# Ngx\_cycle\_t结构讲解

在系统整个运行过程中，需要使用的一些参数、资源需要统一的管理，nginx把这个任务交给了ngx\_cycle\_t

struct ngx\_cycle\_s {

void \*\*\*\*conf\_ctx; //指向模块的配置

ngx\_pool\_t \*pool; //内存池

ngx\_log\_t \*log; //日志

ngx\_log\_t new\_log; //新日志

ngx\_connection\_t \*\*files; //存放socket和connection之间的关系

ngx\_connection\_t \*free\_connections; //空闲连接池

ngx\_uint\_t free\_connection\_n; //空闲连接的个数

ngx\_array\_t listening; //监听者

ngx\_array\_t pathes; //系统所使用的路径集合

ngx\_list\_t open\_files; //打开文件的集合

ngx\_list\_t shared\_memory; //共享内存

ngx\_uint\_t connection\_n; //总的预先创建的connection数目

ngx\_uint\_t files\_n; //总文件数目

ngx\_connection\_t \*connections; //所有的connection

ngx\_event\_t \*read\_events; //读事件

ngx\_event\_t \*write\_events; //写事件

ngx\_cycle\_t \*old\_cycle; //旧的全局信息

ngx\_str\_t conf\_file; //配置文件

ngx\_str\_t conf\_param; //配置参数

ngx\_str\_t conf\_prefix; //配置文件前缀

ngx\_str\_t prefix; //系统安装路径前缀

ngx\_str\_t lock\_file; //锁文件

ngx\_str\_t hostname; //主机名

}

conf\_ctx 指向模块的配置

pool 内存池

log 日志

new\_log 新日志

files 存放socket和connection之间的关系

free\_connections 空闲连接池

free\_connection\_n 空闲连接的个数

listening 监听者，用于监听外部的连接

pathes 整个系统中使用的路径集合，比如 client\_body\_temp ,proxy\_temp 等，这个也是在nginx\_cycle.c 中的ngx\_init\_cycle中分配的，和上面listening分配方式差不多，实际路径的添加是在 ngx\_file.c 中的 ngx\_add\_path 函数中

open\_files 打开文件的集合，当需要打开一个文件时，先在这里找，如果找到直接返回，否则打开文件，放到这个集合中。文件代开在ngx\_conf\_file.c中的ngx\_conf\_open\_file函数中实现的

shared\_memory 系统所有的共享内存，如果往里面添加共享内存，需要检查下是否已经存在了，存在就不用放了，否则就放进去，在函数ngx\_shared\_memory\_add函数中实现。

connection\_n; 总的预先创建的connection数目

files\_n 总的文件数目，不过好像没有设置，待确认

connections 所有的连接，free\_connection也是指向它所指向的空间的

read\_events write\_events 读写事件，他们的个数和connection一样的，并且在初始化的时候，一个connection关联一个读事件、一个写事件。

old\_cycle 旧的全局信息

conf\_file 配置文件,如果系统默认，那么直接从objs/ngx\_auto\_conf.h中读取，也可以在通过参数设定

conf\_param; 配置参数

conf\_prefix; 配置文件前缀

prefix; 系统安装路径前缀

lock\_file; 锁文件

hostname; 主机名