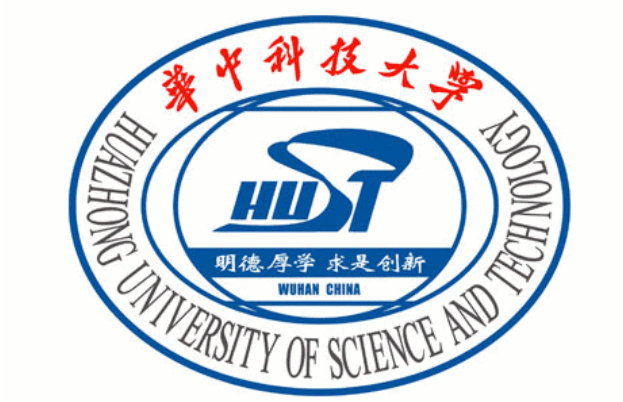
名称	密级
本地文件搜索	开源
版本	共35页
内测版	



本地文件搜索
软件需求规格说明书

拟制	孙经东	日期	2012-10-14
评审	包文博 周宏宽	日期	2012-12-24

修订记录

日期	修订版本	修改章节	修改描述	作者
2012-10-14	Alpha	1-10	完成需求文档基本框架并配图	孙经东
2012-10-19	Alpha	4.1	细化 INDEX 模块需求点	孙经东
2012-11-10	Alpha	4.1	修改 INDEX 模块需求点	孙经东
2012-11-12	Alpha	4.1	增加 INDEX.DATA 需求描述	包文博
2012-12-12	Alpha	2.1 4.1	完成软件介绍和全部需求点描述	孙经东
2012-12-23	Beta	5-10	升级 Beta 版本	孙经东
2012-12-24	Beta	1-10	评审	包文博
		1-10	评审	周宏宽

目录

表目录

图目录

本地文件搜索

软件需求规格说明书

1 简介

1.1 目的

本文描述了本地文件搜索引擎的软件需求规格，用于指导开发人员进行后续的设计、编码和测试工作。本文作为整个软件后续工作的基础。其期望读者是本软件开发团队的所有成员，以及其他相关项目组的配合软件开发人员。

1.2 范围

本软件支持的功能包括：针对本地的文件、目录及文件内的内容进行中英文搜索，索引的建立、压缩及保存，通配符与正则匹配功能，对搜索结果进行高亮、过滤、打开以及提供可选的网络搜索结果建议等后期处理，第三方软件调用等功能，并支持稳定的跨平台特性。

2 总体概述

2.1 软件概述

2.1.1 软件介绍

本地文件搜索，要求软件能够根据用户输入的关键词及相关限制，查找本地磁盘或大容量存储设备中的文件或目录，并通过某种方式将结果反馈给用户。同时，软件还可根据用户的特定需要，扩展相应功能，如通配符搜索、显示结果高亮等等。本地文件搜索涉及到数据结构、数据挖掘、计算机基础、数据库与海量数据处理、图形化编程等方面的内容，是一个综合性比较强、难度较大的软件设计项目。

本地文件搜索主要分为四个部分，分别是索引建立部分、语法分词部分、文件搜索部分及交互界面部分。

- **索引建立**

索引是搜索的基础。建立索引是为了避免过多的目录转移及文件读取工作，加快搜索的速度，同时，好的索引还能通过优秀的分词算法增强搜索的准确性及结果的有效性。本软件选用**倒排索引**的方式，索引包含两大部分，分别为**目录文件索引**，和**文件内容索引**。

目录文件索引通过递归式地遍历本地文件目录建立。由于磁盘容量等限制，本地文件内容有限，目录文件索引的体积和关键词较少（相比于文件内容索引）；并且目录名和文件名表意化更加明显，对分词的要求不高；但是，文件和目录搜索作为基本功能，对索引数据的稳定性和安全性、对搜索的速度和准确度均有较高要求，因此需要灵活的数据结构及个性化的持久化技术。此部分文件目录索引，包括基本接口的设计、数据库的选取及索引数据结构的设计，完全由本组从下至上独立实现，确保对索引的需求功能提供支持。

目录文件索引对应需求项为**SRS.INDEX**，包括**SRS.INDEX.DB**和**SRS.INDEX.DATA**下所有子需求点。

文件内容索引的数据量十分庞大，需要借助中英文（可能需要其他语言）词典对文件内容进行关键词的提取、分词及还原，并保存为压缩度较高的特殊索引文件格式。因此本组**选用Lucene建立**文件内容索引，Lucene是著名的开源全文检索引擎工具包，是Apache软件基金会的子项目，使用Java语言实现。Plucene是其Perl的实现，以Perl模块（Perl Module）的形式发布，并对外提供接口调用。CLucene是其C++实现，PyLucene是其Python实现，可以根据具体编程语言选择相应版本。

文件内容索引对应需求项为**SRS.PLU**，包括其下索引建立相关子需求点。

- **语法分词**

分词是建立索引和搜索都需要进行的关键步骤，是倒排索引建立的基础。由于倒排索引是由分割处理后的关键词对应多个地址的结构，所以，如果关键词分割不当或选择失误，既有可能造成索引数据缺失，又有可能使得索引文件体积过于庞大臃肿。分词需要针对特定语种的特殊特征进行处理，例如，英文分词可以根据空格等天然分隔符进行初步分解，之后再进行小写化和词根还原化等处理；数字则要去掉多余的前置0等。

另外，在本软件的大背景下，还需要考虑不同来源的单词对分词处理的影响。来自于文件名、目录名的单词，具有数据量小、表意明确、代表性强等特征，应该更加谨慎地操作，而来自文件内容的单词，则有数据量大、表意模糊等特点，且单个单词的代表性不如文件名、目录名强，一些连接词和停顿词要忽略，不进行索引。

分词功能对应需求项为**SRS.SPLIT**，包括其下所有子需求点。

- **文件搜索**

文件搜索功能是本次软件的核心功能之一，主要负责的功能范围有两个，一是根据用户输入的搜索条目及相关限制条件，搜索之前建立好的索引数据，找出适配项并反馈给用户；二是针对搜索的结果与用户的输入，进行匹配度或关键度的评定，其结果可以通过关联度评分等方式，统计后反馈给用户。

对于功能一，主要体现的是搜索的基础组织能力。根据搜索的方式，可以分为精确搜索和模糊搜索两种。精确搜索强调可以通过多种搜索选项的输入组合及限制条件，得到一个小范围内的，较为精确的结果；模糊搜索则倾向于通过简单直白的输入，得到大范围的结果，其可能不完全吻合输入项，但是与其有明显的关联。用户可以根据自身需求，灵活选择不同模式，获取搜索结果。

对于功能二，主要体现的是搜索的高级评估能力。因为用户更希望得到与其输入最为相关的结果，搜索结果的关联度评分就显得尤为重要。其功能的实现与索引的建立方式、核心评分公式的设计等均息息相关，在进行评分前，需要首先确定自变量，如词频、关键词吻合率等，再根据统一的公式得到较为公正的结果，并能量化为分数进行排序。

