志的文件。

语法：ErrorLog file-path|syslog[:facility]

如果file-path不是一个以斜杠(/)开头的绝对路径，那么将被认为是一个相对于ServerRoot的相对路径；如果file-path以一个管道符号(|)开头，那么会为它指定一个命令来处理

错误日志，如 ErrorLog "|/usr/local/sbin/cronolog /var/log/httpd/%w/errors\_log"　。

如果系统支持，使用"syslog"替代文件名将通过 syslogd(8)来记载日志。默认将使用系统日志机制local7 ，但您可以用"syslog:facility"语法来覆盖这个设置，其中，facility

的取值为syslog(1)中记载的任何一个名字。

16. LogLevel：用于调整记录在错误日志中的信息的详细程度。

语法：LogLevel level

可以选择下列level，依照重要性降序排列：

emerg 　紧急(系统无法使用)

alert 　必须立即采取措施

crit 　致命情况

error 　错误情况

warn 　警告情况

notice 　一般重要情况

info 　 普通信息

debug 　调试信息

当指定了某个级别时，所有级别高于它的信息也会被同时记录。比如，指定 LogLevel info ，则所有notice和warn级别的信息也会被记录。建议至少使用crit级别。

当错误日志是一个单独分开的正式文件的时候，notice级别的消息总是会被记录下来，而不能被屏蔽。但是，当使用syslog来记录时就没有这个问题。

17. LogFormat：定义访问日志的记录格式。

语法：LogFormat format|nickname [nickname]

LogFormat指令可以使用两种定义格式中的一种。

在第一种格式中，指令只带一个参数，以定义后续的TransferLog指令定义的日志格式。另外它也可以通过下述的方法使用nickname来引用某个之前的LogFormat定义的日志格式。

第二种定义LogFormat指令的格式中，将一个直接的format和一个nickname联系起来。这样在后续的LogFormat或 CustomLog指令中，就不用一再重复整个冗长的格式串。定义别名

的LogFormat指令仅仅用来定义一个nickname ，而不做其它任何事情，也就是说，它只是定义了这个别名，它既没有实际应用这个别名，也不是把它设为默认的格式。因此，它不

会影响后续的 TransferLog指令。另外，LogFormat不能用一个别名来定义另一个别名。nickname不能包含百分号(%)。

关于format的格式，请参见Apache2.2官方文档中的自定义日志格式小节。

18. CustomLog：设定日志的文件名和格式。

语法：CustomLog file|pipe format|nickname [env=[!]environment-variable]

第一个参数指定了日志记录的位置，可以使用以下两种方式来设定：

file 　相对于ServerRoot的日志文件名。

pipe 　管道符"|"后面紧跟着一个把日志输出当作标准输入的处理程序路径。

第二个参数指定了写入日志文件的内容。它既可以是由前面的LogFormat指令定义的nickname ，也可以是直接按Apache2.2官方文档中的自定义日志格式小节所描述的规则定义的

format字符串。

第三个参数是可选的，它根据服务器上特定的环境变量是否被设置来决定是否对某一特定的请求进行日志记录。如果这个特定的环境变量被设置(或者在"env=!name"的情况下未被

设置)，那么这个请求将被记录。可以使用mod\_setenvif和/或mod\_rewrite模块来为每个请求设置环境变量。

19. TransferLog：指定日志文件的位置。

语法：TransferLog file|pipe

本指令除不允许直接定义日志格式或根据条件进行日志记录外，与CustomLog指令有完全相同的参数和功能。实际应用中，日志的格式是由最近的非别名定义的LogFormat指令指定

。如果没有定义任何日志格式，则使用通用日志格式。

20. Alias：映射URL到文件系统的特定区域。

语法：Alias URL-path file-path|directory-path

Alias指令使文档可以被存储在DocumentRoot以外的本地文件系统中。以(%已解码的)url-path路径开头的URL可以被映射到以directory-path开头的本地文件。

如果对在DocumentRoot之外的某个目录建立了一个Alias ，则可能需要通过<Directory>段明确的对目标目录设定访问权限。

21.ScriptAlias：映射一个URL到文件系统并视之为CGI脚本目录。

语法：ScriptAlias URL-path file-path|directory-path

ScriptAlias指令的行为与Alias指令相同，但同时它又标明此目录中含有应该由cgi-script处理器处理的CGI脚本。以URL-path开头的(%已解码的)的URL会被映射到由第二个参数指

定的具有完整路径名的本地文件系统中的脚本。

ScriptSock：在以线程式MPM(worker)运行的Apache中设置用来与CGI守护进程通信的套接字文件名前缀(其后附加父进程 PID组成完整的文件名)。这个套接字将会用启动Apache服

务器的父进程用户权限(通常是root)打开。为了维护与CGI脚本通讯的安全性，不允许其他用户拥有写入套接字所在目录的权限是很重要的。

22. DefaultType：在服务器无法由其他方法确定内容类型时，发送的默认MIME内容类型。

语法：DefaultType MIME-type

默认：DefaultType text/plain

23. AddType：在给定的文件扩展名与特定的内容类型之间建立映射关系。

语法：AddType MIME-type extension [extension] ...

