## 二、运行流程分析

**1、进程模型**

nginx是一个master(主进程)＋多个worker(子进程)的工作模式，nginx启动的过程流程如下:

初始化master🡪初始化模块🡪创建多个worker🡪初始化worker🡪（如果编译时候指定支持线程，则初始化线程、退出线程）🡪循环process🡪收到退出信号，退出工作进程🡪退出master

而在这些子过程内部和子过程之间，又会有读取配置、创建配置、初始化配置、合并配置、http解析、http过滤、http输出、http代理等过程，在这些过程开始前后、过程中、结束前后等时机，nginx调用合适的模块接口完成特定的任务。

一切伟大的壮举都有一个微不足道的开始,我们就从src/core/nginx.c:main()开始吧。

各子目录解释：

auto：当执行./configure做编译前的配置时候，会使用此目录下所依赖的一些自动化脚本来执行，主要用于检查环境和生成几个源码、makefile文件等，其子目录比较多，这里不画出来了。

conf：运行时调用的配置文件模版，比如我们最常用的nginx.conf文件出自这里。

contrib：字符集转换脚本。

html：默认的访问文件存放目录，放一些默认访问的html页面。

man：存放帮助文档。

objs：这里存放编译生成的目标文件，他底下还有一个src目录树，和nginx-1.1.7/src下的目录树结构保持一致，存放的都是编译出来的.o文件，这里也不画出了。另外有一点注意，在objs目录下，configure时会生成三个文件: ngx\_auto\_config.h，ngx\_modules.c，ngx\_auto\_headers.h。这三个文件是做完系统检查和确定要使用那些模块配置后生成的，下一步的编译都会用到这三个文件。

src：核心源代码目录，我们要分析的源码均出自这里，其子目录如下

core，该目录存放核心模块的代码，一些基础数据结构在这里定义，是分析源码的入口。

http，HTTP协议模块，作为WEB服务器和代理服务器运行时的核心模块。

mail，Mail模块，作为pop3/imap/smtp代理服务器运行时的核心模块。

event : 事件响应和处理逻辑。

os : 对各操作系统平台抽象逻辑的封装，解决底层依赖问题，让nginx可以运行在各种平台而不用关心os层面实现。

misc : nginx 的一些utils,定义了test和profiler的一些外围模块。

**2、nginx自定义基础数据结构**

整个nginx源码中，到处充斥着nginx的自定义数据结构类型，连int、char这种c语言最基本的类型都很少出现，更多的是以ngx\_开头的数据类型，有的是struct结构，有的是用typedef重新命名的数据类型，还有队列、数组、string、内存池等结构以及它们的上层操作实现也是nginx自定义的。

nginx 自定义结构体命名规则ngx\_XXX\_t，它们分散在core和os目录下的许多头文件中，而在src/core/ngx\_core.h中把他们大部分都包括进来，而所有的c文件都会包含这个ngx\_core.h头文件，这样就可以使用任何一种数据结构了。

以下展示中，红色字段就是在nginx中经常使用的数据类型名称

src/core/ngx\_config.h:

typedef intptr\_t ngx\_int\_t; //整型

typedef uintptr\_t ngx\_uint\_t; //无符号整型

typedef intptr\_t ngx\_flag\_t; //标志位类型

intptr\_t这些数据类型可能看不懂，其实它们在/usr/include/stdint.h的定义为：

/\* Types for `void \*' pointers. \*/

#if \_\_WORDSIZE == 64

# ifndef \_\_intptr\_t\_defined

typedef long int intptr\_t;

# define \_\_intptr\_t\_defined

# endif

typedef unsigned long int uintptr\_t;

#else

# ifndef \_\_intptr\_t\_defined

typedef int intptr\_t;

# define \_\_intptr\_t\_defined

# endif

typedef unsigned int uintptr\_t;

