华中科技大学计算机科学与技术学院

数据结构课程设计报告

专　　业：<u>　计算机科学与技术　</u>

班　　级：<u>　　1601　　　　　</u>

学　　号：<u>　U201614531　</u>

姓　　名：<u>　　刘本嵩　　　</u>

成　　绩：________________

指导教师：<u>　周时阳　　</u>

2018 年 3 月 29 日

AVL 树

1.引言

1.1 课题背景与意义

平衡二叉树(AVL 树)是数据结构中的重要知识点,在实际应用中多用于在内存中组织数据,对于平衡二叉树最大的应用就是来查找数据,因为它的查找的时间效率为 O(logn),因此就存在要创建平衡二叉树、对其进行插入、删除这三种基本操作。同时因它在动态查找表中的查找效率非常高,在地理信息处理、医学模型处理以及快速成形等技术中都有广泛应用。

1.2 国内外研究现状

In 1962, Georgy Adelson-Velsky, G.; Evgenii Landis intruduced AVL tree in "An algorithm for the organization of information".
In 1972, Rudolf Bayer invented a data structure that was a special order-4 case of a B-tree. These trees maintained all paths from root to leaf with the same number of nodes, creating perfectly balanced trees. However, they were not binary search trees. Bayer called them a "symmetric binary B-tree" in his paper and later they became popular as 2-3-4 trees or just 2-4 trees.
In a 1978 paper, "A Dichromatic Framework for Balanced Trees", Leonidas J. Guibas and Robert Sedgewick derived the red-black tree from the symmetric binary B-tree. The color "red" was chosen because it was the best-looking color produced by the color laser printer available to the authors while working at Xerox PARC. Another response from Guibas states that it was because of the red and black pens available to them to draw the trees.
In 1993, Arne Andersson introduced the idea of right leaning tree to simplify insert and delete operations.
In 1999, Chris Okasaki showed how to make the insert operation purely functional. Its balance function needed to take care of only 4 unbalanced cases and one default balanced case.
The original algorithm used 8 unbalanced cases, but Cormen et al. (2001) reduced that to 6 unbalanced cases. Sedgewick showed that the insert operation can be implemented in just 46 lines of Java code. In 2008, Sedgewick proposed the left-leaning red–black tree, leveraging Andersson's idea that simplified algorithms. Sedgewick originally allowed nodes whose two children are red making his trees more like 2-3-4 trees but later this restriction

was added making new trees more like 2-3 trees. Sedgewick implemented the insert algorithm in just 33 lines, significantly shortening his original 46 lines of code.

1.3 任务与分析

任务：实现平衡二叉树的创建,并且用二叉树树表示集合,并能对集合内的二叉树进行插入删除以及其他一些简单的操作。
分析：平衡二叉树又称 AVL 树。它或者是一棵空树,或者是具有下列性质的二叉树:它的左子树和右子树都是平衡 二叉树,且二叉树上的所有结点的平衡因子绝对值不超过 1。在平衡二叉树上插入或删除结点后,可能使树失去平衡,因此需要对失去平衡的树进行平衡化调整。平衡一个二叉树方法:分别针对不同失衡结构采用 左转、右转、先左转后右转、先右转后左转 4 种。而其他的操作都是建立在如何保证实现平衡,所以这也是整个课设的难点及重点。

2.系统需求分析与总体设计

2.1 系统需求分析

以二叉链表为储存结构,存储数据集并进行基本运算,能够随机生成大量数据,以json 格式保存和读取数据,模拟实际中微博的人际关系,并实现常用功能。

2.2 系统总体设计

系统使用一个 Terminal 风格交互界面,称为 rfaketerm,在 general_ui.hpp 中实现。fake_terminal::go 会阻塞主线程,接收输入,简单 parse 之后通过 callback 函数进行处理。callback 是一个由 ccgen.py 生成代码的 parser(即 reflection,C++20 标准库提供了原生功能),负责将输入翻译到下一层即 relected_impl。reflection_impl 作为本题要求的接口和容器库普遍承认的接口之间的 wrapper,负责管理数据结构对象 r::set<false ...>或 r::multiset<true ...>，which is inherited from avl::tree<_Ty, multi, is_less, is_equal>。它将请求进一步解释,并与后端数据结构进行交互,获取返回值,被 rfaketerm 打印到 stdout。在程序发生未定义行为时,会通过 std::exception 向自身发送 SIGABRT 信号,这有利于通用调试工具的应用。
为了美观,rfaketerm 默认情况下会把所有异常抓下并打印错误信息到 stdout,但脚本模式(stdin 不是有效的 tty)下会立即退出。UserManual 在 rfaketerm 中使用 help 命令即可获得。为了便于 GUI 下的使用,rfaketerm 启动时会自动模拟执行 help 命令。为了获得一个菜单风格的界面，rfaketerm 可以使用 ENABLE_FAKE_MENU 进行编译，以便在 stdin isatty 时自动进行清屏和 callback(std::vector{“help”s})，获得一个菜单界面。

3.系统详细设计

3.1 数据结构定义

代表每一用户的数据结构 person:

```
struct person : public rlib::noncopyable {
 person() {}
      explicit person(id_manager &idgen) : id(idgen.generate(this)),
pmanager(&idgen) {}
  explicit person(id_manager &idgen, person_id id) : id(id), pmanager(&idgen)
{idgen.assign(id, this);}
 person(person &&another) {
  swap(std::move(another));
 }
 const person &operator=(person &&another) {
  swap(std::move(another));
 }

 bool operator==(const person &another) const {
  return id == another.id;
 }
 bool operator!=(const person &another) const {
  return ! operator==(another);
 }
 bool operator<(const person &another) const {
  return id < another.id;
 }
 bool operator>(const person &another) const {
  return ! ( operator<(another) || operator==(another) );
 }

 void show() const {
  rlib::println("Person", id);
  friends.show("Friends");
  followers.show("Followers");
  followings.show("Followings");
 }
```

```
  void swap(person &another) {
    std::swap(pmanager, another.pmanager);
    std::swap(id, another.id);
    if(id) pmanager->update(id, this);
    if(another.id) pmanager->update(another.id, &another);

    friends.swap(another.friends);
    followers.swap(another.followers);
    followings.swap(another.followings);
  }

  person_id id = 0;
  id_manager *pmanager = nullptr;
  r::set<person_id> friends;
  r::set<person_id> followers;
  r::set<person_id> followings;
};
```

负责分配用户身份证号的 id 管理器:

```
class id_manager : public rlib::noncopyable {
public:
  id_manager() : randGen(std::random_device()()) {}
  person_id generate(const person *pp) {
    while(true) {
      person_id trial = randGen();
      if(id_pool.find(trial) == id_pool.end()) {
        id_pool[trial] = pp;
        return trial;
      }
    }
    throw std::runtime_error("Unknown error.");
  }
  const person *check_id(person_id id) const {
    auto pos = id_pool.find(id);
    if(pos == id_pool.cend())
      throw std::invalid_argument("Invalid id.");
    return pos->second;
  }
  void update(person_id id, const person *pp) {
    if(id_pool.find(id) != id_pool.end())
      id_pool[id] = pp;
    else
      throw std::invalid_argument("Invalid id to update.");
  }
```

```
 void assign(person_id id, const person *pp) {
  if(id_pool.find(id) == id_pool.end())
   id_pool[id] = pp;
  else
   throw std::invalid_argument("Id already used.");
 }
 void clear() {
  id_pool.clear();
 }
private:
 std::unordered_map<person_id, const person *> id_pool;
 std::mt19937_64 randGen;
};
```

树的节点（在namespace avl内，unpublic）：

```
 template <typename data_t, bool multi_tree = false>
 struct node {
  using this_type = node<data_t, multi_tree>;
  std::shared_ptr<this_type> left;
  std::shared_ptr<this_type> right;
  std::weak_ptr<this_type> parent;
  data_t data;
  int32_t factor = 0;
  size_t counter = 1;

#ifdef ENABLE_RAVL_FOREACH_NODE
  void for_each_node(std::function<void(const std::shared_ptr<this_type> &)>
func) {}
#endif
 };
```

avl树的定义（在namespace avl内）：

```
     template <typename data_t, bool multi_tree = false, typename
data_equal_func = std::equal_to<data_t>, typename data_less_func =
std::less<data_t>>
 class tree : public rlib::noncopyable {
 public:
         using this_type = tree<data_t, multi_tree, data_equal_func,
data_less_func>;
  using node_type = node<data_t, multi_tree>;

  tree() = default;
  tree(this_type &&another) noexcept {}
  this_type &operator=(this_type &&another) noexcept {}
  class iterator_utils : public rlib::static_class {
  public:
   template <typename cv_node_type>
         static void next(size_t &curr_cter, std::shared_ptr<cv_node_type>
&target, bool _back_tracing = false) {}

   template <typename cv_node_type>
         static void prev(size_t &curr_cter, std::shared_ptr<cv_node_type>
&target, bool _back_tracing = false) {}
  };

  class iterator {};
  class reverse_iterator {};
  class const_iterator {};
  class const_reverse_iterator {};

  iterator begin() {}
  constexpr iterator end() {}
  reverse_iterator rbegin() {}
  constexpr reverse_iterator rend() {}
  const_iterator cbegin() const {}
  constexpr const_iterator cend() const {}
  const_reverse_iterator crbegin() const {}
  constexpr const_reverse_iterator crend() const {}

  auto insert(data_t &&item, bool no_except = false) {}
  auto insert(const data_t &item, bool no_except = false) {}

  void erase(iterator _iter) {}

  template <bool no_except = false>
  const_iterator find(const data_t &item) const {}
  template <bool no_except = false>
  iterator find(const data_t &item) {}

  size_t count(const data_t &item) const noexcept {}
  bool exist(const data_t &item) const noexcept {}
  size_t height() const noexcept {}
  size_t size() const noexcept {}

  void swap(this_type &&another) noexcept {}
  void swap(this_type &another) noexcept {}
  void clear() noexcept {}
```

```
#ifdef ENABLE_RAVL_DEBUG_DUMP
    void dump() {
      // Error if data_t is not printable.
      rlib::println("root: node", root?root->data:data_t());
      for_each_node([](const std::shared_ptr<node_type> &node){
          rlib::printfln("node {}: left {} right {} parent {} factor {} counter
{}", node->data, (node->left?node->left->data:data_t()), (node->right?node-
>right->data:data_t()),
                (node->parent.expired()?data_t():node->parent.lock()->data), node-
>factor, node->counter);
      });
    }
#endif
  private:
#ifdef ENABLE_RAVL_FOREACH_NODE
    void for_each_node(std::function<void(const std::shared_ptr<node_type> &)>
func) {
      if(root) {
        root->for_each_node(func);
        func(root);
      }
    }
#endif

         std::pair<std::shared_ptr<const node_type>, int/*0 if found, 1 if at
right, -1 if at left, -2 if no root*/> do_find(const data_t &item) const
noexcept {}
        std::pair<std::shared_ptr<node_type>, int> do_find(const data_t &item)
noexcept {
                        std::pair<std::shared_ptr<const node_type>, int> &&res =
static_cast<const this_type *>(this)->do_find(item);
            return std::make_pair(std::const_pointer_cast<node_type>(res.first),
res.second);
    };

    template <bool is_inserting>
    int/*Return: grown_height*/ rebalance(std::shared_ptr<node_type> node, int
my_factor_before_insert/*not used if |node.factor| == 2*/) {}

    //rotate without editing any bal factors.
                            void rotate_to_left(std::shared_ptr<node_type> higher,
std::shared_ptr<node_type> lower) {}
                           void rotate_to_right(std::shared_ptr<node_type> higher,
std::shared_ptr<node_type> lower) {}
                 void   rotate_to_left_2(std::shared_ptr<node_type>   higher,
std::shared_ptr<node_type> lower) {}
                void   rotate_to_right_2(std::shared_ptr<node_type>   higher,
std::shared_ptr<node_type> lower) {}

   std::shared_ptr<node_type> root;
   size_t m_size = 0;
 };
```

avl 树中迭代器 常量迭代器 反向迭代器 反向常量迭代器的定义(均为 Bidirectional Iterator)

（均在 namespace avl::tree 内）:

```
   class iterator {
   public:
    using this_type = iterator;
    iterator() = default;
    iterator(const std::shared_ptr<node_type> &item) : target(item) {}

      this_type &operator=(const std::shared_ptr<node_type> &item) {target =
item;}

     this_type &operator++() {iterator_utils::next(curr_cter, target); return
*this;}
                   this_type   operator++(int)  {this_type   backup   =   *this;
iterator_utils::next(curr_cter, target); return backup;}
     this_type &operator--() {iterator_utils::prev(curr_cter, target); return
*this;}
                   this_type   operator--(int)  {this_type   backup   =   *this;
iterator_utils::prev(curr_cter, target); return backup;}

    data_t &operator*() {return target->data;}
    data_t *operator->() {return &target->data;}

    std::shared_ptr<node_type> to_shared() {return target;}
    operator bool() {return static_cast<bool>(target);}
   private:
    size_t curr_cter = 1;
    std::shared_ptr<node_type> target;
   };
   class reverse_iterator {
   public:
    using this_type = reverse_iterator;
    reverse_iterator() = default;
     reverse_iterator(const std::shared_ptr<node_type> &item) : target(item)
{}

      this_type &operator=(const std::shared_ptr<node_type> &item) {target =
item;}

     this_type &operator++() {iterator_utils::prev(curr_cter, target); return
*this;}
                  this_type  operator++(int)  {this_type  backup  =  *this;
iterator_utils::prev(curr_cter, target); return backup;}
     this_type &operator--() {iterator_utils::next(curr_cter, target); return
*this;}
                  this_type  operator--(int)  {this_type  backup  =  *this;
iterator_utils::next(curr_cter, target); return backup;}

    data_t &operator*() {return target->data;}
    data_t *operator->() {return &target->data;}

    std::shared_ptr<node_type> to_shared() {return target;}
    operator bool() {return static_cast<bool>(target);}
  private:
    size_t curr_cter = 1;
    std::shared_ptr<node_type> target;
  };
  class const_iterator {
  public:
    using this_type = const_iterator;
    const_iterator() = default;
            const_iterator(const  std::shared_ptr<const  node_type>  &item)  :
target(item) {}

         this_type  &operator=(const  std::shared_ptr<const  node_type>  &item)
{target = item;}

     this_type &operator++() {iterator_utils::next(curr_cter, target); return
*this;}
                  this_type  operator++(int)  {this_type  backup  =  *this;
iterator_utils::next(curr_cter, target); return backup;}
     this_type &operator--() {iterator_utils::prev(curr_cter, target); return
*this;}
                  this_type  operator--(int)  {this_type  backup  =  *this;
iterator_utils::prev(curr_cter, target); return backup;}
```

```
    const data_t &operator*() {return target->data;}
    const data_t *operator->() {return &target->data;}

    std::shared_ptr<const node_type> to_shared() {return target;}
    operator bool() {return static_cast<bool>(target);}
  private:
    size_t curr_cter = 1;
    std::shared_ptr<const node_type> target;
  };
  class const_reverse_iterator {
  public:
    using this_type = const_reverse_iterator;
    const_reverse_iterator() = default;
      const_reverse_iterator(const std::shared_ptr<const node_type> &item) :
target(item) {}

         this_type &operator=(const std::shared_ptr<const node_type> &item)
{target = item;}

     this_type &operator++() {iterator_utils::prev(curr_cter, target); return
*this;}
                  this_type operator++(int) {this_type backup = *this;
iterator_utils::prev(curr_cter, target); return backup;}
     this_type &operator--() {iterator_utils::next(curr_cter, target); return
*this;}
                  this_type operator--(int) {this_type backup = *this;
iterator_utils::next(curr_cter, target); return backup;}

    const data_t &operator*() {return target->data;}
    const data_t *operator->() {return &target->data;}

    std::shared_ptr<const node_type> to_shared() {return target;}
    operator bool() {return static_cast<bool>(target);}
  private:
    size_t curr_cter = 1;
    std::shared_ptr<const node_type> target;
  };
```

被用作依赖的C++库均在提交的目录中以 git submodule 的形式列出，其中包括
- 自己实现的现代 C++ util 库: rlib https://github.com/recolic/rlib.git
- GNU readline 的 C++ wrapper: cpp-readline https://github.com/Svalorzen/cpp-readline.git
- 优秀的 面向对象的 现代 C++的 json 库: json https://github.com/nlohmann/json.git
- 经典的 doubelc 提供的 GC 库: https://github.com/doublec/gc.git

3.2 算法设计

3.3.1 左旋

```
传入参数为 较高的节点 较低的节点
   较高的节点的右孩子 设置为 较低的节点的左孩子;
   如果 较低的节点的左孩子 较低的节点的左孩子的父节点 设置为 较高的节点;

   较低的节点的左孩子 设置为 较高的节点;
   较低的节点的父节点 设置为 较高的节点的父节点;
   如果 较高的节点的父节点不为空 {
      把 父节点 设置为 较高的节点的父节点;
      如果 较高的节点 是 父节点的右孩子 父节点的右孩子 设置为 较低的节点;
      否则 父节点的左孩子 设置为 较低的节点;
   }
   否则
      根结点 设置为 较低的节点;
   较高的节点的父节点 设置为 较低的节点;
```

3.3.2 右旋

```
传入参数为 较高的节点 较低的节点
   较高的节点的左孩子 设置为 较低的节点的右孩子;
   如果 较低的节点的右孩子 较低的节点的右孩子的父节点 设置为 较高的节点;

   较低的节点的右孩子 设置为 较高的节点;
   较低的节点的父节点 设置为 较高的节点的父节点;
   如果 较高的节点的父节点不为空 {
      把 父节点 设置为 较高的节点的父节点;
      如果 较高的节点 是 父节点的右孩子 父节点的右孩子 设置为 较低的节点;
      否则 父节点的左孩子 设置为 较低的节点;
```

```
        }
        否则
           根结点 设置为 较低的节点;
        较高的节点的父节点 设置为 较低的节点;
```

3.3.3 左右旋

```
传入参数为 较高的节点 较低的节点
        把 另一个较低的节点 设置为 较低的节点的左孩子;
        较高的节点的右孩子 设置为 另一个较低的节点的左孩子;
        如果 另一个较低的节点的左孩子 另一个较低的节点的左孩子的父节点 设置为 较高的节点;
        较低的节点的左孩子 设置为 另一个较低的节点的右孩子;
        如果 另一个较低的节点的右孩子 另一个较低的节点的右孩子的父节点 设置为 较低的节点;

        另一个较低的节点的父节点 设置为 较高的节点的父节点;
        如果 较高的节点的父节点不为空 {
           把 父节点 设置为 较高的节点的父节点;
           如果 较高的节点 是 父节点的右孩子 父节点的右孩子 设置为 另一个较低的节点;
           否则 父节点的左孩子 设置为 另一个较低的节点;
        }
        否则
           根结点 设置为 另一个较低的节点;

        另一个较低的节点的右孩子 设置为 较低的节点;
        另一个较低的节点的左孩子 设置为 较高的节点;
        较低的节点的父节点 设置为 另一个较低的节点;
        较高的节点的父节点 设置为 另一个较低的节点;
```

3.3.4 右左旋

```
传入参数为 较高的节点 较低的节点
        把 另一个较低的节点 设置为 较低的节点的右孩子;
        较高的节点的左孩子 设置为 另一个较低的节点的右孩子;
```

```
 如果 另一个较低的节点的右孩子 另一个较低的节点的右孩子的父节点设置为 较高的节点;
较低的节点的右孩子 设置为 另一个较低的节点的左孩子;
 如果 另一个较低的节点的左孩子 另一个较低的节点的左孩子的父节点设置为 较低的节点;

另一个较低的节点的父节点 设置为 较高的节点的父节点;
如果 较高的节点的父节点不为空 {
   把 父节点 设置为 较高的节点的父节点;
   如果 较高的节点 是 父节点的右孩子 父节点的右孩子 设置为 另一个较低的节点;
   否则 父节点的左孩子 设置为 另一个较低的节点;
}
否则
   根结点 设置为 另一个较低的节点;

另一个较低的节点的左孩子 设置为 较低的节点;
另一个较低的节点的右孩子 设置为 较高的节点;
较低的节点的父节点 设置为 另一个较低的节点;
较高的节点的父节点 设置为 另一个较低的节点;
```

3.3.5 递归获得高度

```
将 当前位置的高度 设置为 0;
将 当前位置 设置为 根结点;
条件循环首(当前位置不是空) {
   增加当前位置的高度;
   如果(当前位置的平衡因子 等于 1)
      当前位置 设置为 当前位置的右孩子;
   否则
      当前位置 设置为 当前位置的左孩子;
}
返回 当前位置的高度;
```

3.3.6 插入

调用 find，获得一个 iterator；

插入一个节点；
调用 rebalance<true>；
复杂度为 O(logn).

3.3.7 删除

调用 find，获得一个 iterator；
删除这个 iterator 指向的节点。如果 iterator 不是叶子，将它与一个合理的节点进行交换后删除；
调用 rebalance<false>();
复杂度为 O(logn).

3.3.8 重新平衡

```
参数： 结点  插入之前的平衡因子
        将 树在过程中长高了的高度 设置为 取绝对值(结点的平衡因子) - 取绝对值(插入之前的平衡因子);
        如果((是否正在插入 并且 树在过程中长高了的高度 小于 0) || (不满足 是否正在插入 并且 树在过程中长高了的高度 大于 0)) 树在过程中长高了的高度 设置为 0;

        如果(取绝对值(结点的平衡因子) 小于 2) {
           如果(树在过程中长高了的高度 不等于 0 并且 不满足 结点的父节点为空) {
              将 父节点 设置为 结点的父节点;
              将 父节点的平衡因子的备份 设置为 父节点的平衡因子;
               父节点的平衡因子 增加 树在过程中长高了的高度 乘以 (结点 等于 父节点的右孩子 ? 1 : -1);
              重新平衡<是否正在插入>(父节点, 父节点的平衡因子的备份);
           }
           返回 树在过程中长高了的高度;
        }
        否则 如果 (取绝对值(结点的平衡因子) 等于 2) {
           如果(结点的平衡因子 大于 0) {
              将 孩子 设置为 结点的右孩子;
              将 w 设置为 孩子的平衡因子;
              如果(w 等于 -1) {
```

```
            将 w2 设置为 孩子的左孩子的平衡因子;
            孩子的左孩子的平衡因子 设置为 w2 等于 1 ? 1 : 0;
            孩子的平衡因子 设置为 w2 等于 -1 ? 1 : 0;
            结点的平衡因子 设置为 w2 等于 1 ? -1 : 0;
            左右旋(结点,孩子);
            返回 0;
         }
         左旋(结点, 孩子);
         结点的平衡因子 设置为 1-w;
         孩子的平衡因子 设置为 w-1;

         如果(取绝对值(结点的平衡因子) 等于 2) {
            重新平衡<是否正在插入>(结点, 2);
         }
         返回 0;
      }
      否则 {
         将 孩子 设置为 结点的左孩子;
         将 w 设置为 孩子的平衡因子;
         如果(w 等于 1) {
            将 w2 设置为 孩子的右孩子的平衡因子;
            孩子的右孩子的平衡因子 设置为 w2 等于 -1 ? -1 : 0;
            孩子的平衡因子 设置为 w2 等于 1 ? -1 : 0;
            结点的平衡因子 设置为 w2 等于 -1 ? -1 : 0;
            右左旋(结点,孩子);
            返回 0;
         }
         结点的平衡因子 设置为 -1-w;
         孩子的平衡因子 设置为 1+w;
         右旋(结点, 孩子);

         如果(取绝对值(结点的平衡因子) 等于 2)
            重新平衡<是否正在插入>(结点, -2);
         返回 0;
      }
   }
   否则
      报错
}
```

3.3.9 获得并集

创建一个新的集合；
迭代地插入两个需要并的集合的所有元素，使用忽略报错模式。
复杂度为 O((m+n)logmn)

3.3.10 获得交集

创建一个新的集合；
迭代地插入第一个集合中的所有元素；
不断从新集合删除第二个集合中的元素，使用忽略报错模式。
复杂度为 O(nlogmn).