MIME-type指明了包含extension扩展名的文件的媒体类型。这个映射关系会添加在所有有效的映射关系上，并覆盖所有相同的extension扩展名映射。

extension参数是不区分大小的，并且可以带或不带前导点。

24. ErrorDocument：批示当遇到错误的时候服务器将给客户端什么样的应答。

语法：ErrorDocument error-code document

error-code 　服务器返回的错误代码

document 　可以由一个斜杠(/)开头来指示一个本地URL(相对于DocumentRoot)，或是提供一个能被客户端解释的完整的URL。此外还能提供一个可以被浏览器显示的消息。比如：

ErrorDocument 500http://www.entage.net/err500.html

ErrorDocument 404 /errors/bad\_urls.html

ErrorDocument 403 "Sorry can't allow you access today"

25. EnableMMAP：指示httpd在递送中如果需要读取一个文件的内容，它是否可以使用内存映射。

语法：EnableMMAP On|Off

当处理一个需要访问文件中的数据的请求时，比如说当递送一个使用mod\_include进行服务器端分析的文件时，如果操作系统支持，Apache将默认使用内存映射。

这种内存映射有时会带来性能的提高，但在某些情况下，您可能会需要禁用内存映射以避免一些操作系统的问题：

在一些多处理器的系统上，内存映射会减低一些httpd的性能；

在挂载了NFS的DocumentRoot上，若已经将一个文件进行了内存映射，则删除或截断这个文件会造成httpd因为分段故障而崩溃。

在可能遇到这些问题的服务器配置过程中，应当使用下面的命令来禁用内存映射：

26. EnableMMAP Off

对于挂载了NFS的文件夹，可以单独在<directory>段中指定禁用内存映射：

<Directory "/path-to-nfs-files">

EnableMMAP Off

</Directory>

27. EnableSendfile：控制httpd是否可以使用操作系统内核的sendfile支持来将文件发送到客户端。

默认情况下，当处理一个请求并不需要访问文件内部的数据时(比如发送一个静态的文件内容)，如果操作系统支持，Apache将使用sendfile将文件内容直接发送到客户端而并不读

取文件。

这个sendfile机制避免了分开的读和写操作以及缓冲区分配，但是在一些平台或者一些文件系统上，最好禁止这个特性来避免一些问题：

一些平台可能会有编译系统检测不到的有缺陷的sendfile支持，特别是将在其他平台上使用交叉编译得到的二进制文件运行于当前对sendfile支持有缺陷的平台时；

在Linux上启用IPv6时，使用sendfile将会触发某些网卡上的TCP校验和卸载bug；

当Linux运行在Itanium处理器上的时候，sendfile可能无法处理大于2GB的文件；

对于一个通过网络挂载了NFS文件系统的DocumentRoot (比如：NFS或SMB)，内核可能无法可靠的通过自己的缓冲区服务于网络文件。

如果出现以上情况，你应当禁用sendfile ：

EnableSendfile Off

针对NFS或SMB，可以单独在<directory>段中指定禁用：

<Directory "/path-to-nfs-files">

EnableSendfile Off

</Directory>

28. include：在服务器配置文件中包含其它配置文件。

语法：Include file-path|directory-path

Shell风格(fnmatch())的通配符可以用于按照字母顺序一次包含多个文件。另外，如果Include指向了一个目录而不是一个文件，Apache将读入该目录及其子目录下的所有文件，并

依照字母顺序将这些文件作为配置文件进行解析。但是并不推荐这么做，因为偶尔会有临时文件在这个目录中生成，从而导致httpd启动失败。

文件的路径可以是一个完整的绝对路径(以一个斜杠开头)，或是相对于ServerRoot目录的相对路径。

29. 设置动态加载的DSO模块

#如果需要提供基于文本文件的认证,加载此模块，否则注释掉

LoadModule authn\_file\_module modules/mod\_authn\_file.so

#如果需要提供基于DBM文件的认证,加载此模块,否则注释掉

#LoadModule authn\_dbm\_module modules/mod\_authn\_dbm.so

#如果需要提供匿名用户认证,加载此模块,否则注释掉

#LoadModule authn\_anon\_module modules/mod\_authn\_anon.so

#如果需要提供基于SQL数据库的认证,加载此模块,否则注释掉

#LoadModule authn\_dbd\_module modules/mod\_authn\_dbd.so

#如果需要在未正确配置认证模块的情况下简单拒绝一切认证信息,加载此模块,否则注释掉

LoadModule authn\_default\_module modules/mod\_authn\_default.so

#此模块提供基于主机名、IP地址、请求特征的访问控制,Allow、Deny指令需要,推荐加载。

LoadModule authz\_host\_module modules/mod\_authz\_host.so

#如果需要使用纯文本文件为组提供授权支持,加载此模块,否则注释掉

#LoadModule authz\_groupfile\_module modules/mod\_authz\_groupfile.so

#如果需要提供基于每个用户的授权支持,加载此模块,否则注释掉

LoadModule authz\_user\_module modules/mod\_authz\_user.so

#如果需要使用DBM文件为组提供授权支持,加载此模块,否则注释掉

#LoadModule authz\_dbm\_module modules/mod\_authz\_dbm.so

#如果需要基于文件的所有者进行授权,加载此模块,否则注释掉

#LoadModule authz\_owner\_module modules/mod\_authz\_owner.so

#如果需要在未正确配置授权支持模块的情况下简单拒绝一切授权请求,加载此模块,否则注释掉

LoadModule authz\_default\_module modules/mod\_authz\_default.so

#如果需要提供基本的HTTP认证,加载此模块,否则注释掉,此模块至少需要同时加载一个认证支持模块和一个授权支持模块

LoadModule auth\_basic\_module modules/mod\_auth\_basic.so

##如果需要提供HTTP MD5摘要认证,加载此模块,否则注释掉,此模块至少需要同时加载一个认证支持模块和一个授权支持模块

#LoadModule auth\_digest\_module modules/mod\_auth\_digest.so

#此模块提供文件描述符缓存支持,从而提高Apache性能,推荐加载,但请小心使用

LoadModule file\_cache\_module modules/mod\_file\_cache.so

#此模块提供基于URI键的内容动态缓存(内存或磁盘),从而提高Apache性能,必须与mod\_disk\_cache/mod\_mem\_cache同时使用,推荐加载

