# 事件驱动

nginx支持多种事件驱动框架，本文主要介绍linux下的epoll事件驱动。nginx中使用的是epoll边缘触发模式。

ngx\_event\_module本身是一个事件驱动的框架。在ngx\_event.h文件中可以找到框架本身提供了一些事件驱动需要的方法，比如ngx\_add\_event、ngx\_del\_event等。具体会使用哪个事件驱动的模块，则是根据配置(没有配置的情况下根据操作系统)来选择。

## 关键数据结构

## 关键函数

### ngx\_event\_module\_init

入参cycle：ngx\_init\_cycle节点创建的数据结构。

出参：正常NGX\_OK，异常NGX\_ERROR。

1. 调用点在ngx\_init\_cycle中，通过ngx\_init\_module函数调用。
2. 创建了一块共享内存用来将记录ngx中各种状态数据，比如accept锁，连接数量等。共享内存是一块连续的内存，各个变量使用内存中不同的偏移记录数据。

### ngx\_event\_process\_init

入参cycle：ngx\_init\_cycle节点创建的数据结构。

出参：正常NGX\_OK，异常NGX\_ERROR。

1. 调用点在ngx\_work\_process\_init（如果是单进程的话是在ngx\_single\_process\_cycle）。
2. for循环遍历所有的NGX\_EVENT\_MODULE找到与use配置相同的一个，并执行init函数。这里关注ngx\_epoll\_init。
3. 创建connections数组，read\_events数组和write\_events数组。这三个数组是等长的，每个connections数组成员对应一个read\_events成员和write\_events成员。
4. do...while循环是将connections数组变成一个单向循环链表。free\_connections是指向数组中的一个空闲连接。
5. 遍历所有的监听端口，建立监听端口和连接之间的关系。设置监听端口对应连接的读事件回调函数并设置epoll读事件。
6. rev->handler = (c->type == SOCK\_STREAM)...给每个socket设置了连接的读事件设置了回调函数，对于TCP socket设置的回调函数是ngx\_event\_accept。对于UDP socket设置的回调函数是ngx\_event\_recvmsg。

### ngx\_get\_connection

入参s：监听端口socket的fd。

入参log：用于日志记录的数据结构。

出参：返回一个可用的ngx\_connection\_t的数据结构。

1. 将当前free\_connections指向的连接作为函数要返回的连接。然后将free\_connections指向下一个空闲连接。
2. connection结构中的fd设置为入参socket的fd。
3. 对rev和wev的instance进行取反操作。通过这个instance后续用来判断事件是否过期。