3.3.11 获得差集

创建一个新的集合；
迭代地插入第一个集合中的所有元素；
迭代地试图从新集合中删除第二个集合中的元素。
复杂度为 O(MAX(nlogn,mlogm))

4.系统实现与测试

4.1 系统实现

程序实现了对用户的管理，共同关注、二度好友及共同好友功能，并实现了数据文件的读写。
我使用的编译环境：Linux 4.15.10-1(ARCH) gcc 8.0.0 g++ 8.0.0 cmake 3.10.3 GNU make 4.2.1 ld 2.29.1 ar 2.29.1
编译环境的要求：Linux x8664 > 2.6, cc 支持 c11， cxx 支持 c++17，cmake>3.2，合适版本的 gnumake ld ar。
为了符合要求，我最终使用的源码和我的开源版本不同，因此没有使用一键编译工具。请使用以下方法进行编译：

```
tar -xvJf hust-ds-homework-all.tar.xz && cd hust-ds-homework-all/
cd hust-ds-homework-avl/ && make bin # make bin-win if building for win32
# use bin/hust-ds for avl/set testing.
cd ..
cd hust-ds-homework-final/ && make bin # make bin-win if building for win32
# use bin/hust-ds for app testing.
cd ..
```

所包含的函数及对应功能为：
main 函数，作为入口点 被__libc_start 调用，调用 rfaketerm::go(parser::parse)来启动程序。
所包含的类接口有(在上面数据结构已经列出的类不再重复列举)：

```
__refl_class__ class reflected_impl : public rlib::nonmovable {
public:
 using stref = const string &;
 using id_t = person_id;

 __refl_func__ void save(stref fname) {}
 __refl_func__ void load(stref fname) {}
 __refl_func__ void clear() {}

 __refl_func__ string new_person() {}
 __refl_func__ string new_person_at(stref id_to_assign) {}
 __refl_func__ string rm_person(stref person_id) {}
 __refl_func__ void get(stref person_id) {}
```

```
  __refl_func__ string add(stref what, stref person_id, stref to_add_id) {}

  __refl_func__ string rm(stref what, stref person_id, stref to_remove_id) {}

  __refl_func__ void common(stref what, stref person1_id, stref person2_id)
{}

  __refl_func__ void indirect(stref what, stref person_id) {}

private:
  r::set<person> buf;
  id_manager id_manager1;
};


class fake_terminal {
#ifdef FORCE_MULTICOLOR_RTERM
  static constexpr bool force_multicolor = true;
#else
  static constexpr bool force_multicolor = false;
#endif
  static void _clear() {rlib::printf("\033[H\033[J");}
public:
  using callback_t = std::function<void (std::vector<std::string>)>;
  static void sigint_handler(int) {}

  [[noreturn]] static void go(const callback_t &callback) {}
private:
  static bool is_scripting() {}
  static string prompt() {}
  static string welcome() {}
  static void showError(const std::string &msg) {}
};


class parser
{
private:
  static void help_msg();
public:
  static void parse(const std::vector<std::string> &to_parse);
};
```

```
namespace rlib {
 namespace _noncp_ {
  class noncopyable
  {
  public:
   noncopyable() = default;
   ~noncopyable() = default;
   noncopyable(const noncopyable &) = delete;
   noncopyable &operator=(const noncopyable &) = delete;
  };
 }
 typedef _noncp_::noncopyable noncopyable;
}

namespace rlib {
 namespace _nonmv_ {
  class nonmovable : private noncopyable
  {
  public:
   nonmovable() = default;
   ~nonmovable() = default;
   nonmovable(const nonmovable &&) = delete;
   nonmovable &operator=(const nonmovable &&) = delete;
  };
 }
 typedef _nonmv_::nonmovable nonmovable;
}

namespace rlib {
 namespace _nonconstructible_ {
  class nonconstructible : private rlib::nonmovable
  {
  public:
   nonconstructible() = delete;
   ~nonconstructible() = delete;
  };
 }
 typedef _nonconstructible_::nonconstructible nonconstructible;
 typedef nonconstructible static_class;
}
```

4.2 系统测试

4.2.1 测试方案

此程序采用“使用小数据测试正确性，使用大数据测试执行效率”的测试方法。

4.2.2 小数据集的测试

先完成编译。
运行 hust-ds-homework-avl/bin/hust-ds，依次进行数据结构测试（你当然可以使用测试脚本自动测试，这也是推荐的做法）

```
#!bin/hust-ds


# avl
# use dump when you want to see the tree, use ls when you want to see the set.
rm 7
insert 3
insert 5
insert 7
ls

rm 3
insert 6
insert 2
insert 1
insert 9
insert 10
insert -7
rm 1
rm 9
rm 10
rm 6
ls
```

```
clear

# set
insert 7
insert 23
insert 4
insert 24
insert 6
insert 6
insert 1
insert 66
insert 4
insert 8
insert -3
insert 11
ls

CreateAVLSet
Select 1
insert 4
insert 7
insert 8
insert 77
insert -2
insert 23
insert 4
ls

Select 0
merge 1

sub 1

common 1

contains 1
contains 0

equal 1
equal 0

exit
```

运行 hust-ds-homework-final/bin/hust-ds，依次进行 app 功能测试（你当然可以使

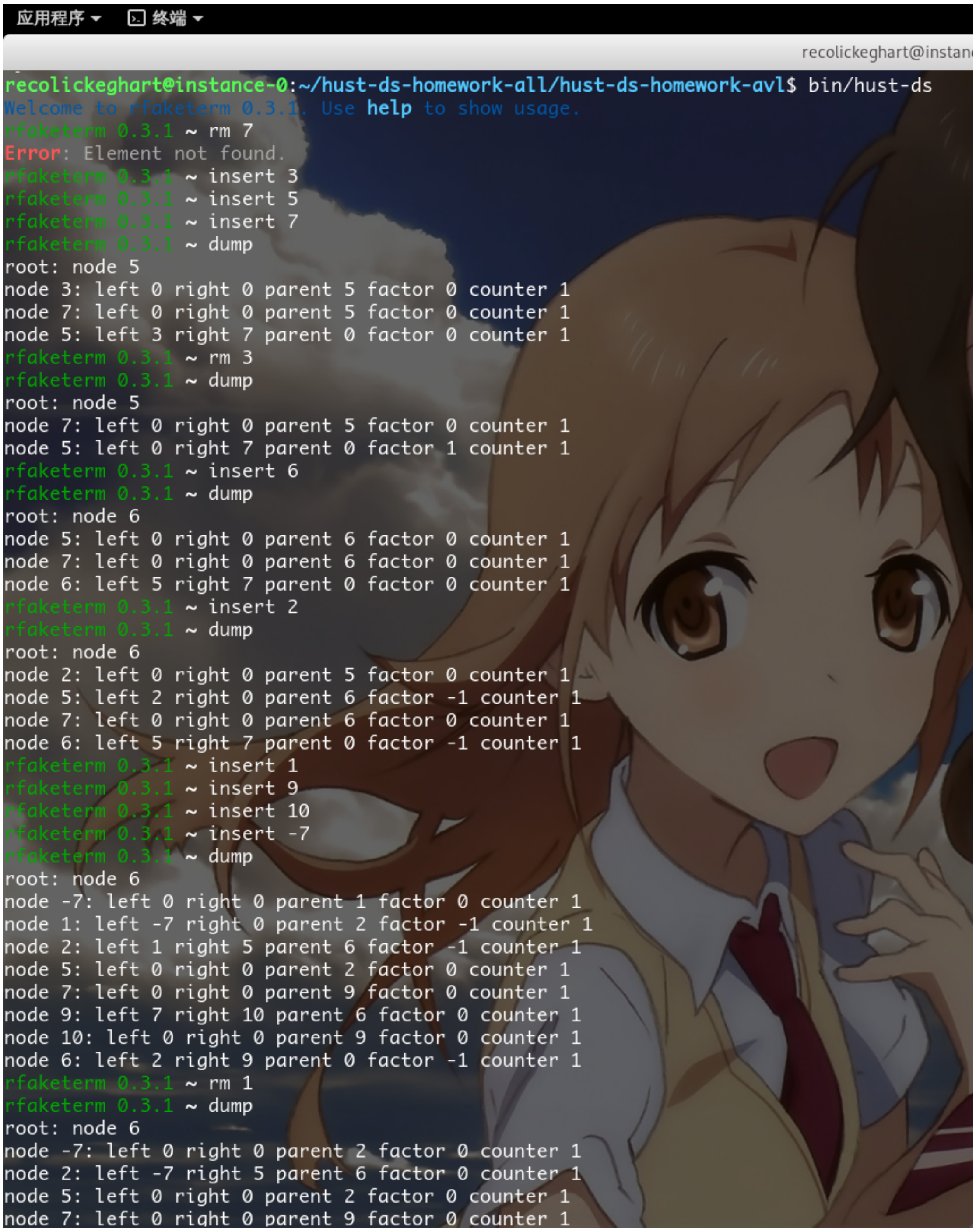
用测试脚本自动测试，这也是推荐的做法）

```
#!bin/hust-ds

new_person_at 1
new_person_at 2
new_person_at 3
new_person_at 4
new_person_at 5
new_person_at 6
new_person_at 7
ls

rm_person 7
ls

add friend 1 2
add friend 1 3
add friend 1 4
add friend 1 6
add friend 2 4
add friend 2 6
add friend 4 6
add friend 5 6
rm friend 1 6
rm friend 3 4

get 1

common friend 1 2

add following 1 2
add following 3 2
common following 1 3

indirect friend 1

# Now save/load data!
save test/naive-1.json

# this load needs about 5.1GiB memory and about 30-40s depending on your harddisk
price. (load took 15.2 sec, but data generate script took 1min13.3sec)
# I re-generated the 1mtest.json so it looked different from the one shown that day.
load test/1mtest.json
```

```
get 1

get 3

indirect 1

# Do anything you want!

exit
```

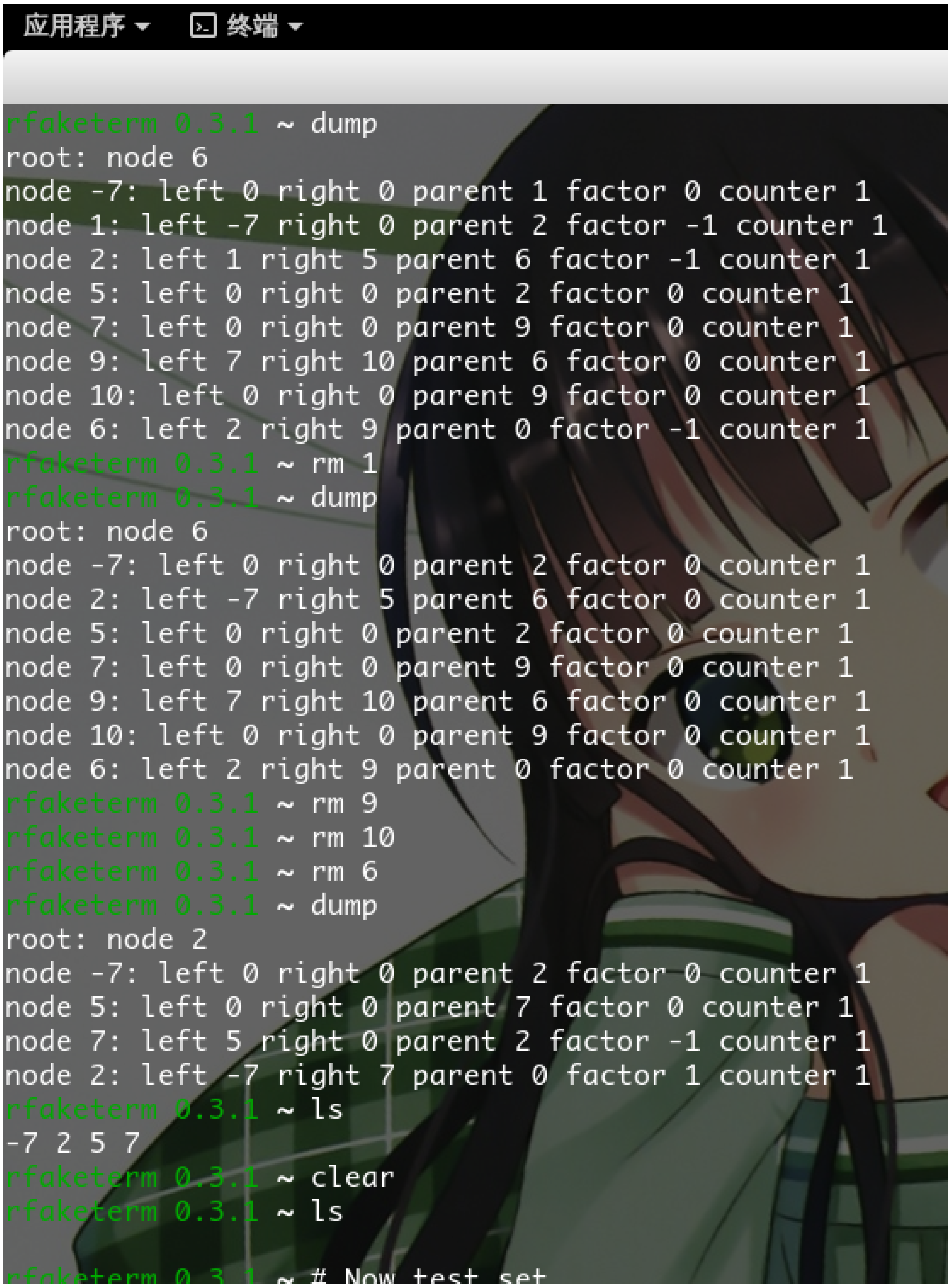
下面为测试时的截图。为了使截图清晰明了(而不是总体积 1GiB)，我关闭了菜单功能。上交的源码默认打开了菜单功能。

1.AVL 基本测试



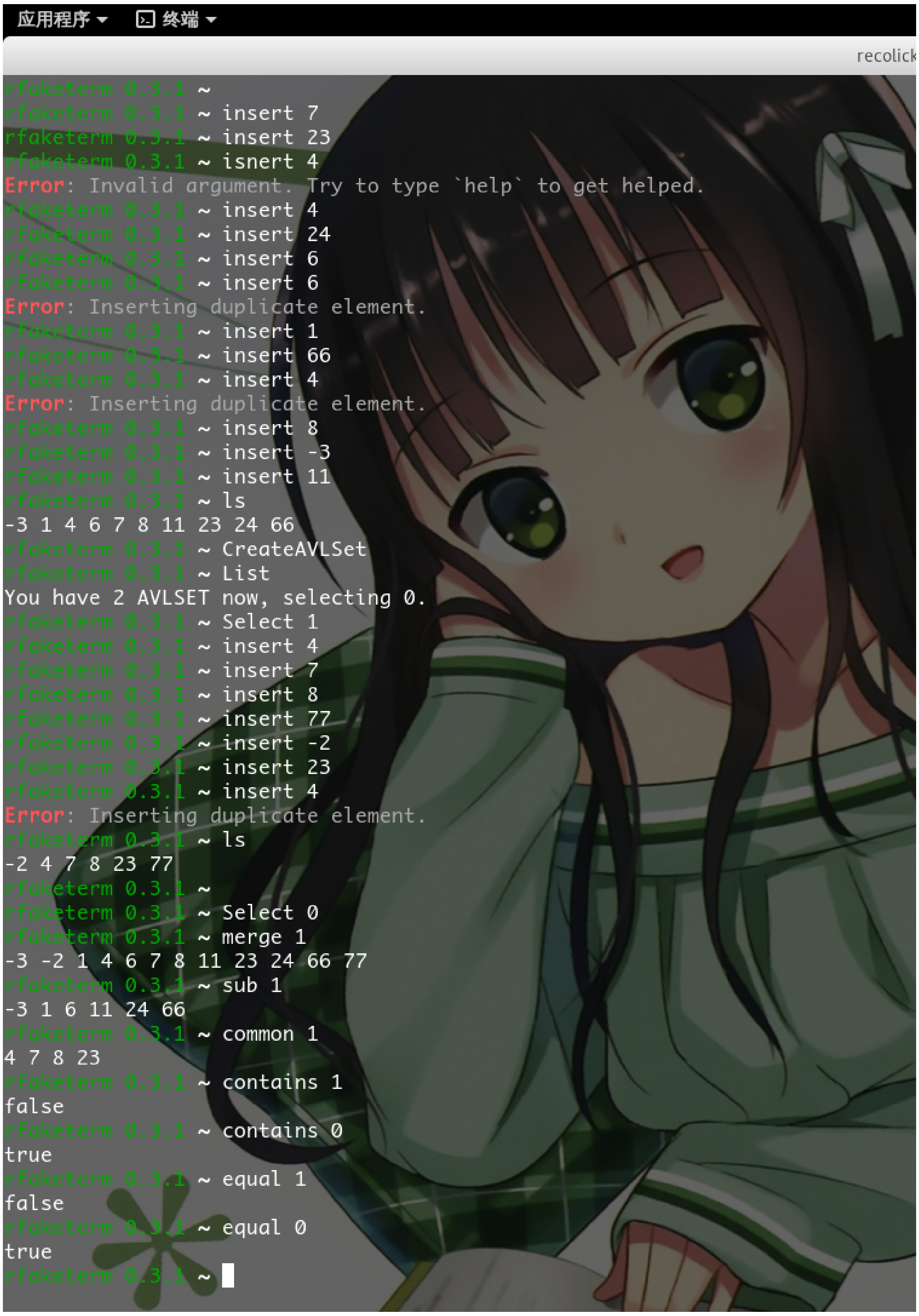
```
recolickeghart@instance-0:~/hust-ds-homework-all/hust-ds-homework-avl$ bin/hust-ds
Welcome to rfaketerm 0.3.1. Use help to show usage.
rfaketerm 0.3.1 ~ rm 7
Error: Element not found.
rfaketerm 0.3.1 ~ insert 3
rfaketerm 0.3.1 ~ insert 5
rfaketerm 0.3.1 ~ insert 7
rfaketerm 0.3.1 ~ dump
root: node 5
node 3: left 0 right 0 parent 5 factor 0 counter 1
node 7: left 0 right 0 parent 5 factor 0 counter 1
node 5: left 3 right 7 parent 0 factor 0 counter 1
rfaketerm 0.3.1 ~ rm 3
rfaketerm 0.3.1 ~ dump
root: node 5
node 7: left 0 right 0 parent 5 factor 0 counter 1
node 5: left 0 right 7 parent 0 factor 1 counter 1
rfaketerm 0.3.1 ~ insert 6
rfaketerm 0.3.1 ~ dump
root: node 6
node 5: left 0 right 0 parent 6 factor 0 counter 1
node 7: left 0 right 0 parent 6 factor 0 counter 1
node 6: left 5 right 7 parent 0 factor 0 counter 1
rfaketerm 0.3.1 ~ insert 2
rfaketerm 0.3.1 ~ dump
root: node 6
node 2: left 0 right 0 parent 5 factor 0 counter 1
node 5: left 2 right 0 parent 6 factor -1 counter 1
node 7: left 0 right 0 parent 6 factor 0 counter 1
node 6: left 5 right 7 parent 0 factor -1 counter 1
rfaketerm 0.3.1 ~ insert 1
rfaketerm 0.3.1 ~ insert 9
rfaketerm 0.3.1 ~ insert 10
rfaketerm 0.3.1 ~ insert -7
rfaketerm 0.3.1 ~ dump
root: node 6
node -7: left 0 right 0 parent 1 factor 0 counter 1
node 1: left -7 right 0 parent 2 factor -1 counter 1
node 2: left 1 right 5 parent 6 factor -1 counter 1
node 5: left 0 right 0 parent 2 factor 0 counter 1
node 7: left 0 right 0 parent 9 factor 0 counter 1
node 9: left 7 right 10 parent 6 factor 0 counter 1
node 10: left 0 right 0 parent 9 factor 0 counter 1
node 6: left 2 right 9 parent 0 factor -1 counter 1
rfaketerm 0.3.1 ~ rm 1
rfaketerm 0.3.1 ~ dump
root: node 6
node -7: left 0 right 0 parent 2 factor 0 counter 1
node 2: left -7 right 5 parent 6 factor 0 counter 1
node 5: left 0 right 0 parent 2 factor 0 counter 1
node 7: left 0 right 0 parent 9 factor 0 counter 1
```

2. AVL 基本测试



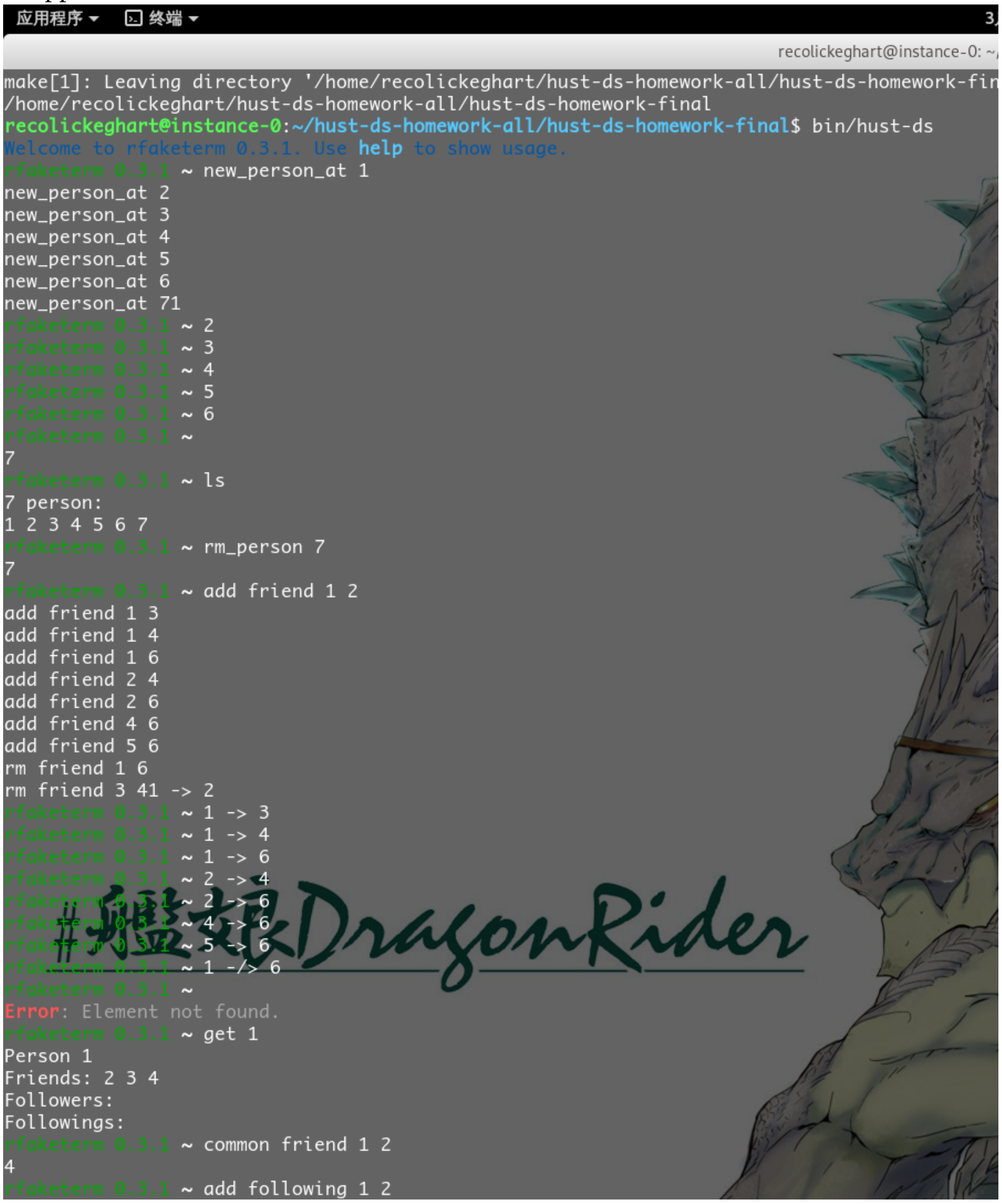
```
rfaketerm 0.3.1 ~ dump
root: node 6
node -7: left 0 right 0 parent 1 factor 0 counter 1
node 1: left -7 right 0 parent 2 factor -1 counter 1
node 2: left 1 right 5 parent 6 factor -1 counter 1
node 5: left 0 right 0 parent 2 factor 0 counter 1
node 7: left 0 right 0 parent 9 factor 0 counter 1
node 9: left 7 right 10 parent 6 factor 0 counter 1
node 10: left 0 right 0 parent 9 factor 0 counter 1
node 6: left 2 right 9 parent 0 factor -1 counter 1
rfaketerm 0.3.1 ~ rm 1
rfaketerm 0.3.1 ~ dump
root: node 6
node -7: left 0 right 0 parent 2 factor 0 counter 1
node 2: left -7 right 5 parent 6 factor 0 counter 1
node 5: left 0 right 0 parent 2 factor 0 counter 1
node 7: left 0 right 0 parent 9 factor 0 counter 1
node 9: left 7 right 10 parent 6 factor 0 counter 1
node 10: left 0 right 0 parent 9 factor 0 counter 1
node 6: left 2 right 9 parent 0 factor 0 counter 1
rfaketerm 0.3.1 ~ rm 9
rfaketerm 0.3.1 ~ rm 10
rfaketerm 0.3.1 ~ rm 6
rfaketerm 0.3.1 ~ dump
root: node 2
node -7: left 0 right 0 parent 2 factor 0 counter 1
node 5: left 0 right 0 parent 7 factor 0 counter 1
node 7: left 5 right 0 parent 2 factor -1 counter 1
node 2: left -7 right 7 parent 0 factor 1 counter 1
rfaketerm 0.3.1 ~ ls
-7 2 5 7
rfaketerm 0.3.1 ~ clear
rfaketerm 0.3.1 ~ ls

rfaketerm 0.3.1 ~ # Now test set
```