LoadModule cache\_module modules/mod\_cache.so

#此模块为mod\_cache提供基于磁盘的缓存管理,推荐加载

LoadModule disk\_cache\_module modules/mod\_disk\_cache.so

#此模块为mod\_cache提供基于内存的缓存管理,推荐加载

LoadModule mem\_cache\_module modules/mod\_mem\_cache.so

#如果需要管理SQL数据库连接，为需要数据库功能的模块提供支持,加载此模块,否则注释掉(推荐)

#LoadModule dbd\_module modules/mod\_dbd.so

#此模块将所有I/O操作转储到错误日志中,会导致在日志中写入及其海量的数据,只建议在发现问题并进行调试的时候使用

#LoadModule dumpio\_module modules/mod\_dumpio.so

#如果需要使用外部程序作为过滤器,加载此模块(不推荐),否则注释掉

#LoadModule ext\_filter\_module modules/mod\_ext\_filter.so

#如果需要实现服务端包含文档(SSI)处理,加载此模块(不推荐),否则注释掉

#LoadModule include\_module modules/mod\_include.so

#如果需要根据上下文实际情况对输出过滤器进行动态配置,加载此模块,否则注释掉

LoadModule filter\_module modules/mod\_filter.so

#如果需要服务器在将输出内容发送到客户端以前进行压缩以节约带宽,加载此模块(推荐),否则注释掉

LoadModule deflate\_module modules/mod\_deflate.so

##如果需要记录日志和定制日志文件格式,加载此模块(推荐),否则注释掉

LoadModule log\_config\_module modules/mod\_log\_config.so

#如果需要对每个请求的输入/输出字节数以及HTTP头进行日志记录,加载此模块,否则注释掉

#LoadModule logio\_module modules/mod\_logio.so

#如果允许Apache修改或清除传送到CGI脚本和SSI页面的环境变量,加载此模块,否则注释掉

LoadModule env\_module modules/mod\_env.so

#如果允许通过配置文件控制HTTP的"Expires:"和"Cache-Control:"头内容,加载此模块(推荐),否则注释掉

LoadModule expires\_module modules/mod\_expires.so

#如果允许通过配置文件控制任意的HTTP请求和应答头信息,加载此模块,否则注释掉

LoadModule headers\_module modules/mod\_headers.so

#如果需要实现RFC1413规定的ident查找,加载此模块(不推荐),否则注释掉

#LoadModule ident\_module modules/mod\_ident.so

#如果需要根据客户端请求头字段设置环境变量,加载此模块,否则注释掉

LoadModule setenvif\_module modules/mod\_setenvif.so

#此模块是mod\_proxy的扩展,提供Apache JServ Protocol支持,只在必要时加载

#LoadModule proxy\_ajp\_module modules/mod\_proxy\_ajp.so

#此模块是mod\_proxy的扩展,提供负载均衡支持,只在必要时加载

#LoadModule proxy\_balancer\_module modules/mod\_proxy\_balancer.so

#如果需要根据文件扩展名决定应答的行为(处理器/过滤器)和内容(MIME类型/语言/字符集/编码),加载此模块,否则注释掉

LoadModule mime\_module modules/mod\_mime.so

#如果允许Apache提供DAV协议支持,加载此模块,否则注释掉

#LoadModule dav\_module modules/mod\_dav.so

#此模块生成描述服务器状态的Web页面,只建议在追踪服务器性能和问题时加载

#LoadModule status\_module modules/mod\_status.so

#如果需要自动对目录中的内容生成列表(类似于"ls"或"dir"命令),加载此模块(会带来安全问题,不推荐),否则注释掉

#LoadModule autoindex\_module modules/mod\_autoindex.so

#如果需要服务器发送自己包含HTTP头内容的文件,加载此模块,否则注释掉

#LoadModule asis\_module modules/mod\_asis.so

#如果需要生成Apache配置情况的Web页面,加载此模块(会带来安全问题,不推荐),否则注释掉

#LoadModule info\_module modules/mod\_info.so

#如果需要在非线程型MPM(prefork)上提供对CGI脚本执行的支持,加载此模块,否则注释掉

#LoadModule cgi\_module modules/mod\_cgi.so

#此模块在线程型MPM(worker)上用一个外部CGI守护进程执行CGI脚本,如果正在多线程模式下使用CGI程序,推荐替换mod\_cgi加载,否则注释掉

#LoadModule cgid\_module modules/mod\_cgid.so

#此模块为mod\_dav访问服务器上的文件系统提供支持,如果加载mod\_dav,则也应加载此模块,否则注释掉

#LoadModule dav\_fs\_module modules/mod\_dav\_fs.so

#如果需要提供大批量虚拟主机的动态配置支持,加载此模块,否则注释掉

#LoadModule vhost\_alias\_module modules/mod\_vhost\_alias.so

#如果需要提供内容协商支持(从几个有效文档中选择一个最匹配客户端要求的文档),加载此模块(推荐),否则注释掉

LoadModule negotiation\_module modules/mod\_negotiation.so

#如果需要指定目录索引文件以及为目录提供"尾斜杠"重定向,加载此模块(推荐),否则注释掉

LoadModule dir\_module modules/mod\_dir.so

#如果需要处理服务器端图像映射,加载此模块,否则注释掉

#LoadModule imagemap\_module modules/mod\_imagemap.so

#如果需要针对特定的媒体类型或请求方法执行CGI脚本,加载此模块,否则注释掉

