WEBCRAWLER

网 络 爬 虫 实 训 项 目

文档版本： 1.0.0.1
编写单位： 达内IT培训集团 C++教学研发部
编写人员： 闵卫
定稿日期： 2015年11月20日 星期五

1. 项目概述

互联网产品形形色色，有产品导向的，有营销导向的，也有技术导向的，但是以技术见长的互联网产品比例相对小些。搜索引擎是目前互联网产品中最具技术含量的产品，如果不是唯一，至少也是其中之一。

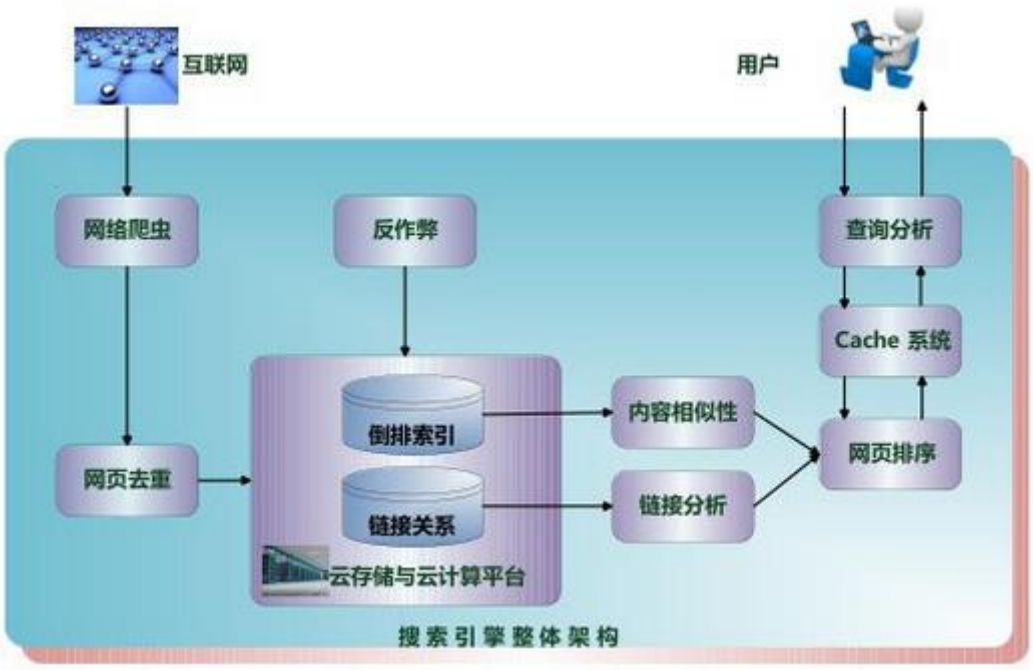
经过十几年的发展，搜索引擎已经成为互联网的重要入口之一，Twitter联合创始人埃文•威廉姆斯提出了“域名已死论”，好记的域名不再重要，因为人们会通过搜索进入网站。搜索引擎排名对于中小网站流量来说至关重要。了解搜索引擎简单界面背后的技术原理其实对每一个希望在互联网行业有所建树的信息技术人员都很重要。

1.1. 搜索引擎

作为互联网应用中最具技术含量的应用之一，优秀的搜索引擎需要复杂的架构和算法，以此来支撑对海量数据的获取、存储，以及对用户查询的快速而准确地响应。从架构层面，搜索引擎需要能够对以百亿计的海量网页进行获取、存储、处理的能力，同时要保证搜索结果的质量。如何获取、存储并计算如此海

量的数据？如何快速响应用户的查询？如何使得搜索结果尽可能满足用户对信息的需求？这些都是搜索引擎的设计者不得不面对的技术挑战。

下图展示了一个通用搜索引擎的基本结构。商业级别的搜索引擎通常由很多相互独立的模块组成，各个模块只负责搜索引擎的一部分功能，相互配合组成完整的搜索引擎：



搜索引擎的信息源来自于互联网网页，通过“网络爬虫”将整个“互联网”的信息获取到本地，因为互联网页面中有相当大比例的内容是完全相同或者近似重复的，“网页去重”模块会对此做出检测，并去除重复内容。

在此之后，搜索引擎会对网页进行解析，抽取网页主体内容，以及页面中包含的指向其它页面的所谓超链接。为了加快用户查询的响应速度，网页内容通过“倒排索引”这种高效查询数据结构来保存，而网页之间的链接关系也会予以保存。之所以要保存链接关系，是因为这种关系在网页相关性排序阶段是可利用的，通过“链接分析”可以判断页面的相对重要性，对于为用户提供准确的搜索结果帮助很大。