3. set 基本测试



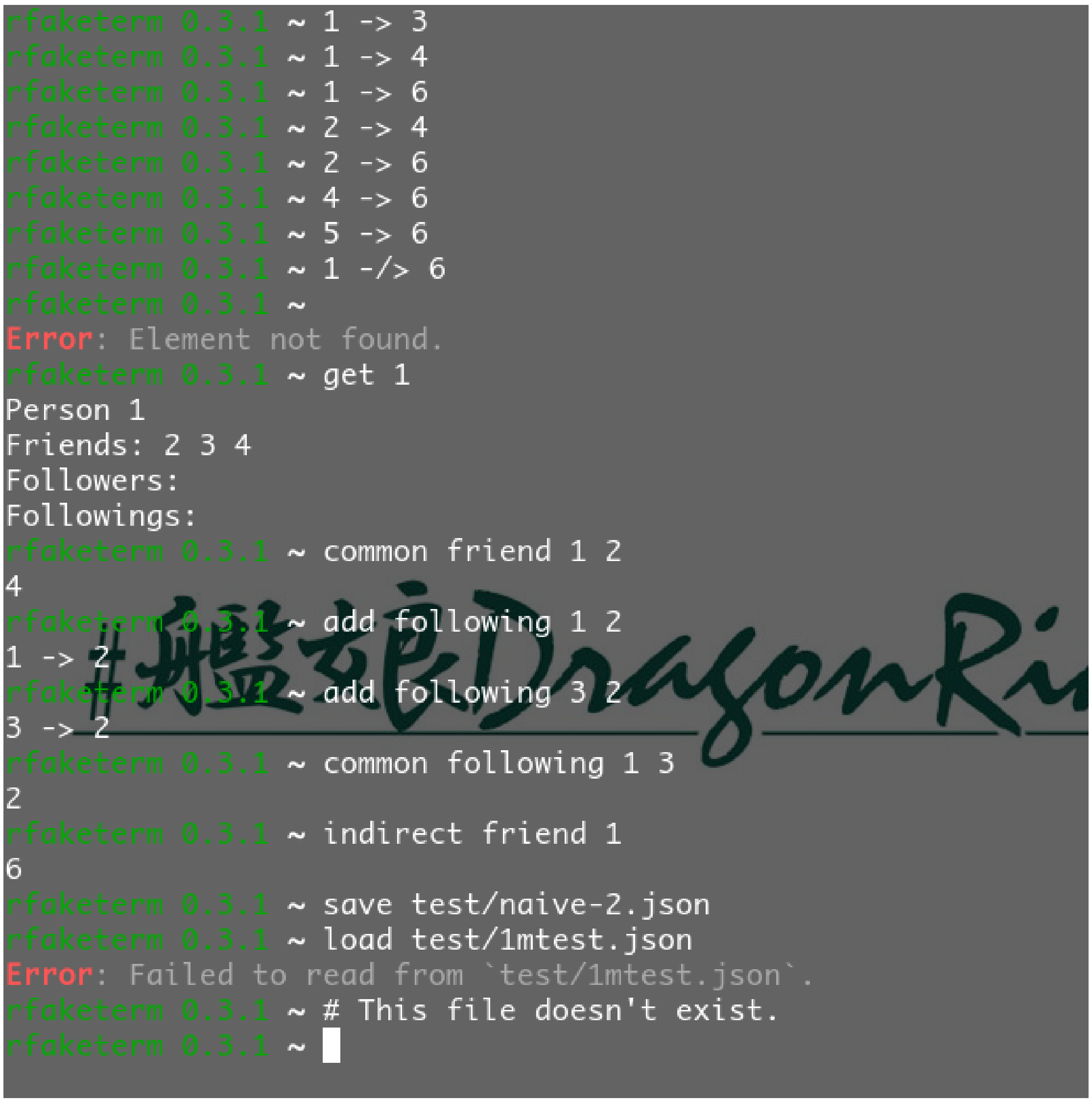
```
rfaketerm 0.3.1 ~
rfaketerm 0.3.1 ~ insert 7
rfaketerm 0.3.1 ~ insert 23
rfaketerm 0.3.1 ~ isnert 4
Error: Invalid argument. Try to type `help` to get helped.
rfaketerm 0.3.1 ~ insert 4
rfaketerm 0.3.1 ~ insert 24
rfaketerm 0.3.1 ~ insert 6
rfaketerm 0.3.1 ~ insert 6
Error: Inserting duplicate element.
rfaketerm 0.3.1 ~ insert 1
rfaketerm 0.3.1 ~ insert 66
rfaketerm 0.3.1 ~ insert 4
Error: Inserting duplicate element.
rfaketerm 0.3.1 ~ insert 8
rfaketerm 0.3.1 ~ insert -3
rfaketerm 0.3.1 ~ insert 11
rfaketerm 0.3.1 ~ ls
-3 1 4 6 7 8 11 23 24 66
rfaketerm 0.3.1 ~ CreateAVLSet
rfaketerm 0.3.1 ~ List
You have 2 AVLSET now, selecting 0.
rfaketerm 0.3.1 ~ Select 1
rfaketerm 0.3.1 ~ insert 4
rfaketerm 0.3.1 ~ insert 7
rfaketerm 0.3.1 ~ insert 8
rfaketerm 0.3.1 ~ insert 77
rfaketerm 0.3.1 ~ insert -2
rfaketerm 0.3.1 ~ insert 23
rfaketerm 0.3.1 ~ insert 4
Error: Inserting duplicate element.
rfaketerm 0.3.1 ~ ls
-2 4 7 8 23 77
rfaketerm 0.3.1 ~
rfaketerm 0.3.1 ~ Select 0
rfaketerm 0.3.1 ~ merge 1
-3 -2 1 4 6 7 8 11 23 24 66 77
rfaketerm 0.3.1 ~ sub 1
-3 1 6 11 24 66
rfaketerm 0.3.1 ~ common 1
4 7 8 23
rfaketerm 0.3.1 ~ contains 1
false
rfaketerm 0.3.1 ~ contains 0
true
rfaketerm 0.3.1 ~ equal 1
false
rfaketerm 0.3.1 ~ equal 0
true
rfaketerm 0.3.1 ~
```

4. app 基本测试



```
make[1]: Leaving directory '/home/recolickeghart/hust-ds-homework-all/hust-ds-homework-fin
/home/recolickeghart/hust-ds-homework-all/hust-ds-homework-final
recolickeghart@instance-0:~/hust-ds-homework-all/hust-ds-homework-final$ bin/hust-ds
Welcome to rfaketerm 0.3.1. Use help to show usage.
rfaketerm 0.3.1 ~ new_person_at 1
new_person_at 2
new_person_at 3
new_person_at 4
new_person_at 5
new_person_at 6
new_person_at 71
rfaketerm 0.3.1 ~ 2
rfaketerm 0.3.1 ~ 3
rfaketerm 0.3.1 ~ 4
rfaketerm 0.3.1 ~ 5
rfaketerm 0.3.1 ~ 6
rfaketerm 0.3.1 ~
7
rfaketerm 0.3.1 ~ ls
7 person:
1 2 3 4 5 6 7
rfaketerm 0.3.1 ~ rm_person 7
7
rfaketerm 0.3.1 ~ add friend 1 2
add friend 1 3
add friend 1 4
add friend 1 6
add friend 2 4
add friend 2 6
add friend 4 6
add friend 5 6
rm friend 1 6
rm friend 3 41 -> 2
rfaketerm 0.3.1 ~ 1 -> 3
rfaketerm 0.3.1 ~ 1 -> 4
rfaketerm 0.3.1 ~ 1 -> 6
rfaketerm 0.3.1 ~ 2 -> 4
rfaketerm 0.3.1 ~ 2 -> 6
rfaketerm 0.3.1 ~ 4 -> 6
rfaketerm 0.3.1 ~ 5 -> 6
rfaketerm 0.3.1 ~ 1 -/> 6
rfaketerm 0.3.1 ~
Error: Element not found.
rfaketerm 0.3.1 ~ get 1
Person 1
Friends: 2 3 4
Followers:
Followings:
rfaketerm 0.3.1 ~ common friend 1 2
4
rfaketerm 0.3.1 ~ add following 1 2
```

5. APP 基本测试



```
rfaketerm 0.3.1 ~ 1 -> 3
rfaketerm 0.3.1 ~ 1 -> 4
rfaketerm 0.3.1 ~ 1 -> 6
rfaketerm 0.3.1 ~ 2 -> 4
rfaketerm 0.3.1 ~ 2 -> 6
rfaketerm 0.3.1 ~ 4 -> 6
rfaketerm 0.3.1 ~ 5 -> 6
rfaketerm 0.3.1 ~ 1 -/> 6
rfaketerm 0.3.1 ~
Error: Element not found.
rfaketerm 0.3.1 ~ get 1
Person 1
Friends: 2 3 4
Followers:
Followings:
rfaketerm 0.3.1 ~ common friend 1 2
4
rfaketerm 0.3.1 ~ add following 1 2
1 -> 2
rfaketerm 0.3.1 ~ add following 3 2
3 -> 2
rfaketerm 0.3.1 ~ common following 1 3
2
rfaketerm 0.3.1 ~ indirect friend 1
6
rfaketerm 0.3.1 ~ save test/naive-2.json
rfaketerm 0.3.1 ~ load test/1mtest.json
Error: Failed to read from `test/1mtest.json`.
rfaketerm 0.3.1 ~ # This file doesn't exist.
rfaketerm 0.3.1 ~
```

4.2.3 大数据集的测试

在 Intel(R) Core(TM) i5-4200H CPU @ 2.80GHz，可用内存 7429MiB DDR3，用 CPU 生成数据进行 AVL 测试，从 tmpfs 类型的磁盘读未结构化的数据进行应用性能测试，结果如下。

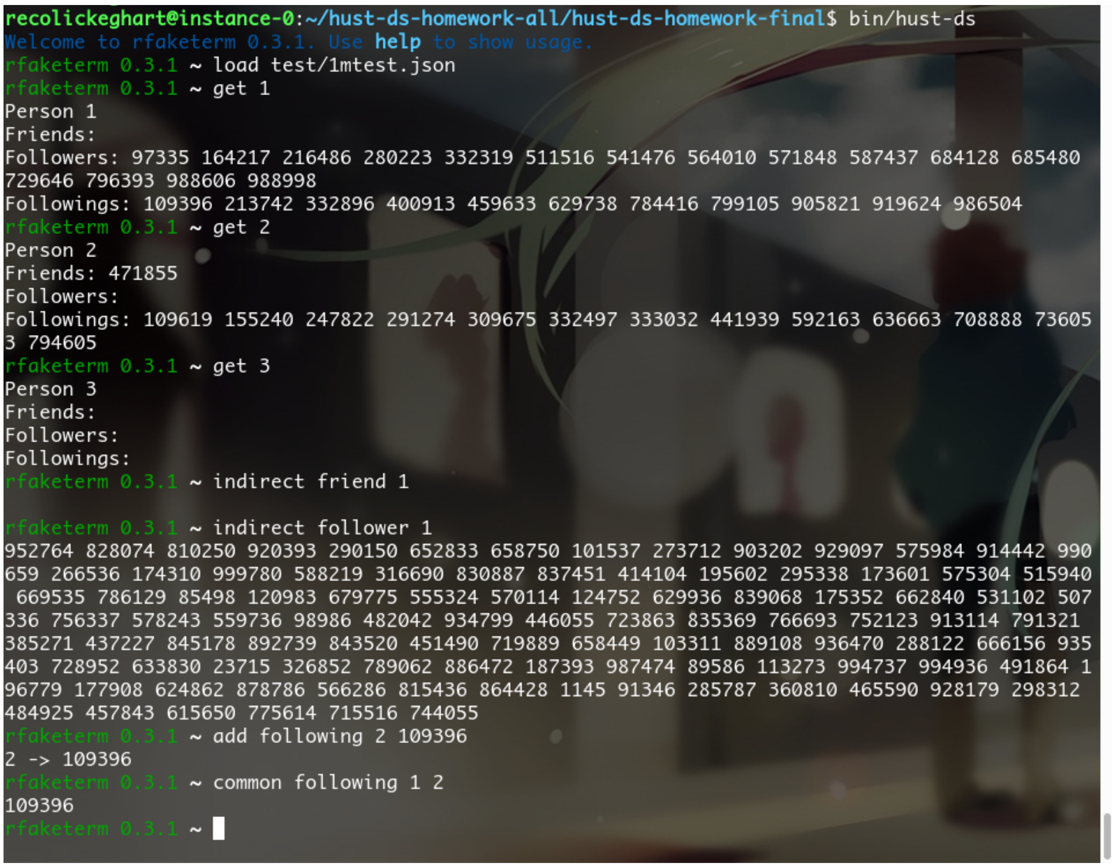
AVL 性能测试(与另一个纯 C 语言实现的 AVL 相比慢约 10%，相对于 shared_ptr 和 weak_ptr 的开销来说，这是合理的)(Intel(R) Core(TM) i5-4200H CPU @ 2.80GHz)：

Insertion(Worst case)	**Deletion**	**Time**
10M	-	3497ms
-	10M	2462ms

应用性能测试(Intel(R) Core(TM) i5-4200H CPU @ 2.80GHz+ddr3+机械硬盘)：

Person Set Size	**Total Operations**	**Time**
1K	25K	0.05s
10K	250K	0.53s
100K	2.5M	5.62s
1M	25M	61.65s

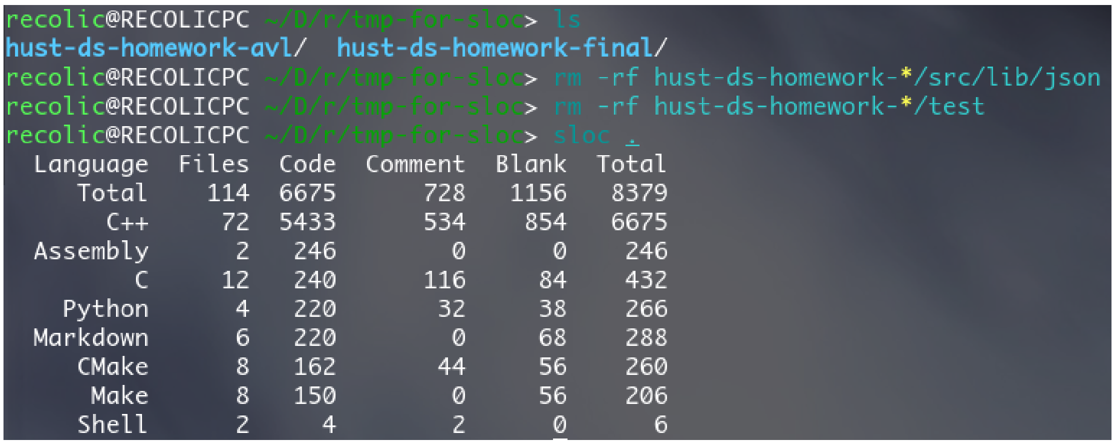
6. 数据 load/save 和大数据情况的测试



```
recolickeghart@instance-0:~/hust-ds-homework-all/hust-ds-homework-final$ bin/hust-ds
Welcome to rfaketerm 0.3.1. Use help to show usage.
rfaketerm 0.3.1 ~ load test/1mtest.json
rfaketerm 0.3.1 ~ get 1
Person 1
Friends:
Followers: 97335 164217 216486 280223 332319 511516 541476 564010 571848 587437 684128 685480
729646 796393 988606 988998
Followings: 109396 213742 332896 400913 459633 629738 784416 799105 905821 919624 986504
rfaketerm 0.3.1 ~ get 2
Person 2
Friends: 471855
Followers:
Followings: 109619 155240 247822 291274 309675 332497 333032 441939 592163 636663 708888 73605
3 794605
rfaketerm 0.3.1 ~ get 3
Person 3
Friends:
Followers:
Followings:
rfaketerm 0.3.1 ~ indirect friend 1

rfaketerm 0.3.1 ~ indirect follower 1
952764 828074 810250 920393 290150 652833 658750 101537 273712 903202 929097 575984 914442 990
659 266536 174310 999780 588219 316690 830887 837451 414104 195602 295338 173601 575304 515940
 669535 786129 85498 120983 679775 555324 570114 124752 629936 839068 175352 662840 531102 507
336 756337 578243 559736 98986 482042 934799 446055 723863 835369 766693 752123 913114 791321
385271 437227 845178 892739 843520 451490 719889 658449 103311 889108 936470 288122 666156 935
403 728952 633830 23715 326852 789062 886472 187393 987474 89586 113273 994737 994936 491864 1
96779 177908 624862 878786 566286 815436 864428 1145 91346 285787 360810 465590 928179 298312
484925 457843 615650 775614 715516 744055
rfaketerm 0.3.1 ~ add following 2 109396
2 -> 109396
rfaketerm 0.3.1 ~ common following 1 2
109396
rfaketerm 0.3.1 ~ █
```

4.3 代码审计

直接看 sloc 结果。



```
recolic@RECOLICPC ~/D/r/tmp-for-sloc> ls
hust-ds-homework-avl/  hust-ds-homework-final/
recolic@RECOLICPC ~/D/r/tmp-for-sloc> rm -rf hust-ds-homework-*/src/lib/json
recolic@RECOLICPC ~/D/r/tmp-for-sloc> rm -rf hust-ds-homework-*/test
recolic@RECOLICPC ~/D/r/tmp-for-sloc> sloc .
  Language  Files  Code  Comment  Blank  Total
     Total    114  6675      728   1156   8379
       C++     72  5433      534    854   6675
  Assembly      2   246        0      0    246
         C     12   240      116     84    432
    Python      4   220       32     38    266
  Markdown      6   220        0     68    288
     CMake      8   162       44     56    260
      Make      8   150        0     56    206
     Shell      2     4        2      0      6
```

(728/8379 = 8.7%)

5.总结与展望

5.1 全文总结

经过本次课程设计，我实现了 avl::tree，avl::multitree，r::set，r::multiset，算法均摊复杂度符合设计要求，更加熟练的掌握了 C++17 的面向对象程序设计，完善了 rlib 和 rfaketerm，学会了从用户的角度考虑数据结构的制作，以实用性为要务来构建算法。
本次课程设计也并非一帆风顺，通过老师的指导，我将 CLI 修改为用户更友好的菜单界面。同时，将测试用的 r::set r::multiset avl::tree 分离出来，使得测试过程更加用户友好。使用更可靠的 r::set 而不是 r::multiset 进行测试。

5.2 工作展望

1.TODO List
完善 rlib 以便支持 meta_template_programming 的一些功能，以及在 rlib 中增加 debug 的支持。增加新的子模块 rlib::meta，rlib::fakterm，rlib::log，rlib::file_descriptor，rlib::debug。
2.在今后的学习生活中，我应记住老师的教导，注重程序实用性。简化用户操作，优化用户体验。

6.体会

本次课程设计，使我对《数据结构》这门课程有了更深入的理解。《数据结构》是一门实践性较强的课程，为了学好这门课程，必须在掌握理论知识的同时，加强上机实践。不断的上网查资料以及翻阅相关书籍，通过不断的模索，测试，发现问题，解决问题和在老师的帮助下一步一步慢慢的正确运行程序，终于完成了这次课程设计，虽然这次课程设计结束了但是总觉 得自已懂得的知识很是不足，学无止境，以后还会更加的努力深入的学习。

我在调试过程中，发生了许多小细节上的问题，它们提醒了自己在以后编程的时候要注意细节，即使是一个括号的遗漏或者一个字符的误写都会造成大量的错误，浪费许多时间去寻找并修改，总结的教训就是写程序的时候，一定要仔细、认真、专注。

我还有一个很深的体会就是格式和注释，由于平时不注意格式和注释这方面的要求，导致有的时候在检查和调试的时候很不方便。有的时候甚至刚刚完成一部分的编辑，结果一不注意，就忘记了这一部分程序的功能。修改的时候也有不小心误删的情况出现。如果注意格式风格，并且养成随手加注释的习惯，就能减少这些不必要的反复和波折。还有一点，就是在修改的时候，要注意修改前后的不同点在哪里，改后调试结果要在原有的基础上更加精确。

参考文献

[1] Eric Alexander. AVL Trees
(http://pages.cs.wisc.edu/~ealexand/cs367/NOTES/AVL-Trees/index.html)

[2] Georgy Adelson-Velsky, G.; Evgenii Landis (1962). "An algorithm for the organization of information"
(http://professor.ufabc.edu.br/~jesus.mena/courses/mc3305-2q-2015/AED2-10-avl-paper.pdf)

[3] wikipedia (https://en.wikipedia.org/wiki/AVL_tree)