#LoadModule actions\_module modules/mod\_actions.so

#如果希望服务器自动纠正URL中的拼写错误,加载此模块(推荐),否则注释掉

LoadModule speling\_module modules/mod\_speling.so

#如果允许在URL中通过"/~username"形式从用户自己的主目录中提供页面,加载此模块,否则注释掉

#LoadModule userdir\_module modules/mod\_userdir.so

#此模块提供从文件系统的不同部分到文档树的映射和URL重定向,推荐加载

LoadModule alias\_module modules/mod\_alias.so

#如果需要基于一定规则实时重写URL请求,加载此模块(推荐),否则注释掉

LoadModule rewrite\_module modules/mod\_rewrite.so

# Apache2 httpd.conf 配置详解(二)

DocumentRoot "/usr/local/apache-2.2.6/htdocs"#设置WEB文档根目录的默认属性

<Directory />

Options FollowSymLinks

AllowOverride None

Order deny,allow

Deny from all

</Directory>

#设置DocumentRoot指定目录的属性

<Directory "/usr/local/apache-2.2.6/htdocs">

Options FollowSymLinks

AllowOverride None

Order allow,deny

Allow from all

</Directory>

#设置默认目录资源列表文件

<IfModule dir\_module>

DirectoryIndex index.html

</IfModule>

#拒绝对.ht开头文件的访问,以保护.htaccess文件

<FilesMatch "^\.ht">

Order allow,deny

Deny from all

Satisfy All

</FilesMatch>

<IfModule log\_config\_module>

#定义访问日志的格式

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

LogFormat "%h %l %u %t \"%r\" %>s %b" common

<IfModule logio\_module>

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio

</IfModule>

#指定访问日志及使用的格式

CustomLog logs/access\_log combined

</IfModule>

#设定默认CGI脚本目录及别名

ScriptAlias /cgi-bin/ "/usr/local/apache-2.2.6/cgi-bin/"

#在以线程式MPM(worker)运行的Apache中设置用来与CGI守护进程通信的套接字文件名前缀

<IfModule cgid\_module>

Scriptsock logs/cgisock

</IfModule>

#设定默认CGI脚本目录的属性

<Directory "/usr/local/apache-2.2.6/cgi-bin">

AllowOverride None

Options None

Order allow,deny

Allow from all

</Directory>

#设定默认MIME内容类型

DefaultType text/plain

<IfModule mime\_module>

##指定MIME类型映射文件

TypesConfig conf/mime.types

#增加.Z .tgz的类型映射

AddType application/x-compress .Z

AddType application/x-gzip .gz .tgz

</IfModule>

#启用内存映射

EnableMMAP on

##使用操作系统内核的sendfile支持来将文件发送到客户端

EnableSendfile on

#指定多路处理模块(MPM)配置文件并将其附加到主配置文件

Include conf/extra/httpd-mpm.conf

#指定多语言错误应答配置文件并将其附加到主配置文件

Include conf/extra/httpd-multilang-errordoc.conf

#指定目录列表配置文件并将其附加到主配置文件

#Include conf/extra/httpd-autoindex.conf

#指定语言配置文件并将其附加到主配置文件

Include conf/extra/httpd-languages.conf

#指定用户主目录配置文件并将其附加到主配置文件

#Include conf/extra/httpd-userdir.conf

#指定用于服务器信息和状态显示的配置文件并将其附加到主配置文件

#Include conf/extra/httpd-info.conf

#指定提供Apache文档访问的配置文件并将其附加到配置文件

#Include conf/extra/httpd-manual.conf

#指定DAV配置文件并将其附加到主配置文件

#Include conf/extra/httpd-dav.conf

#指定与Apache服务自身相关的配置文件并将其附加到主配置文件

Include conf/extra/httpd-default.conf

#指定mod\_deflate压缩模块配置文件并将其附加到主配置文件

Include conf/extra/httpd-deflate.conf

#指定mod\_expires模块配置文件并将其附加到主配置文件

Include conf/extra/httpd-expires.conf

##指定虚拟主机配置文件并将其附加到主配置文件

#Include conf/extra/httpd-vhosts.conf

#指定SSL配置文件并将其附加到主配置文件

Include conf/extra/httpd-ssl.conf

##SSL默认配置

<IfModule ssl\_module>

SSLRandomSeed startup builtin

SSLRandomSeed connect builtin

</IfModule>

#配置多路处理模块(MPM) httpd-mpm.conf

首先说一下原理:

prefork模式

这个多路处理模块(MPM)实现了一个非线程型的、预派生的web服务器，它的工作方式类似于Apache 1.3。它适合于没有线程安全库，需要避免线程兼容性问题的系统。它是要求将