文件名和目录名的搜索对应需求项为**SRS.SEARCH**，包括其下所有子需求点。文件内容搜索对应需求项为**SRS.PLU**，包括其下搜索相关子需求点

- **交互界面**

用户交互界面虽然不是本软件的核心业务模块，但是却有十分重要的作用。因为界面（或称之为软件的“外壳”）同时负责了从用户处获取信息，以及向用户反馈和展示信息。

用户界面的设计需要考虑到且不仅限于以下几个问题：

- 如何让用户使用时可以轻松上手？
- 如何组合和排列搜索功能提供的输入框和选项？
- 控件的数量是否过多或过少以至于造成用户疑惑？
- 交互方式是否仅仅限于输入和点击？
- 搜索结果是否需要一次性全部展示给用户？展示多少合适？
- 不同搜索模式或界面间的切换是否明确直观？如何做到？
- 可以通过何种方式，在用户搜索过程中给予适当的帮助或提示？
- 搜索结果有几种展示方式，是否可以在UI层面进行更深一步的筛选？
- 搜索类软件的核心功能与界面复杂程度的平衡如何把握？
- ……

设计之初，需要积极学习吸收同类产品的设计风格，例如Google Desktop、Everything等，也可以向谷歌、百度、雅虎等主要的网络搜索引擎学习。在保证基本功能点之后，尝试开发和展现拥有自己创意的闪光点。

分词功能对应需求项为**SRS.SHELL**，包括其下所有子需求点。

本软件遵循CMM的V型开发流程规范，主要有五大步骤，依次为：

1) 需求分析（SRS）
2) 概要设计（HLD）
3) 代码编程（CODE）
4) 单元测试（UT）
5) 系统测试（ST）

需要说明的是，由于软件规模和时间长度等原因，详细设计（LLD）作为伪码阶段合入代码编程步骤，不再单独列出。

2.1.2 环境介绍

本软件的开发环境如下：

- Ubuntu Desktop 10.1032-bit(Linux Kernel 2.6.35)
- Microsoft Windows XP Professional / Microsoft Windows 7 Ultimate
- Apple Inc. Mac OSX Leopard (IF POSSIBLE)
- Perl& Tcl dev environment with Tk lib

开发工具：

- Emacs (code editing)
- Vim (code editing on Windows OS)
- Notepad++ (code editing on Windows OS)
- GIMP (images and theme materials)

2.2 软件功能

表1 软件功能列表

功能编号	功能描述
1	本地目录及文件名搜索

2	文本文件（txt、html等）的内容搜索
3	部分二进制文件（doc和xls）文件的内容搜索
4	支持压缩文件（zip）内部搜索
5	搜索支持通配符和正则表达式
~~6~~	~~对索引文件做序列化压缩~~
7	索引文件持久化存储（二进制及文本方式）
8	索引数据MD5校验及配置恢复
9	支持高级搜索和模糊搜索双模式
10	实现搜索结果关联度的量化并排序
11	支持搜索结果组合准则过滤
12	支持搜索结果高亮显示
13	跨平台支持（Linux Unix Windows MacOS）
14	友好、高效的交互界面
15	界面支持手势操作
16	支持直接通过第三方的程序打开结果文件或目录
17	敏感词过滤