附录

下面是代码区。

```
// --- cmake_clean.sh
#!/bin/bash
make clean
rm -rf cmake-build-debug/ cmake_install.cmake Makefile CMakeFiles
CMakeCache.txt
// --- CMakeLists.txt
cmake_minimum_required(VERSION 3.2)
project(hust-ds)

set(CMAKE_CXX_STANDARD 14)
set(CMAKE_C_STANDARD 11)
set(CMAKE_VERBOSE_MAKEFILE ON)

set(CMAKE_CXX_FLAGS_DEBUG "-g -DMALLOC_CHECK_=2")
set(CMAKE_CXX_FLAGS_RELEASE "-O3")

# For codegen.py
add_definitions("-D__refl_func__=")
add_definitions("-D__refl_class__=")
add_definitions("-DENABLE_INSERT_NULL_CHECK")
add_definitions("-DENABLE_RAVL_DEBUG_DUMP")
add_definitions("-DENABLE_RAVL_FOREACH_NODE")

set(THREADS_PREFER_PTHREAD_FLAG ON)
find_package(Threads REQUIRED)

include_directories("/usr/include")
include_directories("/usr/local/include")
include_directories("/usr/include/c++/7.3.0") # Fix clion bug

include_directories(".")
include_directories("./lib")
include_directories("./lib/json/single_include")
include_directories("./lib/gc")
include_directories("./lib/cpp-readline/src")
```

```
add_library(r STATIC lib/rlib/libr.cc)

set(SOURCE_SRC   main.cc   reflected_impl.hpp   general_ui.hpp   parser.hpp
person.hpp db.hpp)
set(LIB_SRC lib/avl.hpp lib/set.hpp)
set(BUILD_SRC ${LIB_SRC} ${SOURCE_SRC})

add_executable(hust-ds-no-menu ${BUILD_SRC})
add_executable(hust-ds ${BUILD_SRC})
target_compile_definitions(hust-ds PRIVATE ENABLE_SILLY_FAKE_MENU=)
IF (WIN32)
      set_target_properties(hust-ds PROPERTIES LINK_FLAGS "-static" )
      set_target_properties(hust-ds-no-menu    PROPERTIES    LINK_FLAGS    "-
static" )
ENDIF ()

target_link_libraries(hust-ds r)
target_link_libraries(hust-ds-no-menu r)
target_link_libraries(hust-ds Threads::Threads)
target_link_libraries(hust-ds-no-menu Threads::Threads)
// --- db.hpp
#ifndef _HUST_DS_DB_HPP
#define _HUST_DS_DB_HPP 1

#include <person.hpp>
#include <nlohmann/json.hpp>
#include <rlib/class_decorator.hpp>
#include <fstream>

struct person_db : rlib::static_class {
 using json = nlohmann::json;

 static json r_set_to_json(const r::set<person_id> &buf) {
  json res;
  std::for_each(buf.cbegin(), buf.cend(), [&](const person_id &i){
   res.push_back(i);
  });
  return std::move(res);
 }

 static void save(/*const*/ r::set<person> &buffer, const std::string &file)
{
  std::ofstream out(file);
  if(!out)
```

```
        throw std::invalid_argument(rlib::format_string("Failed to write to
`{}`.", file));
   json result;
   for(const person &p : buffer) {
    result.push_back({
     {"id", p.id},
     {"friends", r_set_to_json(p.friends)},
     {"followers", r_set_to_json(p.followers)},
     {"followings", r_set_to_json(p.followings)}
    });
   }
   out << result << std::endl;
  }
     static void load(r::set<person> &buffer, const std::string &file,
id_manager &id_manager1) {
   std::ifstream in(file);
   if(!in)
        throw std::invalid_argument(rlib::format_string("Failed to read from
`{}`.", file));
   json content;
   in >> content;
   for(json &p_json : content) {
    person p(id_manager1, p_json["id"]);
    for(const person_id &i : p_json["friends"])
     p.friends.insert(i);
    for(const person_id &i : p_json["followers"])
     p.followers.insert(i);
    for(const person_id &i : p_json["followings"])
     p.followings.insert(i);
    buffer.insert(std::move(p));
   }
  }
};

#endif //_HUST_DS_DB_HPP
// --- general_ui.hpp
#ifndef HUST___GENERAL_UI_HPP_
#define HUST___GENERAL_UI_HPP_

#include <cpp-readline/src/Console.hpp>

#include <functional>
#include <string>
#include <iostream>
```

```
#include <list>

#include <unistd.h>
#include <csignal>
#include <cstdio>

#include <rlib/stdio.hpp>
#include <rlib/terminal.hpp>
#include <rlib/string/string.hpp>

#include <rlib/sys/os.hpp>

using namespace rlib::terminal;
using rlib::splitString;

class fake_terminal {
#ifdef FORCE_MULTICOLOR_RTERM
 static constexpr bool force_multicolor = true;
#else
 static constexpr bool force_multicolor = false;
#endif
 static void _clear() {rlib::printf("\033[H\033[J");}
 //static void _clear() {system("tput clear");}
 //static void _clear() {rlib::printf("\x5b\x1b\x4a\x33\x5b\x1b\x1b\x48\x32\
x5b\x00\x4a");}
public:
 using callback_t = std::function<void (std::vector<std::string>)>;
 static void sigint_handler(int) {
  if(is_scripting())
   exit(127);
  rlib::println();
  if(force_multicolor || rlib::OSInfo::os != rlib::OSInfo::os_t::WINDOWS)
   rlib::printfln("Use {}exit{} to exit.", font_t::bold, clear);
  else
   rlib::println("Use `exit` to exit.");
  rlib::print(prompt());
  std::cout.flush();
 }

 [[noreturn]] static void go(const callback_t &callback) {
  //CppReadline::Console console(prompt());
#ifdef ENABLE_SILLY_FAKE_MENU
  if(!is_scripting()) _clear();
#endif
```

```
   if(!is_scripting()) rlib::println(welcome());
   signal(SIGINT, fake_terminal::sigint_handler);
   while(true) {
#ifdef ENABLE_SILLY_FAKE_MENU
    if(!is_scripting()) callback(std::vector<std::string>{"help"});
#endif
    if(!is_scripting()) rlib::print(prompt());
    try {
     auto cont = rlib::scanln();
#ifdef ENABLE_SILLY_FAKE_MENU
     if(!is_scripting()) _clear();
#endif
     size_t pos = cont.find('#');
     if(pos ≠ std::string::npos)
                    cont  =  cont.substr(0,  pos);  //Remove  comments.  Avoid
rlib::splitString to make it faster.
     callback(splitString(cont));
    }
    catch(std::exception &e) {
     showError(e.what());
     if(is_scripting()) {
      rlib::println("Exiting because of previous error...");
      std::exit(5);
     }
    }
    if(std::cin.eof())
     std::exit(0);
   }
  }
private:
  static bool is_scripting() __attribute__((pure, const)) {
#define WINDOWS 123
#if RLIB_OS_ID == WINDOWS
   return !_isatty(fileno(stdin));
#else
   return !isatty(fileno(stdin));
#endif
#undef WINDOWS
  }
  static string prompt() {
   if(force_multicolor || rlib::OSInfo::os ≠ rlib::OSInfo::os_t::WINDOWS)
     return rlib::format_string("{}rfaketerm 0.3.1{} {}~{} ", color_t::green,
clear, font_t::bold, clear);
   else
```

```
      return "rfaketerm 0.3.1 ~";
  }
  static string welcome() {
    if(force_multicolor || rlib::OSInfo::os != rlib::OSInfo::os_t::WINDOWS)
      return rlib::format_string("{}Welcome to rfaketerm 0.3.1. Use {}help{}{} to show usage.{}", color_t::blue, font_t::bold, clear, color_t::blue, clear);
    else
      return "Welcome to rfaketerm 0.3.1. Use `help` to show usage.";
  }
  static void showError(const std::string &msg) {
    if(force_multicolor || rlib::OSInfo::os != rlib::OSInfo::os_t::WINDOWS)
      rlib::printfln("{}{}Error{}{}: {}{}", color_t::red, font_t::bold, clear, color_t::lightgray, msg, clear);
    else
      rlib::printfln("Error: {}", msg);
  }
};

#endif
// --- lib

cat: lib: 是一个目录
// --- main.cc
#include <general_ui.hpp>
#include <parser.hpp>

reflected_impl impl;
//GCThread gc;

int main() {
  fake_terminal::go(parser::parse);
}// --- parser-codegen.py
#!/usr/bin/python3

import sys
if len(sys.argv) != 2:
  print('Usage: `./ccgen.py code` or `./ccgen.py help`')
  exit(1)

src = 'reflected_impl.hpp'
mode = sys.argv[1]

# DO NOT use macro in func_name! It'll gen wrong code!
```

```
macro_list = [
 ('stref','string'),
 ('void','null'),
]

size_arg = ['size_t']
int_arg = ['int', 'data_t']
string_arg = ['string']

void_ret = ['void', 'null']

def gen_code(line):
 line = line.replace('\t','').replace('\r', '').strip()
 if len(line) == 0:
   return
 ret_type = line.split(' ')[0]
 funcAndArgs = line[len(ret_type):].strip().split('(')
 func_name, args = funcAndArgs[0], funcAndArgs[1].split(')')[0]
 print('//__ccgen_debug__: `ret name(args)` is `{} {}({})`'.format(ret_type,
func_name, args))

 args_string = []
 for arg in args.split(','):
   arg_type = arg.strip().split(' ')[0].replace(' ','')
   if len(arg_type) == 0:
    continue
   if arg_type in size_arg:
       args_string.append('SIZE_ARG({})'.format(len(args_string)+1)) # start
from one
   elif arg_type in int_arg:
        args_string.append('INT_ARG({})'.format(len(args_string)+1)) # start
from one
   elif arg_type in string_arg:
      args_string.append('STRING_ARG({})'.format(len(args_string)+1)) # start
from one
   else:
                  raise RuntimeError('Unclassed arg left here. line={}|
arg_type={}'.format(line, arg_type))
 args_size = len(args_string)
 args_string = ', '.join(args_string)

 print('    IFCMD("{}") {{'.format(func_name))
 print('        WANT_ARG({})'.format(args_size))
 if ret_type not in void_ret:
   print('       HAVE_RETURN_VALUE')
  print('       impl.{}({});'.format(func_name, args_string))
  if ret_type not in void_ret:
   print('       PRINT_RETURN_VALUE')
  print('    }')

def gen_help(line):
  line = line.replace('\t','').replace('\r', '').strip()
  if len(line) == 0:
   return
  ret_type = line.split(' ')[0]
  funcAndArgs = line[len(ret_type):].strip().split('(')
  func_name, args = funcAndArgs[0], funcAndArgs[1].split(')')[0]
#           print('//__ccgen_debug__:  `ret  name(args)`  is  `{}  {}
({})`'.format(ret_type, func_name, args))
  if len(args) == 0:
   print('{} -> {}'.format(func_name, ret_type))
  else:
   print('{} [{}] -> {}'.format(func_name, args, ret_type))

if mode == 'code':
  fuck_a_line = gen_code
  print('//Code generated by ccgen.py below. Do not edit them by hand.')
else:
  fuck_a_line = gen_help
  print('FuncName [Argument ...] -> ReturnValue # Instructions')

with open(src) as fd:
  cont = fd.read()

for line in cont.split('\n'):
  if '__refl_func__' in line:
   line = line.replace('__refl_func__ ', '')
   for _from, _to in macro_list:
    line = line.replace(_from, _to)
   fuck_a_line(line)

if mode != 'code':
  exit(0)

print('''
  IFCMD("exit") {
   rlib::println("bye~");
   ::std::exit(0);
  }
  IFCMD("help") {
    help_msg();
  }
//impl.debug();
//Code generated by ccgen.py above. Do not edit them by hand.
''')
// --- parser.hpp
#ifndef _HUST___PARSER_HPP
#define _HUST___PARSER_HPP 1

#include <reflected_impl.hpp>
#include <list>
#include <string>
#include <iomanip>

#include <rlib/stdio.hpp>
#include <rlib/terminal.hpp>

using namespace rlib::terminal;

class parser
{
private:
  static void help_msg()
  {
    std::string msg = R"_STR_(
rfaketerm 0.3.1 HUST_xxxx special edition

>>> Usage: <Command> [args ...]

>>> Command List:

FuncName [Argument ...] -> ReturnValue # Instructions

help -> null # Show this message.
exit -> int # Exit.
load [string file_name] -> null # Load data from json file.
save [string file_name] -> null # Save data to json file.
clear -> null # Clear data in buffer.

new_person -> string # `new` a person, with a random-generated id.
new_person_at [string id_to_assign] -> string # `new` a person, with a
specified id.
rm_person [string person_id] → string # remove the person specified by id.
get [string person_id] → null # Show all information about the person.
ls → null # List all person_id.

add [string what, string person_id, string to_add_id] → string # add a
friend/follower/following.
rm [string what, string person_id, string to_remove_id] → string # remove a
friend/follower/following.
common [string what, string person1_id, string person2_id] → null # get
common friend/follower/following of two person.
indirect [string what, string person_id] → null # get 2-degree(indirect)
friend/follower/following of a person.)_STR_";
   rlib::println(msg);
  }

public:
  static void parse(const std::vector<std::string> &to_parse)
  {
   if (to_parse.empty())
    return;
   rlib::print(std::boolalpha);

#define AREA_BEGIN if(to_parse.begin()->empty()) {}
#define IFCMD(str) else if(*to_parse.begin() == str)
#define AREA_END else

#define WANT_ARG(n) if(to_parse.size() != n+1) {throw
std::runtime_error(rlib::format_string("{} arguments wanted but {}
provided.", n, to_parse.size()-1));}
#define STRING_ARG(n) to_parse[n]
#define SIZE_ARG(n) std::stoul(to_parse[n])
#define INT_ARG(n) std::stoi(to_parse[n])
#define HAVE_RETURN_VALUE auto ret =
#define PRINT_RETURN_VALUE rlib::println(ret);

   AREA_BEGIN
//__ccgen_managed_begin__

//Code generated by ccgen.py below. Do not edit them by hand.
//__ccgen_debug__: `ret name(args)` is `null save(string fname)`
  IFCMD("save") {
   WANT_ARG(1)
   impl.save(STRING_ARG(1));
  }
```

```
//__ccgen_debug__: `ret name(args)` is `null load(string fname)`
 IFCMD("load") {
  WANT_ARG(1)
  impl.load(STRING_ARG(1));
 }
//__ccgen_debug__: `ret name(args)` is `null clear()`
 IFCMD("clear") {
  WANT_ARG(0)
  impl.clear();
 }
//__ccgen_debug__: `ret name(args)` is `string new_person()`
 IFCMD("new_person") {
  WANT_ARG(0)
  HAVE_RETURN_VALUE
  impl.new_person();
  PRINT_RETURN_VALUE
 }
//__ccgen_debug__:   `ret   name(args)`   is   `string   new_person_at(string
id_to_assign)`
 IFCMD("new_person_at") {
  WANT_ARG(1)
  HAVE_RETURN_VALUE
  impl.new_person_at(STRING_ARG(1));
  PRINT_RETURN_VALUE
 }
//__ccgen_debug__: `ret name(args)` is `string rm_person(string person_id)`
 IFCMD("rm_person") {
  WANT_ARG(1)
  HAVE_RETURN_VALUE
  impl.rm_person(STRING_ARG(1));
  PRINT_RETURN_VALUE
 }
//__ccgen_debug__: `ret name(args)` is `null get(string person_id)`
 IFCMD("get") {
  WANT_ARG(1)
  impl.get(STRING_ARG(1));
 }
//__ccgen_debug__: `ret name(args)` is `null ls()`
 IFCMD("ls") {
  WANT_ARG(0)
  impl.ls();
 }
//__ccgen_debug__:  `ret  name(args)`  is  `string  add(string  what,  string
person_id, string to_add_id)`
```

```
  IFCMD("add") {
    WANT_ARG(3)
    HAVE_RETURN_VALUE
    impl.add(STRING_ARG(1), STRING_ARG(2), STRING_ARG(3));
    PRINT_RETURN_VALUE
  }
//__ccgen_debug__: `ret name(args)` is `string rm(string what, string
person_id, string to_remove_id)`
  IFCMD("rm") {
    WANT_ARG(3)
    HAVE_RETURN_VALUE
    impl.rm(STRING_ARG(1), STRING_ARG(2), STRING_ARG(3));
    PRINT_RETURN_VALUE
  }
//__ccgen_debug__: `ret name(args)` is `null common(string what, string
person1_id, string person2_id)`
  IFCMD("common") {
    WANT_ARG(3)
    impl.common(STRING_ARG(1), STRING_ARG(2), STRING_ARG(3));
  }
//__ccgen_debug__: `ret name(args)` is `null indirect(string what, string
person_id)`
  IFCMD("indirect") {
    WANT_ARG(2)
    impl.indirect(STRING_ARG(1), STRING_ARG(2));
  }

  IFCMD("exit") {
    rlib::println("bye~");
    ::std::exit(0);
  }
  IFCMD("help") {
    help_msg();
  }
//impl.debug();
//Code generated by ccgen.py above. Do not edit them by hand.

//__ccgen_managed_end__
    AREA_END
    {
      throw std::invalid_argument("Invalid argument. Try to type `help` to get
helped.");
    }
  }
};

#endif //_HUST___PARSER_HPP
// --- person.hpp
#ifndef RCPP_PERSON_HPP_
#define RCPP_PERSON_HPP_

#include <string>
#include <random>
#include <unordered_map>
#include <set.hpp>
using std::string;
using person_id = uint64_t;

struct person;

class id_manager : rlib::noncopyable {
public:
  id_manager() : randGen(std::random_device()()) {}
  person_id generate(const person *pp) {
    while(true) {
      person_id trial = randGen();
      if(id_pool.find(trial) == id_pool.end()) {
        id_pool[trial] = pp;
        return trial;
      }
    }
    throw std::runtime_error("Unknown error.");
  }
  const person *check_id(person_id id) const {
    auto pos = id_pool.find(id);
    if(pos == id_pool.cend())
      throw std::invalid_argument("Invalid id.");
    return pos->second;
  }
  void update(person_id id, const person *pp) {
    if(id_pool.find(id) != id_pool.end())
      id_pool[id] = pp;
    else
      throw std::invalid_argument("Invalid id to update.");
  }
  void assign(person_id id, const person *pp) {
    if(id_pool.find(id) == id_pool.end())
      id_pool[id] = pp;
    else
      throw std::invalid_argument("Id already used.");
  }
  void clear() {
    id_pool.clear();
  }
private:
  std::unordered_map<person_id, const person *> id_pool;
  std::mt19937_64 randGen;
};

struct person : public rlib::noncopyable {
  person() {}
        explicit   person(id_manager   &idgen)   :   id(idgen.generate(this)),
pmanager(&idgen) {}
  explicit person(id_manager &idgen, person_id id) : id(id), pmanager(&idgen)
{idgen.assign(id, this);}
  person(person &&another) {
    swap(std::move(another));
  }
  const person &operator=(person &&another) {
    swap(std::move(another));
    return *this;
  }

  bool operator==(const person &another) const {
    return id == another.id;
  }
  bool operator!=(const person &another) const {
    return ! operator==(another);
  }
  bool operator<(const person &another) const {
    return id < another.id;
  }
  bool operator>(const person &another) const {
    return ! ( operator<(another) || operator==(another) );
  }

  void show() const {
    rlib::println("Person", id);
    friends.show("Friends");
    followers.show("Followers");
    followings.show("Followings");
  }
  void swap(person &another) {
    std::swap(pmanager, another.pmanager);
    std::swap(id, another.id);
    if(id) pmanager->update(id, this);
    if(another.id) pmanager->update(another.id, &another);

    friends.swap(another.friends);
    followers.swap(another.followers);
    followings.swap(another.followings);
  }

  person_id id = 0;
  id_manager *pmanager = nullptr;
  r::set<person_id> friends;
  r::set<person_id> followers;
  r::set<person_id> followings;
};

#endif
// --- reflected_impl.hpp
#ifndef HUST___REFLECTED_IMPL_HPP_
#define HUST___REFLECTED_IMPL_HPP_

/*
 * You should NEVER use this code in ANY consequence,
 *     as these code is just to make hust happy.
 */

#include <utility>
#include <functional>
#include <algorithm>
#include <person.hpp>
#include <db.hpp>

#include <rlib/stdio.hpp>
#include <unordered_set>

using std::to_string;
using std::stoull;
using std::string;

__refl_class__ class reflected_impl : public rlib::nonmovable {
```

```
public:
 using stref = const string &;
 using id_t = person_id;

 __refl_func__ void save(stref fname) {
  person_db::save(buf, fname);
 }
 __refl_func__ void load(stref fname) {
  clear();
  person_db::load(buf, fname, id_manager1);
 }
 __refl_func__ void clear() {
  buf.clear();
  id_manager1.clear();
 }

 __refl_func__ string new_person() {
  auto iter = buf.insert(person(id_manager1));
  return to_string(iter->id);
 }
 __refl_func__ string new_person_at(stref id_to_assign) {
  auto iter = buf.insert(person(id_manager1, stoull(id_to_assign)));
  return to_string(iter->id);
 }
 __refl_func__ string rm_person(stref person_id) {
  const person *pp = id_manager1.check_id(stoull(person_id));
  buf.erase(buf.find(*pp));
  return person_id;
 }
 __refl_func__ void get(stref person_id) {
  id_manager1.check_id(stoull(person_id)) -> show();
 }
 __refl_func__ void ls() {
  rlib::println(buf.size(), "person:");
            std::for_each(buf.cbegin(),  buf.cend(),  [](const  person  &p)
{rlib::print(p.id, "");});
  rlib::println();
 }

 __refl_func__ string add(stref what, stref person_id, stref to_add_id) {
                                  person     *pp     =     const_cast<person
*>(id_manager1.check_id(stoull(person_id)));
  id_manager1.check_id(stoull(to_add_id));
```

```
#define OPERATION_(what_) pp->what_.insert(stoull(to_add_id));
   if(what == "friend") {
     OPERATION_(friends)
   }
   else if(what == "follower") {
     OPERATION_(followers)
   }
   else if(what == "following") {
     OPERATION_(followings)
   }
   else
      throw std::invalid_argument(rlib::format_string("Can not understand {}.
Try `help`.", what));
#undef OPERATION_
   return rlib::format_string("{} -> {}", person_id, to_add_id);
  }

  __refl_func__ string rm(stref what, stref person_id, stref to_remove_id) {
                                     person     *pp     =     const_cast<person
*>(id_manager1.check_id(stoull(person_id)));
   id_manager1.check_id(stoull(to_remove_id));

#define                 OPERATION_(what_)                 pp->what_.erase(pp-
>what_.find(stoull(to_remove_id)));
   if(what == "friend") {
     OPERATION_(friends)
   }
   else if(what == "follower") {
     OPERATION_(followers)
   }
   else if(what == "following") {
     OPERATION_(followings)
   }
   else
      throw std::invalid_argument(rlib::format_string("Can not understand {}.
Try `help`.", what));
#undef OPERATION_
   return rlib::format_string("{} -/> {}", person_id, to_remove_id);
  }

  __refl_func__ void common(stref what, stref person1_id, stref person2_id) {
   const person *pp1 = id_manager1.check_id(stoull(person1_id));
   const person *pp2 = id_manager1.check_id(stoull(person2_id));
```

```
#define OPERATION_(what_) (pp1->what_ ^ pp2->what_).show();
   if(what == "friend") {
     OPERATION_(friends)
   }
   else if(what == "follower") {
     OPERATION_(followers)
   }
   else if(what == "following") {