每个请求相互独立的情况下最好的MPM，这样若一个请求出现问题就不会影响到其他请求。

这个MPM具有很强的自我调节能力，只需要很少的配置指令调整。最重要的是将MaxClients设置为一个足够大的数值以处理潜在的请求高峰，同时又不能太大，以致需要使用的内存

超出物理内存的大小。

worker模式

此多路处理模块(MPM)使网络服务器支持混合的多线程多进程。由于使用线程来处理请求，所以可以处理海量请求，而系统资源的开销小于基于进程的MPM。但是，它也使用了多进

程，每个进程又有多个线程，以获得基于进程的MPM的稳定性。

控制这个MPM的最重要的指令是，控制每个子进程允许建立的线程数的ThreadsPerChild指令，和控制允许建立的总线程数的MaxClients指令

#设置prefork多路处理模块

<IfModule mpm\_prefork\_module>

StartServers 5

MinSpareServers 5

MaxSpareServers 10

ServerLimit　　　　8000

MaxClients 8000

MaxRequestsPerChild 0

</IfModule>

StartServers：设置服务器启动时建立的子进程数量。因为子进程数量动态的取决于负载的轻重,所有一般没有必要调整这个参数。

MinSpareServers：设置空闲子进程的最小数量。所谓空闲子进程是指没有正在处理请求的子进程。如果当前空闲子进程数少于 MinSpareServers ,那么Apache将以最大每秒一个的

速度产生新的子进程。只有在非常繁忙机器上才需要调整这个参数。将此参数设的太大通常是一个坏主意。

MaxSpareServers：设置空闲子进程的最大数量。如果当前有超过MaxSpareServers数量的空闲子进程,那么父进程将杀死多余的子进程。只有在非常繁忙机器上才需要调整这个参数

。将此参数设的太大通常是一个坏主意。如果你将该指令的值设置为比MinSpareServers 小,Apache将会自动将其修改成"MinSpareServers+1"。

ServerLimit：服务器允许配置的进程数上限。只有在你需要将MaxClients设置成高于默认值256的时候才需要使用。要将此指令的值保持和MaxClients一样。修改此指令的值必须

完全停止服务后再启动才能生效，以restart方式重启动将不会生效。

MaxClients：用于伺服客户端请求的最大请求数量（最大子进程数），任何超过MaxClients限制的请求都将进入等候队列。默认值是256，如果要提高这个值必须同时提高

ServerLimit的值。笔者建议将初始值设为(以Mb为单位的最大物理内存/2),然后根据负载情况进行动态调整。比如一台4G内存的机器，那么初始值就是4000/2=2000。

MaxRequestsPerChild：设置每个子进程在其生存期内允许伺服的最大请求数量。到达MaxRequestsPerChild的限制后，子进程将会结束。如果MaxRequestsPerChild为"0"，子进程

将永远不会结束。将MaxRequestsPerChild设置成非零值有两个好处：可以防止(偶然的)内存泄漏无限进行而耗尽内存；

给进程一个有限寿命，从而有助于当服务器负载减轻的时候减少活动进程的数量。

如果设置为非零值，建议设为10000-30000之间的一个值。

公式：MaxClients<=ServerLimit

#设置worker多路处理模块

<IfModule mpm\_worker\_module>

StartServers 5

ServerLimit 20

ThreadLimit 200

MaxClients 4000

MinSpareThreads 25

MaxSpareThreads 250

ThreadsPerChild 200

MaxRequestsPerChild 0

</IfModule>

StartServers：设置服务器启动时建立的子进程数量。因为子进程数量动态的取决于负载的轻重,所有一般没有必要调整这个参数。

ServerLimit：服务器允许配置的进程数上限。只有在你需要将MaxClients和ThreadsPerChild设置成需要超过默认值16个子进程的时候才需要使用这个指令。不要将该指令的值设

置的比MaxClients 和ThreadsPerChild需要的子进程数量高。修改此指令的值必须完全停止服务后再启动才能生效，以restart方式重启动将不会生效。

ThreadLimit：设置每个子进程可配置的线程数ThreadsPerChild上限,该指令的值应当和ThreadsPerChild可能达到的最大值保持一致。修改此指令的值必须完全停止服务后再启动

才能生效，以restart方式重启动将不会生效。

MaxClients：用于伺服客户端请求的最大接入请求数量（最大线程数）。任何超过MaxClients限制的请求都将进入等候队列。默认值是"400"，16 (ServerLimit)乘以25

(ThreadsPerChild)的结果。因此要增加MaxClients的时候，你必须同时增加 ServerLimit的值。笔者建议将初始值设为(以Mb为单位的最大物理内存/2),然后根据负载情况进行动

态调整。比如一台4G内存的机器，那么初始值就是4000/2=2000。

MinSpareThreads：最小空闲线程数,默认值是"75"。这个MPM将基于整个服务器监视空闲线程数。如果服务器中总的空闲线程数太少，子进程将产生新的空闲线程。

MaxSpareThreads：设置最大空闲线程数。默认值是"250"。这个MPM将基于整个服务器监视空闲线程数。如果服务器中总的空闲线程数太多，子进程将杀死多余的空闲线程。

MaxSpareThreads的取值范围是有限制的。Apache将按照如下限制自动修正你设置的值：worker要求其大于等于 MinSpareThreads加上ThreadsPerChild的和。

ThreadsPerChild：每个子进程建立的线程数。默认值是25。子进程在启动时建立这些线程后就不再建立新的线程了。每个子进程所拥有的所有线程的总数要足够大，以便可以处理