3 需求建模

3.1 建模工具

使用DFD方法进行需求分析和建模，使用下面的符号：

说明：

数据流图图例

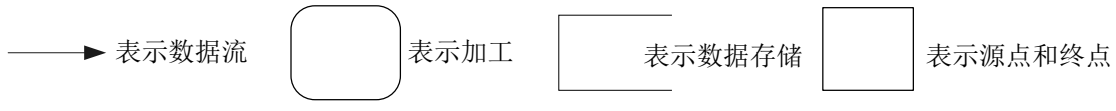


图2 数据流图图例

数据流条目符号说明：

- + 表示“与”
- [|] 表示“或”
- {}表示重复
- () 表示选择
- m .. n 表示界域
- @表示数据存储中的索引
- * 表示注释

蓝色超链接表示需要进一步分解的加工和由上一级继承的加工、数据流。

3.2顶级 DFD 图

顶级图用于描述模块与周边模块的关系。

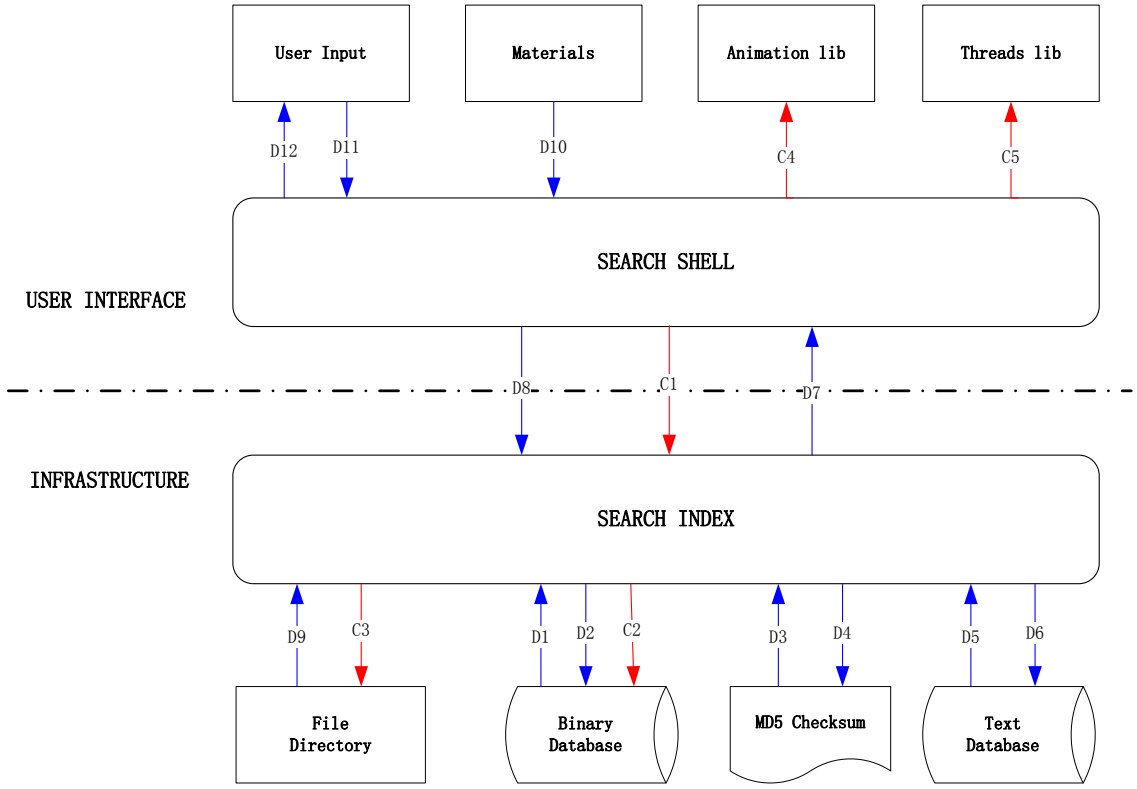


图3 本地文件搜索软件顶级DFD图

对本地文件搜索处理流程进行如下描述（红线所示为控制流，蓝色为数据流）：

控制流描述：

C1：输入关键词后，用户交互界面调用底层接口查询索引数据。

C2：搜索时，底层模块调用接口从数据库中读取索引数据。

C3：索引建立时，底层模块调用接口遍历文件目录建立索引。

C4：用户交互界面调用动画库接口实现动画功能。（Perl或Tcl没有内建的动画库）

C5：用户交互界面调用线程库接口实现多线程。（Perl或Tcl没有对多线程特殊支持）

数据流描述：

D1：搜索过程中，二进制库向底层搜索模块提供索引数据。

D2：内存数据结构与本地二进制库绑定，完成索引数据的添加。

D3：底层索引模块，从校验文件中读取出MD5校验数据。

D4：进程退出时，更新MD5校验数据，写入校验文件。

D5：校验失败后，从文本库进行索引数据的恢复。

D6：进程退出时，将最新索引数据写入文本库。

D7：底层模块返回搜索结果给用户交互界面。

D8：用户交互界面将关键词与搜索配置传递给底层模块。

D9：索引建立时，底层模块读取文件目录和文件内信息，建立索引。

D10：用户交互界面使用素材构建UI。

D11：用户通过交互界面输入搜索关键词及相关搜索选项。

D12：交互界面向用户呈现搜索结果。

4 具体需求

4.1 功能需求

4.1.1 SRS.INDEX.DB.001 初始化本地数据库

1. 介绍

本地数据库用于索引文件的保存与读取。由于索引文件建立耗时较长，所以每次执行搜索命令前都进行一次索引建立开销过大，且效率极低。通过数据库保存已创建的索引文件，便于下次使用前快速加载，可大大提高搜索的效率。

本地数据库根据数据库文件形式分为两种，分别为：二进制文件数据库和文本文件数据库。其对比与优劣可参见下表。

表2 二进制与文本数据库对比表

数据库类型	稳定性	体积	读取速度	存储速度	跨平台性
二进制数据库	差	大	快	快	一般，根据类型有特定平台支持
文本数据库	好	小	慢	慢	好，但需要注意换行符等细节

两类数据库互有优劣，只有结合两者的优点才能提供最稳定、快速的数据服务。

索引文件需同时保存至二进制和文本数据库两种形式，并均可进行读取。从二进制库读取索引，速度快，优先选用。但二进制文件容易损坏，所以还需通过文本库进行备份，并创建一份MD5校验文件，保存上次退出时索引文件的校验和，当校验不通过时，清空二进制库的内容，从文本库进行索引恢复。

初始化本地数据库时，也需要创建二进制数据库、文本数据库和MD5校验文件，数据库内容为空，校验和存在。若原数据库中包含数据，需要进行清空等处理。

2. 输入

1) 本地数据库初始化命令

3. 处理

1) 绑定本地二进制数据库，如果数据库文件不存在，则创建后再进行绑定。绑定本地数据库操作请参见SRS.INDEX.DATA.001 绑定索引与数据库。
2) 清空索引数据，若失败，直接返回，有输出1)。
3) 存储清空后的新二进制库，同时创建文本库备份，以及MD5校验文件。若失败，返回，有输出1)。存储本地数据库操作请参见SRS.INDEX.DB.003 存储本地数据库。
4) 初始化本地数据库完成，返回，有输出2) 。

4. 输出

1) 返回失败，进行资源回退，删除已创建的库及文件，删除本地数据库操作请参见SRS.INDEX.DB.004 删除本地数据库。
2) 返回成功，二进制库、文本库及MD5校验文件初始化成功。

4.1.2 SRS.INDEX.DB.002 读取本地数据库

1. 介绍

读取本地数据库是将数据库中的数据与内存中的数据结构相联系，并读取**部分**数据进入内存（因为索引文件体积庞大，全部读入内存会给主机造成较大负担）。简言之，是建立“可直接操作的内存中的数据结构”与“固态持久化的数据库数据”之间的传输的通道。

读取本地数据库还需要考虑到索引恢复问题，二进制库损坏或数据不可信时，需要清空二进制库，并选择从文本库读取（恢复）索引数据。

2. 输入

1) 本地数据库读取命令

3. 处理

1) 绑定本地二进制数据库。绑定本地数据库操作请参见SRS.INDEX.DATA.001 绑定索引与数据库。
2) 读取MD5校验文件，将校验文件中的校验和与二进制库中数据的校验和进行比较：
   a) 若校验和相等，则直接返回，有输出1) 。
   b) 若校验和不相等，则清空二进制库中的数据，并从文本库进行索引数据的恢复，有输出2) 。
   c) 若从文本库进行索引恢复失败，则进行资源回退，删除全部数据，删除本地数据库操作请参见SRS.INDEX.DB.004 删除本地数据库。返回，有输出3) 。

4. 输出

1) 返回成功，读取二进制库成功。
2) 返回成功，读取文本库成功。
3) 返回失败，删除已创建的库及文件。

4.1.3 SRS.INDEX.DB.003 存储本地数据库

1. 介绍

存储本地数据库最重要的功能是更新文本库及MD5校验文件，对于二进制库，由于其与内存中的数据结构相绑定，所以在索引的增加和删除过程中，二进制库内容便会同步更新。

存储本地数据库的操作发生在以下两种情况：

- 进程退出时，更新本次索引数据的MD5校验文件，更新文本库。
- 初始化本地数据库时，二进制库初始化完毕后通过本操作建立对应的MD5校验文件和文本库。

重建索引文件时不需要进行存储操作，以保证主要功能点流畅。

2. 输入

1) 本地数据库存储命令

3. 处理

1) 绑定本地二进制数据库。绑定本地数据库操作请参见SRS.INDEX.DATA.001 绑定索引与数据库。
2) 对索引数据进行MD5校验，得到校验和。
3) 读取MD5校验文件，将校验文件中的校验和与二进制库中数据的校验和进行比较：
   a) 若相等，有输出1)，继续向下执行。
   b) 若不相等，将最新MD5校验和写入MD5校验文件，继续向下执行。
4) 若重建数据库时产生的临时库存在，则删除临时库。
5) 更新文本库中的索引数据，保证其为最新版本，返回，有输出2) 。

