
华中科技大学计算机科学与技术学院

C 语言课程设计报告

专 业： 计算机科学与技术

班 级： 1601

学 号： U201614531

姓 名： 刘本嵩

成 绩：

指导教师： 甘早斌

完成日期： 2017 年 9 月 11 日



目 录

一、系统需求分析.....	1
二、总体设计.....	5
三、详细设计.....	12
四、系统实现.....	18
五、运行测试与结果分析.....	73
六、总结.....	81
七、参考文献.....	83



一、系统需求分析

现有某菜农，在给定面积的菜地里种植各种蔬菜，每年种植的蔬菜种类不完全相同，各种的收成也不一样，请设计一个程序，帮助该菜农管理每年种植的蔬菜品种、蔬菜秧苗数量（株）、收获蔬菜的数量（斤）。

1、需要处理的基础数据

对该菜农种植的蔬菜信息进行管理，主要包括蔬菜种类信息，蔬菜基本信息，菜农种植信息表等三类信息。

(1) 蔬菜种类信息表

中文字段名	类型及长度	举例
分类编码	char	'1'~'5'
分类名称	char[8]	5个分类名称：根茎类、果菜类、瓜类、叶菜类、菌类

根茎类：白萝卜，胡萝卜，大葱，小葱，蒜，洋葱，莴笋，山药，马铃薯、红薯，等等

果菜类：菜椒，青椒，尖椒，甜椒，朝天椒，线椒，南瓜，丝瓜，扁豆等等

瓜类：西瓜，甜瓜，白瓜，黄瓜，苦瓜等等

叶菜类：大白菜，小白菜，生菜，菠菜，韭菜，芹菜，空心菜等等

菌类：木耳，银耳，平菇，草菇，金针菇，香菇，等等

(2) 蔬菜基本信息表

中文字段名	类型及长度	举例
蔬菜编号	int	自增长(顺序增加)
蔬菜名称	char[20]	“白萝卜”
分类码	char	'1' //表示根茎类蔬菜
营养成分	Char[20]	指蔬菜中含义的矿物质成分，比如 菠菜含有铁

辣椒含有胡萝卜素、钙、铁；扁豆含有胡萝卜素、维生素 C、钾、磷、铁、锌；西红柿含有钙、磷、铁、硼、锰、铜；萝卜含有蛋白质、葡萄糖、维生素 C；胡萝卜含有胡萝卜素；空心菜含有蛋白质、胡萝卜素、维生素 B1、B2、C；苦瓜含有维生素 B2、维生素 C、钙、铁、磷；香菇含有多糖、糖种酶。等等，其他蔬菜的营养成分学生可自行查阅。

(3) 菜农种植信息表

中文字段名	类型及长度	举例
编号	Int	自增长
蔬菜编号	int	同蔬菜基本信息表中的蔬菜编号
种植面积	Int	2：表示 2 分地
收获重量	float	公斤
种植年份	char[5]	“2015”2015 年

2、系统基本功能

本系统需要实现数据维护，数据查询和数据统计三个主要功能模块，另外根据情况添加辅助功能模块。下面给出了三个主要模块的功能需求，辅助功能模块根据各人的理解和分析自己设计（鼓励！）。

(1) . 数据维护

本模块实现蔬菜种类信息、蔬菜基本信息、蔬菜种植信息等三方面基本信息的数据维护功能，又分为三个子模块。

- 蔬菜种类信息维护：包括对蔬菜种类信息的录入、修改和删除等功能。
- 蔬菜基本信息维护：包括对蔬菜基本信息的录入、修改和删除等功能。
- 蔬菜种植信息维护：包括对蔬菜种植信息的录入、修改和删除等功能。

(2) . 数据查询

本模块实现蔬菜种类信息(五个分类：根茎类、果菜类、瓜类、叶菜类、菌类)，蔬菜基本信息，蔬菜种植信息等三方面基本信息的数据查询功能，又分为三个子模块。

1) 蔬菜种类信息查询功能



以分类编码为条件来查找并显示满足条件的蔬菜分类信息。例如，查找并显示分类编码为'3'的蔬菜分类信息。

2) 蔬菜基本信息查询功能

- 蔬菜名称中文字符子串为条件查找，并显示蔬菜中包含指定子串的蔬菜基本信息。例如，查找并显示果菜类蔬菜名称中包含为“椒”的果菜类蔬菜基本信息。
 - 以分类码和营养成分为条件查找并显示满足条件的蔬菜基本信息。例如，查找并显示分类码为'4'（叶菜类）且营养成分中含有铁的所有蔬菜基本信息。

3) 蔬菜种植信息查询功能

- 蔬菜部分名称（模糊查找）和种植年份为条件查找并显示满足条件的所有蔬菜种植信息。
 - 例如，查找并显示蔬菜名称中含有椒的蔬菜且在“2015”年种植的所有蔬菜信息。
 - 以蔬菜名称为条件查找并显示满足条件的蔬菜种植信息。例如，查找并显示蔬菜名称为“菠菜”的所有年份种植的菠菜信息（重量），并按年份分别显示。

(3) . 数据统计

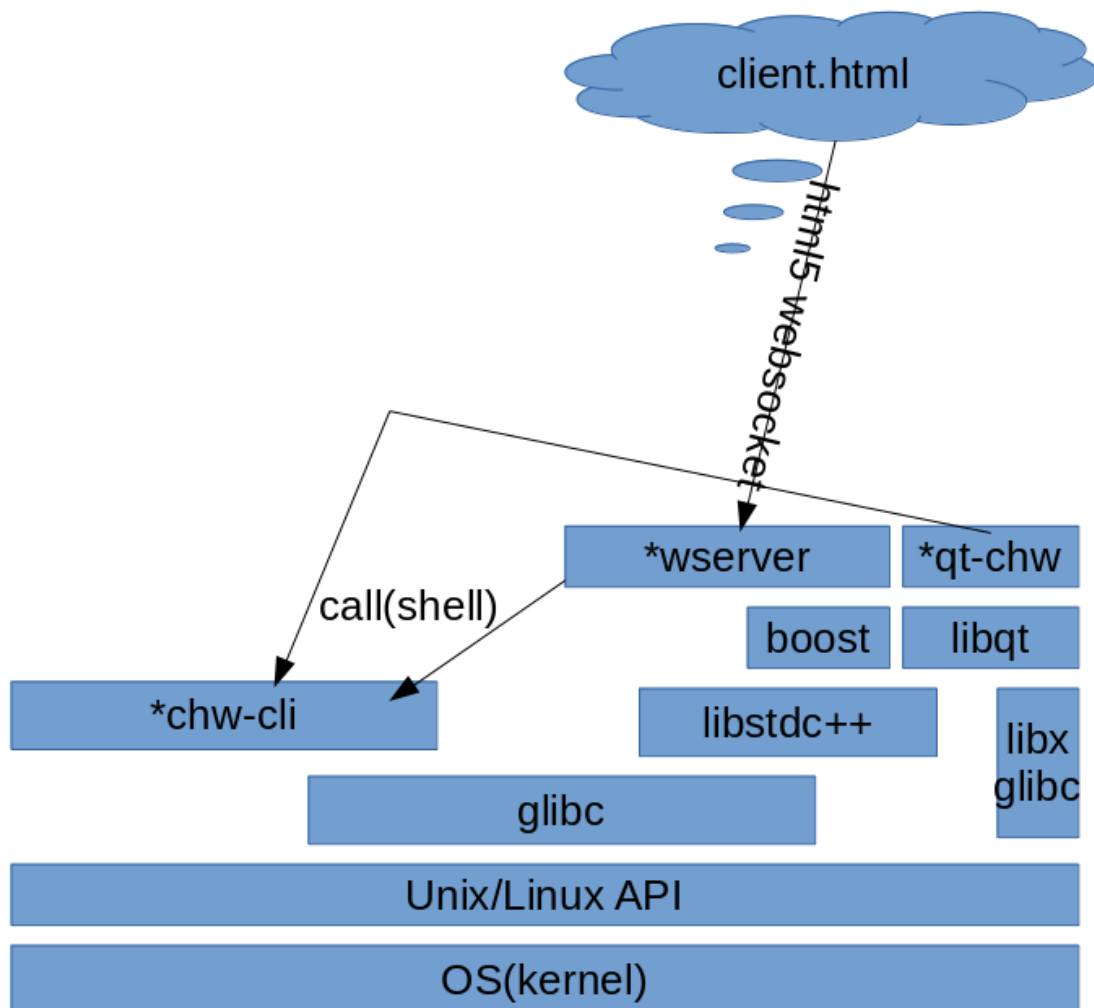
本模块实现五个方面的数据统计功能，前四个功能需求已给出，第三个自行设计。

- 分别统计某年各类蔬菜（如叶菜类、根茎类等）种植总面积、收获总重量，按总重量降序排序后，输出分类名称、种植面积、收获总重量（按种类统计）。
- 以所输入的起止年份为条件（如 2015-2017），按蔬菜名称（如菠菜、黄瓜等）统计该三年内所种各种蔬菜总面积、收获总重量，按总重量降序排序后，输出蔬菜名称、分类名称、种植面积、总重量。
 - 分别统计某种类蔬菜的已有的数量。
- 给定某个营养成分，程序自动判定含有该营养成分的蔬菜，并且显示输出所有含有该营养成分的蔬菜名称。
 - 有关该菜农蔬菜种植信息的其他方面数据统计。

二、总体设计

总体设计是在功能需求分析的基础上，设计系统的功能结构和数据结构，即按照功能要求构造系统的功能框架，并确定数据的描述规范和数据之间的关系。通过总体设计建立目标系统的逻辑关系，系统功能框架通常用模块层次结构图来描述，能够直接地体现功能模块之间的调用关系。

2.1 系统整体设计



图中，在 released bin 中呈现的可执行文件为 chw-cli(Backend), wserver(websocket server to support client.html frontend), qt-chw(frontend implemented by libqt). chw-cli 通过 shell 传参给 main, 通过 stdout(/dev/fd/1) 输出信息，通过 stderr(/dev/fd/2) 输出错误。shell 为一切进程间通信提供了高效而简洁的接口。专业的前端设计语言有利于加速开发和减少开发成本。模块间的低耦合性为代码的可维护性提供了根本保证。

此报告将认为代码的价值满足，可维护性>可扩展性>正确性>运行时间开销>运行内存开销>硬盘/其他外存储设备开销>编译开销。

2.2、数据结构设计

数据结构设计即确定数据的逻辑结构和物理结构，数据的逻辑结构是对数据之间关系的描述，数据的物理结构是数据逻辑结构在计算机中的表示。

按任务要求，系统需要处理的信息有三种：蔬菜种类(veg_class)，蔬菜信息(veg_info)，工作信息(job_info)。

另外，出于数据规范化、便于处理和节省存储空间等目的，我们还可以对信息中某些项进行编码，用固定长度的代码来表示长度不一的信息。

下面分别为本系统所涉及的代码数据、基础数据和生成数据的数据结构，以及数据在内存外存中的存储结构。

```
>>> ./src/app/data.h

typedef struct _veg_class {
    int code;
    char name[64];
} veg_class;

typedef struct _veg_info {
    int no; //Must self-inc autoly.
    char name[64];
    const veg_class *pclass;
    int __class_code; //Unused (!!Unmaintained!!). Just to make
the doc happy. //Defined as pclass->code
    char nutr_info[64];
```

```
} veg_info;

typedef struct _job_info {
    int no;
    const veg_info *pveg;
    int __veg_no; //Unused (!!Unmaintained!!). Just to make the
doc happy. //Defined as pveg->no
    int area;
    float weight;
    int year;
    char __year[5]; //Unused (!!Unmaintained!!). Just to make the
doc happy. //Defined as string(year)
} job_info;
<<< end data.h
```

1. 蔬菜种类

数据结构名称：蔬菜种类 数据结构标识：_veg_class					
数据项名称	数据项标识	数据类型	数据长度	取值范围	示例
代码	code	int	sizeof(int)	int	1
名字	name	CString	64	01-FF	"veg_cls_a"

2. 蔬菜信息

数据结构名称：蔬菜信息 数据结构标识：_veg_info					
数据项名称	数据项标识	数据类型	数据长度	取值范围	示例
序号	no	int	sizeof(int)		1
名称	name	CString	64		"vega"
种类	__class_c ode	int	sizeof(int)		1
营养	nutr_info	CString	64		"vc/Hg"

3. job

数据结构名称：job 数据结构标识：_job_info					
数据项名称	数据项标识	数据类型	数据长度	取值范围	示例
job 编号	no	int	sizeof(int)		1
蔬菜	__veg_no	int	sizeof(int)		2
种植面积	area	int	sizeof(int)		9999
种植质量	weight	float	sizeof(f)	0-	123.456
种植年份	__year	int	sizeof(int)	0-	1970

链表使用伪泛型实现，talk is cheap, I'll just show key code.