可能的请求高峰。

MaxRequestsPerChild：设置每个子进程在其生存期内允许伺服的最大请求数量。到达MaxRequestsPerChild的限制后，子进程将会结束。如果MaxRequestsPerChild为"0"，子进程

将永远不会结束。将MaxRequestsPerChild设置成非零值有两个好处：可以防止(偶然的)内存泄漏无限进行而耗尽内存；

给进程一个有限寿命，从而有助于当服务器负载减轻的时候减少活动进程的数量。

如果设置为非零值，建议设为10000-30000之间的一个值。

公式：

ThreadLimit >= ThreadsPerChild

MaxClients <= ServerLimit \* ThreadsPerChild 必须是ThreadsPerChild的倍数

MaxSpareThreads >= MinSpareThreads+ThreadsPerChild

#配置Apache服务器默认设置 httpd-default.conf

Timeout 300

KeepAlive Off

MaxKeepAliveRequests 100

KeepAliveTimeout 5

UseCanonicalName Off

AccessFileName .htaccess

ServerTokens Prod

ServerSignature Off

HostnameLookups Off

Timeout：设置服务器在断定请求失败前等待的秒数。默认值300。

KeepAlive：设置是否启用HTTP持久链接，On 代表打开，Off 代表关闭。如果用于同一页面包含大量静态文件的应用，设置为On，以提高性能；如果用于主要为动态页面的应用，

设置为Off，以节约内存资源；如果服务器前跑有squid或者其它七层设备，设置为On。

MaxKeepAliveRequests：限制当启用KeepAlive时，每个连接允许的请求数量。如果将此值设为"0"，将不限制请求的数目。笔者建议将此值设为100-500之间的一个值，以确保最优

的服务器性能。

KeepAliveTimeout：设置持久链接中服务器在两次请求之间等待的秒数。对于高负荷服务器来说，KeepAliveTimeout值较大会导致一些性能方面的问题：超时值越大，与空闲客户

端保持连接的进程就越多。

UseCanonicalName：配置服务器如何确定它自己的域名，可选值为On | Off | DNS。DNS用于为大量基于IP的虚拟主机支持那些古董级的不提供"Host:"头的浏览器使用。笔者建议

设置为Off。

AccessFileName：设置分布式配置文件的名字，默认为.htaccess。如果为某个目录启用了分布式配置文件功能，那么在向客户端返回其中的文档时，服务器将在这个文档所在的各

级目录中查找此配置文件，因此会带来性能问题，笔者建议关闭分布式配置文件功能。

ServerTokens：控制服务器回应给客户端的"Server:"应答头是否包含关于服务器操作系统类型和编译进的模块描述信息，同时还控制着 ServerSignature指令的显示内容。可选值

为Full | OS | Minor | Minimal | Major | Prod。笔者建议设置为显示最少信息的Prod。

ServerSignature：配置服务器生成页面的页脚，可选值为On | Off | EMail。采用On会简单的增加一行关于服务器版本和正在伺服的虚拟主机的ServerName，而EMail设置会额外

创建一个指向 ServerAdmin的"mailto:"部分。建议使用默认值Off。

HostnameLookups：设置是否启用对客户端IP的DNS查找，可选值为On | Off | Double。DNS查询会造成明显的时间消耗，建议设置为Off。

#配置mod\_deflate压缩模块 httpd-deflate.conf

mod\_deflate模块提供了DEFLATE输出过滤器，允许服务器在将输出内容发送到客户端以前进行压缩，以节约带宽。

编辑mod\_deflate压缩模块设置文件：

<IfModule mod\_deflate.c>

DeflateMemLevel 9

DeflateWindowSize 15

DeflateBufferSize 8096

DeflateCompressionLevel 8

## 调试时去掉下面5行前的注释符，用以记录压缩日志

#DeflateFilterNote Input instream

#DeflateFilterNote Output outstream

#DeflateFilterNote Ratio ratio

#LogFormat '"%r" %b %{outstream}n/%{instream}n (%{ratio}n%%) "%{User-agent}i"' deflate

#CustomLog logs/deflate\_log deflate

## 插入DEFLATE过滤器，以启用输出压缩

　SetOutputFilter DEFLATE

　## Netscape 4.x 有一些问题，仅压缩txt/html类型

BrowserMatch ^Mozilla/4 gzip-only-text/html

　## Netscape 4.06-4.08不能处理任何压缩内容，不进行压缩

BrowserMatch ^Mozilla/4.0[678] no-gzip

## MSIE 会伪装成 Netscape ，但是事实上它没有问题

BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

## 设置图片、mp3、压缩格式文件等不需要进行压缩内容

SetEnvIfNoCase Request\_URI .(?:gif|jpe?g|png)$ no-gzip dont-vary

SetEnvIfNoCase Request\_URI .(?:exe|t?gz|zip|bz2|sit|rar)$ no-gzip dont-vary

SetEnvIfNoCase Request\_URI .(?:pdf|mov|avi|mp3|mp4|rm)$ no-gzip dont-vary

## 确保代理不会发送错误的内容

Header append Vary User-Agent env=!dont-vary

</IfModule>

DeflateMemLevel：指定zlib在压缩时最多可以使用多少内存，取值范围在1到9之间。

DeflateWindowSize：定义zlib压缩窗口(compression window)的大小，取值范围在1到15之间，通常窗口越大压缩效果越好。

DeflateBufferSize：定义zlib一次压缩的片断的字节数，默认为8096。

DeflateCompressionLevel：设定压缩程度，越高的压缩程度就会有越好的压缩效果，同时也意味着占用越多的CPU资源。取值范围在 1(最低压缩率) 到 9(最高压缩率)之间，如果