4. 输出

1) 调试信息输出“No Changes need to be saved!”。
2) 返回成功，本地数据库存储成功。

4.1.4 SRS.INDEX.DB.004 删除本地数据库

1. 介绍

删除本地数据库包括删除二进制库，删除文本库，及删除MD5校验文件。因为在进程运行时二进制库与内存中的索引哈希表绑定，故删除二进制库可以通过清空索引哈希表实现。

删除本地数据的操作发生在以下三种情况：

- 载入数据库时，初始化数据库失败，则删除本地数据库。
- 初始化本地数据库时，首先进行删除本地数据库操作，清空所有索引文件及数据，然后重新建立本地数据库。
- 读取本地数据库时，MD5校验和不相等，同时文本库不存在导致文本恢复错误，则必须进行删除本地数据库操作。

2. 输入

2) 本地数据库删除命令

3. 处理

1) 绑定本地二进制数据库。绑定本地数据库操作请参见SRS.INDEX.DATA.001 绑定索引与数据库。
2) 清空索引哈希表，具体操作请参见SRS.INDEX.DATA.003 清空索引数据。
3) 读取MD5校验文件，将校验文件中的校验和与二进制库中数据的校验和进行比较：

c) 若相等，有输出1)，继续向下执行。

d) 若不相等，将最新MD5校验和写入MD5校验文件，继续向下执行。

4) 若重建数据库时产生的临时库存在，则删除临时库。

5) 更新文本库中的索引数据，保证其为最新版本，返回，有输出2) 。

4. 输出

1) 调试信息输出“No Changes need to be saved!”。

2) 返回成功，本地数据库存储成功。

4.1.5 SRS.INDEX.DB.005 重建本地数据库

1. 介绍

重建本地数据库即是对现有索引数据进行更新，故重复需求点不再出现在INDEX需求部分。重建本地库操作仅会重建二进制库，不会更改文本库和MD5校验文件，因为重建过程可能出现错误,资源回退较为繁琐;同时文本文件读写效率较低,会对性能产生较大影响。文本库和二进制库的更新会在进程结束时统一进行。

重建数据库有两种思路：

- 将现有数据库文件更名并另存备份（如在原文件名后加.bk后缀），然后以原库名新建一个空数据库，绑定后添加新索引。若成功，则删除原备份数据库；若失败，删除新建数据库（存在情况下），并把备份数据库恢复回原名。
- 原有数据库不变，以不同名称新建空数据库，绑定该新数据库后添加新索引。若成功，删除原数据库，将新数据库更名；若失败，直接删除新建立的数据库。

这里采用后一种方法，主要考虑到搜索过程与数据库重建过程可能重叠，必须考虑两者并发的情况下需要利用旧数据库提供索引数据。搜索相较与重建数据库更为频繁，如果采用第一种思路，将导致每次搜索时频繁判断和切换需要绑定的数据库，造成过多浪费。

2. 输入

1) 本地数据库重建命令

3. 处理

1) 绑定新建的二进制数据库。绑定本地数据库操作请参见SRS.INDEX.DATA.001 绑定索引与数据库。

2) 初始化新二进制库，初始化本地数据库操作请参见SRS.INDEX.DB.001 初始化本地数据库。

3) 在新二进制库中建立索引，索引建立操作请参见SRS.INDEX.DATA.002添加新索引。

4) 若处理步骤2)和3)均成功，则删除原二进制库文件，将新二进制库更名为原二进制库的名字。返回，有输出1)。

5) 若处理步骤2)和3)之中有任何一个步骤失败，保留原二进制库不变，删除新建的二进制库，返回，有输出2) 。

4. 输出

1) 返回成功，调试信息输出“Rebuilding success...”。

2) 返回失败，调试信息输出“Rebuilding error, keep old database.”。

4.1.6 SRS.INDEX.DB.006 载入本地数据库

1. 介绍

载入本地数据库发生在进程启动时，判断进行本地数据库初始化或读取操作。判断标准唯一，即MD5校验文件是否存在。原因如下：

- 若MD5校验文件存在，则会进行读取本地数据库操作。在读取过程中会对二进制库进行校验；若校验没有通过，则从文本库进行恢复；若文本库也不存在，则清空一切数据。
- 若MD5校验文件不存在，则会进行初始化本地数据库操作。因为校验文件损坏或丢失，可以认为本地数据库中的信息均是不可信任的，所以此时无论二进制库和文本库是否存在，均会被删除，然后重新建立新的空数据库。

整个过程逻辑无误，没有遗漏。

2. 输入

1) 本地数据库载入命令

3. 处理

1) 判断MD5校验文件是否存在：

a) 若存在，读取本地数据库，请参见SRS.INDEX.DB.002 读取本地数据库。

b) 若不存在，初始化本地数据库，请参见SRS.INDEX.DB.001 初始化本地数据库。

2) 返回，有输入1)。

4. 输出

1) 返回成功，载入本地数据库完成。

4.1.7 SRS.INDEX.DATA.001 绑定索引与数据库

1. 介绍

绑定索引与数据库操作的目的是为了建立内存中数据结构与本地磁盘上数据库之间数据交换的桥梁。即，当软件执行过程中，写入数据到内存数据结构中，同时也会将数据写入到本次磁盘的数据库中（过程中可能在内存中有一定缓存然后一次刷入磁盘）；当需要从本地磁盘的数据库中获取索引数据时，可以直接向内存中的数据结构获取，通过内存中数据结构的组织形式获取格式化后的索引数据。

绑定索引与数据库是数据库操作前提和基础，SRS.INDEX.DB下的所有需求点均需要在绑定索引与数据库的前提下进行，包括：

- 初始化本地数据库 SRS.INDEX.DB.001~SRS.INDEX.DB.001
- 读取本地数据库 SRS.INDEX.DB.001~SRS.INDEX.DB.002
- 存储本地数据库 SRS.INDEX.DB.001~SRS.INDEX.DB.003
- 删除本地数据库 SRS.INDEX.DB.001~SRS.INDEX.DB.004
- 重建本地数据库 SRS.INDEX.DB.001~SRS.INDEX.DB.005
- 载入本地数据库 SRS.INDEX.DB.001~SRS.INDEX.DB.006

即SRS.INDEX.DB.001~SRS.INDEX.DB.006。

需要注意的是，绑定操作仅限于本地二进制库。因为文本库仅作为索引数据恢复使用，而且不支持与内存数据直接交换。

2. 输入

1) 绑定索引变量

2) 当前索引状态

3. 处理

1) 判断当前索引状态。

a) 如果索引状态为空闲，将目标数据库设为本地二进制库

b) 如果索引状态为正在重建索引，则将目标数据库设为临时二进制库。

2) 将内存中的索引数据结构绑定到目标数据库上。若失败，有输出1)，返回。若成功，有输出2)，返回。

4. 输出

1) 返回失败，数据库绑定失败。

2) 返回成功，数据库绑定成功。

4.1.8 SRS.INDEX.DATA.002添加新索引

1. 介绍

添加新索引操作相当于向索引数据库中添加条目。其过程主要分为两步，分别为：遍历目录树与文件/目录名处理。

第一步：遍历目录树。会通过深度优先搜索方法，从指定根目录开始，递归式地遍历全部目录及文件。遍历过程中如果遇到文件夹（或压缩包），则递归式地进入遍历；如果为文件，进行第二步文件/目录名处理。