#endif

src/core/ngx\_string.h:

typedef struct {…} ngx\_str\_t; //字符串类型

src/core/ngx\_queue.h:

typedef struct ngx\_queue\_s ngx\_queue\_t; //队列

struct ngx\_queue\_s {...};

src/core/ngx\_list.h:

typedef struct ngx\_list\_part\_s ngx\_list\_part\_t; //列表

struct ngx\_list\_part\_s {...};

src/core/ngx\_core.h:

typedef struct ngx\_module\_s ngx\_module\_t; //模块结构

typedef struct ngx\_conf\_s ngx\_conf\_t; //配置文件结构

typedef struct ngx\_cycle\_s ngx\_cycle\_t; //循环结构，处理配置信息，循环载入和初始化模块的时候用到。

typedef struct ngx\_pool\_s ngx\_pool\_t; //池结构，多用于内存池，也有其他类型使用。

typedef struct ngx\_chain\_s ngx\_chain\_t; //链结构

typedef struct ngx\_log\_s ngx\_log\_t; //日志类型

typedef struct ngx\_array\_s ngx\_array\_t; //数组类型

typedef struct ngx\_open\_file\_s ngx\_open\_file\_t; //打开的文件

typedef struct ngx\_command\_s ngx\_command\_t; //内部命令字定义

typedef struct ngx\_file\_s ngx\_file\_t; //文件描述符封装

typedef struct ngx\_event\_s ngx\_event\_t; //事件机制结构封装

typedef struct ngx\_event\_aio\_s ngx\_event\_aio\_t; //事件IO结构

typedef struct ngx\_connection\_s ngx\_connection\_t; //socket连接封装

ngx\_core.h中这些结构的实现，在下面文件中

src/core/ngx\_conf\_file.h:

struct ngx\_module\_s {…}

struct ngx\_conf\_s {…}

struct ngx\_open\_file\_s {…}

struct ngx\_command\_s {…}

src/core/ngx\_cycle.h:

struct ngx\_cycle\_s {…}

src/core/ngx\_palloc.h:

struct ngx\_ pool\_s {…}

src/core/ngx\_buf.h:

struct ngx\_chain\_s {…}

src/core/ngx\_log.h:

struct ngx\_log\_s {…}

src/core/ngx\_array.h:

struct ngx\_array\_s {…}

src/core/ngx\_file.h:

struct ngx\_file\_s {…}

src/event/ngx\_event.h:

struct ngx\_event\_s {…}

struct ngx\_event\_aio\_s {…}

src/core/ngx\_connection.h:

struct ngx\_connection\_s {…}

关于各数据结构的详细信息，以后单独做一个版面，这里先把它们列出来，以方便我们后边分析的时候，可以知道这些基本结构都是做什么用的。

**3、自动化脚本**

自动化脚本用于在安装nginx前，使用configure命令来生成Makefile文件，以方便后边的编译和安装。值得一提的是自动脚本不是用autoconf生成，而是手工编写的，对于编写自动脚本很有借鉴。

自动化脚本包括两部分，一个是根目录下的configure文件，另外就是auto目录下的所有脚本。configure是总的驱动，它通过组合auto目录下不同功能的脚本来完成以下工作：

* 检查编译环境
* 生成源码和makefile文件

环境检查主要是三个部分：auto/cc用来检查编译器版本支持、auto/os检查操作系统版本支持、auto/lib第三方库支持。自动编译用于检查某个特性的代码片段，根据编译器的输出情况判定是否支持该种特性，他最终会生成一个简单的nginx程序。根据检查的情况，如果环境足以支持运行一个简单版本的nginx，就会生成 Makefile和c代码，放到objs目录下。当然，也可能会失败，假如对于不支持的特性，如果没有屏蔽掉相关的模块，自动脚本就会失败。

生成的c源码有三个文件：

1.ngx\_auto\_config.h 根据环境检查结果声明的一些平台相关的宏定义，这个头文件被ngx\_core.h引用，所有的源码都可以使用到，这样可移植在不同的编译器和操作系统平台上；

2.ngx\_auto\_headers.h 声明一些系统头文件是否存在；

3.ngx\_modules.c 默认的模块的声明，如果想去掉一些模块，只要修改这个文件即可。

auto目录下的脚本：

1.auto/options脚本，设定配置选项。

2.auto/init设置初始变量。

3.auto/sources源码相关变量脚本

…

还有一些检查环境、模块的，这里不列出来了，会有单独的板块详细解释。

## 二、