由于网页数量太多，搜索引擎不仅需要保存网页的原始信息，还要保存一些中间处理结果，使用单台或者少量的计算机明显是不现实的。Google等商业搜索引擎提供商，为此开发了一整套云存储与云计算平台，使用数以万计的普通PC

搭建了海量信息的可靠存储与计算架构，以此作为搜索引擎及其相关应用的基础支撑。优秀的云存储与云计算平台已经成为大型商业搜索引擎的核心竞争力。

以上所述是搜索引擎如何获取并存储海量的网页相关信息。这些功能因为不需要实时计算，所以可以被看作是搜索引擎的后台计算系统。搜索引擎的首要目标当然是为用户提供准确而全面的搜索结果，因此响应用户查询并实时提供准确结果便构成了搜索引擎的前台计算系统。

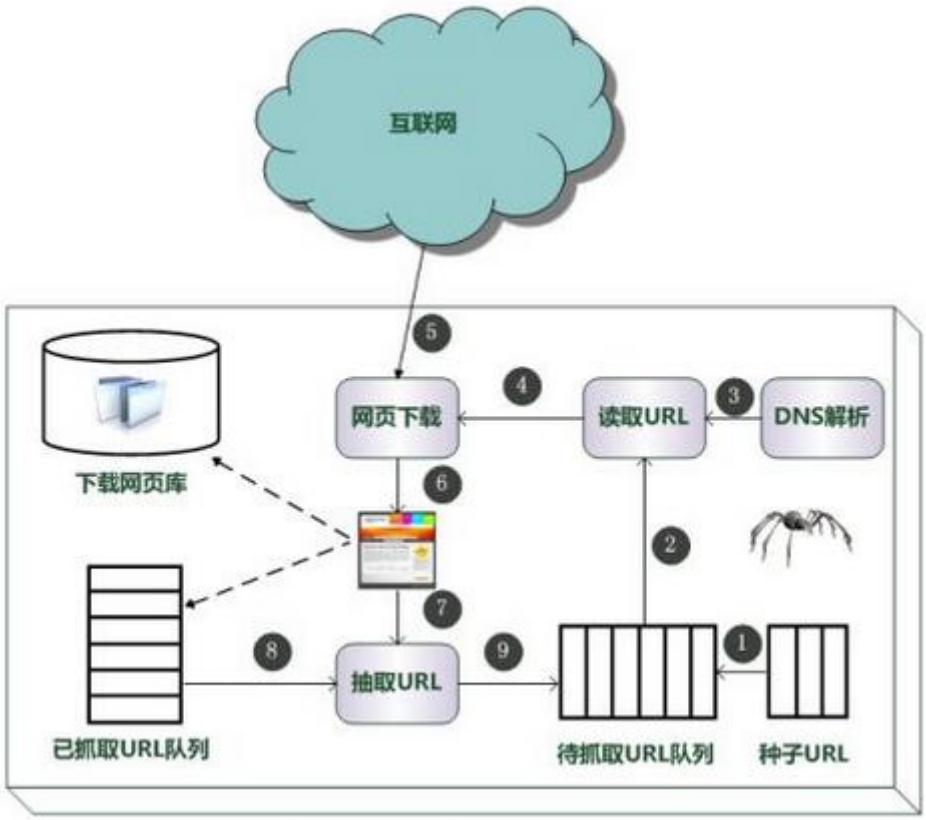
当搜索引擎接收到用户的查询请求后，首先需要对查询词进行分析，通过与用户信息的结合，正确推导出用户的真实搜索意图。此后，先在“Cache系统”所维护的缓存中查找。搜索引擎的缓存存储了不同的搜索意图及其相对应的搜索结果。如果在缓存中找到满足用户需求的信息，则直接将搜索结果返回给用户。这样既省掉了重复计算对资源的消耗，又加快了整个搜索过程的响应速度。而如果在缓存中没有找到满足用户需求的信息，则需要通过“网页排序”，根据用户的搜索意图，实时计算哪些网页是满足用户需求的，并排序输出作为搜索结果。而网页排序最重要的两个参考因素，一个是“内容相似性”，即哪些网页是和用户的搜索意图密切相关的；一个是网页重要性，即哪些网页是质量较好或相对重要的，而这往往可以从“链接分析”的结果中获得。综合以上两种考虑，前台系统对网页进行排序，作为搜索的最终结果。