第二步：文件/目录名处理。获取文件后，会对文件名依次执行文件名分词、关键词重复项合并、索引数据添加等操作。

索引数据需要包含**关键词**、**目录地址**和**词频**三项信息。其对应关系为：

- 一个关键词对应多个目录地址，一个目录地址可能包含于多个关键词
- 同一关键词下的一个目录地址对应一个词频，记录关键词在目录地址下出现的次数

添加新索引的时间较长，索引数据通过持久化保存后可以重复使用。

2. 输入

1) 指定根目录

3. 处理

1) 遍历目录树。会通过深度优先搜索方法，从指定根目录开始，递归式地遍历全部目录及文件。遍历过程中如果遇到文件夹（或压缩包），则递归式地进入遍历；如果为文件，进行第二步文件/目录名处理。

2) 文件/目录名处理。获取文件后，会对文件名依次执行文件名分词、关键词重复项合并、索引数据添加等操作。参见SRS.SPLIT.WORD.001 分词处理 及SRS.INDEX.DATA.004 索引关键词的合并。

3) 对文本文件的内容建立索引。非文本文件尝试转换为文本文件后，再建立索引。

4) 对于zip压缩文件，视之为一个目录，进行一层递归搜索，但是不建立压缩文件内文件的内容索引。参见SRS.INDEX.DATA.005 建立压缩文件索引。

5) 全部处理完成后有输出1)。

4. 输出

1) 新索引建立成功，添加到索引数据库中。

4.1.9 SRS.INDEX.DATA.003 清空索引数据

1. 介绍

清空索引数据为清空索引数据库中的索引条目，而不删除数据库文件。清空索引数据发生在初始化本地数据库和删除本地数据库时。

2. 输入

1) 清空索引数据命令

3. 处理

1) 打开二进制库文件，读取其内容

2) 如果二进制库中索引数据为空，直接返回

3) 如果二进制库中索引数据存在，则删除文本库中的所有内容，并返回。

4. 输出

1) 二进制库数据被清空

4.1.10 SRS.INDEX.DATA.004 索引关键词的合并

1. 介绍

建立倒排索引时，首先需要对原字串进行分词处理，处理完成后得到一个或多个关键词。关键词有重复时，需要进行合并操作，合并后的关键词指向多个“地址”（即文件名或目录名）。指向的地址越多，则证明该关键词出现的情况越普遍，那么在给搜索“地址”打分时，该关键词所占的权重就要适当降低。

于此相对应的，某个关键词在一个“地址”中出现的次数称为词频，词频越高，则该关键词的“指向性”越强，那么在打分时其所占权重就要适当提高。

2. 输入

1) 索引关键词合并

3. 处理

1) 在遍历文件目录建立索引过程中，分解出关键词。
2) 判断已建立的索引数据中是否已存在该关键词。
   a) 若该关键词存在，则相同的关键词进行“地址”合并，即该关键词指向的地址群中加入正在解析的“地址”。得到预期结果1)
   b) 若该关键词不存在，则创建新索引关键词，该关键词指向正在解析的“地址”。得到预期结果2)

4. 输出

1) 关键词指向地址数加一，增加的地址即为正在解析的地址。
2) 关键词唯一指向正在解析的地址。

4.1.11 SRS.INDEX.DATA.005 建立压缩文件索引

1. 介绍

由于压缩文件可以视作一个特殊的目录，其中可能包含用户希望搜索到的文件，并且压缩包内的内容较为固定且不经常改变，故对于压缩包内的文件建立索引是很有意义的。目前可先对zip格式的压缩文件建立索引（实验性质）。

在对目录建立索引时，目录下如果有zip格式的压缩文件则需要对其包含的文件或文件夹建立索引，暂时可以不支持压缩包中再包含的压缩包的解析。

2. 输入

1) 类型为zip压缩文件的待解析文件

3. 处理

1) 读取该zip文件，获取压缩包内的文件名列表。
2) 将zip文件作为目录，将压缩包内文件名与父压缩包进行路径合并。
3) 对新合并后的路径建立索引，有输出1)。

4. 输出

1) 返回zip压缩包内文件的索引，其中zip压缩包等同于目录。

4.1.12 SRS.SPLIT.WORD.001 分词处理

1. 介绍

对于一个完整的文件或目录名，并不是直接将之作为关键词保存为索引数据的，这样既会无谓增大索引数据的体积，也不会提高搜索的效率，更无从评估出搜索结果的关联度。本软件采用倒排索引的方式，每个关键词都对应了多个文件、目录地址，所以需要对该文件、目录名进行分词处理，分离出来的元素经过处理之后成为关键词，可以作为搜索的依据。

在获取到名称后，需要把该名称中不同类型的元素提取出来，例如：数字、中文、英文。之后交由不同的处理流程进行特定处理。一般而言，一个名称可以提取出一个或多个元素，所以会对应一个或多个关键词。提取区分的过程可以通过逐字符读取匹配或正则表达式匹配等方式，特定处理的过程需要结合文件、目录名称及语种、类型的特点进行处理。

2. 输入

1) 文件或目录名称

3. 处理

1) 通过正则表达式或逐字符匹配的方式，提取出不同类型的元素。
2) 逐个按类型对元素进行处理：
   a) 若元素类型为数字，处理参见SRS.SPLIT.KEYWORD.001 针对数字分词
   b) 若元素类型为英文，处理参见SRS.SPLIT.KEYWORD.002 针对英文分词
   c) 若元素类型为中文，处理参见SRS.SPLIT.KEYWORD.003 针对中文分词

   有输出1)
3) 每个元素处理完成后得到的关键词可以添加入索引，具体操作流程请参见SRS.INDEX.DATA.002添加新索引。

4. 输出

1) 原文件或目录名称被分解为一个或多个可加入索引的关键词。

4.1.13 SRS.SPLIT.KEYWORD.001 针对数字分词

1. 介绍

对于文件或目录名中的数字部分，采取以下原则进行处理：

- 若数字为多个0，如000000，则合并为一个0，得到0
- 若数字有多个前置0，如000009，则删除全部前置0，得到9
- 小数视为以小数点隔开的两个整数，如12.34，得到12和34两个关键词

2. 输入

1) 数字

3. 处理

1) 按照数字分词处理原则进行处理，得到输出1)

4. 输出

1) 得到可添加入索引的数字关键词。

4.1.14 SRS.SPLIT.KEYWORD.002 针对英文分词

1. 介绍

对于文件或目录名中的英文部分，采取以下原则进行处理：

- 以空格、点等休止符天然划分为多个词
- 删除后置描述符，如’s、‘m、n’t等中的’会删除，剩余项合并入前项
- 全部转化为小写字母
- 对于大小写混合的字符串，也应该进行分割，例如TixBook应该分解为tix和book