不设置此指令，默认为zlib的默认值。

DeflateFilterNote：在日志中放置压缩率标记。

语法为：DeflateFilterNote type notename

notename就表示这个压缩率标记的名字，你可以为了某种统计目的将这个标记的名字添加到访问日志中。

type指定notename标记所记录的数据类型：Input，在标记中存储过滤器输入流的字节数；Output，在标记中存储过滤器输出流的字节数；Ratio，在标记中存储过滤器的压缩比(输

出/输入\*100)，这是type的默认值。

SetOutputFilter：设置用于在服务器应答发送到客户端之前使用的过滤器。如果多于一个过滤器，必须按照处理内容的顺序用分号(;)分隔。

BrowserMatch：基于User-Agent头有条件地设置环境变量。

SetEnvIfNoCase：根据大小写无关的客户端请求属性设置环境变量。

Headers：配置HTTP应答头。这个指令可以替换、合并、删除HTTP应答头。应答头紧跟在内容处理器和输出过滤器完工之后生成，这时候才能对头进行修改。

#配置mod\_expires模块

这个模块控制服务器应答时的Expires头内容和Cache-Control头的max-age指令。有效期(expiration date)可以设置为相对于源文件的最后修改时刻或者客户端的访问时刻。这些

HTTP头向客户端表明了文档的有效性和持久性。如果有缓存，文档就可以从缓存(除已经过期)而不是从服务器读取。接着，客户端考察缓存中的副本，看看是否过期或者失效，以

决定是否必须从服务器获得更新。

编辑mod\_expires设置文件：httpd-expires.conf

## mod\_expires设置文件

<IfModule mod\_expires.c>

ExpiresActive on

　 ExpiresDefault A300

ExpiresByType text/html A300

ExpiresByType image/gif A2592000

ExpiresByType image/jpeg A2592000

ExpiresByType image/png A2592000

ExpiresByType text/css A2592000

ExpiresByType text/js A2592000

ExpiresByType text/javascript A2592000

ExpiresByType application/x-javascript A2592000

</IfModule>

ExpiresActive：对其作用范围内的文档启用或禁用产生Expires和Cache-Control头的功能。若设置为 Off 则不会为其作用范围内的任何文档生成Expires和Cache-Control头(除非

被更低一层的规则改写，比如.htaccess文件)。若设置为 On 则会按照ExpiresByType和ExpiresDefault指令定义的标准为其作用范围内的文档生成Expires和Cache-Control 头。

注意，这个指令并不保证Expires或Cache-Control头一定会产生。如果定义的标准不规范，将不会产生这两个头，其效果是好像从未设置过这个指令一样。

ExpiresByType：为指定MIME类型的文档配置Expires头的值。

语法为：ExpiresByType MIME-type <code>seconds

seconds参数设置了添加到基准时间以构造有效期限的秒数。Cache-Control: max-age的计算方法是从有效期减去当前请求时间并转化为秒数。

基准时刻可以是源文件的最后修改时刻或者客户端对源文件的访问时刻，至于使用那一个则由<code>指定。"M"表示源文件的最后修改时刻，"A"表示客户端对源文件的访问时刻。

需要注意的是<code>和seconds之间没有空格。如果使用"M"，所有当前缓存中的文档副本都将在同一时刻过期，这个可能对定期更新的URL(比如位于同一位置的每周通告)很有好处

。如果使用"A"，则每个客户端所得到的有效期是不一样的，这个可能对那些几乎不更新的图片文件很有好处，特别是对于一组都引用了相同图片的相关文档。

ExpiresDefault：设置其作用范围内的所有文档的默认有效期的计算方法，它可以被ExpiresByType指令基于MIME类型被改写。

语法为：ExpiresDefault <code>seconds

<code>seconds同ExpiresByType一致。

#最后配置一下虚拟主机 httpd-vhost.conf

#这里重点介绍一下基于多端口的配置

NameVirtualHost \*:81

NameVirtualHost \*:82

NameVirtualHost \*:83

<VirtualHost \*:81>

ServerAdmin jafy@jafy00.com

DocumentRoot /www1

ServerName www.jafy00.com

ErrorLog logs/www1-error.log

CustomLog logs/www1-access\_log common

</VirtualHost>

<VirtualHost \*:82>

<Directory "/www2" >

Options FollowSymLinks

#Deny from all

Allow from all

</Directory>

ServerAdmin jafy@jafy00.com

DocumentRoot /www2

ServerName www.jafy00.com

DirectoryIndex index.php index.html

ErrorLog logs/www2-error.log

CustomLog logs/www2-access\_log combined

</VirtualHost>

# （总结）Nginx配置文件nginx.conf中文详解

#定义Nginx运行的用户和用户组

user www www; #nginx进程数，建议设置为等于CPU总核心数。

worker\_processes 8;

#全局错误日志定义类型，[ debug | info | notice | warn | error | crit ]

error\_log /var/log/nginx/error.log info;

#进程文件

pid /var/run/nginx.pid;