     OPERATION_(followings)
   }
   else
      throw std::invalid_argument(rlib::format_string("Can not understand {}.
Try `help`.", what));
#undef OPERATION_
  }

  __refl_func__ void indirect(stref what, stref person_id) {
   std::unordered_set<id_t> buffer;
   const person *pp = id_manager1.check_id(stoull(person_id));

#define OPERATION_(what_) \
     std::for_each(pp->what_.cbegin(), pp->what_.cend(), [&buffer, self=this]
(const id_t &id){ \
     const person *pp = self->id_manager1.check_id(id); \
      std::for_each(pp->what_.cbegin(), pp->what_.cend(), [&buffer](const id_t
&id){ \
       buffer.insert(id); \
     }); \
   }); \
     std::for_each(pp->what_.cbegin(), pp->what_.cend(), [&buffer](const id_t
&id) { \
     buffer.erase(id); \
   }); \
   buffer.erase(pp->id);

   if(what == "friend") {
     OPERATION_(friends)
   }
   else if(what == "follower") {
     OPERATION_(followers)
   }
   else if(what == "following") {
     OPERATION_(followings)
   }
```

```
    else
      throw std::invalid_argument(rlib::format_string("Can not understand {}.
Try `help`.", what));
#undef OPERATION_

    std::for_each(buffer.cbegin(), buffer.cend(), [&buffer](const id_t &id){
      rlib::print(id, "");
    });
    rlib::println();
  }

private:
  r::set<person> buf;
  id_manager id_manager1;
};

extern reflected_impl impl;

#endif
// --- lib/avl.hpp
#ifndef R_AVL_HPP_
#define R_AVL_HPP_

#include <rlib/require/cxx14>
#include <rlib/class_decorator.hpp>
#include <rlib/string/string.hpp>
#include <memory>
#include <functional>
#include <utility>

#include <rlib/stdio.hpp>

namespace avl {
  template <typename data_t, bool multi_tree = false>
  struct node {
    using this_type = node<data_t, multi_tree>;
    std::shared_ptr<this_type> left;
    std::shared_ptr<this_type> right;
    std::weak_ptr<this_type> parent;
    data_t data;
    int32_t factor = 0;
    size_t counter = 1;

#ifdef ENABLE_RAVL_FOREACH_NODE
```

```
  void for_each_node(std::function<void(const std::shared_ptr<this_type> &)>
func) {
    if(left)
      left->for_each_node(func);
    if(!parent.expired()) {
      auto par = parent.lock();
      auto self_ptr = par->left;
      if(par->right.get() == this)
        self_ptr = par->right;
      //root ignored because I can not get self_ptr

      for(size_t cter = 0; cter < counter; ++cter)
        func(self_ptr);
    }
    if(right)
      right->for_each_node(func);
  }
#endif
 };

     template  <typename  data_t,  bool  multi_tree  =  false,  typename
data_equal_func   =   std::equal_to<data_t>,   typename   data_less_func   =
std::less<data_t>>
 class tree : public rlib::noncopyable {
 public:
         using  this_type  =  tree<data_t,  multi_tree,  data_equal_func,
data_less_func>;
  using node_type = node<data_t, multi_tree>;

  tree() = default;
  tree(this_type &&another) noexcept {
    swap(std::move(another));
  }
  this_type &operator=(this_type &&another) noexcept {
    swap(std::move(another));
    return *this;
  }

  class iterator_utils : public rlib::static_class {
  public:
    template <typename cv_node_type>
        static  void  next(size_t  &curr_cter,  std::shared_ptr<cv_node_type>
&target, bool _back_tracing = false) {
      if(!target)
```

```
        throw std::invalid_argument("Can not ++ null iterator.");
      if(_back_tracing) { //target->right already fucked.
        if(target->parent.expired()) {
          target.reset();
          return;
        }
        auto par = target->parent.lock();
        if(par->left && par->left == target) {
          target = par;
          curr_cter = 1;
          return;
        }
        target = par;
        return next(curr_cter, target, true);
      }

      if(target->counter > curr_cter) {
        ++curr_cter;
        return;
      }
      if(target->right) {
        target = target->right;
        while(target->left) target = target->left;
        curr_cter = 1;
        return;
      }
      else
        return next(curr_cter, target, true);
    }

    template <typename cv_node_type>
          static  void  prev(size_t  &curr_cter,  std::shared_ptr<cv_node_type>
&target, bool _back_tracing = false) {
      if(!target)
        throw std::invalid_argument("Can not -- null iterator.");
      if(_back_tracing) { //target->left already fucked.
        if(target->parent.expired()) {
          target.reset();
          return;
        }
        auto par = target->parent.lock();
        if(par->right && par->right == target) {
          target = par;
          curr_cter = 1;
```

```
            return;
          }
          target = par;
          return prev(curr_cter, target, true);
        }

        if(target->counter > curr_cter) {
          ++curr_cter;
          return;
        }
        if(target->left) {
          target = target->left;
          while(target->right) target = target->right;
          curr_cter = 1;
          return;
        }
        else
          return prev(curr_cter, target, true);
      }
    };

    class iterator {
    public:
      using this_type = iterator;
      iterator() = default;
      iterator(const std::shared_ptr<node_type> &item) : target(item) {}

      this_type &operator=(const std::shared_ptr<node_type> &item) {target = item;}

      this_type &operator++() {iterator_utils::next(curr_cter, target); return *this;}
      this_type operator++(int) {this_type backup = *this; iterator_utils::next(curr_cter, target); return backup;}
      this_type &operator--() {iterator_utils::prev(curr_cter, target); return *this;}
      this_type operator--(int) {this_type backup = *this; iterator_utils::prev(curr_cter, target); return backup;}

      data_t &operator*() {return target->data;}
      data_t *operator->() {return &target->data;}

      std::shared_ptr<node_type> to_shared() {return target;}
      operator bool() {return static_cast<bool>(target);}
```

```
  private:
   size_t curr_cter = 1;
   std::shared_ptr<node_type> target;
  };
  class reverse_iterator {
  public:
   using this_type = reverse_iterator;
   reverse_iterator() = default;
    reverse_iterator(const std::shared_ptr<node_type> &item) : target(item)
{}

    this_type &operator=(const std::shared_ptr<node_type> &item) {target =
item;}

    this_type &operator++() {iterator_utils::prev(curr_cter, target); return
*this;}
              this_type  operator++(int)  {this_type  backup  =  *this;
iterator_utils::prev(curr_cter, target); return backup;}
    this_type &operator--() {iterator_utils::next(curr_cter, target); return
*this;}
              this_type  operator--(int)  {this_type  backup  =  *this;
iterator_utils::next(curr_cter, target); return backup;}

   data_t &operator*() {return target->data;}
   data_t *operator->() {return &target->data;}

   std::shared_ptr<node_type> to_shared() {return target;}
   operator bool() {return static_cast<bool>(target);}
  private:
   size_t curr_cter = 1;
   std::shared_ptr<node_type> target;
  };
  class const_iterator {
  public:
   using this_type = const_iterator;
   const_iterator() = default;
        const_iterator(const  std::shared_ptr<const  node_type>  &item)  :
target(item) {}

      this_type  &operator=(const  std::shared_ptr<const  node_type>  &item)
{target = item;}

    this_type &operator++() {iterator_utils::next(curr_cter, target); return
*this;}
```

```
                  this_type  operator++(int)  {this_type  backup  =  *this;
iterator_utils::next(curr_cter, target); return backup;}
     this_type &operator--() {iterator_utils::prev(curr_cter, target); return
*this;}
                  this_type  operator--(int)  {this_type  backup  =  *this;
iterator_utils::prev(curr_cter, target); return backup;}

    const data_t &operator*() {return target->data;}
    const data_t *operator->() {return &target->data;}

    std::shared_ptr<const node_type> to_shared() {return target;}
    operator bool() {return static_cast<bool>(target);}
  private:
    size_t curr_cter = 1;
    std::shared_ptr<const node_type> target;
  };
  class const_reverse_iterator {
  public:
    using this_type = const_reverse_iterator;
    const_reverse_iterator() = default;
      const_reverse_iterator(const std::shared_ptr<const node_type> &item) :
target(item) {}

         this_type  &operator=(const  std::shared_ptr<const  node_type>  &item)
{target = item;}

     this_type &operator++() {iterator_utils::prev(curr_cter, target); return
*this;}
                  this_type  operator++(int)  {this_type  backup  =  *this;
iterator_utils::prev(curr_cter, target); return backup;}
     this_type &operator--() {iterator_utils::next(curr_cter, target); return
*this;}
                  this_type  operator--(int)  {this_type  backup  =  *this;
iterator_utils::next(curr_cter, target); return backup;}

    const data_t &operator*() {return target->data;}
    const data_t *operator->() {return &target->data;}

    std::shared_ptr<const node_type> to_shared() {return target;}
    operator bool() {return static_cast<bool>(target);}
  private:
    size_t curr_cter = 1;
    std::shared_ptr<const node_type> target;
  };
  iterator begin() {
    if(!root) return iterator();
    auto curr = root;
    while(curr->left)
      curr = curr->left;
    return iterator(curr);
  }
  constexpr iterator end() {
    return iterator();
  }
  reverse_iterator rbegin() {
    if(!root) return reverse_iterator();
    auto curr = root;
    while(curr->right)
      curr = curr->right;
    return reverse_iterator(curr);
  }
  constexpr reverse_iterator rend() {
    return reverse_iterator();
  }
  const_iterator cbegin() const {
    if(!root) return const_iterator();
    auto curr = std::const_pointer_cast<const node_type>(root);
    while(curr->left)
      curr = curr->left;
    return const_iterator(curr);
  }
  constexpr const_iterator cend() const {
    return const_iterator();
  }
  const_reverse_iterator crbegin() const {
    if(!root) return const_reverse_iterator();
    auto curr = std::const_pointer_cast<const node_type>(root);
    while(curr->right)
      curr = curr->right;
    return curr;
  }
  constexpr const_reverse_iterator crend() const {
    return const_reverse_iterator();
  }

  auto insert(data_t &&item, bool no_except = false) {
#ifdef ENABLE_INSERT_NULL_CHECK
```

```
    if(data_equal_func()(item, data_t()))
          throw std::invalid_argument("Sorry but you can not insert a null
item.");
#endif
    std::shared_ptr<node_type> iter;
    int pos;
    std::tie(iter, pos) = do_find(item);

    if(!multi_tree && pos == 0) {
      if(no_except)
        return iterator();
      else
        throw std::runtime_error("Inserting duplicate element.");
    }

    ++m_size;
    if(multi_tree && pos == 0) {
      iter->counter ++;
      return iterator(iter);
    }

    auto new_node = std::make_shared<node_type>();
    new_node->data = std::move(item);

    decltype(iter->factor) iter_factor_backup;
    switch(pos) {
      case -2:
        root = new_node;
        break;
      case -1:
        new_node->parent = iter;
        iter->left = new_node;
        iter_factor_backup = iter->factor;
        iter->factor -= 1;
        rebalance<true>(iter, iter_factor_backup);
        break;
      case 1:
        new_node->parent = iter;
        iter->right = new_node;
        iter_factor_backup = iter->factor;
        iter->factor += 1;
        rebalance<true>(iter, iter_factor_backup);
        break;
      default:
         throw std::runtime_error(rlib::format_string("do_find returns invalid
 pos {}.", pos));
     }
     return iterator(new_node);
    }
    auto insert(const data_t &item, bool no_except = false) {
     data_t copied(item);
     return insert(std::move(copied), no_except);
    }

    void erase(iterator _iter) {
     auto iter = _iter.to_shared();
     exchange_again:
     if(multi_tree && (iter->counter > 1)) {
       --iter->counter;
       --m_size;
       return;
     }

     auto to_erase = iter;
     if(iter->right) {
       to_erase = iter->right;
       while(to_erase->left)
        to_erase = to_erase->left;
     }
     else if(iter->left) {
       to_erase = iter->left;
       while(to_erase->right)
        to_erase = to_erase->right;
     }

     if(iter != to_erase) {
       iter->data = std::move(to_erase->data);
       if(multi_tree)
        iter->counter = to_erase->counter;

       // fake recursion: erase(to_erase)
       iter = to_erase;
       goto exchange_again;
     }

     // do erase
     --m_size;
     if(to_erase->parent.expired())
       root.reset();
     else {
       auto par = to_erase->parent.lock();
       auto parent_factor_backup = par->factor;
       if(par->right == to_erase) {
        par->right.reset();
        par->factor -= 1;
       }
       else {
        par->left.reset();
        par->factor += 1;
       }
       rebalance<false>(par, parent_factor_backup);
     }
    }

    template <bool no_except = false>
    const_iterator find(const data_t &item) const {
     std::shared_ptr<const node_type> iter;
     int pos;
     std::tie(iter, pos) = do_find(item);

     if(no_except || pos == 0)
       return const_iterator(iter);
     else
       throw std::out_of_range("Element not found.");
    }
    template <bool no_except = false>
    iterator find(const data_t &item) {
     std::shared_ptr<node_type> iter;
     int pos;
     std::tie(iter, pos) = do_find(item);

     if(no_except || pos == 0)
       return iterator(iter);
     else
       throw std::out_of_range("Element not found.");
    }

    size_t count(const data_t &item) const noexcept {
     std::shared_ptr<const node_type> iter;
     int pos;
     std::tie(iter, pos) = do_find(item);
```

```
    if(pos != 0)
      return 0;
    if(!multi_tree)
      return 1;
    return iter->counter;
  }
  bool exist(const data_t &item) const noexcept {
    return count(item) != 0;
  }
  size_t height() const noexcept {
    size_t curr_height = 0;
    std::shared_ptr<const node_type> curr = root;
    while(curr) {
      ++curr_height;
      if(curr->factor == 1)
        curr = curr->right;
      else
        curr = curr->left;
    }
    return curr_height;
  }
  size_t size() const noexcept {
    return m_size;
  }

  void swap(this_type &&another) noexcept {
    root.swap(another.root);
    std::swap(m_size, another.m_size);
  }
  void swap(this_type &another) noexcept {
    root.swap(another.root);
    std::swap(m_size, another.m_size);
  }
  void clear() noexcept {
    root.reset();
    m_size = 0;
  }

#ifdef ENABLE_RAVL_DEBUG_DUMP
  void dump() {
    // Error if data_t is not printable.
    rlib::println("root: node", root?root->data:data_t());
    for_each_node([](const std::shared_ptr<node_type> &node){
        rlib::printfln("node {}: left {} right {} parent {} factor {} counter
{}", node->data, (node->left?node->left->data:data_t()), (node->right?node->right->data:data_t()),
            (node->parent.expired()?data_t():node->parent.lock()->data), node->factor, node->counter);
    });
  }
#endif
 private:
#ifdef ENABLE_RAVL_FOREACH_NODE
  void for_each_node(std::function<void(const std::shared_ptr<node_type> &)> func) {
    if(root) {
      root->for_each_node(func);
      func(root);
    }
  }
#endif

     std::pair<std::shared_ptr<const node_type>, int/*0 if found, 1 if at right, -1 if at left, -2 if no root*/> do_find(const data_t &item) const noexcept {
    static auto is_less = data_less_func();
    static auto is_equal = data_equal_func();

    auto curr = root;
    auto return_par = root;
    auto return_pos = -2;
    while(curr) {
      if(is_equal(curr->data, item)) {
        return {curr, 0};
      }
      return_par = curr;
      if(is_less(curr->data, item)) {
        return_pos = 1;
        curr = curr->right;
      }
      else {
        return_pos = -1;
        curr = curr->left;
      }
    }
    return {return_par, return_pos};
  }
    std::pair<std::shared_ptr<node_type>, int> do_find(const data_t &item)
```

```
noexcept {
            std::pair<std::shared_ptr<const node_type>, int> &&res =
static_cast<const this_type *>(this)->do_find(item);
      return std::make_pair(std::const_pointer_cast<node_type>(res.first),
res.second);
  };

  template <bool is_inserting>
  int/*Return: grown_height*/ rebalance(std::shared_ptr<node_type> node, int
my_factor_before_insert/*not used if |node.factor| == 2*/) {
   int grown_up = abs(node->factor) - abs(my_factor_before_insert);
    if((is_inserting && grown_up < 0) || (!is_inserting && grown_up > 0))
grown_up = 0;

   if(abs(node->factor) < 2) {
    if(grown_up != 0 && !node->parent.expired()) {
     auto par = node->parent.lock();
     auto par_factor_backup = par->factor;
     par->factor += grown_up * (node == par->right ? 1 : -1); // +-1 * +-1
     rebalance<is_inserting>(par, par_factor_backup);
    }
    return grown_up;
   }
   else if (abs(node->factor) == 2) {
    if(node->factor > 0) { // my factor is 2
     auto child = node->right;
     int w = child->factor;
     if(w == -1) {
      int w2 = child->left->factor;
      child->left->factor = w2 == 1 ? 1 : 0;
      child->factor = w2 == -1 ? 1 : 0;
      node->factor = w2 == 1 ? -1 : 0;
      rotate_to_left_2(node,child);
      return 0;
     }
     rotate_to_left(node, child);
     node->factor = 1-w;
     child->factor = w-1;

     if(abs(node->factor) == 2) {
         rebalance<is_inserting>(node, 2); // Always return 0 because its
factor is +-2
     }
     return 0;
```

```
      }
      else { // my factor is -2
        auto child = node->left;
        int w = child->factor;
        if(w == 1) {
          int w2 = child->right->factor;
          child->right->factor = w2 == -1 ? -1 : 0;
          child->factor = w2 == 1 ? -1 : 0;
          node->factor = w2 == -1 ? -1 : 0;
          rotate_to_right_2(node,child);
          return 0; // Warn: not carefully checked.
        }
        node->factor = -1-w;
        child->factor = 1+w; // 0 1
        rotate_to_right(node, child);

        if(abs(node->factor) == 2)
            rebalance<is_inserting>(node, -2); // Always return 0 because its
factor is +-2
        return 0;
      }
    }
    else
        throw std::runtime_error("There's a node whose factor is greater than
2.");
  }

  //rotate without editing any bal factors.
                  void   rotate_to_left(std::shared_ptr<node_type>   higher,
std::shared_ptr<node_type> lower) {
    higher->right = lower->left;
    if(lower->left) lower->left->parent = higher;

    lower->left = higher;
    lower->parent = higher->parent;
    if(!higher->parent.expired()) {
      auto par = higher->parent.lock();
      if(higher == par->right) par->right = lower;
      else par->left = lower;
    }
    else //parent is root
      root = lower;
    higher->parent = lower;
  }
```

```
                  void   rotate_to_right(std::shared_ptr<node_type>   higher,
std::shared_ptr<node_type> lower) {
    higher->left = lower->right;
    if(lower->right) lower->right->parent = higher;

    lower->right = higher;
    lower->parent = higher->parent;
    if(!higher->parent.expired()) {
     auto par = higher->parent.lock();
     if(higher == par->right) par->right = lower;
     else par->left = lower;
    }
    else //parent is root
     root = lower;
    higher->parent = lower;
   }
                 void   rotate_to_left_2(std::shared_ptr<node_type>   higher,
std::shared_ptr<node_type> lower) {
    auto lower2 = lower->left;
    higher->right = lower2->left;
    if(lower2->left) lower2->left->parent = higher;
    lower->left = lower2->right;
    if(lower2->right) lower2->right->parent = lower;

    lower2->parent = higher->parent;
    if(!higher->parent.expired()) {
     auto par = higher->parent.lock();
     if(higher == par->right) par->right = lower2;
     else par->left = lower2;
    }
    else
     root = lower2;

    lower2->right = lower;
    lower2->left = higher;
    lower->parent = lower2;
    higher->parent = lower2;
   }
                void   rotate_to_right_2(std::shared_ptr<node_type>   higher,
std::shared_ptr<node_type> lower) {
    auto lower2 = lower->right;
    higher->left = lower2->right;
    if(lower2->right) lower2->right->parent = higher;
    lower->right = lower2->left;
```

```
      if(lower2->left) lower2->left->parent = lower;

      lower2->parent = higher->parent;
      if(!higher->parent.expired()) {
        auto par = higher->parent.lock();
        if(higher == par->right) par->right = lower2;
        else par->left = lower2;
      }
      else
        root = lower2;

      lower2->left = lower;