说明：这里没有把停词(stop words)，即be、are、the、is等去掉，主要考虑到两个方面的原因，一是文件名称或目录名称“寸土寸金”，停词可能并非是无意义的词语，相反可能是用户特意如此命名的；二是考虑到关联度评估时需要的数据越多越准确，保留停词可以在某种程度上提高评估的准确性。

在进行索引数据压缩时，会删除停词部分的索引。

在建立内容索引时，停词肯定要删去，并不会不加入索引，因为出现次数过多且无实际意义。

2. 输入

1) 英文

3. 处理

1) 按照英文分词处理原则进行处理，得到输出1)

4. 输出

1) 得到可添加入索引的英文关键词

4.1.15 SRS.SPLIT.KEYWORD.003 针对中文分词

1. 介绍

对于文件或目录名中的中文部分，采取以下原则进行处理：

- 删除中文标点符号，如（）、，。“” ‘’等，剩余部分合并为一个完整关键词。

说明：对中文没有进行过多处理，一是考虑到在目录或者文件名中出现的中文一般都具有比较完整的意义，继续进行分词不会对搜索结果的准确度有提升，反而可能会增加处理的复杂度；二是考虑到中文分词的困难性，一种比较简单的中文分词方法是“二字法”，例如“我爱天安门”可以两两拆分为“我爱”、“爱天”、“天安”、“安门”四个关键词，这种分词方法较为简单，但是很显然会导致搜索结果不完整，有些符合的结果搜索不到，若选用更为复杂的分词方式，则需要字典支持（例如庖丁解牛中文包），又进一步提高了复杂度，但是经过测试其效果反而不如中文不分词的情况（再次提醒，操作域为目录名、文件名，比较有特点），不过可以考虑在后续版本中提供。

2. 输入

1) 中文

3. 处理

1) 按照英文分词处理原则进行处理，得到输出1)

4. 输出

1) 得到可添加入索引的中文关键词

4.1.16 SRS.SPLIT.EXTEND.001 针对扩展名处理

1. 介绍

扩展名是个很纠结的东西，因为这个东西只存在于Windows系统，在Linux中文件类型

的划分都不是通过扩展名区分的。同时，有的文件可能套换上完全不对应的扩展名，例如一些病毒文件，往往为XXX.jpg.exe的格式，用户在隐藏扩展名的情况下，容易误认为是图片而不加防范。

针对文件类型的处理需要根据不同平台单独完成。在Windows平台下，可以通过扩展名实现类型识别，在Linux和Unix系统中，可以使用系统“file”命令，分析其返回值完成类型识别。

2. 输入

无

3. 处理

无

4. 输出

无

4.1.17 SRS.SEARCH.SPECIFIC.001 精确搜索模式

1. 介绍

精确搜索模式强调搜索结果的准确性和搜索过程的可定制性，通过向用户提供多种输入和匹配选项，实现多角度定位目标文件，得到最准确的结果。

该模式提供五种搜索备选项：

i. 待搜索字符串（必选） include any
ii. 必包含字符串（可选） include all
iii. 必不包含字符串（可选） include not
iv. 大小写敏感（可选） case sensitive
v. 严格模式（可选） be strict

待搜索字符串是必选项，即用户最直接的希望搜索到的东西，用户可能输入多个单词或词语，搜索结果中应该至少包含其中一种，包含词语个数越多、匹配越准确，则关联度越高。

必包含字符串是可选项，用户可以在此项中输入已保证存在的信息，不符合此项的结果不会显示出来。

必不包含字符串是可选项，用户可以在此项中输入已确认不会存在的信息，符合此项的结果不会显示出来。

大小写敏感是可选项，确认在搜索过程中是否区分字母大小写。

严格模式是可选项，若选中此项，则只有完全吻合输入各关键词的结果才会显示，搜索速度最快，要求最严格。

2. 输入

无

3. 处理

无

4. 输出

无

4.1.18 SRS.SEARCH.FUZZY.001 模糊搜索模式

1. 介绍

模糊搜索模式强调搜索过程的便捷性，它只提供一个输入框，用户输入其中的信息会作为搜索关键字进行进一步的分词和匹配处理，得到的结果也是广泛的，即模糊搜索会尝试寻求关键字中的最小词元，只要匹配到最小词元的地址均会显示到结果中，只是关联度会比较低导致排名在较后的位置。

模糊搜索模式希望用户能够通过最少、最方便的操作，得到尽可能准确且丰富的结果。同时，当用户无法准确给出目标文件的具体特征时，也适于用模糊搜索进行初步的定位。

2. 输入

无

3. 处理

无

4. 输出

无

4.1.19 SRS.SEARCH.SCORE.001 搜索结果关联度评估

1. 介绍

搜索结果关联度的评估是搜索过程中的重要过程，也是体现搜索软件质量优劣的重要指

标。关联度体现的是搜索结果与用户输入之间的匹配度，其计算素材的来源主要是用户输入的字串以及数据库中的索引。比较显而易见的关联因素，如词频等；另外，索引地址的总数等隐含在索引数据中的变量也会间接影响关联度。若某个地址自身匹配搜索，同时其父地址也匹配搜索，那么父地址的关联度会以一定比例加乘到子地址中。

在参考了相关文献，并经过大量实验之后，总结出如下关联度计算公式：

$$\begin{cases} R_i = \dfrac{1}{r_i}\left(f_i \times log\dfrac{N_{\mathrm{t}}}{N_i}\right) \\ R = \displaystyle\sum_{i=1}^{n} R_i + \sigma R_j \end{cases}$$

其中，

R 为整个地址的综合关联度

R_i 为单个关键词给地址贡献的关联度

R_j 为该地址包含的且匹配搜索关键词的父地址的关联度

σ 为父地址关联度对地址总关联度的贡献比例

N_{t} 为整个索引数据中关键词的总个数

N_i 为包含有该关键词的地址数

f_i 为该索引关键词在地址中出现的次数，即词频

r_i 为搜索串与索引关键词的匹配度（$r_i \geq 1$）

其直观表达为：如果某个关键词在该地址中出现的次数越多，关联度越大；如果该关键词包含在过多地址中，则说明比较普通常见，关联度要降低；在搜索过程中，该关键词与倒排索引的Key值相差的字符数越少，关联度越高；一个地址的综合关联度为其包含的各个关键词关联度之和。

2. 输入

1) 索引数据中关键词的总个数、包含有该关键词的地址数、词频、关键词的匹配度

3. 处理

1) 根据公式计算单个关键词关联度

2) 将一个地址中包含的关联度叠加，并按照比例加成如匹配父地址关联度，输出1)