#一个nginx进程打开的最多文件描述符数目，理论值应该是最多打开文件数（系统的值ulimit -n）与nginx进程数相除，但是nginx分配请求并不均匀，所以建议与ulimit -n的值保持一致。

worker\_rlimit\_nofile 65535;

#工作模式与连接数上限

events

{

#参考事件模型，use [ kqueue | rtsig | epoll | /dev/poll | select | poll ]; epoll模型是Linux 2.6以上版本内核中的高性能网络I/O模型，如果跑在FreeBSD上面，就用kqueue模型。

use epoll;

#单个进程最大连接数（最大连接数=连接数\*进程数）

worker\_connections 65535;

}

#设定http服务器

http

{

include mime.types; #文件扩展名与文件类型映射表

default\_type application/octet-stream; #默认文件类型

#charset utf-8; #默认编码

server\_names\_hash\_bucket\_size 128; #服务器名字的hash表大小

client\_header\_buffer\_size 32k; #上传文件大小限制

large\_client\_header\_buffers 4 64k; #设定请求缓

client\_max\_body\_size 8m; #设定请求缓

sendfile on; #开启高效文件传输模式，sendfile指令指定nginx是否调用sendfile函数来输出文件，对于普通应用设为 on，如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络I/O处理速度，降低系统的负载。注意：如果图片显示不正常把这个改成off。

autoindex on; #开启目录列表访问，合适下载服务器，默认关闭。

tcp\_nopush on; #防止网络阻塞

tcp\_nodelay on; #防止网络阻塞

keepalive\_timeout 120; #长连接超时时间，单位是秒

#FastCGI相关参数是为了改善网站的性能：减少资源占用，提高访问速度。下面参数看字面意思都能理解。

fastcgi\_connect\_timeout 300;

fastcgi\_send\_timeout 300;

fastcgi\_read\_timeout 300;

fastcgi\_buffer\_size 64k;

fastcgi\_buffers 4 64k;

fastcgi\_busy\_buffers\_size 128k;

fastcgi\_temp\_file\_write\_size 128k;

#gzip模块设置

gzip on; #开启gzip压缩输出

gzip\_min\_length 1k; #最小压缩文件大小

gzip\_buffers 4 16k; #压缩缓冲区

gzip\_http\_version 1.0; #压缩版本（默认1.1，前端如果是squid2.5请使用1.0）

gzip\_comp\_level 2; #压缩等级

gzip\_types text/plain application/x-javascript text/css application/xml;

#压缩类型，默认就已经包含text/html，所以下面就不用再写了，写上去也不会有问题，但是会有一个warn。

gzip\_vary on;

#limit\_zone crawler $binary\_remote\_addr 10m; #开启限制IP连接数的时候需要使用

upstream blog.ha97.com {

#upstream的负载均衡，weight是权重，可以根据机器配置定义权重。weigth参数表示权值，权值越高被分配到的几率越大。

server 192.168.80.121:80 weight=3;

server 192.168.80.122:80 weight=2;

server 192.168.80.123:80 weight=3;

}

#虚拟主机的配置

server

{

#监听端口

listen 80;

#域名可以有多个，用空格隔开

server\_name www.ha97.com ha97.com;

index index.html index.htm index.php;

root /data/www/ha97;

location ~ .\*.(php|php5)?$

{

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

#图片缓存时间设置

location ~ .\*.(gif|jpg|jpeg|png|bmp|swf)$

{

expires 10d;

}

#JS和CSS缓存时间设置

location ~ .\*.(js|css)?$

{

expires 1h;

}

#日志格式设定

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

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

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

#定义本虚拟主机的访问日志

access\_log /var/log/nginx/ha97access.log access;

#对 "/" 启用反向代理

location / {

proxy\_pass http://127.0.0.1:88;

proxy\_redirect off;

proxy\_set\_header X-Real-IP $remote\_addr;

#后端的Web服务器可以通过X-Forwarded-For获取用户真实IP

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

#以下是一些反向代理的配置，可选。

proxy\_set\_header Host $host;

client\_max\_body\_size 10m; #允许客户端请求的最大单文件字节数

client\_body\_buffer\_size 128k; #缓冲区代理缓冲用户端请求的最大字节数，

proxy\_connect\_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)

proxy\_send\_timeout 90; #后端服务器数据回传时间(代理发送超时)

proxy\_read\_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)

proxy\_buffer\_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小

proxy\_buffers 4 32k; #proxy\_buffers缓冲区，网页平均在32k以下的设置

proxy\_busy\_buffers\_size 64k; #高负荷下缓冲大小（proxy\_buffers\*2）

proxy\_temp\_file\_write\_size 64k;

#设定缓存文件夹大小，大于这个值，将从upstream服务器传

}

#设定查看Nginx状态的地址

location /NginxStatus {

stub\_status on;

access\_log on;

auth\_basic "NginxStatus";

auth\_basic\_user\_file conf/htpasswd;

#htpasswd文件的内容可以用apache提供的htpasswd工具来产生。

}

#本地动静分离反向代理配置

#所有jsp的页面均交由tomcat或resin处理

location ~ .(jsp|jspx|do)?$ {

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_pass http://127.0.0.1:8080;

}

#所有静态文件由nginx直接读取不经过tomcat或resin

location ~ .\*.(htm|html|gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt|pdf|xls|mp3|wma)$

{ expires 15d; }

location ~ .\*.(js|css)?$

{ expires 1h; }

}

}

更详细的模块参数请参考：http://wiki.nginx.org/Main