Linux网络爬虫系统概要

1、爬虫是什么？

获取互联网资源的一种软件，是搜索引擎的第一个子系统，数据入口之一。

2、爬虫能做什么？

网页下载器

搜索引擎的基础应用

抓取大数据的一种手段

网店秒杀

3、关于项目

在linux下实现关于爬虫的需求、设计、编码、测试、集成。

需求：是要做什么。

设计分为业务设计和技术设计。

业务设计：

业务层的方案，对于软件工程师来说，系统要完成工作的步骤和流程。

技术设计：

关于系统的框架和处理流程。注意：要采用面向对象的思维方式。

编码：使用基本的技术细节进行系统实现。

测试：是对我们是同可靠的一个监测。

集成：将每个程序员开发的模块和子系统合成一个完整的系统。

思维方式：线性思维方式与发散思维方式。

冷更新：先停机再更新。

热更新：不停机更新（一般使用插件）。

软件开发要掌握的思想：模块思维、渐进式开发。

代码结构：

Spider：

spider.conf:配置文件，系统入口

spider：

Makefile:make执行文件

download:下载文件夹

modules：模块文件夹

src：源代码文件夹

4、项目需求：自动榨取网络资源的软件

1、资源是什么？网页、图片、音乐、视频等。

2、自动化是什么样子？一旦运行就不需要更多的干预。

生成需求说明文档。

内容：对需求不明确或不完善的说明进行解释。功能点、附加要求、性能要求等。

5、设计：

业务设计：

如何思考并完成设计：

注意：从顶层开始思考并设计，避免过早的陷入细节。

系统最粗浅的数据流

设计处理流程：

1. 得到爬取的种子（URL）
2. 根据URL下载资源（页面）
3. 解析页面，提取更多的URL
4. 对页面做持久化操作
5. 根据提取的URL再进行下载操作
6. 重复第2到5步。

系统结构图

系统设计：

注意：设计阶段必不可少。

设计可以使我们的思路更清晰，可以提高工作效率，可以提高代码质量。

详细设计：

注意：对系统设计要有输入和输出。

控制器模块



控制模块由三个模块组成：

1. 配置文件处理模块：从配置文件中读取配置项，提供配置项的提取接口。
2. URL维护模块：负责维护URL库，提供如下功能：
   1. 输入新的URL
   2. 输出一个未被抓取的URL
   3. 负责维护URL的抓取状态
3. 任务调度模块：
   1. 负责协调控制器的流程
   2. 负责调用其他系统模块完成工作
4. 维护URL列表数据结构







6、控制器设计：

配置文件解析模块：

配置文件以文件形式保存程序进行时必要时的繁琐过程。

文本类型是文本文件，内容一般以键值形式出现。

概要设计：

配置文件内容：

key=value 形式

注释规则：注释字符串前以“#”标记。

配置项设置：

|  |  |
| --- | --- |
| 项目 | 字段 |
| 并发任务数 | Job\_num |
| url种子 | Seed |
| 抓取深度 | Deeps |
| 输出日志的等级 | Log\_level |
| 模块存放路径（唯一） | Module\_path |
| 模块名称（模块文件名，可以多个） | Module\_name |
| 允许抓取的资源类型（多个，文件后缀） | File\_type |
|  |  |

模块详细设计：

1. 读取配置文件
2. 得到配置文件选项的值（键值）
3. 初始化

//类设计伪代码

class ConfigParser

{

public://共有成员函数（外部接口）

ConfigParper();

load();

对应key的操作();

操作1();

操作2();

…

private://私有成员变量

job\_num

seed

deeps

log\_level

Module\_path

Module\_name

file\_type

};

技术点：

1. 字符串操作（分割字符串）
2. 按行读取fgets()
3. 消除注释
4. 消除空格

URL维护模块：

分析：

url格式：<http://192.168.40.150/docs/index.html>

结构：域名或IP地址，路径，文件名

设计URL的数据结构

|  |  |
| --- | --- |
| 项目 | 字段名称 |
| 完整的URL | url |
| 协议类型 | protocal |
| 域名 | Sitename |
| 资源路径 | Path |
| 文件名 | Filename |
| 当前url处理状态（0-未抓取，1-抓取成功，2-抓取失败） | State |
| 当前url深度 | Deep |
| 当前资源类型 | Filetype |

http协议请求页面时的流程：

1. 输入网址
2. 向DNS发送解析请求
3. DNS返回给我们一个对应的IP地址
4. 通过IP地址向资源所在的主机发送请求
5. 如果资源存在，主机返回200状态，同时返回数据部分
6. 本地http客户端（一般来说是浏览器）接收数据
7. 得到资源

页面抓取的处理流程：

1. 得到一个新的URL
2. URL进入抓取队列等待抓取
3. 从队列中得到一个URL，把其分配给一个下载器的实例
4. 得到下载器的处理状态（URL处理状态需要被改写，得到当前URL深度，得到当前资源类型假如下载成功）
5. 得到当前页面中存在的下一级URL列表



URL维护模块

URL维护模块操作（对外接口）：

1. 添加新URL
2. 使URL进入抓取队列
3. 从抓取队列中移除一个URL
4. 修改URL库中某一个URL的值
5. 添加新URL的列表

class Url

{

public:

//初始化

//set

//get

private:

//

url

protocal

sitename

path

filename

state

deep

filetype

};

class UrlManager

{

public://外部接口

addUrl();

addUrlList();

getUrlForQueue();

removeUrlForQueue();

//setUrlValue();

private://内部接口

findurl()

private:

list<Url>Urls;

map<string,Url\*> UrlMap;

queue<Url\*> Urlqueue;

};