除了上述功能模块，搜索引擎的“反作弊”模块近年来越来越受到重视。搜索引擎作为互联网用户上网的入口，对于网络流量的引导和分流至关重要，甚至可以说起着决定性的作用。因此，各种“作弊”方式也逐渐流行起来，通过各种手段将网页的搜索排名提前到与其网页质量不相称的位置，这会严重影响用户的搜索体验。所以，如何自动发现作弊网页并对其给于相应的惩罚，就成了搜索引擎非常重要的功能之一。

1.2. 网络爬虫

通用搜索引擎的处理对象是互联网网页，截至目前的网页数量数以百万计，所以搜索引擎首先面临的问题就是如何能够设计出高效的下载系统，将如此海量的网页数据传送到本地，在本地形成互联网网页的镜像备份。网络爬虫即扮演如此角色。它是搜索引擎中及其关键的基础构件。

网络爬虫的一般工作原理如下图所示：



❶ 从互联网网页中选择部分网页的链接作为“种子URL”，放入“待抓取URL队列”；

❷ 爬虫从“待抓取URL队列”中依次“读取URL”；

❸ 爬虫通过“DNS解析”将读到的URL转换为网站服务器的IP地址；

❹ 爬虫将网站服务器的IP地址、通信端口、网页路径等信息交给“网页下载”器；

❺ “网页下载”器负责从“互联网”上下载网页内容；

❻ 对于已经下载到本地的网页内容，一方面将其存储到“下载页面库”中，等待建立索引等后续处理，另一方面将其URL放入“已抓取URL队列”，后者显然是为了避免网页被重复抓取；

❼ 对于刚刚下载到本地的网页内容，还需要从中“抽取URL”；

❽ 在“已抓取URL队列”中检查所抽取的URL是否已被抓取过；

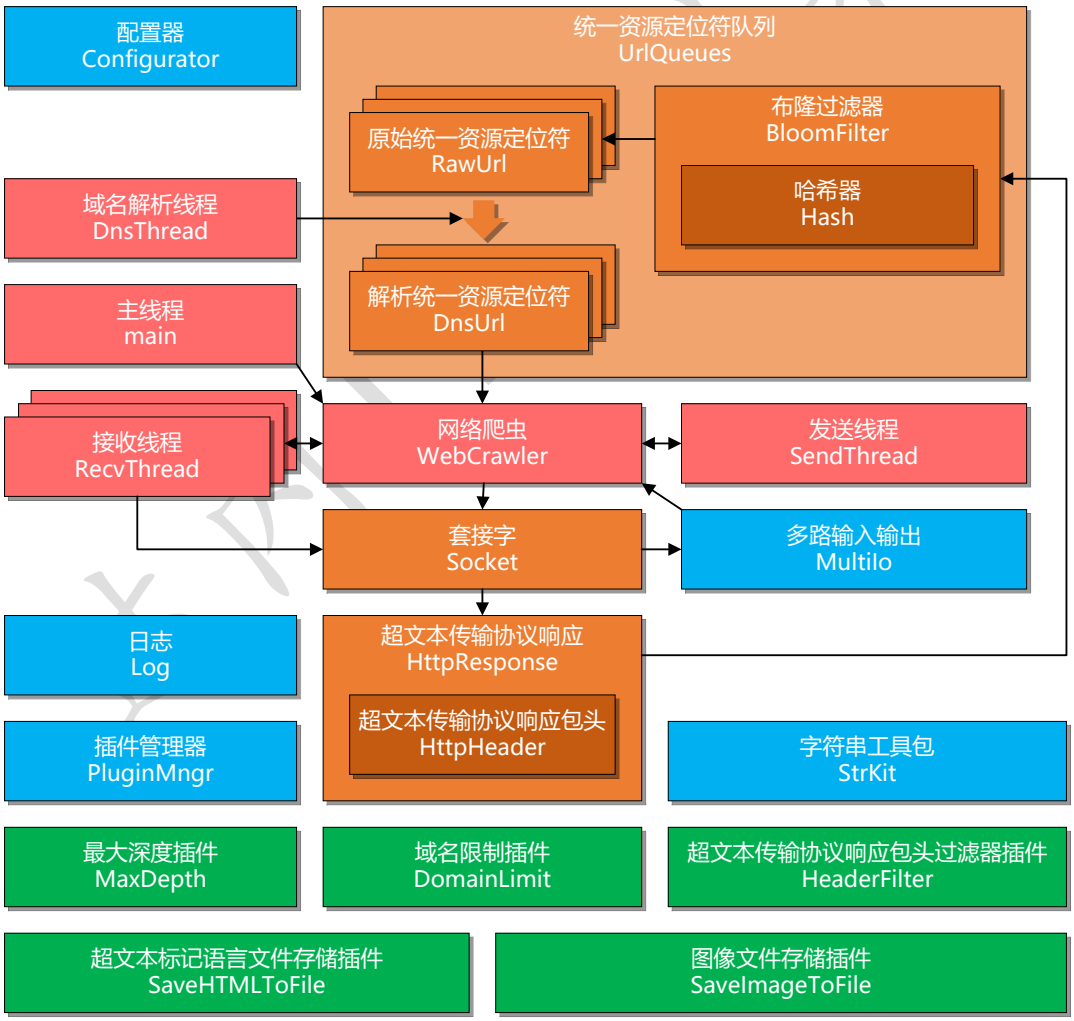
❾ 如果所抽取的URL没有被抓取过，则将其排入“待抓取URL队列”末尾，在之后的抓取调度中重复第❷步，下载这个URL所对应的网页。如此这般，形成