>>> job.c (Incomplete code)

```
#include "list.h"
rList jbuf;
jbuf = rList_newlist();
read_from_disk();
rList_for_each(iter, jbuf)
{
    job_info *p = (job_info *)iter->data;
    //Do some operation, print result to some pipe/fd.
}
write_to_disk();
rList_destructor(jbuf);
<<< end job.c
```

veg-class.c, veg.c 表现与 job.c 非常相似，代码复用率高。这是使用抽象数据结构的好处。

下面是链表库的设计，高度复用了 glibc++ std::list 的源码。接口按标准风格封装，命名和 std::list 保持严格一致。

```
>>> list.h
#ifndef SRC_LIBR_LIST
#define SRC_LIBR_LIST
```

```
struct rListNode{
    struct rListNode* prev;
    struct rListNode* next;
    void* data;
};

struct rListDescriptor{
    struct rListNode* first;
    struct rListNode* last;
};

struct rListNode* rList_constructor();
struct rListDescriptor* rList_newlist();
void rList_push_back(struct rListDescriptor* desc,void* data,unsigned
int length);
void rList_pop_back(struct rListDescriptor* desc);
void rList_push_front(struct rListDescriptor* desc,void*
data,unsigned int length);
void rList_pop_front(struct rListDescriptor* desc);
void rList_eraseElem(struct rListDescriptor* desc,struct rListNode*
target);
void rList_destructor(struct rListDescriptor* desc);
typedef struct rListDescriptor rListDescriptor;
typedef struct rListNode* rListIterator;
typedef rListDescriptor *rList;
#define rList_for_each(_var_iterator, _var_list) for(rListIterator
_var_iterator = _var_list->first; _var_iterator != nullptr;
_var_iterator = _var_iterator->next)
#endif
<<< end list.h
```

数据在外存中的存储由 librlist/fmtio.c 实现。这是简单的二进制 dump。

```
>>> fmtio.c
```



```
#include "fmtio.h"
#include <unistd.h>

int fmt_read_file(fd fin, rList dataBuf, size_t sizeofObj, const void
*delimiter)
// Pass nullptr as delimiter to use 0 bytes as delimiter. //Return
read obj counter, neg if error.
{
    bool mustFree = delimiter == nullptr;
    if(mustFree)
        delimiter = calloc(1, sizeofObj);
    int counter = 0;
    while(true)
    {
        void *mem = malloc(sizeofObj);
        auto bytes = read(fin, mem, sizeofObj);
        if(bytes == -1) die("read fin error. current counter=%d",
counter);
        if(bytes < sizeofObj) break;
        if(0 == memcmp(mem, delimiter, sizeofObj)) break;
        rList_push_back(dataBuf, mem, sizeofObj);
        ++counter;
        free(mem);
    }
    if(mustFree)
        free((void *)delimiter);
    return counter;
}

int fmt_write_file(fd fout, rList dataBuf, size_t sizeofObj, const
void *delimiter)
```



```
// Pass nullptr as delimiter to use 0 bytes as delimiter. //Return
read obj counter, neg if error.
{
    bool mustFree = delimiter == nullptr;
    if(mustFree)
        delimiter = calloc(1, sizeofObj);
    int counter = 0;
    for(rListIterator it = dataBuf->first; it != nullptr; it = it-
>next)
    {
        auto ret = write(fout, it->data, sizeofObj);
        if(ret < sizeofObj) die("write fout error: write() return %d,
current counter=%d", ret, counter);
        ++counter;
    }
    auto ret = write(fout, delimiter, sizeofObj);
    if(ret != sizeofObj)
        die("write delimiter failed. write()=%d, counter=%d", ret,
counter);
    if(mustFree)
        free((void *)delimiter);
    return counter;
}
<<< fmtio.c
```

uthash 开源库使用纯宏实现，不涉及 struct，按规则不予列举。

三、详细设计

RealTime Status: master  release 

主要介绍 chw-cli。

首先，我必须介绍 project 的文件结构，以便进行整体结构的介绍。

建议直接在 <https://github.com/recolic/chw> 获得源文件。

recolic@RECOLICPC ~/CL/chw (master)> tree .

. #master 分支:没有对 c++代码进行预编译，打包和加密的纯代码分支。

- ├── .travis.yml #指导 travis ci 对每一次提交进行自动的在线编译，并自动更新文档中的 build status。

- ├── bin #预先在 Linux x86_64 下编译，可以直接部署的二进制文件。

- ├── check.sh #编译时环境检查，验证程序要求的依赖是否满足。

- ├── chw-cli #后端

- ├── clear_database.sh #清空数据库。这个功能没有要求，因此单独实现

- ├── gui #web gui files

- ├── client.min.html #web gui here!

- ├── favicon.ico

- ├── form.min.css

- └── sheet.min.js

- ├── qt-gui #用 qt 重写的 gui

- ├── README #说明

- ├── runtime_check.sh #检查预先编译的程序是否能正常运行在这台主机

- └── wserver #web socket server

- ├── LICENSE #开源许可证:修改过的 Mozilla 许可证，对抄袭 C 课设的责任认定和 Fair use 的界定做了特别规定和修改。

- ├── README.md #编译说明文档 项目介绍文档

- └── src #源文件

- ├── app #与逻辑有关的源文件

- ├── cmd.c #命令行参数解析器



- | |—— cmd.h
- | |—— data.h #定义数据结构
- | |—— job.c #定义 job 有关的数据库操作
- | |—— job.h
- | |—— main.c #定义 main 函数，调用伪 c++ 类的构造函数和析构函数
- | |—— shit-op.c #定义其他操作
- | |—— shit-op.h
- | |—— veg.c #定义蔬菜有关的数据库操作
- | |—— veg-class.c #定义蔬菜种类有关的数据库操作
- | |—— veg-class.h
- | |—— veg.h
- |—— cclib #Simple C Container Library. 已经停止维护并且 buggy.
- | |—— bitstrings.c
- | |—— bloom.c
- | |—— buffer.c
- | |—— ccl_internal.h
- | ... 此处省略 cclib 源文件
- | |—— wstrcollection.c
- | |—— wstringlist.c
- | |—— wstringlist.h
- |—— check.sh #检查依赖和环境，由 CMakeLists.txt 调用并进行系统自检
- |—— clear_database.sh #清空数据库。这个功能没有要求，因此单独实现
- |—— cmake_clean.sh #清理 cmake 缓存
- |—— CMakeLists.txt #CMake 主要配置文件，指定了一切编译选项，编译条件和编译方法，用于自动生成 Makefile，指导整个项目的编译与发布。
- |—— common.h #定义了一些对 C 语言的常用扩展
- |—— error.h #定义了错误处理和异常，使用 php die 风格。
- |—— librlist #自己实现的链表库
- | |—— CMakeLists.txt #子编译配置文件，指导 cmake
- | |—— fmtio.c #定义了内存中链表和外存相交换的方法
- | |—— fmtio.h



- | |—— list.c #定义了链表的基本操作
- | |—— list.h
- | |—— search.c #泛型链表算法，未使用
- | |—— search.h
- | |—— sort.c #泛型链表算法，未使用
- | |—— sort.h
- |—— libut #纯宏实现的 C Container Lib
- | |—— LICENSE #libut 的开源许可证
- | |—— utarray.h
- | |—— uthash.h
- | |—— utlist.h #未使用 utlist
- | |—— utringbuffer.h
- | |—— utstring.h
- |—— mod
- | |—— console.c #定义了更好的 system 函数，与 shell 交互更便利。
- | |—— console.h
- | |—— filter.c #定义了类似 gnu grep 的过滤器，为 ls 后的可选参数提供支持。
- | |—— filter.h
- | |—— _fmtio.c #已弃用 整合进 rList
- | |—— _fmtio.h
- | |—— fstr.c #动态的格式化字符串函数 C 语言版本已停止维护
- | |—— fstr.h
- | |—— hash.c #mod33 哈希函数
- | |—— hash.h
- |—— note.md
- |—— qt #qt 部分与此段无关 不予介绍
- | |—— build-untitled-unknown-Debug
- | | |—— inputer.o
- | | |—— main.o
- | | |—— mainwindow.o



- | | |—— Makefile
- | | |—— moc_inputer.cpp
- | | |—— moc_inputer.o
- | | |—— moc_mainwindow.cpp
- | | |—— moc_mainwindow.o
- | | |—— moc_predefs.h
- | | |—— ui_inputer.h
- | | |—— ui_mainwindow.h
- | | |—— untitled
- | |—— inputer
- | | |—— inputer.cpp
- | | |—— inputer.h
- | | |—— inputer.ui
- | |—— untitled
- | |—— cmdr.h
- | |—— main.cpp
- | |—— mainwindow.cpp
- | |—— mainwindow.h
- | |—— mainwindow.ui
- | |—— untitled.pro
- | |—— untitled.pro.user
- |—— runtime_check.sh #从源码编译的情形无需进行 runtime 检查
- |—— tests
- | |—— a.out
- | |—— libccl.a
- | |—— test-console.c
- | |—— test-ht.c
- | |—— test-lib.c
- | |—— test.sh #压力测试脚本 进行几万次的自动压力测试，保证可靠性
- |—— web #与 web 前端有关的组件
- | |—— client # web 前端

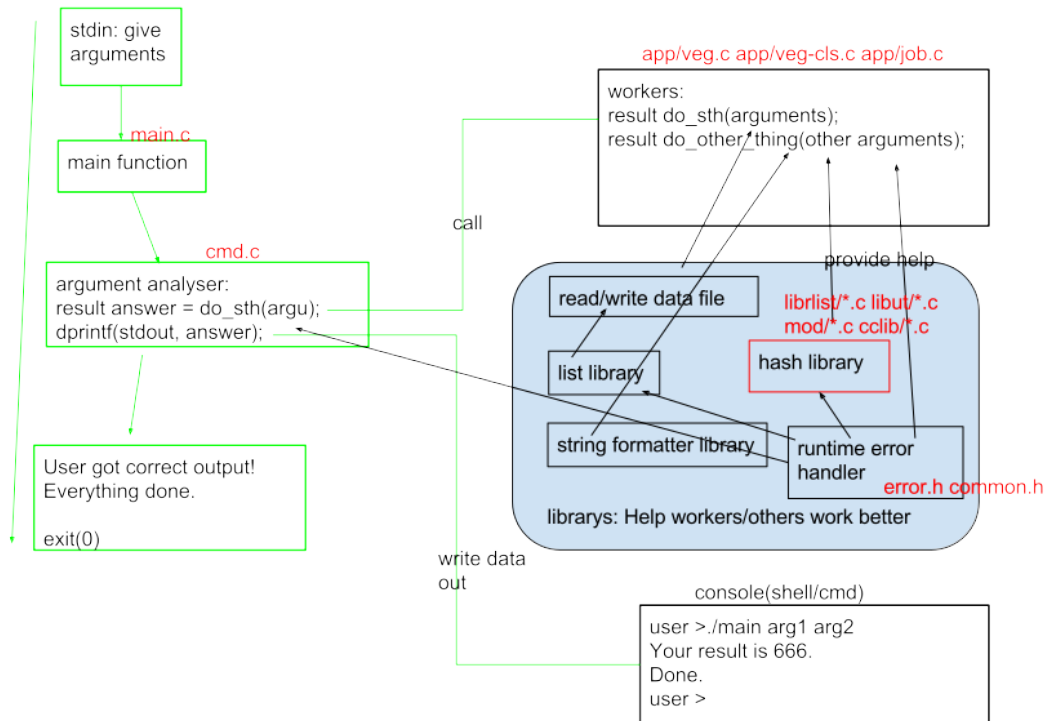


- | |—— client.html
- | |—— favicon.ico
- | |—— form.css
- | |—— min
 - | |—— client.min.html
 - | |—— favicon.ico
 - | |—— form.min.css
 - | |—— sheet.min.js
- |—— sheet.js
- server #web socket 服务器
 - bin.h #C++到 C 的转换自动生成的文件
 - cmake_clean.sh #清理子目录的 cmake 缓存
 - CMakeLists.txt #指导此项目的编译 要求 C++14
 - libwebsocket++ #现代 C++实现的 websocket 纯头库
 - websocketpp
 - base64
 - base64.hpp
 - client.hpp
 - CMakeLists.txt
 - 此处省略冗长无意义的列举
 - utilities.hpp
 - version.hpp
 - main.cc #main 函数所在地
 - make_trick.py #执行转换
 - mod
 - console.cc #修改 console.cc 使之适合 C++
 - console.h
 - rlib #完全自己实现的 C++库 不再进行说明
 - c
 - fake_cpp.h #向 C 语言中引入 C++类,vector 和简单的 tree.
 - macro.hpp

```
├── Makefile
├── noncopyable.hpp
├── nonmovable.hpp
├── print.hpp
├── README.md
├── require
│   ├── cxx11
│   ├── cxx14
│   ├── cxx17
│   ├── gcc
│   ├── linux
│   └── win
├── scope_guard_buffer.hpp
├── scope_guard.hpp
├── string
│   ├── fstr.hpp
│   └── string.hpp
├── sys
│   ├── cc_codegen.py
│   ├── cc_list
│   ├── compiler_detector
│   ├── fdset.hpp
│   ├── os.hpp
│   ├── rwlock.hpp
│   └── sio.hpp
├── terminal.hpp
└── threadPool.hpp
```

下面是程序运行的大致思路。绿色部分表示程序运行流程，黑色箭头表示依赖树(或依赖图)，红色文字表示对应的实现的大致源文件位置。

run this program from console



由于 filter 的存在，ls 操作将会稍有不同。这将在下一节说明。

下面是被要求的 librlist 流程图。由于只有 `rList_push_back`, `rList_for_each` 和 `rList_remove_item` 被使用过，因此只列举这些方法的流程图，它们将在下一小节的末尾出现。

四、系统实现

只列出 chw-cli 的逻辑部分的源代码。由于在 office 中应用代码高亮会产生极大的性能开销，此处不得不黑白打印。

首先应指明 cmake 要求的编译环境和编译选项。

>>> CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5) #Boost::system syntax support
project(chw)

if(CMAKE_CXX_COMPILER_ID STREQUAL "GNU")
    if(CMAKE_CXX_COMPILER_VERSION VERSION_LESS "7.0.0")
        message(FATAL_ERROR "Insufficient gcc version")
    endif()
else()
    message(FATAL_ERROR "gcc is required")
endif()

# This is to satisfy silly CLion syntax checker.
add_definitions(-D_ENABLE_CXX1X_AUTO_FIX)
set(CMAKE_CXX_FLAGS_DEBUG "-g -DMALLOC_CHECK_=2 -D__DEBUG")
set(CMAKE_CXX_FLAGS_RELEASE "-O3")
set(CMAKE_VERBOSE_MAKEFILE ON)

set(SOURCE_MOD mod/console.c mod/filter.c mod/hash.c mod/fstr.c)
set(SOURCE_APP_CUI app/cmd.c app/main.c app/veg-class.c app/veg.c app/job.c
app/shit-op.c)
set(SOURCE_APP_GUI app/cmd.c app/gui-main.c app/veg-class.c app/veg.c
app/job.c app/shit-op.c)
```