      lower2->right = higher;
      lower->parent = lower2;
      higher->parent = lower2;
    }

    std::shared_ptr<node_type> root;
    size_t m_size = 0;
  };

}

#endif
// --- lib/cpp-readline

cat: lib/cpp-readline: 是一个目录
// --- lib/gc

cat: lib/gc: 是一个目录
// --- lib/json

cat: lib/json: 是一个目录
// --- lib/rlib

cat: lib/rlib: 是一个目录
// --- lib/set.hpp
#ifndef R_SET_
#define R_SET_

#include <avl.hpp>
#include <algorithm>

namespace r
```

```
{
    template<typename T, bool __multi = false, typename _equal_func = std::equal_to<T>, typename _less_func = std::less<T>>
 class set : public avl::tree<T, __multi, _equal_func, _less_func>
 {
 public:
  using this_type = set<T, __multi>;
  using base_type = avl::tree<T, __multi, _equal_func, _less_func>;

  this_type operator^(const this_type &another) const noexcept
  {
   this_type result;
   auto curr_a = base_type::cbegin();
   auto curr_b = another.cbegin();
   static auto is_less = _less_func();
   static auto is_equal = _equal_func();

   while (curr_a && curr_b)
   {
    if (is_equal(*curr_a, *curr_b))
    {
     if (!__multi)
      result.insert(*curr_a);
     else
     {
      size_t shift_time = base_type::count(*curr_a);
      size_t insert_time = std::min(shift_time, another.count(*curr_a));
      for (size_t cter = 0; cter < insert_time; ++cter)
       result.insert(*curr_a);
      for (size_t cter = 1; cter < shift_time; ++cter)
       ++curr_a;
     }
     ++curr_a;
     ++curr_b;
     continue;
    }
    if (is_less(*curr_a, *curr_b))
     ++curr_a;
    else
     ++curr_b;
   }
   return std::move(result);
  }
  this_type operator+(const this_type &another) const noexcept
  {
   this_type result;
   for (auto iter = base_type::cbegin(); iter; ++iter)
   {
    result.insert(*iter, true);
   }
   for (auto iter = another.cbegin(); iter; ++iter)
   {
    result.insert(*iter, true);
   }
   return std::move(result);
  }

  this_type operator-(const this_type &another) const noexcept
  {
   this_type result;
   for (auto iter = base_type::cbegin(); iter; ++iter)
   {
    if (!__multi)
    {
     if (!another.exist(*iter))
      result.insert(*iter);
    } else
    {
     size_t shift_time = base_type::count(*iter);
     int insert_time = shift_time - another.count(*iter);
     for (int cter = 0; cter < insert_time; ++cter)
      result.insert(*iter);
     for (size_t cter = 1; cter < shift_time; ++cter)
      ++iter;
    }
   }
   return std::move(result);
  }

  this_type clone() const noexcept
  {
   this_type result;
    std::for_each(base_type::cbegin(), base_type::cend(), [&result](const T
&dat) { result.insert(dat); });
   return std::move(result);
  }
  bool operator==(const this_type &another) const noexcept
  {
    if (base_type::size() != another.size())
      return false;
      for (auto itera = base_type::cbegin(), iterb = another.cbegin(); itera
&& iterb; ++itera, ++iterb)
    {
      if (!_equal_func()(*itera, *iterb))
        return false;
    }
    return true;
  }

  bool operator!=(const this_type &another) const noexcept
  {
    return !operator==(another);
  }

  bool contain(const this_type &another) const noexcept
  {
    for (auto iter = another.cbegin(); iter; ++iter)
    {
          if (base_type::count(*iter) < another.count(*iter)) //Warning:
Unnecessary `another.count` is O(log(n)) slow.
        return false;
    }
    return true;
  }

  void show(const std::string &name) const noexcept
  {
    rlib::printf("{}: ", name);
    show();
  }
  void show() const noexcept
  {
    std::for_each(base_type::cbegin(), base_type::cend(), [](const int &i) {
rlib::print(i, ""); });
    rlib::println();
  }
  constexpr const this_type &crange() const noexcept {
    return *this;
  }
 };
```

```
    template<typename T, typename _equal_func = std::equal_to<T>, typename
_less_func = std::less<T>>
  class multiset : public set<T, true, _equal_func, _less_func>
  {};

}
#endif
// --- lib/rlib/class_decorator.hpp
#ifndef RLIB_CLASS_DECO_HPP_
#define RLIB_CLASS_DECO_HPP_

#include <rlib/require/cxx11>

namespace rlib {
  namespace _noncp_ {
    class noncopyable
    {
    public:
      noncopyable() = default;
      ~noncopyable() = default;
      noncopyable(const noncopyable &) = delete;
      noncopyable &operator=(const noncopyable &) = delete;
    };
  }
  typedef _noncp_::noncopyable noncopyable;
}

namespace rlib {
  namespace _nonmv_ {
    class nonmovable : private noncopyable
    {
    public:
      nonmovable() = default;
      ~nonmovable() = default;
      nonmovable(const nonmovable &&) = delete;
      nonmovable &operator=(const nonmovable &&) = delete;
    };
  }
  typedef _nonmv_::nonmovable nonmovable;
}

namespace rlib {
  namespace _nonconstructible_ {
   class nonconstructible : private rlib::nonmovable
   {
   public:
     nonconstructible() = delete;
     ~nonconstructible() = delete;
   };
  }
  typedef _nonconstructible_::nonconstructible nonconstructible;
  typedef nonconstructible static_class;
}

#endif// --- lib/rlib/c-with-class.h
#ifndef RLIB_CWITHCLASS_H_
#define RLIB_CWITHCLASS_H_

//TODO: clean namespace.
//TODO: use macro to type class_name only once.
//#error c_with_class not completed yet

#define     RCPP_NEW(type,name,constructor_arg)     struct     type     name
__attribute__((cleanup(type##_rcpp_destructor)));type##_rcpp_constructor(&na
me,constructor_arg)
#define               RCPP_CALL(i_objectname,i_funcname,               ...)
i_objectname.i_funcname(&i_objectname, ##__VA_ARGS__) //ONLY static public
function can be called directly!!!
#define     RCPP_PCALL(p_objectname,i_funcname,     ...)     p_objectname-
>i_funcname(p_objectname, ##__VA_ARGS__)

#define RCPP_CLASS_DECL(class_name) struct class_name;
#define RCPP_CLASS_METHOD_DECL_1(class_name, method_name, return_type, ...)
typedef return_type (* class_name##method_name##_rcpp_t)(struct class_name
*this, ##__VA_ARGS__); //VAARGS is `int arg1, float arg2, ...`
#define RCPP_CLASS_BEGIN(class_name) struct class_name {
#define           RCPP_CLASS_METHOD_DECL_2(class_name,           method_name)
RCPP_CLASS_MEMBER_DECL(class_name##method_name##_rcpp_t, method_name)
#define RCPP_CLASS_MEMBER_DECL(type, name) type name;
#define RCPP_CLASS_END() };
#define RCPP_CLASS_METHOD_IMPL(class_name, method_name, return_type, ...)
return_type class_name##method_name##_rcpp_impl(struct class_name *this,
##__VA_ARGS__) //VAARGS is `int arg1, float arg2, ...`
#define             RCPP_CLASS_CONSTRUCTOR_IMPL(class_name)             void
class_name##_rcpp_constructor(struct class_name *this, void *arg) //TODO:
Register all methods.
#define     RCPP_CLASS_METHOD_REGISTER(class_name,     method_name)     this-
```

```
>method_name = &class_name##method_name##_rcpp_impl;
#define              RCPP_CLASS_DESTRUCTOR_IMPL(class_name)              void
class_name##_rcpp_destructor(struct class_name *this)

#endif
// --- lib/rlib/functional.hpp
#ifndef RLIB_FUNCTIONAL_HPP_
#define RLIB_FUNCTIONAL_HPP_

#include <rlib/require/cxx14>
#include <rlib/class_decorator.hpp>

#include <type_traits>
#include <list>
#include <functional>
#include <chrono>

namespace rlib {
  template <class operation_t, typename ... args_t>
   static inline double timed_func(::std::function<operation_t> f, args_t ...
args)
  {
   auto begin = std::chrono::high_resolution_clock::now();
   f(args ...);
   auto end = std::chrono::high_resolution_clock::now();
   return ::std::chrono::duration<double>(end - begin).count();
  }

  template <class operation_t, typename ... args_t>
    static  inline  typename  ::std::result_of<operation_t(args_t  ...)>::type
repeat(size_t count, operation_t f, args_t ... args)
  {
   for(size_t cter = 0; cter < count - 1; ++cter)
    f(args ...);
   return ::std::move(f(args ...));
  }
  template <class operation_t, typename ... args_t>
                                                                        static
inline ::std::list<typename ::std::result_of<operation_t(args_t ...)>::type>
repeat_and_return_list(size_t count, operation_t f, args_t ... args)
  {
   ::std::list<typename ::std::result_of<operation_t(args_t ...)>::type> ret;
   for(size_t cter = 0; cter < count; ++cter)
    ret.push_back(std::move(f(args ...)));
   return std::move(ret);
  }
}
#endif
// --- lib/rlib/librr.cc
namespace rlib {
  bool enable_endl_flush = true;
}// --- lib/rlib/LICENSE
MIT License

Copyright (c) 2017 Recolic Keghart

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to
deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE
SOFTWARE.
// --- lib/rlib/macro.hpp
#ifndef R_MACRO_HPP
#define R_MACRO_HPP

#ifndef MACRO_DECAY
#define MACRO_DECAY(m) (m)
#endif

#ifndef _R_MACRO_ENSTRING
#define _R_MACRO_ENSTRING(_s) #_s
#endif
```

```
#ifndef MACRO_TO_CSTR
#define MACRO_TO_CSTR(m) _R_MACRO_ENSTRING(m)
#endif

#ifndef MACRO_EQL
#define MACRO_EQL(a, b) (MACRO_TO_CSTR(a) == MACRO_TO_CSTR(b))
#endif

#ifndef MACRO_CAT
#define MACRO_CAT(a, b) _MACRO_CAT_I(a, b)
#define _MACRO_CAT_I(a, b) _MACRO_CAT_II(~, a ## b)
#define _MACRO_CAT_II(p, res) res
#endif
#ifndef MAKE_UNIQUE_NAME
#define MAKE_UNIQUE_NAME(base) MACRO_CAT(base, __COUNTER__)
#endif

#endif
// --- lib/rlib/Makefile
CXX ?= g++
CC ?= gcc
AR ?= ar
CXXFLAGS = -O3
CFLAGS =
ARFLAGS = rcs

def: compile_library

compile_library:
	$(CXX) $(CXXFLAGS) -c libr.cc -o libr.o
	$(AR) $(ARFLAGS) libr.a libr.o

install_header:
	[ ! -d /usr/include/rlib ] || rm -rf /usr/include/rlib
	cp -r . /usr/include/rlib
	rm -rf /usr/include/rlib/test /usr/include/rlib/.git

install_library: compile_library
	cp libr.a /usr/lib/

install: install_header install_library

uninstall:
	rm -rf /usr/include/rlib
	rm /usr/lib/librlib.a

clean:
	rm *.o *.a
// --- lib/rlib/opt.hpp
/*
This opt_parser works well for correct cmd args,
but not guaranteed to works well in all condition
(for example, some ill formed argument).

It's possible to read wrong information rather than
raise an exception on some rare ill formed arguments.
*/
#ifndef R_OPT_HPP
#define R_OPT_HPP

#include <rlib/require/cxx14>
#include <rlib/class_decorator.hpp>
#include <rlib/string/string.hpp>
#include <rlib/scope_guard.hpp>

#include <string>
#include <vector>
#include <algorithm>
#include <stdexcept>

namespace rlib {
  class opt_parser : private noncopyable
  {
  public:
    opt_parser() = delete;
    opt_parser(size_t arglen, char **argv) {
      for(size_t cter = 1; cter < arglen; ++cter)
        args.push_back(std::move(std::string(argv[cter])));
    }

    std::string getValueArg(const std::string &argName, bool required = false)
    { //If required argument not exist, I'll throw. Else, return "" if arg is
not read.
      bool useEqualSym = false;
          auto pos = std::find_if(args.cbegin(), args.cend(), [&](auto &ele)-
>bool{
        if(ele == argName) return true;
        if(ele.size() > argName.size() && ele.substr(0, argName.size()+1) ==
argName + "=") {
       useEqualSym = true;
       return true;
      }
      return false;
    });
    if(required && pos == args.cend())
        throw std::invalid_argument(format_string("Required argument '{}' not
provided.", argName));
    if(pos == args.cend())
      return std::move(std::string(""));
    defer(([&, pos]{if(!useEqualSym) args.erase(pos+1); args.erase(pos);}));
    if(useEqualSym)
      return std::move(pos->substr(argName.size() + 1));
    else
    {
      if(++pos == args.cend())
        throw std::invalid_argument(format_string("Argument '{}' must provide
value.", argName));
      return *pos;
    }
  }

          std::string getValueArg(const std::string &argName, const char
*pAnotherCStr)
   { //getValueArg("--long", "-l") may be converted to getValueArg("--long",
true).
    return std::move(getValueArg(argName, pAnotherCStr, false));
  }

  bool getBoolArg(const std::string &argName)
  { //Return if it's defined.
    auto pos = std::find(args.cbegin(), args.cend(), argName);
    if(pos == args.cend()) return false;
    args.erase(pos);
    return true;
  }

     std::string getValueArg(const std::string &longName, const std::string
&shortName, bool required = false)
  {
    std::string valueL = getValueArg(longName);
    std::string valueS = getValueArg(shortName);
    std::string value = valueL.empty() ? valueS : valueL;
    if(required && value.empty())
        throw std::invalid_argument(format_string("Required argument '{}/{}'
not provided.", longName, shortName));
    return value;
   }

   bool getBoolArg(const std::string &longName, const std::string &shortName)
   {
    return getBoolArg(longName) || getBoolArg(shortName);
   }

   bool allArgDone() const
   {
    return args.empty();
   }
  private:
   std::vector<std::string> args;
  };
}

#endif
// --- lib/rlib/README.md
# rlib

Here is recolic's private library...
// --- lib/rlib/require

cat: lib/rlib/require: 是一个目录
// --- lib/rlib/scope_guard_buffer.hpp
/*
scope_guards scope_exit, scope_fail;

action1();
scope_exit += [](){ cleanup1(); };
scope_fail += [](){ rollback1(); };

action2();
scope_exit += [](){ cleanup2(); };
scope_fail += [](){ rollback2(); };

//...
```

```
scope_fail.dismiss();
*/

#ifndef R_SCOPE_GUARD_BUFFER_HPP
#define R_SCOPE_GUARD_BUFFER_HPP

#include <rlib/require/cxx11>
#include <functional>
#include <deque>
#include <rlib/class_decorator.hpp>

namespace rlib {
    class  scope_guards  :  public  std::deque<std::function<void()>>,  private
noncopyable
  {
  public:
    template<class Callable>
    scope_guards& operator += (Callable && undo_func) {
      emplace_front(std::forward<Callable>(undo_func));
    }

    ~scope_guards() {
      for(auto &f : *this) f(); // must not throw
    }

    void dismiss() noexcept {
      clear();
    }
  };
}

#endif
// --- lib/rlib/scope_guard.hpp
/* Exception safe usage:
*
* reinforce_scope_begin(_gname, [](){do_sth();})
* 1+1;
* 1+1=2;
* 2+2=4;
* reinforce_scope_end(_gname)
*
*/

#ifndef R_SCOPE_GUARD
```

```
#define R_SCOPE_GUARD

#include <rlib/require/cxx11>
#include <functional>
#include <rlib/class_decorator.hpp>

namespace rlib {
 class scope_guard : private noncopyable
 {
 public:
  template<class Callable>
   scope_guard(Callable && undo_func) : f(std::forward<Callable>(undo_func))
{}

  scope_guard(scope_guard && other) : f(std::move(other.f)) {
   other.f = nullptr;
  }

  ~scope_guard() {
   if(f) f(); // must not throw
  }

  void dismiss() noexcept {
   f = nullptr;
  }

  void force_call() noexcept {
   if(f) f();
   dismiss();
  }

 private:
  std::function<void()> f;
 };
}


#ifndef defer
#include <rlib/macro.hpp>
#define  defer(callable)  ::rlib::scope_guard  MAKE_UNIQUE_NAME(_guarder_id_)
(callable)
#endif

#define reinforce_scope_begin(guarderName, callable) scope_guard guarderName
```

```
= callable; try{
#define         reinforce_scope_end(guarderName)         }         catch(...)
{ guarderName.force_call(); throw;}


#endif
// --- lib/rlib/stdio.hpp
#ifndef R_STDIO_HPP
#define R_STDIO_HPP

#include <rlib/require/cxx11>
// Must link libr.a
#include <string>
#include <iostream>
#include <rlib/string/string.hpp>

namespace rlib {
 template<typename PrintFinalT>
 void print(PrintFinalT reqArg);
 template<typename Required, typename... Optional>
 void print(Required reqArgs, Optional... optiArgs);
 template<typename... Optional>
 void println(Optional... optiArgs);
 void println();

 template<typename Iterable, typename Printable>
 void print_iter(Iterable arg, Printable spliter);
 template<typename Iterable, typename Printable>
 void println_iter(Iterable arg, Printable spliter);
 template<typename Iterable>
 void print_iter(Iterable arg);
 template<typename Iterable>
 void println_iter(Iterable arg);

 template<typename... Args>
 size_t printf(const std::string &fmt, Args... args);
 template<typename... Args>
 size_t printfln(const std::string &fmt, Args... args);

 inline std::string scanln()
 {
  ::std::string line;
  ::std::getline(::std::cin, line);
  return std::move(line);
 }

// Implements.
 extern bool enable_endl_flush;
 template< class CharT, class Traits >
  std::basic_ostream<CharT, Traits>& endl(std::basic_ostream<CharT, Traits>&
os) {
   os << '\n';
   if(enable_endl_flush)
    os.flush();
   return os;
 }

 template<typename PrintFinalT>
 void print(PrintFinalT reqArg)
 {
  ::std::cout << reqArg;
 }
 template<typename Required, typename... Optional>
 void print(Required reqArgs, Optional... optiArgs)
 {
  ::std::cout << reqArgs << ' ';
  print(optiArgs ...);
 }
 template<typename... Optional>
 void println(Optional... optiArgs)
 {
  print(optiArgs ...);
  println();
 }
 inline void println()
 {
  ::std::cout << ::rlib::endl;
 }

 template<typename Iterable, typename Printable>
 void print_iter(Iterable arg, Printable spliter)
 {
  for(const auto & i : arg)
   ::std::cout << i << spliter;
 }
 template<typename Iterable, typename Printable>
 void println_iter(Iterable arg, Printable spliter)
 {
   print_iter(arg, spliter);
   ::std::cout << ::rlib::endl;
 }
 template<typename Iterable>
 void print_iter(Iterable arg)
 {
   for(const auto & i : arg)
     ::std::cout << i << ' ';
 }
 template<typename Iterable>
 void println_iter(Iterable arg)
 {
   print_iter(arg);
   ::std::cout << ::rlib::endl;
 }

 template<typename... Args>
 size_t printf(const std::string &fmt, Args... args)
 {
   std::string to_print = format_string(fmt, args...);
   ::std::cout << to_print;
   return to_print.size();
 }
 template<typename... Args>
 size_t printfln(const std::string &fmt, Args... args)
 {
   size_t len = ::rlib::printf(fmt, args...);
   ::std::cout << ::rlib::endl;
   return len + 1;
 }
}


#endif
// --- lib/rlib/string

cat: lib/rlib/string: 是一个目录
// --- lib/rlib/sys

cat: lib/rlib/sys: 是一个目录
// --- lib/rlib/terminal.hpp
#ifndef R_STD_COLOR_HPP
#define R_STD_COLOR_HPP
```

```
#include <rlib/require/cxx11>
#include <rlib/sys/os.hpp>

#include <iostream>
#include <string>
#include <stdexcept>
#include <exception>
using std::string;
using std::basic_ostream;

namespace rlib::terminal {
  enum class color_t {color_unset = 10, black = 0, red, green, brown, blue,
magenta, cyan, lightgray};
   enum class font_t {font_unset = 0, bold = 1, underline = 4, dark = 2,
background = 7, striked = 9}; //Edit line53 if (int)font_t may >= 10 !!
  class clear_t {} clear;

  class fontInfo
  {
  public:
    fontInfo(color_t text_color) : textColor(text_color) {}
    fontInfo(font_t font_type) : fontType(font_type) {}
      fontInfo(color_t text_color, font_t font_type) : textColor(text_color),
fontType(font_type) {}
    fontInfo(const clear_t &) : clear(true) {}
    fontInfo() = default;
    string toString() const
    {
      if(rlib::OSInfo::os == rlib::OSInfo::os_t::WINDOWS)
        return std::move(std::string());
      else
         return std::move(clear ? std::string("\033[0m") : (color_to_string() +
font_to_string()));
    }
  private:
    color_t textColor = color_t::color_unset;
    font_t fontType = font_t::font_unset;
    bool clear = false;
  private:
    constexpr static int color_to_int(const color_t &_ct)
    {
      return static_cast<int>(_ct);
    }
    constexpr static int font_to_int(const font_t &_ft)
    {
      return static_cast<int>(_ft);
    }
    constexpr static char color_to_char(const color_t &_ct)
    {
       return _ct == color_t::color_unset ? '\0' : '0' + color_to_int(_ct);
//Return '\0' if unset.
    }
    constexpr static char font_to_char(const font_t &_ft)
    {
      return _ft == font_t::font_unset ? '\0' :'0' + font_to_int(_ft);
    }
    string color_to_string() const
    {
      if(textColor == color_t::color_unset)
        return std::move(std::string());
      char toret[] = "\033[3?m";
      toret[3] = color_to_char(textColor);
      return std::move(std::string(toret));
    }
    string font_to_string() const
    {
      if(fontType == font_t::font_unset)
        return std::move(std::string());
      char toret[] = "\033[?m";
      toret[2] = font_to_char(fontType);
      return std::move(std::string(toret));
    }
  };

    struct _rosi_font {_rosi_font(const fontInfo &_ref_fi) : _ref_fi(_ref_fi)
{} const fontInfo &_ref_fi;};
  inline _rosi_font setfont(const fontInfo &__fi) {return _rosi_font(__fi);}

  template<typename _CharT, typename _Traits>
    inline basic_ostream<_CharT, _Traits>&
    operator<<(basic_ostream<_CharT, _Traits>& __os, const fontInfo &__f)
    {
      __os << __f.toString();
      return __os;
    }

  template<typename _CharT, typename _Traits>
    inline basic_ostream<_CharT, _Traits>&
```

```
    operator<<(basic_ostream<_CharT, _Traits>& __os, _rosi_font __rosi_f)