循环，直到“待抓取URL队列”空，这表示爬虫已将所有能够被抓取的网页尽数抓完，完成一轮完整的抓取过程。

以上所述仅仅是网络爬虫的一般性原理，具体实现过程中还可以有很多优化的空间，比如将“网页下载”以多线索（进程或线程）并发的方式实现，甚至将“DNS解析”也处理为并发的过程，以避免爬虫系统的I/O吞吐率受到网站服务器和域名解析服务器的限制。而对于“已抓取URL队列”则可以采用布隆排重表的方式加以优化，以降低其时间和空间复杂度。

2. 总体架构

本项目总体架构如下图所示：



2.1. 基础设施

2.1.1. 字符串工具包(StrKit)

常用字符串处理函数。

2.1.2. 日志(Log)

分等级，带格式的日志文件打印。

2.1.3. 配置器(Configurator)

从指定的配置文件中加载配置信息。

2.1.4. 多路输入输出(MultiIo)

封装epoll多路I/O系统调用，提供增加、删除和等待操作接口。

2.1.5. 插件管理器(PluginMngr)

加载插件并接受其注册，维护插件对象容器并提供调用其处理函数的外部接口。

2.2. 网络通信

2.2.1. 哈希器(Hash)

封装各种哈希算法函数。

2.2.2. 布隆过滤器(BloomFilter)

基于布隆算法，对欲加入队列的原始统一资源定位符进行过滤，以防止已被抓取过的URL再次入队，降低冗余开销同时避免无限循环。

2.2.3. 原始统一资源定位符(RawUrl)

提供原始形态的统一资源定位符字符串的简单包装，以及规格化等辅助支持。

2.2.4. 解析统一资源定位符(DnsUrl)

将原始形态的统一资源定位符字符串，解析为服务器域名、资源路径、服务器IP地址，乃至服务器通信端口等。

2.2.5. 统一资源定位符队列(UrlQueues)

封装原始统一资源定位符队列和解析统一资源定位符队列，提供线程安全的入队、出队操作，通过统一资源定位符过滤器排重，同时支持基于正则表达式的统一资源定位符抽取功能。

2.2.6. 套接字(Socket)

发送/接收超文本传输协议请求/响应，发送成功将套接字描述符加入多路I/O，接收成功抽取统一资源定位符压入队列。

2.2.7. 超文本传输协议响应包头(HttpHeader)

状态码和内容类型等关键信息。

2.2.8. 超文本传输协议响应(HttpResponse)

服务器统一资源定位符和超文本传输协议包头、包体及长度的简单封装。

2.3. 流程控制

2.3.1. 域名解析线程(DnsThread)

从原始统一资源定位符队列中弹出RawUrl对象，借助域名解析系统（DNS）获取服务器的IP地址，构造DnsUrl对象压入解析统一资源定位符队列。

2.3.2. 发送线程(SendThread)

通过WebCrawler对象启动新的抓取任务，从解析统一资源定位符队列中弹出DnsUrl对象，向HTTP服务器发送HTTP请求，并将套接字描述符放入MultiIo对象。

2.3.3. 接收线程(RecvThread)

由WebCrawler对象在从MultiIo对象中等到套接字描述符可读时动态创建，通过Socket对象接收超文本传输协议响应。

2.3.4. 网络爬虫(WebCrawler)