```
# do check.sh

execute_process(COMMAND ./check.sh OUTPUT_VARIABLE CHECK_SH_0)

message("${CHECK_SH_0}")

set(CHECK_OK)

string(FIND "${CHECK_SH_0}" "fatal" CHECK_OK)

if(NOT "${CHECK_OK}" STREQUAL "-1")

    message(FATAL_ERROR "NEEDED dependency can't satisfy. Stopping
configure...")

endif()

# compiler options

# set(SOURCE_FILES ${SOURCE_RLIST} ${SOURCE_MOD} ${SOURCE_APP})

set(CMAKE_C_STANDARD 11)

include_directories(.)

include_directories(/mod)

include_directories(/librlist)

add_subdirectory(librlist)

# cclib custom makefile configurations

# set the output destination

set(CCLIB_A cclib/libccl.a)

# create a custom target called build_scintilla that is part of ALL

# and will run each time you type make

add_custom_target(build_ccl ALL

    COMMAND make libccl.a

    WORKING_DIRECTORY cclib

    COMMENT "Calling cclib makefile to build libccl.a")

# now create an imported static target

add_library(libccl STATIC IMPORTED)

# Import target "scintilla" for configuration ""
```



```
set_property(TARGET libccl APPEND PROPERTY IMPORTED_CONFIGURATIONS NOCONFIG)
set_target_properties(libccl PROPERTIES
    IMPORTED_LOCATION_NOCONFIG "${CCLIB_A}")
# now you can use scintilla as if it were a regular cmake built target in
your project
add_dependencies(libccl build_ccl)

# setup cli
add_executable(chw-cli ${SOURCE_MOD} ${SOURCE_APP_CUI})
target_link_libraries(chw-cli libccl rlistlib)
add_custom_command(TARGET chw-cli POST_BUILD COMMAND mv chw-cli ../bin/
COMMENT "Moving exec to bin...")
add_custom_command(TARGET chw-cli POST_BUILD COMMAND rm -rf ../bin/gui)
add_custom_command(TARGET chw-cli POST_BUILD COMMAND cp -r
web/client/min ../bin/gui COMMENT "Moving gui to bin...")
add_custom_command(TARGET chw-cli POST_BUILD COMMAND cp *.sh ../bin/ COMMENT
"Moving scripts to bin...")
add_custom_command(TARGET chw-cli POST_BUILD COMMAND rm
../bin/cmake_clean.sh COMMENT "Do cleanup...")

# build wserver
add_subdirectory(web/server)

# deleted # # frontend will be impl seperately.
# deleted # # import gtk3
# deleted # find_package(PkgConfig REQUIRED)
# deleted # pkg_check_modules(GTK3 REQUIRED gtk+-3.0)
# deleted # include_directories(${GTK3_INCLUDE_DIRS})
# deleted # link_directories(${GTK3_LIBRARY_DIRS})
# deleted # add_definitions(${GTK3_CFLAGS_OTHER})
```



```
# deleted #  
# deleted # # setup gui  
# deleted # add_executable(chw-gui ${SOURCE_MOD} ${SOURCE_APP_GUI})  
# deleted # target_link_libraries(chw-gui libcccl rlistlib ${GTK3_LIBRARIES})
```

<<< end CMakeLists.txt

下面是错误处理和一些有用的小 macro。unpack_arg 是为了让参数中的其他 macro 有机会展开。

>>> common.h

```
#ifndef _SRC_COMMON_DECL_H  
#define _SRC_COMMON_DECL_H 1  
  
#include <stddef.h>  
#include <stdbool.h>  
#include <stdlib.h>  
#include <stdint.h>  
#include <stddef.h>  
#include "error.h"  
  
#define __unpack_arg(arg) arg  
  
#define decltype(type_t) typeof(__unpack_arg(type_t))  
#ifdef _ENABLE_CXX1X_AUTO_FIX  
#define auto __auto_type  
#endif  
  
#define nullptr NULL  
  
#define __new(data_type) (data_type *)malloc(sizeof(data_type))  
#define new(data_type) __new(__unpack_arg(data_type))  
  
#define __min(a, b) ((a) < (b) ? (a) : (b))
```



```
#define min(a, b) __min(__unpack_arg(a), __unpack_arg(b))

typedef int fd;

#endif

<<< end common.h

>>> error.h

#ifndef _SRC_ERROR_H
#define _SRC_ERROR_H 1

#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>

#define __unpack_arg_2(arg) arg
#define _do_except(msg, ...) do{ \
    printf("Exception at %s:%d | errno=%d:%s >", \
    __FILE__, __LINE__, errno, strerror(errno)); \
    printf(msg, ##__VA_ARGS__); \
    printf("\n\n"); \
    quick_exit(errno?errno:126); \
}while(0)

#define except(msg, ...) _do_except(msg, ##__VA_ARGS__)

#define die except

#ifdef __DEBUG
#define RECORD printf("R> %s:line%d\n", __FILE__, __LINE__);
#define debug_if(sth) if(sth)
#else
#define RECORD
#define debug_if(sth) if(false)
#endif
```




```
#endif  
  
#define debug debug_if(true)
```

```
#endif
```

```
<<< end error.h
```

下面是 app 文件夹中的所有逻辑代码。介绍以注释形式出现。共计大约 30 页。
我已经在上一小节介绍了每一个源文件的作用，此处不再赘述。

```
>>> app/shit-op.c  
  
#include "shit-op.h"  
  
#include "list.h"  
  
#include "data.h"  
  
#include "console.h"  
  
#include      "cclib/containers.h"      //http://www.cs.virginia.edu/~lcc-  
win32/ccl/ccl.html  
  
  
#include      "libut/uthash.h"  
  
//http://troydhanson.github.io/uthash/userguide.html  
  
  
#include <unistd.h>  
  
#include <stdio.h>  
  
  
extern rList jbuf; //constructed.  
  
  
/** Comment for C oj report  
A structure for uthash. refer to libut document for details please.  
***/  
/** Special comment end  
typedef struct __to_print_1  
{  
    int classCode; //K  
    char className[64];  
    int sumArea;
```



```
float sumWeight;

UT_hash_handle hh;          /* makes this structure hashable */
} infa;
typedef infa * HashTable_infa;
/** Comment for C oj report
job_report_by_veg_class implemented here.
***/ Special comment end

void      _impl_job_report_by_veg_class(fd      fout)      // #print
className+sumArea+sumWeight
{
    HashTable_infa buf = NULL;

    // HashTable *buf = iHashTable.Create(sizeof(infa));

    /** Comment for C oj report
遍历链表。hash map 负责存储 veg_class_no 到 __to_print_1 的映射。
***/ Special comment end

    for(rListIterator it = jbuf->first; it != nullptr; it = it->next)
    {
        job_info *p = (job_info *)it->data;
        infa dat;
        dat.classCode = p->pveg->pclass->code;
        strcpy(dat.className, p->pveg->pclass->name);
        dat.sumArea = p->area;
        dat.sumWeight = p->weight;

        infa *ele;
        HASH_FIND_INT(buf, &dat.classCode, ele);

        // infa *ele = iHashTable.GetElement(buf, &dat.classCode,
sizeof(int));

        if(ele == nullptr)
        {
```



```
        infa *toadd = malloc(sizeof(infa));
        memcpy(toadd, &dat, sizeof(infa));
        HASH_ADD_INT(buf, classCode, toadd);
        //iHashTable.Add(buf, &dat.classCode, sizeof(int), &dat);
    }
    else
    {
        ele->sumArea += dat.sumArea;
        ele->sumWeight += dat.sumWeight;
    }
}
/*
Iterator * it = iHashTable.NewIterator(buf);
infa *dat;
for(dat = it->GetFirst(it); dat != nullptr; dat = it->GetNext(it))
{
    if(dprintf(fout, "%s|%d|%f\n", dat->className, dat->sumArea, dat-
>sumWeight) < 0)
        die("dprintf failed.");
}
iHashTable.DeleteIterator(it);*/
**** Comment for C oj report
遍历 hash 表，取出所有求和完毕的项，打印到指定的文件。
***/// Special comment end

infa *current, *tmp;
HASH_ITER(hh, buf, current, tmp)
{
    if(dprintf(fout, "%s|%d|%f\n", current->className, current->sumArea,
current->sumWeight) < 0)
        die("dprintf failed.");
}
```



```
/** Comment for C oj report
```

清理，结束。

```
***/// Special comment end
```

```
    HASH_CLEAR(hh, buf);
```

```
    return;
```

```
}
```

```
/** Comment for C oj report
```

原理同上 保存准备打印的信息。

```
***/// Special comment end
```

```
typedef struct __to_print_2
```

```
{
```

```
    const veg_info *pveg;
```

```
    int sumArea;
```

```
    float sumWeight;
```

```
    UT_hash_handle hh;          /* makes this structure hashable */
```

```
} infb;
```

```
typedef infb * HashTable_infb;
```

```
/** Comment for C oj report
```

函数名写的很清楚。做 sort 的一个过滤器。

```
***/// Special comment end
```

```
void __filter_sort(fd fin, fd fout)
```

```
{
```

```
    doCmd(fin, fout, FD_DEFAULT, "sort --field-separator='|' -k4");
```

```
    return;
```

```
}
```

```
/** Comment for C oj report
```

实现 job_report_by_year_and_veg_name。老手段，hash 表，遍历打印结果。如果要投入生产环境，必须进一步提高代码复用率。

```
***/// Special comment end
```

```
void __impl_job_report_by_year_and_veg_name(fd fout, int year_begin, int
```



```
year_end/*Not                                included*/)                // #print
vegName+className+sumArea+sumWeight(decrease)
{
    HashTable_infb buf = NULL;
    //HashTable *buf = iHashTable.Create(sizeof(infb));
    for(rListIterator it = jbuf->first; it != nullptr; it = it->next)
    {
        job_info *p = (job_info *)it->data;
        if(p->year < year_begin || p->year >= year_end)
            continue;
        infb dat;
        dat.pveg = p->pveg;
        dat.sumArea = p->area;
        dat.sumWeight = p->weight;

        infb *ele;
        HASH_FIND_INT(buf, &dat.pveg->no, ele);
        //infb *ele = iHashTable.GetElement(buf, &dat.pveg->no,
sizeof(int));
        if(ele == nullptr)
        {
            infb *toadd = malloc(sizeof(infb));
            memcpy(toadd, &dat, sizeof(infb));
            HASH_ADD_INT(buf, pveg->no, toadd);
        }
        //iHashTable.Add(buf, &dat.pveg->no, sizeof(int), &dat);
    else
    {
        ele->sumArea += dat.sumArea;
        ele->sumWeight += dat.sumWeight;
    }
}
```



```
}
/*
Iterator * it = iHashTable.NewIterator(buf);
infb *dat;
for(dat = it->GetFirst(it); dat != nullptr; dat = it->GetNext(it))
{
    if(dprintf(fout, "%s|%s|%d|%f\n", dat->pveg->name, dat->pveg->
    >pclass->name, dat->sumArea, dat->sumWeight) < 0)
        die("dprintf failed.");
}
iHashTable.DeleteIterator(it);*/
infb *current, *tmp;
HASH_ITER(hh, buf, current, tmp)
{
    if(dprintf(fout, "%s|%s|%d|%f\n", current->pveg->name, current->
    >pveg->pclass->name, current->sumArea, current->sumWeight) < 0)
        die("dprintf failed.");
}
HASH_CLEAR(hh, buf);
return;
}
**** Comment for C oj report
比三下划线的禁止调用版本多了一次 sort，这是题目要求的操作。
***/// Special comment end
void __impl_job_report_by_year_and_veg_name_decrease(fd fout, int year_begin,
int year_end/*Not included*/)
{
    fd pipefd[2];
    if(pipe(pipefd) == -1) die("pipe failed");
    __impl_job_report_by_year_and_veg_name(pipefd[1], year_begin,
year_end);
}
```



```
close(pipefd[1]);
__filter_sort(pipefd[0], 1);
close(pipefd[0]);
return;
}

/** Comment for C oj report
对指定类的重量求和。简单的遍历。
***/
Special comment end

void _impl_check_weight_by_veg_class(fd fout, int classNum) // #print
sumWeight
{
    float sumWeight = 0;
    for(rListIterator it = jbuf->first; it != nullptr; it = it->next)
    {
        job_info *p = it->data;
        if(p->pveg->pclass->code == classNum)
            sumWeight += p->weight;
    }
    if(dprintf(fout, "%f\n", sumWeight) < 0)
        die("dprintf failed.");
    return;
}

<<< end shit-op.c
>>> app/veg-class.c

#include "veg-class.h"
#include "list.h"
#include "fmtio.h"
#include <fcntl.h>
```



```
#include <unistd.h>

#define FILE_PATH DATA_DUMP_PATH_HEAD "vcls"

/* Usage tip.
 *   for (myListIterator it = goodsList->first; it != NULL; it=it->next)
 *   {
 *       Goods good = *(Goods*)(it->data);
 *       do something...
 *   }
 */
rList vcbuf;

**** Comment for C oj report
dump 内存链表到外存，由析构函数调用。出错时 on_exit 不会被调用(quick_exit)。
****/// Special comment end
void veg_class_dump()
{
    fd cfgFile = open(FILE_PATH, O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR |
S_IWUSR | S_IRGRP | S_IROTH);
    if(cfgFile == -1) die("open file %s for write failed.", FILE_PATH);
    if(fmt_write_file(cfgFile, vcbuf, sizeof(veg_class), nullptr) < 0)
die("fmt_write_file failed.");
    close(cfgFile);
}