4. 输出

1) 获得一个地址的综合关联度

4.1.20 SRS.KINO.BUILD.001 文件内容索引的建立

1. 介绍

文件内容的数据量非常大，而且要经过严格的筛选和分词处理。在有限的软件开发时间内，很难在兼顾其他功能点的情况下，独立完成内容索引的建立。然而，现在有很多很成熟的全文索引工具，例如solr、elastic search、coreseek、sphinx等，前面两个工具都是基于一个十分著名的索引库Lucene。

Lucene是Apache下的一个开源全文检索引擎工具包，原基于Java，但是也针对不同语言有不同实现，例如PyLucene、Plucene、CLucene等。Lucene提供了一套完整的索引建立与查询API，同时也给予使用者很多自由，例如选择索引的词域，是否可以被索引或搜索，自行设定被索引的内容等。

借助Lucene，可以很方便地创建文件内容的索引，但需要自行确定被索引的文件对象及索引内容。分词器选择自带的Simple Analyzer或Standard Analyzer即可。

2. 输入

1) 指定根目录

3. 处理

1) 提取该目录及其子目录下的目标文件，目标文件可以是文本文件或HTML文件
2) 创建analyzer准备分词解析处理
3) 创建Writer对象，实现向索引数据添加Document
4) 创建Document作为一个独立索引的容器
5) 根据文件类型，选择相关内容加入content域，其类型为TEXT，可以被索引
6) 将该文件的路径加入filename域，其类型为KEYWORD，唯一且不可被索引
7) 将填充完整的Document添加入全部索引数据。
8) 销毁Writer、Document等相关对象，得到输入1）

4. 输出

1) 指定根目录下的文本文件内容被索引入全文数据库

4.1.21 SRS.PLU.BUILD.002 文本和PDF文件内容索引处理

1. 介绍

对于全文索引，其基本功能是对文本文件索引。对于其他类型的文档，例如pdf或doc

格式，也是通过各种方式转换为文本数据输入Lucene建立索引，可以说，文本数据是Lucene索引数据的基础。

2. 输入

1) 文本文件、PDF文件

3. 处理

1) 根据文件类型

a) 若为文本文件，读取文件内容，输入document的content域建立索引，具操作请参见 SRS.KINO.BUILD.001 文件内容索引的建立，有输出1）

b) 若为PDF文件，则通过模块接口将其内容转化为文本格式的字符串，输入document的content域建立索引，具操作请参见 SRS.KINO.BUILD.001 文件内容索引的建立，有输出2）

4. 输出

1) 文本文件的内容被索引入全文数据库

2) PDF文件的内容被索引入全文数据库

4.1.22 SRS.PLU.BUILD.003 HTML文件内容索引处理

1. 介绍

HTML是一类特殊的文本文件，其通过比较松散的规则组织各式静态元素，可以被浏览器解析，从而呈现出丰富美观的信息。然而，HTML文件中并非所有数据都是有效且应该被索引的，例如，HTML文件中的各种标签，如<p>、<tr>、<li>、<href>、等，仅仅为了组织排列数据或强调某个字串，并没有明显意义，而且HTML头部的信息很多也没有特征，例如编码格式是UTF-8或gb2312。

对于HTML类型文件索引，需要提取出其直观的信息，即解析后呈现给用户的信息。这一原则也适用于其他类似文件类型，如XML。

2. 输入

1) HTML文件

3. 处理

1) 根据HTML文件，建立HTML树形结构

2) 自根节点开始向下遍历，遍历过程中收集显示信息

3) 将收集到的数据输入document的content域建立索引，具操作请参见SRS.KINO.BUILD.001 文件内容索引的建立，有输出1）

4. 输出

1) HTML文件的可视内容被索引入全文数据库

4.1.23 SRS.PLU.SEARCH.001 文件内容的搜索

1. 介绍

与文件内容索引的建立方式类型，文件内容搜索也需要借助Lucene自带的API实现。

文本内容的搜索与索引建立的过程息息相关，在索引建立时，会给一个Document设定不同的Field，这些Field用相同或不同的特性，有的可以索引、有的可以搜索、有的仅仅作为标识。搜索时，从目标Field中查找目标，再通过同一document中保存文件或目录名的Field获取结果文件名，这一过程可以通过Hit Collector收集，最后一并交给使用者。

此外，在搜索时也需要进行分词解析等操作，为了保证能查找到相关结果，搜索时选择的分词器和建立索引时选择的分词器要统一。

Lucene自身十分强大，除了针对文件内容建立索引外，还可以将文件名加入另一个Field同样索引，并支持搜索。但是由于文件和目录的特殊性，其搜索结果范围往往偏窄，即要求十分匹配才能获得结果，经测试，对于文件名和目录名的搜索，Lucene的表现并不好，精确搜索结果过少，且很难实现模糊搜索。所以我们仅仅使用Lucene完成文件内容的索引与搜索。

2. 输入

1) 用户输入的搜索字串

3. 处理

1) 根据分词原则，对原始字串进行初步分词，具体请参见 SRS.SPLIT.WORD.001 分词处理

2) 新建解析器QueryParser并选择与索引建立时相同的Analyzer，设置搜索域

3) 新建IndexSearcher对象，将初步分词后的关键词逐个输入IndexSearcher

4) 创建HitCollector对象收集搜索结果，并把各个关键词对应的结果地址进行合并

5) 销毁QueryParser IndexSearcher HitCollector等对象，返回搜索结果，有输出1）

4. 输出

1) 获取到地址列表，即为包含用户输入关键词的文件地址

4.1.24 SRS.SHELL.GENERAL.001 交互界面整体设计

1. 介绍

交互界面是软件与用户沟通的桥梁，既负责通过合理渠道从用户处获取信息输入软件处理，又负责将处理后的结果以适当方式呈现给用户，所以对于用户体验有至关重要的作用。

交互界面需要至少支持以下特性：

- 交互界面涵盖精确搜索、模糊搜索功能，并给出相关控件进行操作
- 交互界面支持自动识别本地编码格式并对不同编码格式间进行转换，不出现乱码
- 搜索结果提供至少文件名、路径、标识图标、修改时间和大小等信息
- 搜索结果支持按照关联度、大小、修改时间、名称，进行升序和降序排列
- 搜索结果支持按照不同类型对结果文件和文件夹进行显示/隐藏的操作
- 搜索结果支持即时过滤，并高亮过滤匹配字符，支持过滤操作后结果的还原
- 搜索结果支持双击直接打开相应文件或文件夹
- 提供帮助信息，数据库信息，索引目录，搜索结果统计信息
- 搜索时根据用户输入，可以即时提供关键词建议
- 界面支持主题切换，主题采用插件思想，可以独立制作，打包发布
- 界面支持手势操作