代表整个应用程序的逻辑对象，构建并维护包括日志、配置器、多路I/O、插件管理器、统一资源定位符队列、域名解析线程等在内的多个底层设施，提供诸如初始化、执行多路输入输出循环、启动抓取任务等外部接口。

2.3.5. 主线程(main)

主函数，处理命令行参数，初始化应用程序对象，进入多路I/O循环。

2.4. 外围扩展

2.4.1. 最大深度插件(MaxDepth)

根据配置文件的MAX_DEPTH配置项，对被抓取超链接的最大递归深度进行限制。

2.4.2. 域名限制插件(DomainLimit)

根据配置文件的INCLUDE_PREFIXES和EXCLUDE_PREFIXES配置项，对被抓取超链接的前缀进行限制。

2.4.3. 超文本传输协议响应包头过滤器插件(HeaderFilter)

根据配置文件的ACCEPT_TYPE配置项，对超文本传输协议响应的内容类型进行限制。

2.4.4. 超文本标记语言文件存储插件(SaveHTMLToFile)

将用超文本标记语言描述的页面内容保存到磁盘文件中。

2.4.5. 图像文件存储插件(SaveImageToFile)

将页面内容中引用的图像资源保存到磁盘文件中。

3. 工作流程

3.1. 主事件流

进程入口函数在进行必要的命令行参数处理和系统初始化以后，进入网络爬虫的多路输入输出循环，一旦发现某个与服务器相连的套接字有数据可读，即创

建接收线程，后者负责抓取页面内容，而前者继续于多路输入输出循环中等待其它套接字上的I/O事件。

3.2. 解析事件流

独立的域名解析线程实时监视原始统一资源定位符队列的变化，并将其中的每一条新近加入的原始统一资源定位符，借助域名解析系统转换为解析统一资源定位符，并压入解析统一资源定位符队列。

3.3. 发送事件流

不断从解析统一资源定位符队列弹出解析统一资源定位符，创建套接字，根据服务器的IP地址和通信端口发起连接请求，建立TCP连接，发送超文本传输协议请求包，并将套接字放入多路输入输出对象，由主事件流等待其数据到达事件。

3.4. 接收事件流

每个超文本传输线程通过已明确有数据可读的套接字接收来自服务器的超文本传输协议响应，并交由统一资源定位符队列进行超链接抽取和布隆排重过滤，直至压入原始统一资源定位符队列。在压入原始统一资源定位符队列之前，以及接收到超文本传输协议包头和包体之后，分别执行统一资源定位符插件、超文本传输协议包头插件和超文本标记语言插件的处理过程。

以上四个事件流，需要平行且独立地并发运行，并在共享资源和执行步调上保持适度的同步。

4. 目录结构

本项目的目录结构如下所示：

```
WebCrawler/
├── bin/
│   ├── WebCrawler
│   ├── WebCrawler.cfg
│   └── WebCrawler.scr
├── docs/
│   ├── 概要设计.pdf
```

```
│   └── 详细设计.pdf
├── download/
├── plugins/
│   ├── DomainLimit.cpp
│   ├── DomainLimit.h
│   ├── DomainLimit.mak
│   ├── DomainLimit.so
│   ├── HeaderFilter.cpp
│   ├── HeaderFilter.h
│   ├── HeaderFilter.mak
│   ├── HeaderFilter.so
│   ├── MaxDepth.cpp
│   ├── MaxDepth.h
│   ├── MaxDepth.mak
│   ├── MaxDepth.so
│   ├── SaveHTMLToFile.cpp
│   ├── SaveHTMLToFile.h
│   ├── SaveHTMLToFile.mak
│   ├── SaveHTMLToFile.so
│   ├── SaveImageToFile.cpp
│   ├── SaveImageToFile.h
│   ├── SaveImageToFile.mak
│   ├── SaveImageToFile.so
│   └── mkall
└── src/
    ├── BloomFilter.cpp
    ├── BloomFilter.h
    ├── Configurator.cpp
    ├── Configurator.h
    ├── DnsThread.cpp
    ├── DnsThread.h
    ├── Hash.cpp
    ├── Hash.h
    ├── Http.h
    ├── Log.cpp
    ├── Log.h
    ├── Main.cpp
    ├── Makefile
    ├── MultiIo.cpp
    ├── MultiIo.h
    ├── Plugin.h
    ├── PluginMngr.cpp
    ├── PluginMngr.h
    ├── Precompile.h
    ├── RecvThread.cpp
    ├── RecvThread.h
    ├── SendThread.cpp
    ├── SendThread.h
    ├── Socket.cpp
    ├── Socket.h
```

```
├── StrKit.cpp
├── StrKit.h
├── Thread.cpp
├── Thread.h
├── Url.cpp
├── Url.h
├── UrlFilter.h
├── UrlQueues.cpp
├── UrlQueues.h
├── WebCrawler.cpp
└── WebCrawler.h
```