**** Comment for C oj report
析构函数。
****/// Special comment end
void veg_class_destructor()
{
    veg_class_dump();
}
```




```
    rList_destructor(vcbuf);
}

/** Comment for C oj report
封装闭包。
***/
/** Special comment end
void __vc_exitfunc(int __ar, void *__ar2) {veg_class_destructor();}
/** Comment for C oj report
构造函数。会读外存数据进链表。
***/
/** Special comment end
void veg_class_constructor()
{
    vcbuf = rList_newlist();
    fd cfgFile = open(FILE_PATH, O_RDONLY);
    if(errno == 2) goto file_not_exist;
    if(cfgFile == -1) die("open file %s for read failed.", FILE_PATH);
    if(fmt_read_file(cfgFile, vcbuf, sizeof(veg_class), nullptr) < 0)
die("fmt_read_file failed.");
    close(cfgFile);
file_not_exist:
    on_exit(__vc_exitfunc, nullptr);
}

/** Comment for C oj report
链表 set
***/
/** Special comment end
void veg_class_set(veg_class to_push)
{
    veg_class *toReplace = nullptr; RECORD
    for(rListIterator it = vcbuf->first; it != nullptr; it = it -> next)
    {RECORD
        veg_class *p = (veg_class *) it->data;RECORD
        if(p->code == to_push.code)
```



```
        toReplace = p;

    }RECORD

    if(toReplace)

        *toReplace = to_push;

    else

        rList_push_back(vcbuf, &to_push, sizeof(to_push));RECORD
}

/**/ Comment for C oj report

链表 ls

***/ Special comment end

void veg_class_ls(fd fout)

{RECORD

    for(rListIterator it = vcbuf->first; it != nullptr; it = it -> next)

    {RECORD

        veg_class *p = (veg_class *)it->data;RECORD

        dprintf(fout, "%d|s\n", p->code, p->name);RECORD

    }RECORD

}

/**/ Comment for C oj report

链表 rm

***/ Special comment end

bool veg_class_rm(int code)

{

    for(rListIterator it = vcbuf->first; it != nullptr; it = it -> next)

    {

        veg_class *p = (veg_class *)it->data;

        if(p->code == code)

        {

            rList_eraseElem(vcbuf, it);

            return true;

        }

    }

}
```



```
    }  
    return false;  
}  
  
/** Comment for C oj report  
看函数名  
***/// Special comment end  
const veg_class *code_to_veg_class(int code)  
{  
    for(rListIterator it = vcbuf->first; it != nullptr; it = it -> next)  
    {  
        veg_class *p = (veg_class *)it->data;  
        if(p->code == code)  
            return p;  
    }  
    return nullptr;  
}  
  
<<< end veg-class.c  
>>> app/data.h  
#ifndef _SRC_DATA_H  
#define _SRC_DATA_H 1  
  
#include "common.h"  
  
#define DATA_DUMP_PATH_HEAD "/tmp/.chw/dat-"  
#define MAX_OBJECT 1024  
  
/** Comment for C oj report  
这里是数据结构，早已在上一小节介绍过。不再赘述。  
***/// Special comment end  
typedef struct _veg_class {
```



```
    int code;

    char name[64];
} veg_class;

typedef struct _veg_info {
    int no; //Must self-inc autoly.
    char name[64];
    const veg_class *pclass;
    int __class_code; //Unused(!!Unmaintained!!). Just to make the silly
doc happy. //Defined as pclass->code
    char nutr_info[64];
} veg_info;

typedef struct _job_info {
    int no;
    const veg_info *pveg;
    int __veg_no; //Unused(!!Unmaintained!!). Just to make the silly doc
happy. //Defined as pveg->no
    int area;
    float weight;
    int year;
    char __year[5]; //Unused(!!Unmaintained!!). Just to make the silly doc
happy. //Defined as string(year)
} job_info;

#endif

<<< end data.h
>>> app/job.h

#ifndef _CHW_JOB_H
#define _CHW_JOB_H 1

#include "common.h"
```



```
#include "data.h"

void job_set(job_info to_push);

void job_ls(fd fout);

bool job_rm(int no);

/** Comment for C oj report
头文件是接口定义。详情见.c
***/

***/ Special comment end

#endif // _CHW_JOB_H

<<< end job.h

>>> app/veg.h

#ifndef _CHW_VEG_H
#define _CHW_VEG_H 1

#include "common.h"

#include "data.h"

void veg_info_set(veg_info to_push);

void veg_info_ls(fd fout);

bool veg_info_rm(const char *name);

/** Comment for C oj report
头文件是接口定义。详情见.c
***/

***/ Special comment end

const veg_info *no_to_veg_info(int no);

#endif // _CHW_VEG_H

<<< end veg.h

>>> app/job.c

#include "job.h"

#include "list.h"

#include "veg.h"

#include "fmtio.h"
```



```
#include <fcntl.h>

#include <unistd.h>

#define FILE_PATH DATA_DUMP_PATH_HEAD "jinf"

#define streql(sa, sb) (strlen(sa) == strlen(sb) && strcmp(sa, sb) == 0)

rList jbuf;

/** Comment for C oj report
函数名即为注释
***/
void job_dump()
{
    fd cfgFile = open(FILE_PATH, O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR |
S_IWUSR | S_IRGRP | S_IROTH);
    if(cfgFile == -1) die("open file %s for write failed.", FILE_PATH);
    rList_for_each(it, jbuf)
    { //trick for dump
        job_info *p = it->data;
        /** Comment for C oj report
将指针还原成 code，存储依然有效
***/
        p->pveg = (const veg_info *)p->pveg->no;
    }
    if(fmt_write_file(cfgFile, jbuf, sizeof(job_info), nullptr) < 0)
die("fmt_write_file failed.");
    close(cfgFile);
}

/** Comment for C oj report
函数名即为注释
***/
```



```
void job_destructor()
{
    job_dump();
    rList_destructor(jbuf);
}

/**/ Comment for C oj report
函数名即为注释
***/ Special comment end

void __j_exitfunc(int __ar, void *__ar2) {job_destructor();}

/**/ Comment for C oj report
函数名即为注释
***/ Special comment end

void job_constructor()
{
    jbuf = rList_newlist();
    fd cfgFile = open(FILE_PATH, O_RDONLY);
    if(errno == 2) goto file_not_exist;
    if(cfgFile == -1) die("open file %s for read failed.", FILE_PATH);
    if(fmt_read_file(cfgFile, jbuf, sizeof(job_info), nullptr) < 0)
die("fmt_read_file failed.");
    close(cfgFile);
    rList_for_each(it, jbuf)
    {
        job_info *p = it->data;
        debug printf(">%d", (int)p->pveg);
    }

    /**/ Comment for C oj report
    将存盘时转换成 code 的指针重新拿回来。找不到会报错。数据损坏。
    ***/ Special comment end

    p->pveg = no_to_veg_info((int)p->pveg);
    if(p->pveg == nullptr) die("pveg_info not found. data boom.");
}
```



```
file_not_exist:
    on_exit(___j_exitfunc, nullptr);
}

/**/ Comment for C oj report
函数名即为注释
***/ Special comment end

void job_set(job_info to_push)
{
    job_info *toReplace = nullptr;
    for(rListIterator it = jbuf->first; it != nullptr; it = it -> next)
    {
        job_info *p = (job_info *) it->data;
        if(to_push.no == p->no)
            toReplace = p;
    }
    if(toReplace)
        *toReplace = to_push;
    else
        rList_push_back(jbuf, &to_push, sizeof(to_push));
}

/**/ Comment for C oj report
函数名即为注释
***/ Special comment end

void job_ls(fd fout)
{RECORD
    for(rListIterator it = jbuf->first; it != nullptr; it = it -> next)
    {RECORD
        job_info *p = (job_info *)it->data;RECORD
        debug printf("%d", p->pveg->no);
        RECORD
        dprintf(fout, "%d|%d|%s|%d|%f|%d\n", p->no, p->pveg->no, p->pveg-
```




```
>name, p->area, p->weight, p->year);
    }RECORD
}

/** Comment for C oj report
函数名即为注释
***/ Special comment end

bool job_rm(int no)
{
    for(rListIterator it = jbuf->first; it != nullptr; it = it -> next)
    {
        job_info *p = (job_info *)it->data;
        if(p->no == no)
        {
            rList_eraseElem(jbuf, it);
            return true;
        }
    }
    return false;
}

<<< end job.c

>>> app/shit-op.h

#ifndef _CHW_SHIT_OP_H
#define _CHW_SHIT_OP_H 1

#include "common.h"

/** Comment for C oj report
头文件。只是接口。已经在源文件解释过用法。
***/ Special comment end

void          _impl_job_report_by_veg_class(fd          fout);          // #print
className+sumArea+sumWeight
```



```
void _impl_job_report_by_year_and_veg_name_decrease(fd fout, int year_begin,
int          year_end/*Not          included*/);          // #print
vegName+className+sumArea+sumWeight(decrease)
void _impl_check_weight_by_veg_class(fd fout, int classNum); // #print
sumWeight

#endif // _CHW_SHIT_OP_H
```

```
<<< end shit-op.h
```

```
>>> app/veg.c
```

```
#include "veg.h"
#include "veg-class.h"
#include "list.h"
#include "fmtio.h"
#include <fcntl.h>
#include <unistd.h>
```

```
/** Comment for C oj report
```

此源文件采用的所有函数几乎与 veg-info.c 完全相同，含义也完全相同，内部实现的所有细节也完全相同。本该进行一次抽象和复用，但条件不允许。

这里的注释不再赘述，非常感谢谅解！

```
***/// Special comment end
```

```
#define FILE_PATH DATA_DUMP_PATH_HEAD "vinf"
```

```
#define streql(sa, sb) (strlen(sa) == strlen(sb) && strcmp(sa, sb) == 0)
```

```
rList vibuf;
```

```
/** Comment for C oj report
```

函数名即为注释

```
***/// Special comment end
```

```
void veg_info_dump()
```



```
{
    fd cfgFile = open(FILE_PATH, O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR |
S_IWUSR | S_IRGRP | S_IROTH);
    if(cfgFile == -1) die("open file %s for write failed.", FILE_PATH);
    rList_for_each(it, vibuf)
    { //trick for dump
        veg_info *p = it->data;
        p->pclass = (const veg_class *)p->pclass->code;
    }

    if(fmt_write_file(cfgFile, vibuf, sizeof(veg_info), nullptr) < 0)
die("fmt_write_file failed.");
    close(cfgFile);
}

/**** Comment for C oj report
函数名即为注释
****/// Special comment end
void veg_info_destructor()
{
    veg_info_dump();
    rList_destructor(vibuf);
}

/**** Comment for C oj report
函数名即为注释
****/// Special comment end
void __v_exitfunc(int __ar, void *__ar2) {veg_info_destructor();}

/**** Comment for C oj report
函数名即为注释
****/// Special comment end
void veg_info_constructor()
{
    vibuf = rList_newlist();
```



```
fd cfgFile = open(FILE_PATH, O_RDONLY);
if(errno == 2) goto file_not_exist;
if(cfgFile == -1) die("open file %s for read failed.", FILE_PATH);
    if(fmt_read_file(cfgFile, vibuf, sizeof(veg_info), nullptr) < 0)
die("fmt_read_file failed.");
close(cfgFile);
rList_for_each(it, vibuf)
{
    veg_info *p = it->data;
    p->pclass = code_to_veg_class((int)p->pclass);
    if(p->pclass == nullptr) die("pveg_class not found. data boom.");
}
file_not_exist:
    on_exit(___v_exitfunc, nullptr);
}
/** Comment for C oj report
函数名即为注释
***/// Special comment end
void veg_info_set(veg_info to_push)
{
    veg_info *toReplace = nullptr;
    for(rListIterator it = vibuf->first; it != nullptr; it = it -> next)
    {
        veg_info *p = (veg_info *) it->data;
        if(streql(to_push.name, p->name))
            toReplace = p;
    }
    if(toReplace)
    {
        to_push.no = toReplace->no;
        *toReplace = to_push;
    }
}
```



```
    }
    else
    {
        to_push.no = vibuf->first == nullptr ? 1 : ((veg_info *)vibuf->last->data)->no + 1;
        rList_push_back(vibuf, &to_push, sizeof(to_push));
    }
}

/**/ Comment for C oj report
函数名即为注释
***/ Special comment end

void veg_info_ls(fd fout)
{
    for(rListIterator it = vibuf->first; it != nullptr; it = it->next)
    {
        veg_info *p = (veg_info *)it->data;RECORD
        fprintf(fout, "%d|%s|%d|%s\n", p->no, p->name, p->pclass->code, p->nutr_info);RECORD
    }
}

/**/ Comment for C oj report
函数名即为注释
***/ Special comment end

bool veg_info_rm(const char *name)
{
    for(rListIterator it = vibuf->first; it != nullptr; it = it -> next)
    {
        veg_info *p = (veg_info *)it->data;
        if(streql(name, p->name))
        {
            rList_eraseElem(vibuf, it);
        }
    }
}
```



```
        return true;
    }
}

return false;
}

/** Comment for C oj report
函数名即为注释
***/ Special comment end

const veg_info *no_to_veg_info(int no)
{
    for(rListIterator it = vibuf->first; it != nullptr; it = it -> next)
    {
        veg_info *p = (veg_info *)it->data;
        if(p->no == no)
            return p;
    }
    return nullptr;
}

<<< end veg.c

>>> app/main.c

#include "common.h"
#include "cmd.h"
#include <stdio.h>

void veg_class_constructor();
void veg_info_constructor();
void job_constructor();

/** Comment for C oj report
main函数在这里。依次调用构造函数，检查并创建数据文件夹，然后 exec 命令解析器。
***/ Special comment end

int main(int arglen, char **argv)
```



```
{  
    if(sizeof(unsigned char) != 1) die("assertion failed: sizeof uchar !=  
1");  
    system("mkdir /tmp/.chw > /dev/null 2>&1");  
    veg_class_constructor(); veg_info_constructor(); job_constructor();  
    return do_cmd_process(arglen, argv);  
}
```

<<< end main.c

>>> app/veg-class.h