    {
      const fontInfo &__f = __rosi_f._ref_fi;
      return operator<<<_CharT, _Traits>(__os, __f);
    }
}
#endif
// --- lib/rlib/test

cat: lib/rlib/test: 是一个目录
// --- lib/rlib/traits.hpp
#ifndef RLIB_TRAITS_HPP
#define RLIB_TRAITS_HPP

#include <type_traits>

namespace rlib {
  template<typename T>
  struct is_callable_helper {
  private:
    typedef char(&yes)[1];
    typedef char(&no)[2];

    struct Fallback { void operator()(); };
    struct Derived : T, Fallback { };

    template<typename U, U> struct Check;

    template<typename>
    static yes test(...);

    template<typename C>
    static no test(Check<void (Fallback::*)(), &C::operator()>*);

  public:
    static const bool value = sizeof(test<Derived>(0)) == sizeof(yes);
  };
  template<typename T>
  struct is_callable
    : std::conditional<
      std::is_class<T>::value,
      is_callable_helper<T>,
      std::is_function<T>>::type
  {};
```

```
}

#endif
// --- lib/rlib/sys/cc_codegen.py
#!/bin/env python3

def genDefList(idarr):
 s = '#if'
 cter = 1
 for i in idarr:
  s += ' defined(' + i + ')'
  if cter ≠ len(idarr):
   s += ' ||'
  cter += 1
 return s

print('// Generated by cc_codegen.py. Do not edit it by hand.')

with open("cc_list") as fd:
 osarr=fd.read().split('\n')
 for i in osarr:
  if i == '':
   continue
  iarr=i.split(' ')
  if len(iarr) < 2:
   continue
  print('#ifndef RLIB_COMPILER_ID')
  print(genDefList(iarr[:-1:]))
  print('#define RLIB_COMPILER_ID', iarr[-1])
  print('#endif')
  print('#endif')
  print('')

print('#ifndef RLIB_COMPILER_ID')
print('#define RLIB_COMPILER_ID UNKNOWN')
print('#endif')

// --- lib/rlib/sys/cc_list
_ACC_ ACC
__CMB__ ALTIUM_MICROBLAZE
__CHC__ ALTIUM_HARDWARE
__ACK__ AMSTERDAM
__CC_ARM ARMCC
AZTEC_C __AZTEC_C__ AZTEC
```

```
__BORLANDC__ __CODEGEARC__ BORLAND
__CC65__ CC65
__clang__ CLANG
__COMO__ COMEAU
__DECC __DECCXX COMPAQ
__convexc__ CONVEX
__COMPCERT__ COMPCERT
__COVERITY__ COVERITY
_CRAYC CRAY
__DCC__ DIAB
_DICE DICE
__DMC__ DIGITAL_MARS
__SYSC__ DIGNUS
__DJGPP__ DJGPP
__EDG__ EDG
__PATHCC__ EKOPATH
__FCC_VERSION FUJITSU
__ICC __INTEL_COMPILER ICC
__GNUC__ GCC
__ghs__ GREENHILL
__HP_cc HPC
__HP_aCC HPACXX
__IAR_SYSTEMS_ICC__ IARC
__IBMCPP__ __IBMC__ IBMC
__IMAGECRAFT__ IMAGECRAFT
__INTEL_COMPILER __ICL INTELC
__KCC KAICXX
__CA__ __KEIL__ KEIL_CARM
__C166__ KEIL_C166
__C51__ __CX51__ KEIL_C51
__LCC__ LCC
__llvm__ LLVM
__MWERKS__ __CWCC__ METROWERKS
_MSC_VER MSVC
_MRI MICROTEC
__NDPC__ __NDPX__ MICROWAY
__sgi sgi MIPSPRO
MIRACLE MIRACLE
__MRC__ MPW_C MPW_CPLUS MPW
__CC_NORCROFT NORCROFT
__NWCC__ NWCC
__OPEN64__ __OPENCC__ OPEN64
ORA_PROC ORACLE_PROC
__SUNPRO_C __SUNPRO_CC SOLARIS
```

```
__PACIFIC__ PACIFIC
_PACC_VER PLAM
__POCC__ PELLES
__PGI PORTLAND
__RENESAS__ __HITACHI__ RENESAS
SASC __SASC __SASC__ SASC
_SCO_DS SCO_OPENSERVER
SDCC SDCC
__SNC__ SN
__VOSC__ STRATUS_VOS
__SC__ SYMANTEC
__TenDRA__ TENDRA
__TI_COMPILER_VERSION__ _TMS320C6X TEXAS
THINKC3 THINKC4 THINK
__TINYC__ TINYC
__TURBOC__ TURBOC
_UCC UCC
__USLC__ USLC
__VBCC__ VBCC
__WATCOMC__ WATCOM
__ZTC__ ZORTECH
// --- lib/rlib/sys/compiler_detector
// Generated by cc_codegen.py. Do not edit it by hand.
#ifndef RLIB_COMPILER_ID
#if defined(_ACC_)
#define RLIB_COMPILER_ID ACC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__CMB__)
#define RLIB_COMPILER_ID ALTIUM_MICROBLAZE
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__CHC__)
#define RLIB_COMPILER_ID ALTIUM_HARDWARE
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__ACK__)
#define RLIB_COMPILER_ID AMSTERDAM
```

```
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__CC_ARM)
#define RLIB_COMPILER_ID ARMCC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(AZTEC_C) || defined(__AZTEC_C__)
#define RLIB_COMPILER_ID AZTEC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__BORLANDC__) || defined(__CODEGEARC__)
#define RLIB_COMPILER_ID BORLAND
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__CC65__)
#define RLIB_COMPILER_ID CC65
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__clang__)
#define RLIB_COMPILER_ID CLANG
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__COMO__)
#define RLIB_COMPILER_ID COMEAU
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__DECC) || defined(__DECCXX)
#define RLIB_COMPILER_ID COMPAQ
#endif
#endif
```

```
#ifndef RLIB_COMPILER_ID
#if defined(__convexc__)
#define RLIB_COMPILER_ID CONVEX
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__COMPCERT__)
#define RLIB_COMPILER_ID COMPCERT
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__COVERITY__)
#define RLIB_COMPILER_ID COVERITY
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(_CRAYC)
#define RLIB_COMPILER_ID CRAY
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__DCC__)
#define RLIB_COMPILER_ID DIAB
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(_DICE)
#define RLIB_COMPILER_ID DICE
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__DMC__)
#define RLIB_COMPILER_ID DIGITAL_MARS
#endif
#endif

#ifndef RLIB_COMPILER_ID
```

```
#if defined(__SYSC__)
#define RLIB_COMPILER_ID DIGNUS
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__DJGPP__)
#define RLIB_COMPILER_ID DJGPP
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__EDG__)
#define RLIB_COMPILER_ID EDG
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__PATHCC__)
#define RLIB_COMPILER_ID EKOPATH
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__FCC_VERSION)
#define RLIB_COMPILER_ID FUJITSU
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__ICC) || defined(__INTEL_COMPILER)
#define RLIB_COMPILER_ID ICC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__GNUC__)
#define RLIB_COMPILER_ID GCC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__ghs__)
#define RLIB_COMPILER_ID GREENHILL
```

```
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__HP_cc)
#define RLIB_COMPILER_ID HPC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__HP_aCC)
#define RLIB_COMPILER_ID HPACXX
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__IAR_SYSTEMS_ICC__)
#define RLIB_COMPILER_ID IARC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__IBMCPP__) || defined(__IBMC__)
#define RLIB_COMPILER_ID IBMC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__IMAGECRAFT__)
#define RLIB_COMPILER_ID IMAGECRAFT
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__INTEL_COMPILER) || defined(__ICL)
#define RLIB_COMPILER_ID INTELC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__KCC)
#define RLIB_COMPILER_ID KAICXX
#endif
#endif
```

```
#ifndef RLIB_COMPILER_ID
#if defined(__CA__) || defined(__KEIL__)
#define RLIB_COMPILER_ID KEIL_CARM
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__C166__)
#define RLIB_COMPILER_ID KEIL_C166
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__C51__) || defined(__CX51__)
#define RLIB_COMPILER_ID KEIL_C51
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__LCC__)
#define RLIB_COMPILER_ID LCC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__llvm__)
#define RLIB_COMPILER_ID LLVM
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__MWERKS__) || defined(__CWCC__)
#define RLIB_COMPILER_ID METROWERKS
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(_MSC_VER)
#define RLIB_COMPILER_ID MSVC
#endif
#endif

#ifndef RLIB_COMPILER_ID
```

```
#if defined(_MRI)
#define RLIB_COMPILER_ID MICROTEC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__NDPC__) || defined(__NDPX__)
#define RLIB_COMPILER_ID MICROWAY
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__sgi) || defined(sgi)
#define RLIB_COMPILER_ID MIPSPRO
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(MIRACLE)
#define RLIB_COMPILER_ID MIRACLE
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__MRC__) || defined(MPW_C) || defined(MPW_CPLUS)
#define RLIB_COMPILER_ID MPW
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__CC_NORCROFT)
#define RLIB_COMPILER_ID NORCROFT
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__NWCC__)
#define RLIB_COMPILER_ID NWCC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__OPEN64__) || defined(__OPENCC__)
#define RLIB_COMPILER_ID OPEN64
```

```
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(ORA_PROC)
#define RLIB_COMPILER_ID ORACLE_PROC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__SUNPRO_C) || defined(__SUNPRO_CC)
#define RLIB_COMPILER_ID SOLARIS
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__PACIFIC__)
#define RLIB_COMPILER_ID PACIFIC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(_PACC_VER)
#define RLIB_COMPILER_ID PLAM
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__POCC__)
#define RLIB_COMPILER_ID PELLES
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__PGI)
#define RLIB_COMPILER_ID PORTLAND
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__RENESAS__) || defined(__HITACHI__)
#define RLIB_COMPILER_ID RENESAS
#endif
#endif
```

```
#ifndef RLIB_COMPILER_ID
#if defined(SASC) || defined(__SASC) || defined(__SASC__)
#define RLIB_COMPILER_ID SASC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(_SCO_DS)
#define RLIB_COMPILER_ID SCO_OPENSERVER
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(SDCC)
#define RLIB_COMPILER_ID SDCC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__SNC__)
#define RLIB_COMPILER_ID SN
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__VOSC__)
#define RLIB_COMPILER_ID STRATUS_VOS
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__SC__)
#define RLIB_COMPILER_ID SYMANTEC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__TenDRA__)
#define RLIB_COMPILER_ID TENDRA
#endif
#endif

#ifndef RLIB_COMPILER_ID
```

```
#if defined(__TI_COMPILER_VERSION__) || defined(_TMS320C6X)
#define RLIB_COMPILER_ID TEXAS
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(THINKC3) || defined(THINKC4)
#define RLIB_COMPILER_ID THINK
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__TINYC__)
#define RLIB_COMPILER_ID TINYC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__TURBOC__)
#define RLIB_COMPILER_ID TURBOC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(_UCC)
#define RLIB_COMPILER_ID UCC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__USLC__)
#define RLIB_COMPILER_ID USLC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__VBCC__)
#define RLIB_COMPILER_ID VBCC
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__WATCOMC__)
#define RLIB_COMPILER_ID WATCOM
```

```
#endif
#endif

#ifndef RLIB_COMPILER_ID
#if defined(__ZTC__)
#define RLIB_COMPILER_ID ZORTECH
#endif
#endif

#ifndef RLIB_COMPILER_ID
#define RLIB_COMPILER_ID UNKNOWN
#endif
// --- lib/rlib/sys/fdset.hpp
#ifndef R_FDSET_HPP
#define R_FDSET_HPP

#include <unistd.h>
#include <sys/types.h>
namespace rlib{
 class FileDescriptorSet
 {
 public:
  using fd=int;
           FileDescriptorSet() : m_size(0), maxFileDescriptor(NULL)
{FD_ZERO(&m_fds_data);}
     void push(fd FileDescriptor) {FD_SET(FileDescriptor, &m_fds_data); +
+m_size; maxFileDescriptor = (maxFileDescriptor > FileDescriptor ?
maxFileDescriptor : FileDescriptor);}
     void pop(fd FileDescriptor) {FD_CLR(FileDescriptor, &m_fds_data); --
m_size;} //It will break maxFileDescriptor.(for performance reason).
  void clear() {FD_ZERO(&m_fds_data); m_size = 0;maxFileDescriptor = 0;}
        bool check(fd FileDescriptor) {return FD_ISSET(FileDescriptor,
&m_fds_data);}
  size_t size() const {return m_size;}
  int getMaxFileDescriptor() const {return maxFileDescriptor;}
  fd_set *getptr() {return &m_fds_data;}
 private:
  fd_set m_fds_data;
  size_t m_size;
  int maxFileDescriptor;
 };
}
#endif
// --- lib/rlib/sys/os.hpp
```

```
#ifndef R_OS_HPP
#define R_OS_HPP

#ifndef RLIB_OS_ID
#if   defined(_Windows)   ||   defined(__WIN32__)   ||   defined(_WIN64)   ||
defined(WIN32)
#   define RLIB_OS_ID WINDOWS
#elif defined(__linux__) || defined(__linux)
#   define RLIB_OS_ID LINUX
#elif defined(__APPLE__)
#   include "TargetConditionals.h"
#   if TARGET_IPHONE_SIMULATOR
#   define RLIB_OS_ID IOS
#   elif TARGET_OS_IPHONE
#   define RLIB_OS_ID IOS
#   elif TARGET_OS_MAC
#   define RLIB_OS_ID MACOS
#   else
#   define RLIB_OS_ID UNKNOWN_UNIX
#   endif
#elif defined(__ANDROID__)
#   define RLIB_OS_ID ANDROID
#elif defined(__unix__) || defined(__unix)
#   define RLIB_OS_ID UNKNOWN_UNIX
#else
#   define RLIB_OS_ID UNKNOWN
#endif
#endif

#include "compiler_detector"
// Define RLIB_COMPILER_ID and RLIB_COMPILER_VER

#if __cplusplus ⩾ 201103L
namespace rlib {
 class OSInfo
 {
 public:
      enum class os_t {UNKNOWN, WINDOWS, LINUX, MACOS, BSD, IOS, ANDROID,
UNKNOWN_UNIX};
    enum class compiler_t {UNKNOWN, GCC, CLANG, MSVC, INTELC, BORLAND, IARC,
SOLARIS, ZAPCC}; //Compiler which not supports cxx1x yet is not listed here.
201708.
   static constexpr os_t os =
 #if defined(RLIB_OS_ID)
 os_t::RLIB_OS_ID;
 #else
 os_t::UNKNOWN;
 #endif
  static constexpr compiler_t compiler =
 #if defined(RLIB_COMPILER_ID)
 compiler_t::RLIB_COMPILER_ID;
 #else
 compiler_t::UNKNOWN;
 #endif
  static constexpr auto compiler_version =
 #if defined(RLIB_COMPILER_VER)
 RLIB_COMPILER_VER;
 #else
 0;
 #endif
 };
}

#endif

#endif
// --- lib/rlib/sys/rwlock.hpp
#ifndef R_SWLOCK_HPP
#define R_SWLOCK_HPP

#include <pthread.h>
namespace rlib {
 class RWLock
 {
 public:
  RWLock() : isFree(true) {pthread_rwlock_init(&m_lock, NULL);}
  ~RWLock() {pthread_rwlock_destroy(&m_lock);}
  void acquireShared() {pthread_rwlock_rdlock(&m_lock);isFree = false;}
  void acquireExclusive() {pthread_rwlock_wrlock(&m_lock);isFree = false;}
  void release() {pthread_rwlock_unlock(&m_lock);isFree = true;}
 //     bool tryAcquireShared() {return pthread_rwlock_tryrdlock(&m_lock) ==
0;}
 //     bool tryAcquireExclusive() {return pthread_rwlock_trywrlock(&m_lock)
== 0;}
 private:
  pthread_rwlock_t m_lock;
  bool isFree;
 };
```

```
}

#endif// --- lib/rlib/sys/sio.hpp
#ifndef R_SIO_HPP
#define R_SIO_HPP

#include <cerrno>
#include <cstdlib>
#include <unistd.h>
#include <string>
#include <stdexcept>

#ifndef WIN32
#include <sys/socket.h>
//POSIX Version
namespace rlib {
 class fdIO
 {
 public:
  static ssize_t readn(int fd, void *vptr, size_t n) noexcept //Return -1 on
error, read bytes on success, blocks until nbytes done.
  {
   size_t  nleft;
   ssize_t nread;
   char   *ptr;

   ptr = (char *)vptr;
   nleft = n;
   while (nleft > 0) {
    if ( (nread = read(fd, ptr, nleft)) < 0) {
     if (errno == EINTR)
      nread = 0;      /* and call read() again */
     else
      return (-1);
    } else if (nread == 0)
     return (-1);              /* EOF */

    nleft -= nread;
    ptr += nread;
   }
   return (n);         /* return success */
  }
      static ssize_t writen(int fd, const void *vptr, size_t n) noexcept
//Return -1 on error, read bytes on success, blocks until nbytes done.
```

```
  {
    size_t nleft;
    ssize_t nwritten;
    const char *ptr;

    ptr = (const char *)vptr;
    nleft = n;
    while (nleft > 0) {
      if ( (nwritten = write(fd, ptr, nleft)) <= 0) {
        if (nwritten < 0 && errno == EINTR)
          nwritten = 0;   /* and call write() again */
        else
          return (-1);    /* error */
       }

       nleft -= nwritten;
       ptr += nwritten;
    }
    return (n);
  }
     static ssize_t readall(int fd, void **pvptr, size_t initSize) noexcept
//Return -1 on error, read bytes on success. pvptr must be a malloc/calloced
buffer, I'll malloc one if *pvptr is NULL.
  {
    size_t current = initSize ? initSize : 1024;
    void *vptr = *pvptr;
    if(vptr == NULL)
      vptr = malloc(current);
    void *currvptr = vptr;

    {
      ssize_t ret = read(fd, currvptr, current / 2);
      if(ret == -1) return -1;
      if(ret < current / 2)
      {
        *pvptr = vptr;
        return ret;
      }
      currvptr = (char *)vptr + current / 2;
    }

    while(true)
    {
      ssize_t ret = read(fd, currvptr, current / 2);
      if(ret == -1) return -1;
      if(ret < current)
      {
       *pvptr = vptr;
       return ret + current / 2;
      }

      current *= 2;
      void *vptrBackup = vptr;
      if((vptr = realloc(vptr, current)) == NULL) {
       free(vptrBackup);
       errno = EMSGSIZE;
       return -1;
      }
      currvptr = (char *)vptr + current / 2;
     }
    }
    static void readn_ex(int fd, void *vptr, size_t n) //never return error.
    {
     auto ret = readn(fd, vptr, n);
     if(ret == -1) throw std::runtime_error("readn failed.");
    }
    static void writen_ex(int fd, const void *vptr, size_t n)
    {
     auto ret = writen(fd, vptr, n);
     if(ret == -1) throw std::runtime_error("writen failed.");
    }
     static ssize_t readall_ex(int fd, void **pvptr, size_t initSize) //never
return -1
    {
     auto ret = readall(fd, pvptr, initSize);
     if(ret == -1) throw std::runtime_error("readall failed.");
     return ret;
    }
   };

   class sockIO
   {
   public:
     static ssize_t recvn(int fd, void *vptr, size_t n, int flags) noexcept
//Return -1 on error, read bytes on success, blocks until nbytes done.
    {
     size_t  nleft;
     ssize_t nread;
    char   *ptr;

    ptr = (char *)vptr;
    nleft = n;
    while (nleft > 0) {
      if ( (nread = recv(fd, ptr, nleft, flags)) < 0) {
        if (errno == EINTR)
          nread = 0;      /* and call read() again */
        else
          return (-1);
      } else if (nread == 0)
        return -1;              /* EOF */

      nleft -= nread;
      ptr += nread;
    }
    return (n);         /* return success */
  }
    static ssize_t sendn(int fd, const void *vptr, size_t n, int flags) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.
  {
    size_t nleft;
    ssize_t nwritten;
    const char *ptr;

    ptr = (const char *)vptr;
    nleft = n;
    while (nleft > 0) {
      if ( (nwritten = send(fd, ptr, nleft, flags)) <= 0) {
        if (nwritten < 0 && errno == EINTR)
          nwritten = 0;   /* and call write() again */
        else
          return (-1);    /* error */
       }

       nleft -= nwritten;
       ptr += nwritten;
    }
    return (n);
  }
   static ssize_t recvall(int fd, void **pvptr, size_t initSize, int flags) noexcept //Return -1 on error, read bytes on success. pvptr must be a malloc/calloced buffer, I'll malloc one if *pvptr is NULL.
  {
    size_t current = initSize ? initSize : 1024;
    void *vptr = *pvptr;
    if(vptr == NULL)
      vptr = malloc(current);
    void *currvptr = vptr;

    {
      ssize_t ret = recv(fd, currvptr, current / 2, flags);
      if(ret == -1) return -1;
      if(ret < current / 2)
      {
        *pvptr = vptr;
        return ret;
      }
      currvptr = (char *)vptr + current / 2;
    }

    while(true)
    {
      ssize_t ret = recv(fd, currvptr, current / 2, flags);
      if(ret == -1) return -1;
      if(ret < current)
      {
        *pvptr = vptr;
        return ret + current / 2;
      }

      current *= 2;
      void *vptrBackup = vptr;
      if((vptr = realloc(vptr, current)) == NULL) {
        free(vptrBackup);
        errno = EMSGSIZE;
        return -1;
      }
      currvptr = (char *)vptr + current / 2;
    }
  }
    static void recvn_ex(int fd, void *vptr, size_t n, int flags) //return
read bytes.
  {
    auto ret = recvn(fd, vptr, n, flags);
    if(ret == -1) throw std::runtime_error("recvn failed.");
  }
```

```
   static ssize_t sendn_ex(int fd, const void *vptr, size_t n, int flags)
   {
    auto ret = sendn(fd, vptr, n, flags);
    if(ret == -1) throw std::runtime_error("sendn failed.");
    return ret;
   }
      static ssize_t recvall_ex(int fd, void **pvptr, size_t initSize, int
flags) //never return -1
   {
    auto ret = recvall(fd, pvptr, initSize, flags);
    if(ret == -1) throw std::runtime_error("recvall failed.");
    return ret;
   }
  };
}
#else
#include <winsock2.h>
//WINsock version
namespace rlib {
  class sockIO
  {
  private:
   static int WSASafeGetLastError()
   {
    int i;
    WSASetLastError(i = WSAGetLastError());
    return i;
   }
  public:
       static ssize_t recvn(SOCKET fd, char *vptr, size_t n, int flags)
noexcept //Return -1 on error, read bytes on success, blocks until nbytes
done.
   {
    size_t  nleft;
    ssize_t nread;
    char   *ptr;

    ptr = (char *)vptr;
    nleft = n;
    while (nleft > 0) {
     if ( (nread = recv(fd, ptr, nleft, flags)) == SOCKET_ERROR) {
      if (WSASafeGetLastError() == WSAEINTR)
       nread = 0;      /* and call read() again */
      else
```

```
        return (-1);
      } else if (nread == 0)
       return (-1);              /* EOF */

      nleft -= nread;
      ptr += nread;
    }
    return (n);         /* return >= 0 */
  }
```