其中bin目录存放可执行程序文件、启动画面文件和配置文件，docs目录存放项目文档，download目录存放爬虫下载的网页文件和图像文件，plugins目录存放扩展插件的源代码和共享库文件，src目录存放项目主体部分的源代码文件。

在教学环境下，以上目录结构可分别放在teacher和student两个子目录中。其中teacher目录包含完整的程序源码和资料文档，以为学生开发时提供参考和借鉴。student目录中的源代码是不完整的，部分类或者函数的实现只给出了基本框架，但代码中的注释和teacher目录下对应的部分完全相同，其中缺失的内容，需要学生在理解整体设计思路和上下文逻辑的前提下予以补全。需要学生参与补全的源代码文件详见开发计划。

5. 开发计划

本项目拟在四个工作日内完成：

工作日	模块	子模块	代码文件
第一天	基础设施	预编译头 Precompile	Precompile.h
		字符串工具包 StrKit	StrKit.h
			StrKit.cpp
		日志 Log	Log.h
			Log.cpp
		配置器 Configurator	Configurator.h
			Configurator.cpp

		多路输入输出 MultiIo	MultiIo.h
			MultiIo.cpp
		插件接口 Plugin	Plugin.h
		插件管理器 PluginMngr	PluginMngr.h
			PluginMngr.cpp
第二天	网络通信	哈希器 Hash	Hash.h
			Hash.cpp
		统一资源定位符过滤器接口 UrlFilter	UrlFilter.h
		布隆过滤器 BloomFilter	BloomFilter.h
			BloomFilter.cpp
		原始统一资源定位符 RawUrl	Url.h
		解析统一资源定位符 DnsUrl	***Url.cpp***
		统一资源定位符队列 UrlQueues	UrlQueues.h
			UrlQueues.cpp
		套接字 Socket	Socket.h
			Socket.cpp
		超文本传输协议响应包头 HttpHeader	Http.h
		超文本传输协议响应 HttpResponse	
第三天	流程控制	线程 Thread	Thread.h
			Thread.cpp
		域名解析线程 DnsThread	DnsThread.h
			DnsThread.cpp
		发送线程 SendThread	SendThread.h
			SendThread.cpp

		接收线程 RecvThread	RecvThread.h
			RecvThread.cpp
		网络爬虫 WebCrawler	WebCrawler.h
			WebCrawler.cpp
		主线程 main	Main.cpp
		构建脚本 Makefile	Makefile
第四天	外围扩展	最大深度插件 MaxDepth	MaxDepth.h
			MaxDepth.cpp
			MaxDepth.mak
		域名限制插件 DomainLimit	DomainLimit.h
			DomainLimit.cpp
			DomainLimit.mak
		超文本传输协议响应包头过滤器插件 HeaderFilter	HeaderFilter.h
			HeaderFilter.cpp
			HeaderFilter.mak
		超文本标记语言文件存储插件 SaveHTMLToFile	SaveHTMLToFile.h
			SaveHTMLToFile.cpp
			SaveHTMLToFile.mak
		图像文件存储插件 SaveImageToFile	SaveImageToFile.h
			SaveImageToFile.cpp
			SaveImageToFile.cpp
		构建脚本 mkall	mkall

其中被***突出显示***的代码文件中，包含需要学生添加的内容，注意源文件中形如“// 此处添加代码”的注释。

6. 知识扩展

为了能在实训环节，进一步强化学生独立思考、独立解决问题的能力，本项目有意涵盖了一些前期课程中不曾涉及或只作为一般性了解的知识和技巧。具体包括：

- 预编译头文件
- std::string
- 变长参数表
- 基于epoll的多路I/O
- 哈希算法和布隆表
- URL、DNS、HTTP和HTML
- 正则表达式
- 线程封装
- 精灵进程和I/O重定向
- Makefile

对于上述内容，建议项目指导教师根据学生的接受能力，结合项目中的具体应用，在项目正式启动之前，先做概要性介绍，同时提供进一步详细学习和研究的线索，包括man手册、参考书、网络链接或其它媒体资源，尽量让学生通过自己的实践和探索找到解决问题的方法，这才是项目实训的意义所在！