```
#ifndef _PROJECT_VEG_CLASS_H  
#define _PROJECT_VEG_CLASS_H 1  
#include "common.h"  
#include "data.h"  
void veg_class_set(veg_class to_push);  
void veg_class_ls(fd fout);  
bool veg_class_rm(int code);  
  
const veg_class *code_to_veg_class(int code);  
  
#endif //_PROJECT_VEG_CLASS_H
```

<<< end veg-class.h

>>> app/cmd.c

```
#include "cmd.h"  
#include "data.h"  
#include "veg-class.h"  
#include "veg.h"  
#include "job.h"  
#include "filter.h"  
#include "shit-op.h"
```



```
#include <unistd.h>

#include <stdio.h>

#include <string.h>

/** Comment for C oj report
打印帮助信息。exeName 会被写进帮助信息，传 NULL 也可以。
***/

void usage(const char *exeName)
{
    if(exeName == NULL)
        exeName = "./thisBin";

    #if defined nl || defined exe
    #error __FILE__:__LINE__ alias conflict. Please repair.
    #endif

    #define nl "\n"
    #define exe ,exeName);printf(
    printf(
        "Usage:" nl
        "%s <sub-command> <required arguments ...> [optional arguments ...]"
exe nl nl
        "sub-commands:" nl
        "veg-class-set <classNum> <className>" nl
        "veg-class-ls [--code=...]" nl
        "veg-class-rm <classNum>" nl
        "veg-info-set <vegName> <classNum> <neuInfoStr>" nl
        "veg-info-ls [--name-keyword=...] [--neu-keyword=...] [--class-
code=...]" nl
        "veg-info-rm <vegName>" nl
        "job-set <jobNum> <vegNum> <area> <weight> <year>" nl
        "job-ls [--veg-name-keyword=...] [--veg-name=...] [--year=...]" nl
        "job-rm <jobNum>" nl nl
}
```




```
//Shit below: Must be implemented by bash script.
"job-report-by-veg-class #print className+sumArea+sumWeight" nl
"job-report-by-year-and-veg-name <yearBegin> <yearEnd(NOT included)>
#print vegName+className+sumArea+sumWeight(decrease)" nl
"check-weight-by-veg-class <classNum> #print sumWeight" nl
"Please use veg-info-ls to print veg info by neu keyword." nl nl
);
#undef exe
#undef nl
return;
}
```

/** Comment for C oj report

为字符串处理做一些准备，毕竟c本不擅长做这个。

***/// Special comment end

```
#define streql(sa, sb) (strlen(sa) == strlen(sb) && strcmp(sa, sb) == 0)
#define strheadeql(__sa, __sb) ({int __la=strlen(__sa);int
__lb=strlen(__sb);!strncmp(__sa,__sb,min(__la,__lb));})
//Warning: NO alias unpack.
#define ifcmd(t) if(streql(subcmd, t))
```

```
void deal_veg_info_ls(size_t, char **);
```

```
void deal_job_ls(size_t, char **);
```

```
//TODO: finish deal_ls.
```

/** Comment for C oj report

这里是真正的main函数，语法解析器。本该使用gnu为linux设计的库getopt，因为那更稳定，用户更友好，更节约开发时间，更可维护。

***/// Special comment end

```
int do_cmd_process(size_t arglen, char **argv)
{
```



```
if(arglen < 2)
{
    usage(arglen == 1 ? argv[0] : NULL);
    return 1;
}

const char *binName = argv[0];
const char *subcmd = argv[1];

/** Comment for C oj report
快速抛出异常，快速检查，减少代码量，增加可读性/维护性
***/
#define BAD_ARGUMENT do {usage(binName); die("bad_argument");} while(0)
#define REQUIRE_ARGLEN(n) if(arglen < n) BAD_ARGUMENT;

/** Comment for C oj report
下面是简单的构造。ls 需要特殊对待。
***/
ifcmd("veg-class-set")
{
    REQUIRE_ARGLEN(4)
    veg_class vc;
    vc.code = atoi(argv[2]);
    strncpy(vc.name, argv[3], 63);
    veg_class_set(vc);
}
else ifcmd("veg-class-ls")
{
    REQUIRE_ARGLEN(2)
    if(arglen == 2)
        veg_class_ls(1);
    else
    { //dirty.
        if(strheadeql(argv[2], "--code="))
```



```
{
    const char *codeArg = argv[2] + 7;
    fd pipefd[2];
    if(-1 == pipe(pipefd))
        die("failed to init pipe.");
    /*** Comment for C oj report
    ls 将输出打印到 pipefd
    ***/
    /*** Special comment end
        veg_class_ls(pipefd[1]);
        close(pipefd[1]);
    /*** Comment for C oj report
    按要求进行过滤，送到 1 号管道，即 stdout
    ***/
    /*** Special comment end
        filter(pipefd[0], 1, true, codeArg, VEG_CLASS_CODE);
        close(pipefd[0]);
    }
    else
        BAD_ARGUMENT;
}

}

else ifcmd("veg-class-rm")
{
    REQUIRE_ARGLEN(3)
    if(veg_class_rm(atoi(argv[2])) == false)
        printf("Failed to remove record. Confirm if it do exists.");
}

else ifcmd("veg-info-set")
{
    REQUIRE_ARGLEN(5)
    veg_info vi;
    strncpy(vi.name, argv[2], 63);
```



```
vi.pclass = code_to_veg_class(atoi(argv[3]));
if(vi.pclass == nullptr) die("v_class not exist.");
strncpy(vi.nutr_info, argv[4], 63);
veg_info_set(vi);
}
else ifcmd("veg-info-ls")
{RECORD
    REQUIRE_ARGLEN(2) RECORD
    deal_veg_info_ls(arglen, argv);
}
else ifcmd("veg-info-rm")
{
    REQUIRE_ARGLEN(3)
    if(veg_info_rm(argv[2]) == false)
        printf("Failed to remove record. Confirm if it do exists.");
}
else ifcmd("job-set")
{
    REQUIRE_ARGLEN(7)
    job_info ji;
    ji.no = atoi(argv[2]);
    ji.pveg = no_to_veg_info(atoi(argv[3]));
    if(ji.pveg == nullptr) die("v_info not exist.");
    ji.area = atoi(argv[4]);
    ji.weight = (float)atof(argv[5]);
    ji.year = atoi(argv[6]);
    job_set(ji);
}
else ifcmd("job-ls")
{
    REQUIRE_ARGLEN(2)
```



```
        deal_job_ls(arglen, argv);
    }
    else ifcmd("job-rm")
    {
        REQUIRE_ARGLEN(3)
        if(job_rm(atoi(argv[2])) == false)
            printf("Failed to remove record. Confirm if it do exists.");
    }
    else ifcmd("job-report-by-veg-class")
    {
        REQUIRE_ARGLEN(2)
        _impl_job_report_by_veg_class(1);
    }
    else ifcmd("job-report-by-year-and-veg-name")
    {
        REQUIRE_ARGLEN(4)
        _impl_job_report_by_year_and_veg_name_decrease(1, atoi(argv[2]),
        atoi(argv[3]));
    }
    else ifcmd("check-weight-by-veg-class")
    {
        REQUIRE_ARGLEN(3)
        _impl_check_weight_by_veg_class(1, atoi(argv[2]));
    }
    else
        BAD_ARGUMENT;
#undef REQUIRE_ARGLEN
    return 0;
}

/*
```



```
*      "job-report-by-veg-class #print className+sumArea+sumWeight" nl
      "job-report-by-year-and-veg-name <yearBegin> <yearEnd(NOT included)>
#print vegName+className+sumArea+sumWeight(decrease)" nl
      "check-weight-by-veg-class <classNum> #print sumWeight" nl
      "Please use veg-info-ls to print veg info by neu keyword." nl nl

      "veg-info-ls [--name-keyword=...] [--neu-keyword=...] [--class-
code=...]" nl
      "job-ls [--veg-name-keyword=...] [--veg-name=...] [--year=...]" nl

if(arglen == 2)
    veg_class_ls(1);
else
{
    if(strheadeql(argv[2], "--code="))
    {
        const char *codeArg = argv[2] + 7;
        fd pipefd[2];
        if(-1 == pipe(pipefd))
            die("failed to init pipe.");
        veg_class_ls(pipefd[1]);
        filter(pipefd[0], 1, true, codeArg, VEG_CLASS_CODE);
    }
    else
        BAD_ARGUMENT;
}

*/
```

/** Comment for C oj report

这里为另外几个 ls 提供了过滤器的实现。依然比较 dirty。

没什么黑科技。继续。

```
***/// Special comment end

void deal_veg_info_ls(size_t arglen, char **argv)
{RECORD
    debug printf("%d\n", (int)arglen);
    const char *binName = argv[0];
    fd pipefd[2];
    if(-1 == pipe(pipefd))
        die("failed to init pipe.");
    veg_info_ls(pipefd[1]);
    close(pipefd[1]);RECORD
    for(size_t cter = 2; cter < arglen; ++cter)
    {
        if(strheadeql(argv[cter], "--name-keyword="))
        {RECORD
            fd tmpPipe[2];
            if(-1 == pipe(tmpPipe))
                die("failed to init pipe.");
            filter(pipefd[0], tmpPipe[1], false, argv[cter] + 15, VEG_NAME);
            close(tmpPipe[1]);
            close(pipefd[0]);
            pipefd[0] = tmpPipe[0];
            continue;
        }
        if(strheadeql(argv[cter], "--neu-keyword="))
        {RECORD
            fd tmpPipe[2];RECORD
            if(-1 == pipe(tmpPipe))
                die("failed to init pipe.");
            filter(pipefd[0], tmpPipe[1], false, argv[cter] + 14,
VEG_NEU);RECORD
```



```
        debug printf("tp0=%d, p0=%d\n", tmpPipe[0], pipefd[0]);RECORD
        close(tmpPipe[1]);
        close(pipefd[0]);
        pipefd[0] = tmpPipe[0];
        continue;
    }
    if(strheadeql(argv[cter], "--class-code="))
    {
        fd tmpPipe[2];
        if(-1 == pipe(tmpPipe))
            die("failed to init pipe.");
        filter(pipefd[0], tmpPipe[1], true, argv[cter] + 13,
VEG_CLASS_CODE_BY_INFO);
        close(tmpPipe[1]);
        close(pipefd[0]);
        pipefd[0] = tmpPipe[0];
        continue;
    }
    BAD_ARGUMENT;
}

debug printf("p0=%d\n",pipefd[0]);
printFd(pipefd[0]);
close(pipefd[0]);
return;
}

/**/ Comment for C oj report
函数名即为注释
***/ Special comment end

void deal_job_ls(size_t arglen, char **argv)
{RECORD
    const char *binName = argv[0];
```




```
fd pipefd[2];
if(-1 == pipe(pipefd))
    die("failed to init pipe.");
job_ls(pipefd[1]);
close(pipefd[1]);
for(size_t cter = 2; cter < arglen; ++cter)
{
    if(strheadeql(argv[cter], "--veg-name-keyword="))
    {
        fd tmpPipe[2];
        if(-1 == pipe(tmpPipe))
            die("failed to init pipe.");
        filter(pipefd[0], tmpPipe[1], false, argv[cter] + 19,
VEG_NAME_BY_JOB);
        close(tmpPipe[1]);
        close(pipefd[0]);
        pipefd[0] = tmpPipe[0];
        continue;
    }
    if(strheadeql(argv[cter], "--veg-name="))
    {
        fd tmpPipe[2];
        if(-1 == pipe(tmpPipe))
            die("failed to init pipe.");
        filter(pipefd[0], tmpPipe[1], true, argv[cter] + 11,
VEG_NAME_BY_JOB);
        close(tmpPipe[1]);
        close(pipefd[0]);
        pipefd[0] = tmpPipe[0];
        continue;
    }
}
```



```
    if(strheadeql(argv[cter], "--year="))
    {
        fd tmpPipe[2];
        if(-1 == pipe(tmpPipe))
            die("failed to init pipe.");
        filter(pipefd[0], tmpPipe[1], true, argv[cter] + 7, JOB_YEAR);
        close(tmpPipe[1]);
        close(pipefd[0]);
        pipefd[0] = tmpPipe[0];
        continue;
    }
    BAD_ARGUMENT;
}

printFd(pipefd[0]);
close(pipefd[0]);
return;
}
```

```
#undef BAD_ARGUMENT

<<< end cmd.c

>>> app/cmd.h

#ifndef _SRC_CMD_H
#define _SRC_CMD_H

#include "common.h"