static ssize_t sendn(SOCKET fd, const char *vptr, size_t n, int flags) noexcept //Return -1 on error, read bytes on success, blocks until nbytes done.

```
  {
    size_t nleft;
    ssize_t nwritten;
    const char *ptr;

    ptr = (const char *)vptr;
    nleft = n;
    while (nleft > 0) {
      if ( (nwritten = send(fd, ptr, nleft, flags)) <= 0) {
       if (nwritten == SOCKET_ERROR && WSASafeGetLastError() == WSAEINTR)
        nwritten = 0;   /* and call write() again */
       else
        return (-1);    /* error */
      }

      nleft -= nwritten;
      ptr += nwritten;
    }
    return (n);
  }
```

static ssize_t recvall(SOCKET fd, void **pvptr, size_t initSize, int flags) noexcept //Return -1 on error, read bytes on success. pvptr must be a malloc/calloced buffer, I'll malloc one if *pvptr is NULL.

```
  {
    size_t current = initSize ? initSize : 1024;
    void *vptr = *pvptr;
    if(vptr == NULL)
      vptr = malloc(current);
    void *currvptr = vptr;

    {
    _retry_1:
      ssize_t ret = recv(fd, (char *)currvptr, current / 2, flags);
      if(ret == SOCKET_ERROR) {
        if(WSASafeGetLastError() == WSAEINTR)
          goto _retry_1;
        return SOCKET_ERROR;
      }
      if(ret < current / 2)
      {
        *pvptr = vptr;
        return ret;
      }
      currvptr = (char *)vptr + current / 2;
    }

    while(true)
    {
      ssize_t ret = recv(fd, (char *)currvptr, current / 2, flags);
      if(ret == SOCKET_ERROR) {
        if(WSASafeGetLastError() == WSAEINTR)
          continue; //retry
        return SOCKET_ERROR;
      }
      if(ret < current)
      {
        *pvptr = vptr;
        return ret + current / 2;
      }

      current *= 2;
      void *vptrBackup = vptr;
      if((vptr = realloc(vptr, current)) == NULL) {
        free(vptrBackup);
        WSASetLastError(WSAEMSGSIZE);
        return SOCKET_ERROR;
      }
      currvptr = (char *)vptr + current / 2;
    }
  }
   static void recvn_ex(SOCKET fd, char *vptr, size_t n, int flags) //never
return error.
  {
    auto ret = recvn(fd, vptr, n, flags);
    if(ret == -1) throw std::runtime_error("recvn failed.");
  }
  static ssize_t sendn_ex(SOCKET fd, const char *vptr, size_t n, int flags)
  {
   auto ret = sendn(fd, vptr, n, flags);
   if(ret == -1) throw std::runtime_error("recvn failed.");
                   return ret;
  }
   static ssize_t recvall_ex(SOCKET fd, void **pvptr, size_t initSize, int
flags) //never return -1
  {
   auto ret = recvall(fd, pvptr, initSize, flags);
   if(ret == -1) throw std::runtime_error("recvn failed.");
   return ret;
  }
 };

 class fdIO
 {
 public:
  static ssize_t readn(SOCKET fd, void *vptr, size_t n) noexcept //Return -1
on error, read bytes on success, blocks until nbytes done.
  {
   return sockIO::recvn(fd, (char *)vptr, n, 0);
  }
    static ssize_t writen(SOCKET fd, const void *vptr, size_t n) noexcept
//Return -1 on error, read bytes on success, blocks until nbytes done.
  {
   return sockIO::sendn(fd, (const char *)vptr, n, 0);
  }
      static ssize_t readall(SOCKET fd, void **pvptr, size_t initSize)
noexcept //Return -1 on error, read bytes on success. pvptr must be a
malloc/calloced buffer, I'll malloc one if *pvptr is NULL.
  {
   return sockIO::recvall(fd, pvptr, initSize, 0);
  }
  static void readn_ex(SOCKET fd, void *vptr, size_t n) //return read bytes.
  {
   return sockIO::recvn_ex(fd, (char *)vptr, n, 0);
  }
  static ssize_t writen_ex(SOCKET fd, const void *vptr, size_t n)
  {
   return sockIO::sendn_ex(fd, (const char *)vptr, n, 0);
  }
     static ssize_t readall_ex(SOCKET fd, void **pvptr, size_t initSize)
//never return -1
```

```
   {
     return sockIO::recvall_ex(fd, pvptr, initSize, 0);
   }
  };
}

#endif



#endif
// --- lib/rlib/string/string.hpp
#ifndef R_STRING_HPP
#define R_STRING_HPP

#include <vector>
#include <string>
#include <cstdarg>
#include <cstdio>
#include <cstdlib>
#include <stdexcept>
#include <sstream>
#include <type_traits>

namespace rlib {
      std::vector<std::string> splitString(const std::string &toSplit, const
char &divider = ' ');
      std::vector<std::string> splitString(const std::string &toSplit, const
std::string &divider);
  template <class ForwardIterator>
      std::string joinString(const char &toJoin, ForwardIterator begin,
ForwardIterator end);
  template <class ForwardIterator>
    std::string joinString(const std::string &toJoin, ForwardIterator begin,
ForwardIterator end);
  template <class ForwardIterable>
      std::string joinString(const char &toJoin, ForwardIterable begin,
ForwardIterable end);
  template <class ForwardIterable>
    std::string joinString(const std::string &toJoin, ForwardIterable begin,
ForwardIterable end);

    size_t replaceSubString(std::string& str, const std::string &from, const
std::string& to);
   bool replaceSubStringOnce(std::string& str, const std::string& from, const
std::string& to);
  template<typename... Args>
  std::string format_string_c(const std::string &fmt, Args... args);
  template<typename... Args>
  std::string format_string(const std::string &fmt, Args... args);


//Implements.
  char *_format_string_c_helper(const char *fmt, ...);
  template<typename... Args>
  std::string format_string_c(const std::string &fmt, Args... args)
  {
   char *res = _format_string_c_helper(fmt.c_str(), args ...);
   std::string s = res;
   free(res);
   return std::move(s);
  }

  template<typename StdString>
  void _format_string_helper(std::stringstream &ss, const StdString &fmt) {
       static_assert(std::is_same<StdString, std::string>::value, "incorrect
argument type to _format_string_helper");
   ss << fmt;
  }
  template<typename Arg1, typename... Args>
   void _format_string_helper(std::stringstream &ss, const std::string &fmt,
Arg1 arg1, Args... args) {
   size_t pos = 0;
   while((pos = fmt.find("{}")) != std::string::npos) {
     if(pos != 0 && fmt[pos-1] == '\\') {
      ++pos;
      continue;
     }
     ss << fmt.substr(0, pos) << arg1;
     _format_string_helper(ss, fmt.substr(pos + 2), args ...);
     return;
   }
       _format_string_helper(ss, fmt);
  }
  template<typename... Args>
  std::string format_string(const std::string &fmt, Args... args) {
   std::stringstream ss;
   _format_string_helper(ss, fmt, args...);
   return ss.str();
  }

     inline  std::vector<std::string>  splitString(const  std::string
&toSplit, const char &divider)
     {
   std::vector<std::string> buf;
   size_t curr = 0, prev = 0;
   while((curr = toSplit.find(divider, curr)) ≠ std::string::npos) {
     buf.push_back(toSplit.substr(prev, curr - prev));
     ++curr; // skip divider
     prev = curr;
   }
   buf.push_back(toSplit.substr(prev));
   return std::move(buf);
     }
     inline std::vector<std::string> splitString(const std::string &toSplit,
const std::string &divider)
     {
   std::vector<std::string> buf;
   size_t curr = 0, prev = 0;
   while((curr = toSplit.find(divider, curr)) ≠ std::string::npos) {
     buf.push_back(toSplit.substr(prev, curr - prev));
     curr += divider.size(); // skip divider
     prev = curr;
   }
   buf.push_back(toSplit.substr(prev));
   return std::move(buf);
     }
  template <class ForwardIterator>
     std::string  joinString(const  char  &toJoin,  ForwardIterator  begin,
ForwardIterator end) {
   std::string result;
   for(ForwardIterator iter = begin; iter ≠ end; ++iter) {
     if(iter ≠ begin)
      result += toJoin;
     result += *iter;
   }
   return std::move(result);
  }
  template <class ForwardIterator>
   std::string joinString(const std::string &toJoin, ForwardIterator begin,
ForwardIterator end) {
   std::string result;
   for(ForwardIterator iter = begin; iter ≠ end; ++iter) {
    if(iter ≠ begin)
     result += toJoin;
    result += *iter;
   }
   return std::move(result);
  }
  template <class ForwardIterable>
  std::string joinString(const std::string &toJoin, ForwardIterable buf) {
   auto begin = buf.begin();
   auto end = buf.end();
   return std::move(joinString(toJoin, begin, end));
  }
  template <class ForwardIterable>
  std::string joinString(const char &toJoin, ForwardIterable buf) {
   auto begin = buf.begin();
   auto end = buf.end();
   return std::move(joinString(toJoin, begin, end));
  }

  inline size_t replaceSubString(std::string& str, const std::string &from,
const std::string& to)
  {
   if(from.empty())
    return 0;
   size_t start_pos = 0;
   size_t times = 0;
   while((start_pos = str.find(from, start_pos)) ≠ std::string::npos)
   {
    ++times;
    str.replace(start_pos, from.length(), to);
        start_pos += to.length(); // In case 'to' contains 'from', like
replacing 'x' with 'yx'
   }
   return times;
  }
  inline bool replaceSubStringOnce(std::string& str, const std::string& from,
const std::string& to)
  {
   size_t start_pos = str.find(from);
   if(start_pos == std::string::npos)
    return false;
   str.replace(start_pos, from.length(), to);
   return true;
  }
  inline char *_format_string_c_helper(const char *fmt, ... )
  {
    int n;
    int size = 100;     /* Guess we need no more than 100 bytes */
    char *p, *np;
    va_list ap;

    if ((p = (char *)malloc(size)) == NULL)
      throw std::runtime_error("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)
        throw std::runtime_error("vsnprintf returns " + std::to_string(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);
        throw std::runtime_error("make_message realloc failed.");
      } else {
        p = np;
      }
    }
  }

}
```

```
#endif
// --- lib/rlib/require/cxx11
#ifndef R_CXX11_REQUIRED
#define R_CXX11_REQUIRED

#if __cplusplus < 201103L
#error C++11 is required.
#endif

#endif// --- lib/rlib/require/cxx14
#ifndef R_CXX14_REQUIRED
#define R_CXX14_REQUIRED

#include <bits/c++14_warning.h>

#endif// --- lib/rlib/require/cxx17
#ifndef R_CXX17_REQUIRED
#define R_CXX17_REQUIRED

#include <bits/c++17_warning.h>

#endif// --- lib/rlib/require/gcc
#ifndef R_GCC_REQUIRED
#define R_GCC_REQUIRED

#ifdef RLIB_MARK_NAMESPACE_POLLUTED
#error RLIB_MARK_NAMESPACE_POLLUTED must not be defined in global namespace.
#endif

#include <rlib/sys/os.hpp>

#ifndef GCC
#define GCC 9876
#define RLIB_MARK_NAMESPACE_POLLUTED
#endif

#if __COMPILER_ID__ != GCC
#error Gcc is required but not detected.
#endif

#ifdef RLIB_MARK_NAMESPACE_POLLUTED
#undef GCC
#undef RLIB_MARK_NAMESPACE_POLLUTED
#endif
```

```
#endif
// --- lib/rlib/require/linux
#ifndef R_LINUX_REQUIRED
#define R_LINUX_REQUIRED

#ifdef RLIB_MARK_NAMESPACE_POLLUTED
#error RLIB_MARK_NAMESPACE_POLLUTED must not be defined in global namespace.
#endif

#include <rlib/sys/os.hpp>

#ifndef LINUX
#define LINUX 9876
#define RLIB_MARK_NAMESPACE_POLLUTED
#endif

#if __OS_ID__ ≠ 9876
#error Linux is required but not detected.
#endif

#ifdef RLIB_MARK_NAMESPACE_POLLUTED
#undef LINUX
#undef RLIB_MARK_NAMESPACE_POLLUTED
#endif

#endif
// --- lib/rlib/require/win
#ifndef R_WINDOWS_REQUIRED
#define R_WINDOWS_REQUIRED

#ifdef RLIB_MARK_NAMESPACE_POLLUTED
#error RLIB_MARK_NAMESPACE_POLLUTED must not be defined in global namespace.
#endif

#include <rlib/sys/os.hpp>

#ifndef WINDOWS
#define WINDOWS 9876
#define RLIB_MARK_NAMESPACE_POLLUTED
#endif

#if __OS_ID__ ≠ WINDOWS
#error Windows is required but not detected.
```

```
#endif

#ifdef RLIB_MARK_NAMESPACE_POLLUTED
#undef WINDOWS
#undef RLIB_MARK_NAMESPACE_POLLUTED
#endif

#endif

// --- lib/gc/gc.cpp
// Copyright (C) 2009 Chris Double. All Rights Reserved.
// See the license at the end of this file
#include <iostream>
#include <boost/date_time/posix_time/posix_time.hpp>
#include "gc.h"

using namespace std;

// GCObject
GCObject::GCObject() :
 mMarked(false) {
 GarbageCollector::GC.addObject(this);
}

GCObject::GCObject(GCObject const&) :
 mMarked(false) {
 GarbageCollector::GC.addObject(this);
}

GCObject::~GCObject() {
}

void GCObject::mark() {
 if (!mMarked) {
  mMarked = true;
  markChildren();
 }
}

void GCObject::markChildren() {
}

// GCMemory
GCMemory::GCMemory(int size) : mSize(size) {
 mMemory = new unsigned char[size];
}

GCMemory::~GCMemory() {
 delete [] mMemory;
}

unsigned char* GCMemory::get() {
 return mMemory;
}

int GCMemory::size() {
 return mSize;
}

// GarbageCollector
GarbageCollector GarbageCollector::GC;

void GarbageCollector::collect(bool verbose) {
 using namespace boost::posix_time;
      unsigned    int    start    =    (microsec_clock::universal_time()    -
ptime(min_date_time)).total_milliseconds();

 // Mark root objects
 for (ObjectSet::iterator it = mRoots.begin();
   it != mRoots.end();
   ++it)
  (*it)->mark();

 // Mark pinned objects
 for (PinnedSet::iterator it = mPinned.begin();
   it != mPinned.end();
   ++it)
  (*it).first->mark();

 if (verbose) {
  cout << "Roots: " << mRoots.size() << endl;
  cout << "Pinned: " << mPinned.size() << endl;
  cout << "GC: " << mHeap.size() << " objects in heap" << endl;
 }

 sweep(verbose);

 if (verbose) {
```

```
          unsigned   int   end   =   (microsec_clock::universal_time()   -
ptime(min_date_time)).total_milliseconds();
  cout << "GC: " << (end-start) << " milliseconds" << endl;
 }
}

void GarbageCollector::addRoot(GCObject* root) {
 mRoots.insert(root);
}

void GarbageCollector::removeRoot(GCObject* root) {
 mRoots.erase(root);
}

void GarbageCollector::pin(GCObject* o) {
 PinnedSet::iterator it = mPinned.find(o);
 if (it == mPinned.end()) {
  mPinned.insert(make_pair(o, 1));
 }
 else {
  (*it).second++;
 }
}

void GarbageCollector::unpin(GCObject* o) {
 PinnedSet::iterator it = mPinned.find(o);
 assert(it != mPinned.end());

 if (--((*it).second) == 0)
  mPinned.erase(it);
}

void GarbageCollector::addObject(GCObject* o) {
 mHeap.insert(o);
}

void GarbageCollector::removeObject(GCObject* o) {
 mHeap.erase(o);
}

void GarbageCollector::sweep(bool verbose) {
 unsigned int live = 0;
 unsigned int dead = 0;
 unsigned int total = 0;
 vector<ObjectSet::iterator> erase;
 for (ObjectSet::iterator it = mHeap.begin();
   it != mHeap.end();
   ++it) {
  GCObject* p = *it;
  total++;
  if (p->mMarked) {
   p->mMarked = false;
   ++live;
  }
  else {
   erase.push_back(it);
  }
 }
 dead = erase.size();
 for (vector<ObjectSet::iterator>::iterator it = erase.begin();
   it != erase.end();
   ++it) {
  GCObject* p = **it;
  mHeap.erase(*it);
  delete p;
 }
 if (verbose) {
  cout << "GC: " << live << " objects live after sweep" << endl;
  cout << "GC: " << dead << " objects dead after sweep" << endl;
 }
}

int GarbageCollector::live() {
 return mHeap.size();
}

// Copyright (C) 2009 Chris Double. All Rights Reserved.
// The original author of this code can be contacted at:
chris.double@double.co.nz
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
met:
//
// 1. Redistributions of source code must retain the above copyright notice,
//    this list of conditions and the following disclaimer.
//
// 2. Redistributions in binary form must reproduce the above copyright
```

```
notice,
//       this list of conditions and the following disclaimer in the
documentation
//    and/or other materials provided with the distribution.
//
// THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED
WARRANTIES,
// INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND
// FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
// DEVELOPERS AND CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS;
// OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
// WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
// OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
// ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// --- lib/gc/gc.h
// Copyright (C) 2009 Chris Double. All Rights Reserved.
// See the license at the end of this file
#if !defined(GC_H)
#define GC_H

#include <set>
#include <map>

// Base class for all objects that are tracked by
// the garbage collector.
class GCObject {
 public:

  // For mark and sweep algorithm. When a GC occurs
  // all live objects are traversed and mMarked is
  // set to true. This is followed by the sweep phase
  // where all unmarked objects are deleted.
  bool mMarked;

 public:
  GCObject();
  GCObject(GCObject const&);
 virtual ~GCObject();

 // Mark the object and all its children as live
 void mark();

 // Overridden by derived classes to call mark()
 // on objects referenced by this object. The default
 // implemention does nothing.
 virtual void markChildren();
};

// Wrapper for an array of bytes managed by the garbage
// collector.
class GCMemory : public GCObject {
public:
 unsigned char* mMemory;
 int   mSize;

public:
 GCMemory(int size);
 virtual ~GCMemory();

 unsigned char* get();
 int size();
};

// Garbage Collector. Implements mark and sweep GC algorithm.
class GarbageCollector {
public:
 // A collection of all active heap objects.
 typedef std::set<GCObject*> ObjectSet;
 ObjectSet mHeap;

 // Collection of objects that are scanned for garbage.
 ObjectSet mRoots;

 // Pinned objects
 typedef std::map<GCObject*, unsigned int> PinnedSet;
 PinnedSet mPinned;

 // Global garbage collector object
 static GarbageCollector GC;

public:
 // Perform garbage collection. If 'verbose' is true then
 // GC stats will be printed to stdout.
 void collect(bool verbose = false);

 // Add a root object to the collector.
 void addRoot(GCObject* root);

 // Remove a root object from the collector.
 void removeRoot(GCObject* root);

 // Pin an object so it temporarily won't be collected.
 // Pinned objects are reference counted. Pinning it
 // increments the count. Unpinning it decrements it. When
 // the count is zero then the object can be collected.
 void pin(GCObject* o);
 void unpin(GCObject* o);

 // Add an heap allocated object to the collector.
 void addObject(GCObject* o);

 // Remove a heap allocated object from the collector.
 void removeObject(GCObject* o);

 // Go through all objects in the heap, unmarking the live
 // objects and destroying the unreferenced ones.
 void sweep(bool verbose);

 // Number of live objects in heap
 int live();
};

#endif
// Copyright (C) 2009 Chris Double. All Rights Reserved.
// The original author of this code can be contacted at:
chris.double@double.co.nz
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
met:
//
// 1. Redistributions of source code must retain the above copyright notice,
//    this list of conditions and the following disclaimer.
//
// 2. Redistributions in binary form must reproduce the above copyright
```

```
notice,
//       this list of conditions and the following disclaimer in the
documentation
//    and/or other materials provided with the distribution.
//
// THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED
WARRANTIES,
// INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND
// FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
// DEVELOPERS AND CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS;
// OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
// WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
// OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
// ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// --- lib/gc/makefile
UNAME=$(shell uname -s)

ifeq "$(UNAME)" "Darwin"
INCLUDE=-I/opt/local/include
LIB=-L/opt/local/lib
endif

INCLUDE=
LIB=
CFLAGS=-g

all: libgc.a

libgc.a: gc.o
	ar r libgc.a gc.o

gc.o: gc.cpp gc.h
	g++ $(INCLUDE) $(CFLAGS) -c -o gc.o gc.cpp

clean:
	rm gc.o
	rm libgc.a
```

```
// --- lib/gc/README
C++ Garbage Collection Library
==============================

This is a library to manage memory in C++ programs using a garbage
collector. It uses a mark and sweep algorithm.

All objects that are to be managed by the collector should be derived
from GCObject:

 class Test : public GCObject {
  ...
 };

If the object maintains references to other GC managed objects it
should override 'markChildren' to call 'mark' on those objects:

 class Test2 : public GCObject {
  private:
   Test* mTest;

  public:
   virtual void markChildren() {
    mTest->mark();
   }
  ...
 };

Periodic calls to GarbageCollector::GC.collect() should be made to
delete unreferenced objects and free memory. This call will call
'mark' on all the root objects, ensuring that they and their children
are not deleted, and and remaining objects are destroyed.

To add an object as a root, call GarbageCollector::GC.addRoot().

License
=======
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
 this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
 this list of conditions and the following disclaimer in the documentation
 and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
DEVELOPERS AND CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Feedback
========
I host this on githib:
 http://github.com/doublec/gc

Feel free to clone, hack away, suggest patches, etc. I can be reached
via email: chris.double@double.co.nz

// --- lib/cpp-readline/src/CMakeLists.txt
#    /cpp-readline/src/CMakeLists.txt
#
#    @author zmij
#    @date May 17, 2016

cmake_minimum_required(VERSION 2.6)

set(cpp_readline_SRCS
  Console.cpp
)

add_library(${lib_name} SHARED ${cpp_readline_SRCS})
set_target_properties(
  ${lib_name} PROPERTIES
  VERSION ${PROJECT_VERSION}
  SOVERSION 1
)
target_link_libraries(${lib_name} ${Readline_LIBRARY})
// --- lib/cpp-readline/src/Console.cpp
#include "Console.hpp"
```

```
#include <iostream>
#include <fstream>
#include <functional>
#include <algorithm>
#include <iterator>
#include <sstream>
#include <unordered_map>

#include <cstdlib>
#include <cstring>
#include <readline/readline.h>
#include <readline/history.h>

namespace CppReadline {
 namespace {

  Console* currentConsole         = nullptr;
  HISTORY_STATE* emptyHistory     = history_get_history_state();

 }  /* namespace  */

 struct Console::Impl {
                                          using      RegisteredCommands      =
std::unordered_map<std::string,Console::CommandFunction>;

   ::std::string       greeting_;
   // These are hardcoded commands. They do not do anything and are catched
manually in the executeCommand function.
  RegisteredCommands  commands_;
  HISTORY_STATE*      history_    = nullptr;

  Impl(::std::string const& greeting) : greeting_(greeting), commands_() {}
  ~Impl() {
   free(history_);
  }

  Impl(Impl const&) = delete;
  Impl(Impl&&) = delete;
  Impl& operator = (Impl const&) = delete;
  Impl& operator = (Impl&&) = delete;
 };

 // Here we set default commands, they do nothing since we quit with them
 // Quitting behaviour is hardcoded in readLine()
```

```
Console::Console(std::string const& greeting)
  : pimpl_{ new Impl{ greeting } }
{
  // Init readline basics
  rl_attempted_completion_function = &Console::getCommandCompletions;

  // These are default hardcoded commands.
  // Help command lists available commands.
  pimpl_->commands_["help"] = [this](const Arguments &){
    auto commands = getRegisteredCommands();
    std::cout << "Available commands are:\n";
    for ( auto & command : commands ) std::cout << "\t" << command << "\n";
    return ReturnCode::Ok;
  };
  // Run command executes all commands in an external file.
  pimpl_->commands_["run"] =  [this](const Arguments & input) {
        if ( input.size() < 2 ) { std::cout << "Usage: " << input[0] << "
script_filename\n"; return 1; }
    return executeFile(input[1]);
  };
  // Quit and Exit simply terminate the console.
  pimpl_->commands_["quit"] = [this](const Arguments &) {
    return ReturnCode::Quit;
  };

  pimpl_->commands_["exit"] = [this](const Arguments &) {
    return ReturnCode::Quit;
  };
}

Console::~Console() = default;

void Console::registerCommand(const std::string & s, CommandFunction f) {
  pimpl_->commands_[s] = f;
}

std::vector<std::string> Console::getRegisteredCommands() const {
  std::vector<std::string> allCommands;
  for ( auto & pair : pimpl_->commands_ ) allCommands.push_back(pair.first);

  return allCommands;
}

void Console::saveState() {
  free(pimpl_->history_);
  pimpl_->history_ = history_get_history_state();
}

void Console::reserveConsole() {
  if ( currentConsole == this ) return;

  // Save state of other Console
  if ( currentConsole )
    currentConsole->saveState();

  // Else we swap state
  if ( ! pimpl_->history_ )
    history_set_history_state(emptyHistory);
  else
    history_set_history_state(pimpl_->history_);

  // Tell others we are using the console
  currentConsole = this;
}

void Console::setGreeting(const std::string & greeting) {
  pimpl_->greeting_ = greeting;
}

std::string Console::getGreeting() const {
  return pimpl_->greeting_;
}

int Console::executeCommand(const std::string & command) {
  // Convert input to vector
  std::vector<std::string> inputs;
  {
    std::istringstream iss(command);
    std::copy(std::istream_iterator<std::string>(iss),
        std::istream_iterator<std::string>(),
        std::back_inserter(inputs));
  }

  if ( inputs.size() == 0 ) return ReturnCode::Ok;

  Impl::RegisteredCommands::iterator it;
         if ( ( it = pimpl_->commands_.find(inputs[0]) ) != end(pimpl_-
>commands_) ) {
    return static_cast<int>((it->second)(inputs));
  }

  std::cout << "Command '" << inputs[0] << "' not found.\n";
  return ReturnCode::Error;
}

int Console::executeFile(const std::string & filename) {
  std::ifstream input(filename);
  if ( ! input ) {
    std::cout << "Could not find the specified file to execute.\n";
    return ReturnCode::Error;
  }
  std::string command;
  int counter = 0, result;

  while ( std::getline(input, command)  ) {
    if ( command[0] == '#' ) continue; // Ignore comments
    // Report what the Console is executing.
    std::cout << "[" << counter << "] " << command << '\n';
    if ( (result = executeCommand(command)) ) return result;
    ++counter; std::cout << '\n';
  }

  // If we arrived successfully at the end, all is ok
  return ReturnCode::Ok;
}

int Console::readLine() {
  reserveConsole();

  char * buffer = readline(pimpl_->greeting_.c_str());
  if ( !buffer ) {
      std::cout << '\n'; // EOF doesn't put last endline so we put that so
that it looks uniform.
    return ReturnCode::Quit;
  }

  // TODO: Maybe add commands to history only if succeeded?
  if ( buffer[0] != '\0' )
    add_history(buffer);

  std::string line(buffer);
  free(buffer);
  return executeCommand(line);
 }

 char ** Console::getCommandCompletions(const char * text, int start, int) {
  char ** completionList = nullptr;

  if ( start == 0 )
   completionList = rl_completion_matches(text, &Console::commandIterator);

  return completionList;
 }

 char * Console::commandIterator(const char * text, int state) {
  static Impl::RegisteredCommands::iterator it;
  if (!currentConsole)
   return nullptr;
  auto& commands = currentConsole->pimpl_->commands_;

  if ( state == 0 ) it = begin(commands);

  while ( it != end(commands) ) {
   auto & command = it->first;
   ++it;
   if ( command.find(text) != std::string::npos ) {
    return strdup(command.c_str());
   }
  }
  return nullptr;
 }
}
// --- lib/cpp-readline/src/Console.hpp
#ifndef CONSOLE_CONSOLE_HEADER_FILE
#define CONSOLE_CONSOLE_HEADER_FILE

#include <functional>
#include <string>
#include <vector>
#include <memory>

namespace CppReadline {
 class Console {
  public:
   /**
```

```
     * @brief This is the function type that is used to interface with the
Console class.
    *
    * These are the functions that are going to get called by Console
    * when the user types in a message. The vector will hold the
    * command elements, and the function needs to return its result.
    * The result can either be Quit (-1), OK (0), or an arbitrary
    * error (≥1).
    */
    using Arguments = std::vector<std::string>;
    using CommandFunction = std::function<int(const Arguments &)>;

    enum ReturnCode {
     Quit = -1,
     Ok = 0,
     Error = 1 // Or greater!
    };

    /**
    * @brief Basic constructor.
    *
    * The Console comes with two predefined commands: "quit" and
    * "exit", which both terminate the console, "help" which prints a
    * list of all registered commands, and "run" which executes script
    * files.
    *
    * These commands can be overridden or unregistered - but remember
    * to leave at least one to quit ;).
    *
    * @param greeting This represents the prompt of the Console.
    */
    explicit Console(std::string const& greeting);

    /**
    * @brief Basic destructor.
    *
    * Frees the history which is been produced by GNU readline.
    */
    ~Console();

    /**
    * @brief This function registers a new command within the Console.
    *
    * If the command already existed, it overwrites the previous entry.
```

```
     *
     * @param s The name of the command as inserted by the user.
       * @param f The function that will be called once the user writes the
command.
     */
    void registerCommand(const std::string & s, CommandFunction f);

    /**
        * @brief This function returns a list with the currently available
commands.
     *
     * @return A vector containing all registered commands names.
     */
    std::vector<std::string> getRegisteredCommands() const;

    /**
     * @brief Sets the prompt for this Console.
     *
     * @param greeting The new greeting.
     */
    void setGreeting(const std::string & greeting);

    /**
     * @brief Gets the current prompt of this Console.
     *
     * @return The currently set greeting.
     */
    std::string getGreeting() const;

    /**
         * @brief This function executes an arbitrary string as if it was
inserted via stdin.
     *
     * @param command The command that needs to be executed.
     *
     * @return The result of the operation.
     */
    int executeCommand(const std::string & command);

    /**
         * @brief This function calls an external script and executes all
commands inside.
     *
      * This function stops execution as soon as any single command returns
```

```
something
     * different from 0, be it a quit code or an error code.
     *
     * @param filename The pathname of the script.
     *
     * @return What the last command executed returned.
     */
    int executeFile(const std::string & filename);

    /**
         * @brief This function executes a single command from the user via
stdin.
     *
     * @return The result of the operation.
     */
    int readLine();
   private:
    Console(const Console&) = delete;
    Console(Console&&) = delete;
    Console& operator = (Console const&) = delete;
    Console& operator = (Console&&) = delete;

    struct Impl;
    using PImpl = ::std::unique_ptr<Impl>;
    PImpl pimpl_;

    /**
         * @brief This function saves the current state so that some other
Console can make use of the GNU readline facilities.
     */
    void saveState();
    /**
      * @brief This function reserves the use of the GNU readline facilities
to the calling Console instance.
     */
    void reserveConsole();

    // GNU newline interface to our commands.
      using commandCompleterFunction = char**(const char * text, int start,
int end);
    using commandIteratorFunction = char*(const char * text, int state);

    static commandCompleterFunction getCommandCompletions;
    static commandIteratorFunction commandIterator;
```

```
  };
}

#endif
```