交互界面需要遵循以下几点原则：

- 交互界面需要给予用户积极的信息，不排斥用户，以用户为中心
- 界面从整体上，鼓励用户尝试操作，并对不同操作可能导致的后果给予反馈
- 界面需要考虑操作系统平台、屏幕分辨率等硬件因素，并进行适配
- 尽可能减少通过屏幕像素点的绝对位置进行定位和排列空间的操作
- 单个控件的长宽均不小于27像素，否则过小难以识别
- 搜索结果显示多样化，直接呈现的结果界面力求简洁，更多结果界面力求丰富
- 避免在主界面堆砌过多控件，力求直观简洁，用户如果有需要，也可以方便地通过点击按钮等一键操作切换到信息丰富的界面

2. 输入

无

3. 处理

无

4. 输出

无

4.1.25 SRS.SHELL.CMD.001 命令行调试程序设计

1. 介绍

Shell的创建和载入需要较多时间，同时，其显示还需要考虑编码格式转换等因素，不适合使用在开发之初以及后期调试过程中。因此需要开发调试专用的程序。

命令行下的调试程序，通过Terminal终端获取输入并原地打印结果，启动快速，且可以方便地显示程序运行过程中设定的调试信息。例如“Initializing database”、“Reading from binary database”、“Recovery from text database”、“Storing database done”等。对于结果显示，既可以按照界面文本显示方式，按行整齐排列显示，也可以按照内存中的数据结构，例如哈希表等方式直观显示，方便观察调试。

为了减少平台依赖性，本软件在开发过程中不使用任何IDE，故无法打断点调试；但是可以通过命令行调试程序，通过标准错误通道即时打印调试信息或变量值达到相同的目的，故是本软件开发流程中必不可少的组件。

命令行需要基本涵盖界面拥有的功能及选项，例如输入框、包含关系、大小写敏感、精确和模糊搜索模式切换等。由于Terminal条件限制，可以放弃多样化的结果显示，取而代之的是直观的“结果地址-关联度评估分数”一一对应的关系。

命令行需要覆盖的功能至少为搜索功能、索引数据重建功能。特殊命令可以通过“[]”与普通输入区分，例如“[rebuild]”、“[quit]”等

2. 输入

无

3. 处理

无

4. 输出

无

4.2 性能需求

本软件主要针对中小型文件目录的搜索。但是也可以胜任Windows下某个盘符的全盘索引或Linux下/usr或home文件夹的索引。

要求的数据如下：

对于7G左右的目录，30000文件，10000文件夹，预计关键词数量8000，索引数据大小为30MB（文本库）、100MB（二进制库）。要求建立索引时间不超过20分钟，搜索时间不超过30秒。

4.3 外部接口需求

请参见 **https://search.cpan.org**

5 总体设计约束

5.1 标准符合性

1) 遵守CMM开发流程
2) 遵守Perl语言通用编程规范

5.2 硬件约束

无

5.3 技术限制

无

6 软件质量特性

6.1 软件健壮性

本软件保证所开发的功能的健壮性

多个功能需求点组合应用时基本功能正常

在不同硬件设备上基本功能正常。

6.2 软件兼容性

本软件保证所开发的功能在各种类Unix和Windows操作系统环境下工作正常

需保证操作系统的开发环境与本文档描述一致

6.3 软件可靠性

本软件保证所开发的功能在各种常见可靠性测试中基本功能正常

支持进程重启

支持索引数据持久化存储

6.4 软件稳定性

本软件保证所开发的功能在长时间运行、本地文件过多等应用场景下的稳定运行

长时间运行业务后无内存泄漏等异常

执行大量命令后无内存泄漏等异常

6.5 软件可维护性

本软件保证注释量大于或等于总代码量的30%，关键位置均有注释描述说明，保证代码的可读性及软件的可维护性

7 依赖关系

7.1 模块依赖

- 预计需要的Perl模块

  Plucene File::Spec DBM::Deep Digest::MD5 Archive::Zip Encode HTML::TreeBuilder MP4::Info AudioFile::Info CAM::PDF Data::Dumper

- 预计需要的Tcl包

  Tk AniGif

8 其他需求

8.1 数据库

系统支持GDBM数据库，Linux和Unix系统均自带GDBM数据库，Windows系统需额外

安装，可提供独立安装包。

8.2 操作

无。

9 需求分级

表3 需求分级表

Requirement ID 需求ID	Requirement Name 需求名称	Classification 需求分级
SRS.INDEX.DB.001	初始化本地数据库	EASY
SRS.INDEX.DB.002	读取本地数据库	NORMAL
SRS.INDEX.DB.003	存储本地数据库	HARD
SRS.INDEX.DB.004	删除本地数据库	EASY
SRS.INDEX.DB.005	重建本地数据库	NORMAL
SRS.INDEX.DB.006	载入本地数据库	NORMAL
SRS.INDEX.DATA.001	绑定索引与数据库	EASY
SRS.INDEX.DATA.002	添加新索引	HARD
SRS.INDEX.DATA.003	清空索引数据	EASY
SRS.INDEX.DATA.004	索引关键词的合并	EASY
SRS.INDEX.DATA.005	建立压缩文件索引	NORMAL
~~SRS.INDEX.DATA.006~~	~~索引数据的压缩~~	~~EASY~~
SRS.SPLIT.WORD.001	分词处理	NORMAL
SRS.SPLIT.KEYWORD.001	针对数字分词	EASY
SRS.SPLIT.KEYWORD.002	针对英文分词	EASY
SRS.SPLIT.KEYWORD.003	针对中文分词	EASY
SRS.SPLIT.EXTEND.001	针对扩展名处理	EASY
SRS.SEARCH.SPECIFIC.001	精确搜索模式	HARD
SRS.SEARCH.FUZZY.001	模糊搜索模式	HARD
SRS.SEARCH.SCORE.001	搜索结果关联度评估	HARD
SRS.PLU.BUILD.001	文件内容索引的建立	NORMAL
SRS.PLU.BUILD.002	文本和PDF文件内容索引处理	NORMAL
SRS.PLU.BUILD.003	HTML文件内容索引处理	NORMAL
SRS.PLU.SEARCH.001	文件内容的搜索	NORMAL
SRS.SHELL.GENERAL.001	交互界面整体设计	HARD
SRS.SHELL.CMD.001	命令行调试程序设计	NORMAL

10 附录

10.1 附录A 可行性分析结果

无

参考资料清单：

[1] Tck/Tk入门经典（第2版）, J.K.Ousterhout, K.Jones著，张元章译，清华大学出版社

[2] Lucene in Action, E.Hatcher, O.Gospodnetic, M.McCandless著，人民邮电出版社

[3] Advanced Perl Programming, Simon Cozens, O'Relly Publisher

[4] Tk Manual Page http://www.tcl.tk/man/tcl8.4/TkCmd/

[5] Perl CPAN Website http://search.cpan.org/

[6] Perl Programming Documentation http://perldoc.perl.org/

[7] Stack Overflow Community http://stackoverflow.com/

[8] Apache Lucene Home Page http://lucene.apache.org/