/**/ Comment for C oj report

头文件。

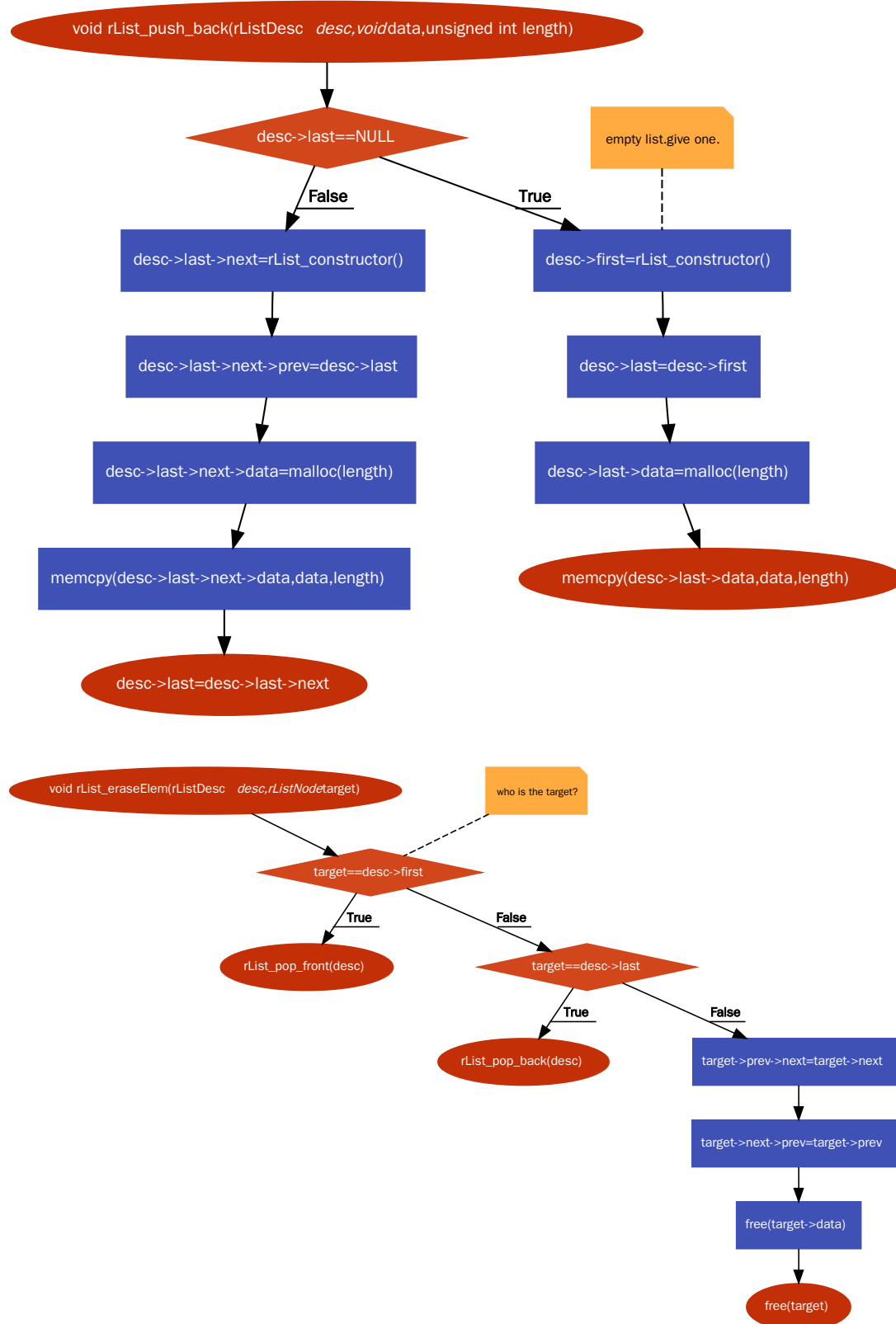
***/ Special comment end

int do_cmd_process(size_t, char **);

#endif

<<< end cmd.h
```

下面应当为'主体功能实现算法'，即'算法'提供流程图。



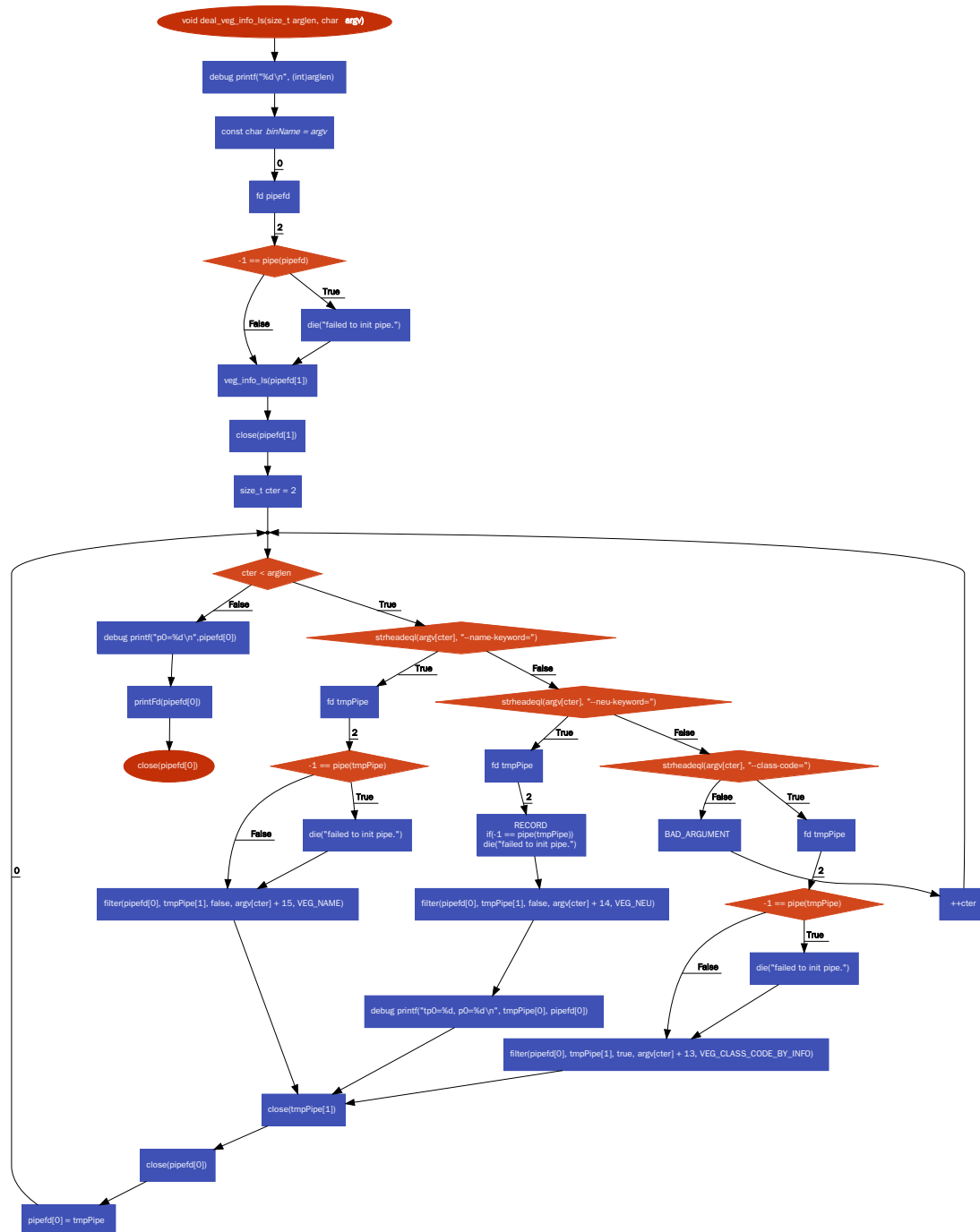
可能值得特别说明的是 `do_cmd_process`。它进行语法解析和子操作分类，并进

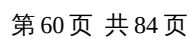
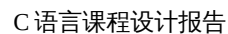


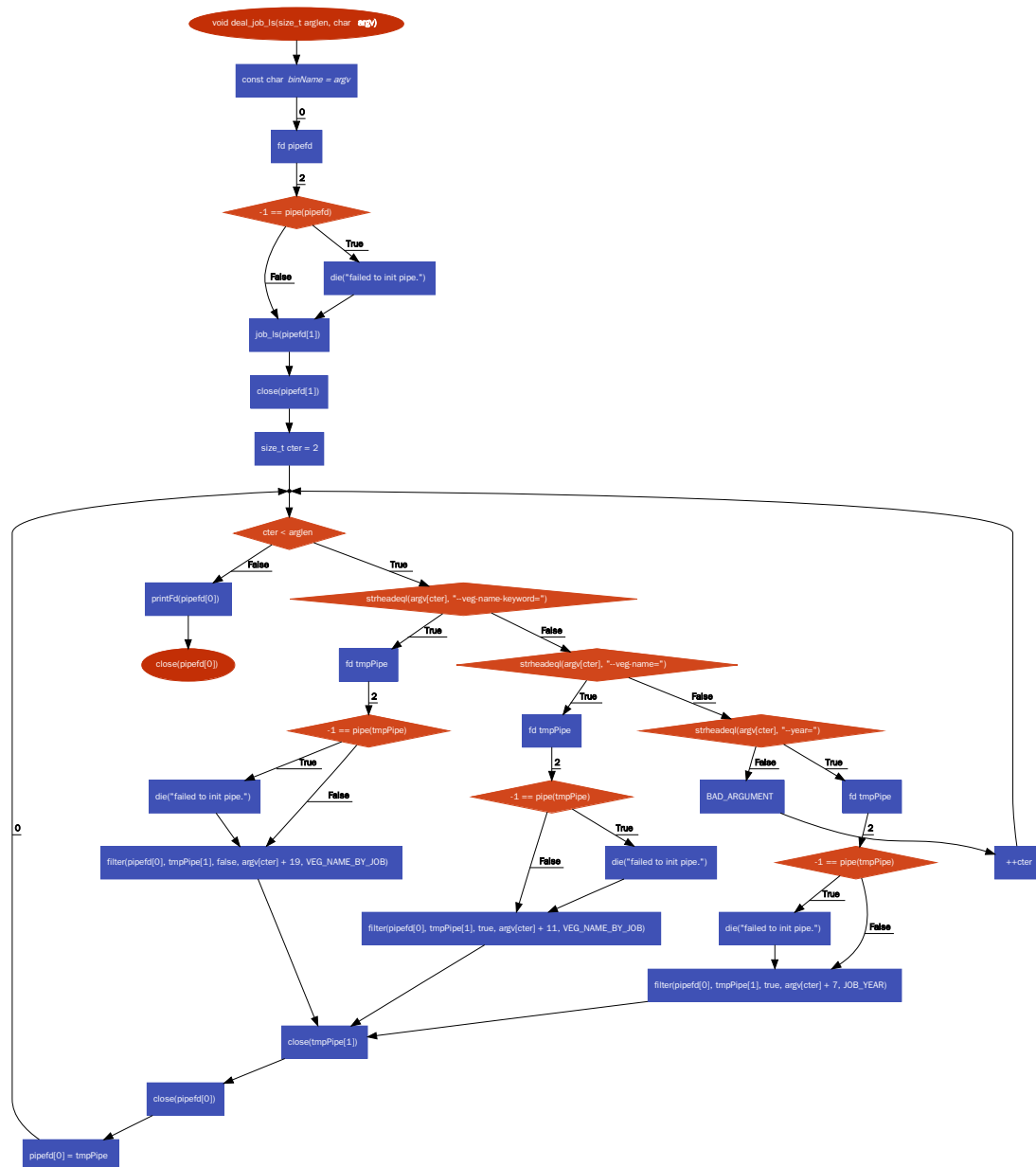
行管道对接和多次过滤等 unix 哲学常用操作。给出流程图。

(可前往 <https://drive.recolic.net/d/a177432eaf/> 获得无法打印清楚的原矢量图)

将上图中的 deal_*_ls 分别予以展开







下面是一些无关紧要的 module 的源码。并不一定全部都被用到

```
>>> mod/console.h
#ifndef _SRC_CONSOLE_H
#define _SRC_CONSOLE_H 1
/** Comment for C oj report
Defines some function for better system(). sync implement.
```



```
***/// Special comment end

#include "common.h"

extern const fd FD_DEFAULT;

int doCmd(fd _stdin, fd _stdout, fd _stderr, const char *cmd);

int doCmd_pure(const char *cmd);

// pass FD_DEFAULT to leave fd as default.

#endif //_SRC_CONSOLE_H

<<< end console.h
>>> mod/console.c

#include "console.h"

#include <string.h>
#include <stdio.h>
#include <unistd.h>

const fd FD_DEFAULT = -1;

#ifdef stdin
#undef stdin
#endif

#ifdef stdout
#undef stdout
#endif

#ifdef stderr
#undef stderr
#endif

//NO INCLUDE BELOW -----

/**/ Comment for C oj report

docmd implement, using bash redirect. Maybe /bin/sh is better.
```




```
***/// Special comment end

int _doCmd(fd stdin, fd stdout, fd stderr, char *cmd, size_t cmdlen) //There
must be 64 bytes pre-allocated after \0. (44 bytes at least.)
{
    char *ins = cmd + cmdlen;
    auto ret = sprintf(ins, " < /dev/fd/%d > /dev/fd/%d 2> /dev/fd/%d",
stdin, stdout, stderr);
    if(ret > 63) die("Memory violate: sprintf prints too many chars(%s)",
ins);RECORD
    //return execl("/bin/bash", "/bin/bash", "-c", cmd, (char *)nullptr);
    return system(cmd);
}

/**** Comment for C oj report
interface
***/// Special comment end

int doCmd(fd _stdin, fd _stdout, fd _stderr, const char *cmd)
{
    if(_stdin == FD_DEFAULT) _stdin = 0;
    if(_stdout == FD_DEFAULT) _stdout = 1;
    if(_stderr == FD_DEFAULT) _stderr = 2;RECORD
    auto cmdlen = strlen(cmd); //Warning: potential DOS.
    auto newcmd = (char *)calloc((cmdlen + 64), sizeof(char));RECORD
    strcpy(newcmd, cmd);RECORD
    auto ret = _doCmd(_stdin, _stdout, _stderr, newcmd, cmdlen);RECORD
    free(newcmd);RECORD
    return ret;RECORD
}

int _doCmd_pure(char *cmd, size_t cmdlen) //There must be 64 bytes pre-
allocated after \0. (44 bytes at least.)
```



```
{
    char *ins = cmd + cmdlen;
    return execl("/bin/bash", "/bin/bash", "-c", cmd, (char *)nullptr);
}

/** Comment for C oj report
again, interface here.
***/
Special comment end
int doCmd_pure(const char *cmd)
{
    auto cmdlen = strlen(cmd); //Warning: potential DOS.
    auto newcmd = (char *)calloc((cmdlen + 64), sizeof(char));
    strcpy(newcmd, cmd);
    auto ret = _doCmd_pure(newcmd, cmdlen);
    free(newcmd);
    return ret;
}

<<< end console.c
>>> mod/hash.c
#include "hash.h"
#include <string.h>
/*
    This is the popular `times 33' hash algorithm which is used by perl
    and that also appears in Berkeley DB. This is one of the best known hash
    functions
    for strings because it is both computed very fast and distributes very
    well.

    This hash function is adapted from Chris Torek's Hash function for text
    in C,
```



```
Usenet message < 27038@mimsy.umd.edu > in comp.lang.c , October, 1990."
in Rich Salz's USENIX 1992 paper about INN
with GLU license.

*/

/** Comment for C oj report
This hash is licensed, and not adapted in this proj.
just packed in.
***/

__hash(unsigned char *str)
{
    unsigned long hash = 5381;
    int c;
    while (c = *str++)
        hash = ((hash << 5) + hash) + c; /* hash * 33 + c */
    return hash;
}

unsigned long ulong_hash(const void *data, size_t size)
{
    unsigned char *strbuf = malloc(size + sizeof(unsigned char));
    memcpy(strbuf, data, size);
    strbuf[size%sizeof(unsigned char)] = 0;
    for(size_t cter = 0; cter < size % sizeof(unsigned char); ++cter)
    {
        if(!strbuf[cter]) strbuf[cter] = 73;
    }
    auto ret = __hash(strbuf);
    free(strbuf);
    return ret;
}
```



```
int int_hash(const void *data, size_t size)
{
    return (int)ulong_hash(data, size);
}

<<< end hash.c

>>> mod/filter.h

#ifndef _SRC_FILTER_H
#define _SRC_FILTER_H 1

#include "../common.h"

/**/ Comment for C oj report
Define filter, which similar as bash/fish/zsh/sh/... '| grep "sth"'
***/ Special comment end
typedef enum {VEG_CLASS_CODE, VEG_NAME, VEG_NEU, VEG_CLASS_CODE_BY_INFO,
VEG_NAME_BY_JOB, JOB_YEAR} FILTER_CLASS;
void filter(fd fin, fd fout, bool fullMatch, const char *keyString,
FILTER_CLASS type);

/**/ Comment for C oj report
print pipe into stdout(1).
***/ Special comment end
void printFd(fd fin);
//print as string.

#endif //_SRC_FILTER_H

<<< end filter.h

>>> mod/hash.h

#ifndef _SRC_HASH_H
```



```
#define _SRC_HASH_H

/** Comment for C oj report
header. no need to instuct
***/ Special comment end

#include "../common.h"

int int_hash(const void *data, size_t size);
unsigned long ulong_hash(const void *data, size_t size);

#endif

<<< end hash.h
>>> mod/fstr.c

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "../common.h"

/*
 * This function is adapted from linux glibc demo.
 * with GNU license.
 *
 * by Recolic Keghart, Jul 2, 2017.
 */

/*
 * This function is adapted to C++, with GNU GPL followed.
 * C edition is abandoned, and should never be used.
 *
 * Recolic Keghart, Aug 24, 2017
 */

/** Comment for C oj report

fstr, used in die(), may cause potential memory leak. however, memory will
be recycled autoly by os on process terminate.

So it doesn't matter. Maybe here's why C++ is recommended.
```



```
***/// Special comment end

char *
fstr(const char *fmt, ...)
{
    int n;

    int size = 100;    /* Guess we need no more than 100 bytes */

    char *p, *np;
    va_list ap;

    /*** Comment for C oj report
    C va_list. as for c++, must be implemented by va template.
    ***/// Special comment end

    if ((p = (char *)malloc(size)) == NULL)
        die("malloc returns null.");

    while (1) {

        /* Try to print in the allocated space */

        va_start(ap, fmt);
        n = vsnprintf(p, size, fmt, ap);
        va_end(ap);

        /* Check error code */

        if (n < 0)
            die("vsnprintf returns %d.", n);

        /* If that worked, return the string */

        if (n < size)
            return p;
    }
}
```



```
/* Else try again with more space */

size = n + 1;      /* Precisely what is needed */

if ((np = (char *)realloc (p, size)) == NULL) {
    free(p);
    die("make_message realloc failed.");
    return NULL;
} else {
    p = np;
}
}

}

<<< end fstr.c
>>> mod/fstr.h

#ifndef _SRC_FSTR_H
#define _SRC_FSTR_H 1

/** Comment for C oj report
header.
***/

/** Special comment end

char *fstr(const char *fmt, ...); //must free after usage.
#endif //SRC_FSTR_H

<<< end fstr.h
>>> mod/filter.c

#include "filter.h"
#include "console.h"
#include <unistd.h>

/*
```



```
* Warning: This source SHOULD NEVER be adapted to industry environment,  
* as implementation is error-prone.  
* ONLY designed for HUST C homework usage.  
* Consider Shell Script if necessary.  
*  
* Recolic Keghart  
*/
```

```
int filterClassToIndex(FILTER_CLASS fc)  
{  
    switch(fc)  
    {  
        case VEG_CLASS_CODE:  
            return 0;  
        case VEG_NAME:  
            return 1;  
        case VEG_NEU:  
            return 3;  
        case VEG_CLASS_CODE_BY_INFO:  
            return 2;  
        case VEG_NAME_BY_JOB:  
            return 2;  
        case JOB_YEAR:  
            return 5;  
        default:  
            die("Failed to assign index: fc=%d", (int)fc);  
    }  
}
```

```
/** Comment for C oj report
```

Here is filter implement. It must ONLY be done by shell script or higher



```
classed grep-devel library.

***/// Special comment end

//TODO: implement filter after data-ls op.

void filter(fd fin, fd fout, bool fullMatch, const char *keyString,
FILTER_CLASS type)
{
    auto index = filterClassToIndex(type);
    auto keyLen = strlen(keyString);
    char *pattern = (char *)calloc(1034, sizeof(char));
    strcpy(pattern, "grep '^");
    if(index*6+keyLen+8 > 1023) die("keyString too long.");
    /*** Comment for C oj report
    Magic logic here, trying to construct a regex string. DO not try to
    understand them...
    ***/// Special comment end

    for(size_t cter = 0; cter < index; ++cter)
        memcpy(pattern + 6 * cter + 6, "[^|]*|", 6);
    if(fullMatch)
    {
        sprintf(pattern + 6 + 6 * index, "%s", keyString);
        sprintf(pattern + 6 + 6 * index + keyLen, "|'");
    }
    else
    {
        sprintf(pattern + 6 + 6 * index, "[^|]*");
        sprintf(pattern + 11 + 6 * index, "%s", keyString);
        sprintf(pattern + 11 + 6 * index + keyLen, "'");
    }
    debug printf("filter cmd=%s\n", pattern);
    debug printf("fout=%d", fout);
    doCmd(fin, fout, FD_DEFAULT, pattern);
}
```



```
    debug doCmd(fin, FD_DEFAULT, FD_DEFAULT, "grep '.');  
    free(pattern);  
    return;  
}
```

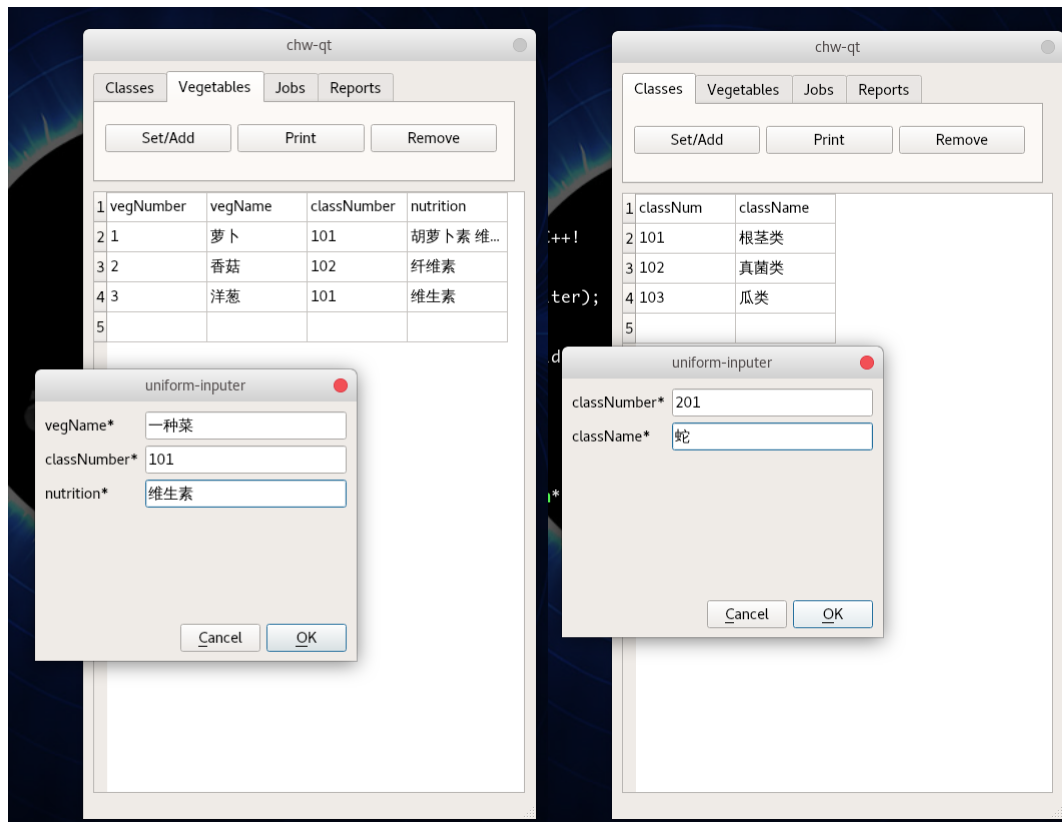
```
/** Comment for C oj report  
grep . is equal to connecting two pipe.  
**/// Special comment end  
void printFd(fd fin)  
/* maybe it's better.  
{  
    filter(fin, 1, false, "", FILTER_NULL);  
}  
*/  
{  
    doCmd(fin, FD_DEFAULT, FD_DEFAULT, "grep '.');  
}
```

<<< end filter.c

Ok. Now everything've been done. Go on into next part.

五、运行测试与结果分析

首先，为了展示系统实现了所要求的所有功能，对前端进行演示。





其次，简单展示 web socket server(wserver), backend(chw-cli), web interface(client.min.html).

Select your operation *

Gen work report by vegetable class

Submit

clear

output here:

200 Operation success.
1|萝卜|101|胡萝卜素 维生素A
2|香菇|102|纤维素 3|洋葱|101|维生素
200 Operation success.
根茎类|17542|1023.220032
真菌类|90|1.200000

Select your operation *

Set(add/edit) a vegetable info

vegName *

金针菇

classNum *

103

neuInfoStr *

纤维素

Submit

clear

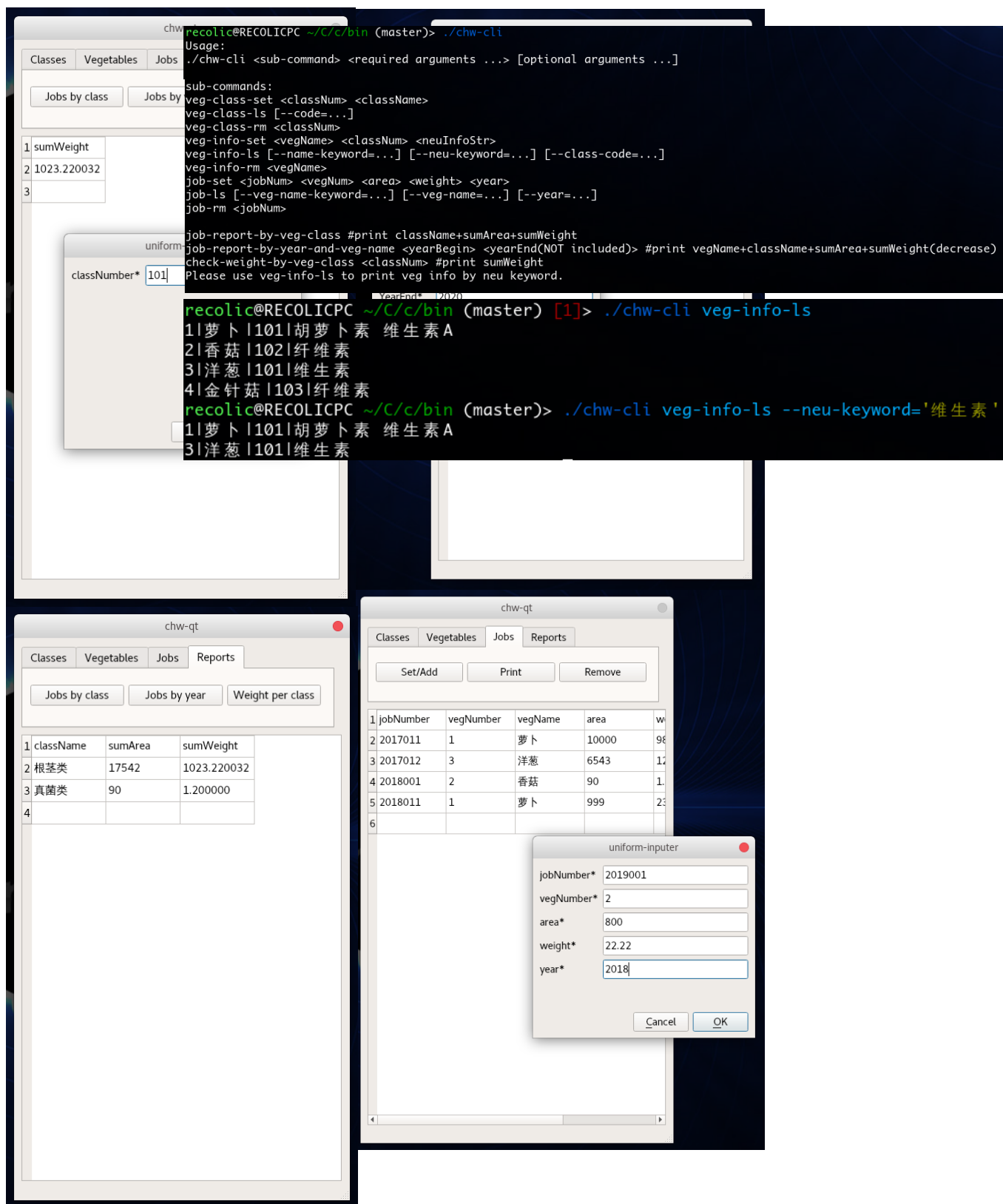
output here:

200 Operation success.

websocket server 与前端的交互:

```
root@kali:~/C/C++/websocket# ./wserver [2017-09-10 20:38:30] [connect] Websocket connection [::ffff:127.0.0.1]:33914 v13 Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.79
CMD ARRIVED>veg-info-ls
[2017-09-10 20:38:38] [frame_header] Dispatching write containing 1 message(s) containing 2 header bytes and 83 payload bytes
[2017-09-10 20:38:38] [frame_header] Header Bytes:
[0] (2) 81 53
[2017-09-10 20:38:38] [frame_payload] Payload Bytes:
[0] (83) [1] 1|萝卜|101|胡萝卜素 维生素A
2|香菇|102|纤维素
3|洋葱|101|维生素
CMD ARRIVED>job-report-by-veg-class
[2017-09-10 20:38:44] [frame_header] Dispatching write containing 1 message(s) containing 2 header bytes and 50 payload bytes
[2017-09-10 20:38:44] [frame_header] Header Bytes:
[0] (2) 81 32
[2017-09-10 20:38:44] [frame_payload] Payload Bytes:
[0] (50) [1] 根茎类|17542|1023.220032
真菌类|90|1.200000
CMD ARRIVED>veg-info-set 金针菇 103 纤维素
[2017-09-10 20:39:27] [frame_header] Dispatching write containing 1 message(s) containing 2 header bytes and 0 payload bytes
[2017-09-10 20:39:27] [frame_header] Header Bytes:
[0] (2) 81 00
[2017-09-10 20:39:27] [frame_payload] Payload Bytes:
[0] (0) [1]
[2017-09-10 20:40:04] [control] Control frame received with opcode 8
[2017-09-10 20:40:04] [frame_header] Dispatching write containing 1 message(s) containing 2 header bytes and 2 payload bytes
[2017-09-10 20:40:04] [frame_header] Header Bytes:
[0] (2) 88 02
[2017-09-10 20:40:04] [frame_payload] Payload Bytes:
[0] (2) [8] 03 E9
[2017-09-10 20:40:04] [error] handle_read_frame error: websocketpp.transport::7 (End of File)
[2017-09-10 20:40:04] [disconnect] Disconnect close local:[1006,End of File] remote:[1001]
```

可以独立运行和测试，便于通过 shell 扩展的后端:



能够直接运行在后端的自动化的压力测试脚本保证了程序的可靠性。

```
>>> src/test/test.sh
```

```
#!/bin/bash
```



```
echo 'Here is test script for recolic/chw, by Recolic Keghart, Mozilla
licensed.'

[[ $1 == '' ]] && echo 'Usage: ./test.sh <bin dir> [disable stdout
redirect]' && exit 1

bindir=$1

[[ $2 != '' ]] && redir="/dev/null" || redir="/dev/fd/1"

function assert () {
    CMD="$*"
    $CMD
    RET_VAL=$?
    if [ $RET_VAL != 0 ]; then
        echo "Assertion failed: $CMD returns $RET_VAL."
        exit $RET_VAL
    fi
}

function randstr4 () {
    < /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-4};echo;
}

function randstr32 () {
    < /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-32};echo;
}

function randint () {
    echo "$(( ( RANDOM % 100 ) + 1 ))"
}
```



```
function test_vcset () {
    ./chw-cli veg-class-set $(randint) $(randstr32)
}

function test_viset () {
    ./chw-cli veg-info-set $(randstr32) $(randint) $(randstr32)
}

function test_jset () {
    ./chw-cli job-set $(randint) $(randint) $(randint) "$(randint).$(randint)" $(randint)
}

function test_vcrm () {
    ./chw-cli veg-class-rm $(randint)
}

function test_virm () {
    ./chw-cli veg-info-rm $(randint)
}

function test_jrm () {
    ./chw-cli job-rm $(randint)
}

function test_ls () {
    echo 'performing ls...'
    echo 'veg-cls->'
    ./chw-cli veg-class-ls
    echo 'veg-inf->'
    ./chw-cli veg-info-ls
}
```



```
    echo 'job->'
    ./chw-cli job-ls
}

assert cd $bindir
assert ./clear_database.sh

echo 'perform 1000 veg class set...'
for i in {1..1000}
do
    echo -ne "\r$i"
    assert test_vcset > $redir
done
echo ' '
assert test_ls

echo 'perform 30 veg class rm, may fail...'
for i in {1..30}
do
    echo -ne "\r$i"
    test_vcrm
done
echo ' '
assert test_ls

echo 'perform 1000 veg class set...'
for i in {1..1000}
do
    echo -ne "\r$i"
    assert test_vcset > $redir
done
```




```
echo ' '

echo 'perform 300 veg info set...'
for i in {1..300}
do
    echo -ne "\r$i"
    test_viset > $redir
done
echo ' '
assert test_ls

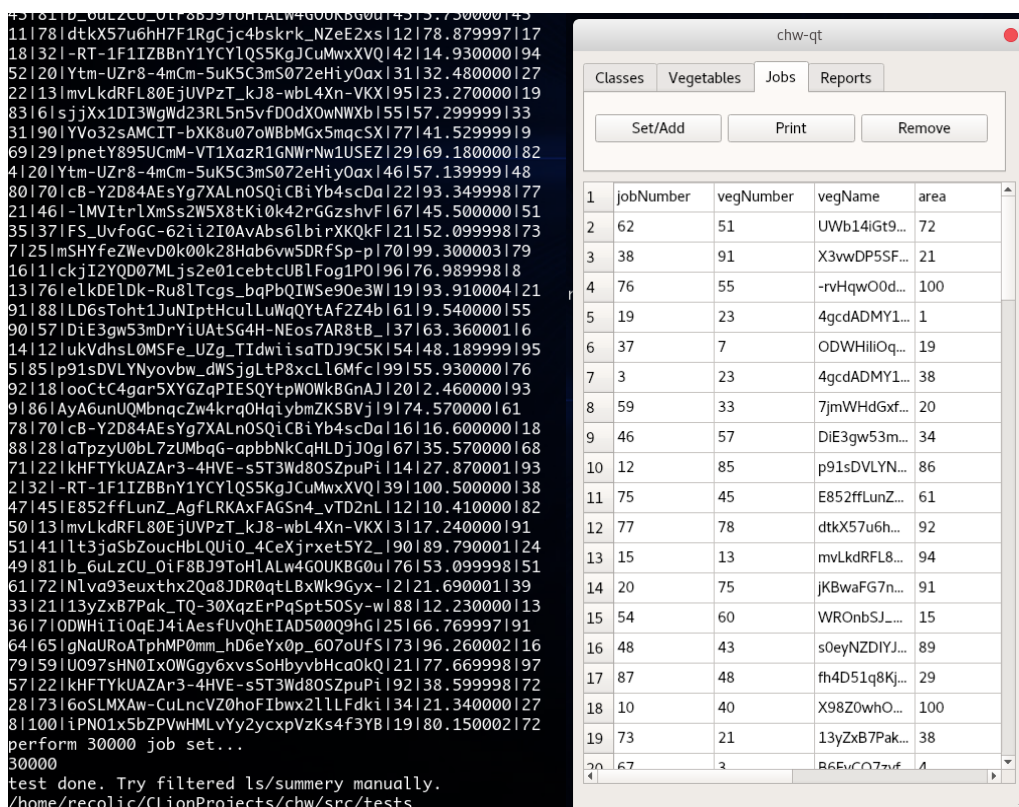
echo 'Because veg name must be given, skipping veg-info-rm-test'

echo 'perform 1000 job set...'
for i in {1..1000}
do
    echo -ne "\r$i"
    test_jset > $redir
done
echo ' '
assert test_ls

echo 'perform 50 job rm, may fail...'
for i in {1..50}
do
    echo -ne "\r$i"
    test_jrm
done
echo ' '
assert test_ls
```

```
echo 'perform 1000 job set...'  
  
for i in {1..1000}  
  
do  
  
    echo -ne "\r$i"  
  
    test_jset > $redir  
  
done  
  
echo ' '  
  
echo 'test done. Try filtered ls/summery manually.'  
  
cd -  
  
<<< test.sh
```

多次随机化的压力测试完全通过。共计 180000 次随机插入删除测试完全通过。
这表明了算法的可靠性。



The terminal window on the left shows the execution of a test script. It prints 'perform 1000 job set...', followed by a loop of 1000 iterations. Each iteration prints a progress indicator (e.g., '\r1', '\r2', etc.) and runs 'test_jset > \$redir'. After the loop, it prints 'test done. Try filtered ls/summery manually.' and 'cd -'. The output of the test script is a large block of text, likely the contents of the file specified by \$redir.

The graphical application window on the right, titled 'chwo-qt', has tabs for 'Classes', 'Vegetables', 'Jobs', and 'Reports'. The 'Jobs' tab is selected, showing a table with columns 'jobNumber', 'vegNumber', 'vegName', and 'area'. The table contains 20 rows of data, including job numbers, vegetable numbers, names, and areas.

jobNumber	vegNumber	vegName	area
62	51	UWb14iGt9...	72
38	91	X3vwDP5SF...	21
76	55	-rvHqW00d...	100
19	23	4gcdADMY1...	1
37	7	ODWHiliOq...	19
3	23	4gcdADMY1...	38
59	33	7jmWHdGxf...	20
46	57	DiE3gw53m...	34
12	85	p91sDVLYN...	86
75	45	E852ffLunZ...	61
77	78	dtkX57u6h...	92
15	13	mvLkdRFL8...	94
20	75	jkBwaFG7n...	91
54	60	WROnbSJ....	15
48	43	s0eyNZDIYJ...	89
87	48	fh4D51q8Kj...	29
10	40	X98Z0whO...	100
73	21	13yZxB7Pak...	38
67	2	B6EvCQ7mf...	4

travis-ci 的实时在线编译测试结果可以反映运行环境的复现较稳定。

六、总结

先放提交记录 <https://github.com/recolic/chw/commits/master>

开发过程 <https://github.com/recolic/chw/graphs/code-frequency>

这是我的第一个以 cmake 来管理编译的 project(以前习惯裸 Makefile)，和第一个试用在线自动测试的 project，因此它在 7 月 5 日完全完成 debug 之后一直在提交新的实验性更改。cmake 和 travis-ci 都是在这个 project 下实验之后才用在生产环境的另一个 project 中的，并表现地令人满意。project 的设计遵从 Mike Gancarz 归纳的 unix 一贯的设计理念，模块之间要求低耦合，前后端严格分离，保证了 project 的 portable 和 extensible，极大的方便了自动测试和生产环境部署(尽管课设并没有为生产环境做过多的考虑，但前后端分离是最基本的)。在开发速度和代码质量之间作出了恰当的权衡，基本遵守 Google C++ 代码风格规范。除了文档要求必须自己实现的数据结构外，最大限度复用已有代码以增加开发速度和程序可靠性与鲁棒性，因此我打包了所需的大多数依赖(除 boost_system, 被静态链接)并用 cmake 依次自动编译，这也是 project 中代码总量 200k 行的原因。其中定义业务逻辑的代码仅 707 行。

发现并最小复现了 libcc1 的 hash 实现的一个 bug，学习了 libut 的纯宏的 template 实现，然后看了一下 glibc 的 list 中的 template 实现，尝试了 C 调用其他编译型语言的 dylib 的 trick(CPython, CGo)和 BashInC 的 trick(最终并没有用到)，这也是 project 中涉及了多种语言代码的原因。



程序设计受到 UNIX Philosophy 的很大影响，下面是它的内容。Unix 非常优美。

Small is beautiful.

Make each program do one thing well.

Build a prototype as soon as possible.

Choose portability over efficiency.

Store data in flat text files.

Use software leverage to your advantage.

Use shell scripts to increase leverage and portability.

Avoid captive user interfaces.

Make every program a filter.

七、参考文献

[0]https://en.wikipedia.org/wiki/Unix_philosophy

[1]Google C++ Style

Guide:<https://google.github.io/styleguide/cppguide.html>

[2]libccl:URL:<https://github.com/Lupus/ccl>

[3]libccl:URL:<http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1625.pdf>

[4]libccl:URL:<http://www.cs.virginia.edu/~lcc-win32/ccl/ccl.html>

[5]libut:URL:<https://troydhanson.github.io/uthash/>

[6]URL:<http://en.cppreference.com/w/>

[7]URL:<http://www.cplusplus.com/reference/>

[8]ISO/IEC 9899:201x:<http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1548.pdf>

[9]ISO/IEC 14882:2011:<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2012/n3337.pdf>

[10]ISO/IEC 14882:2014:<https://www.iso.org/obp/ui/#iso:std:iso-iec:14882:ed-4:v1:en>

[11]ISO/IEC 14882:2017:<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4659.pdf>

[12]IEEE Std 1003.1-2008, 2016

Edition:<http://pubs.opengroup.org/onlinepubs/9699919799/functions/contents.html>



[13]Linux Manual (man 2, man 3)

[14]URL:<https://linux.die.net/>

[15]URL:<http://man7.org/linux/man-pages/>

[16]URL:<https://www.qt.io/>

[17]URL:<https://en.wikipedia.org/wiki/WebSocket>

[18]URL:<https://tools.ietf.org/html/rfc6455>

[19]URL:<https://github.com/zaphoyd/websocketpp>

[20]URL:<https://www.gnu.org/software/bash/manual/bash.html>

[21]URL:https://wiki.gentoo.org/wiki/Main_Page

[22]URL:https://wiki.archlinux.org/index.php/Main_page

[23]URL:<http://libcello.org>