Lists are lists

Lieven de Cock
www.codeblocks.org
www.cppdriven.com

lieven@cppdriven.com

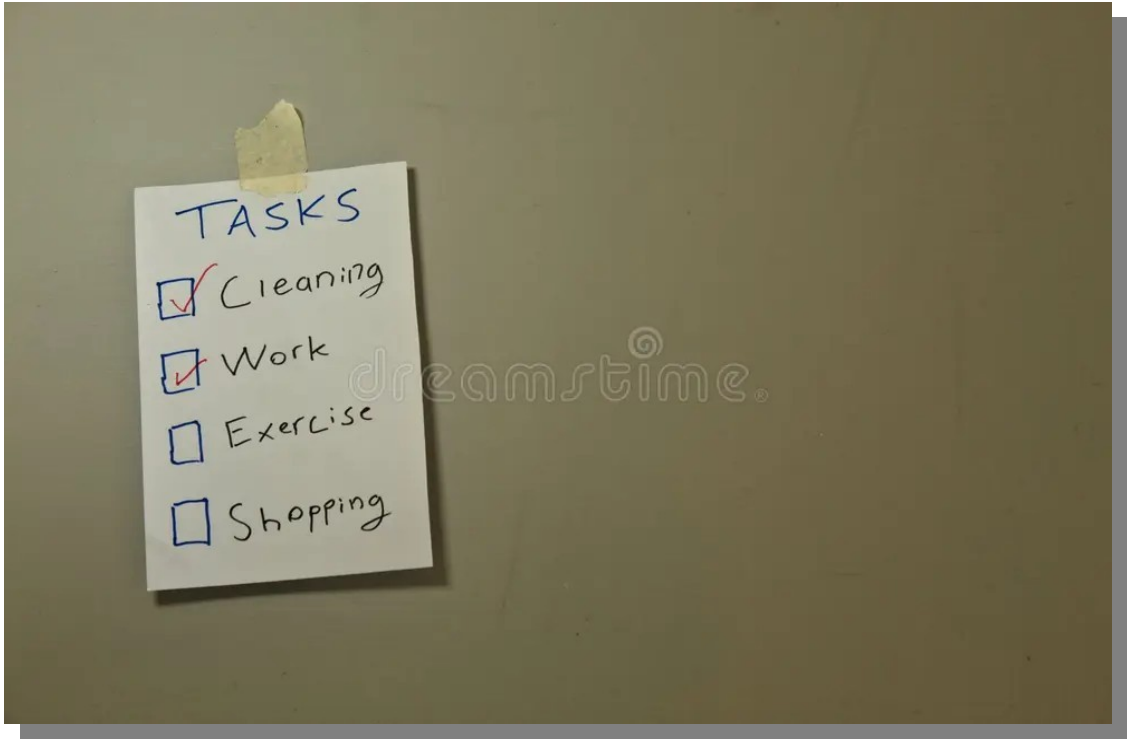
Grumpy Old Man Complaining

Lists

- Shopping Lists
- To-do Lists

Let’s inspect some

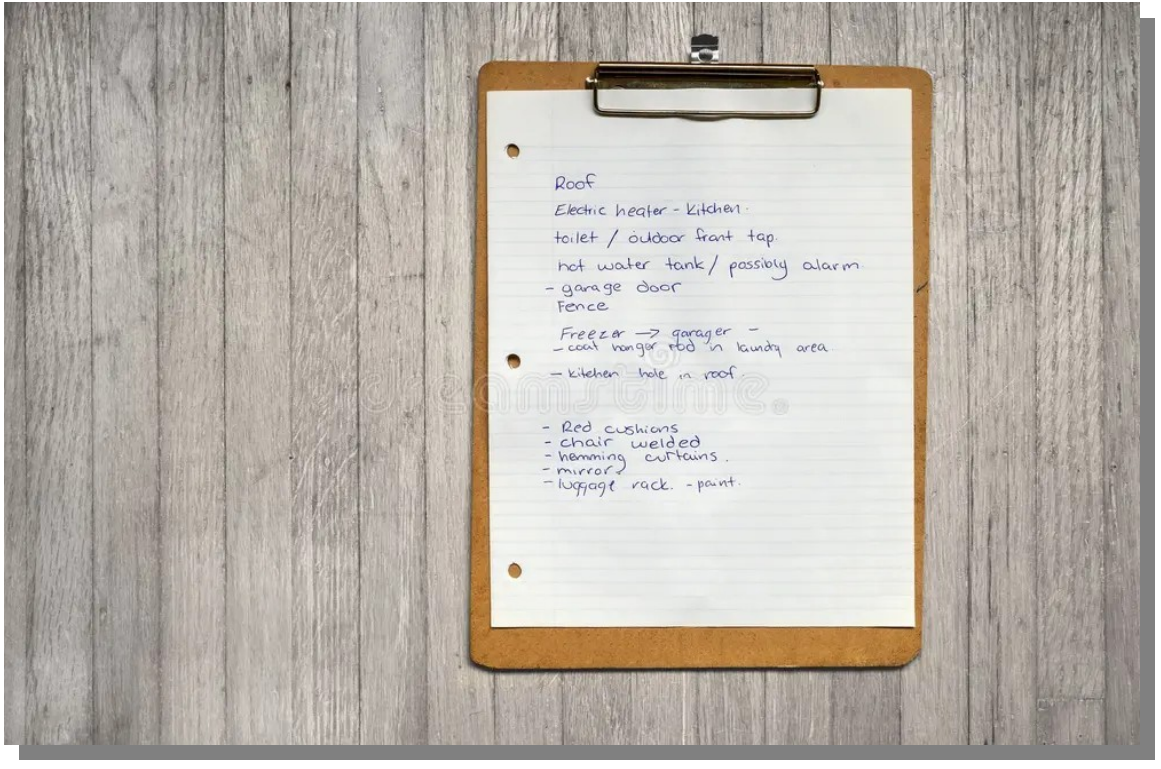
Real Life Lists



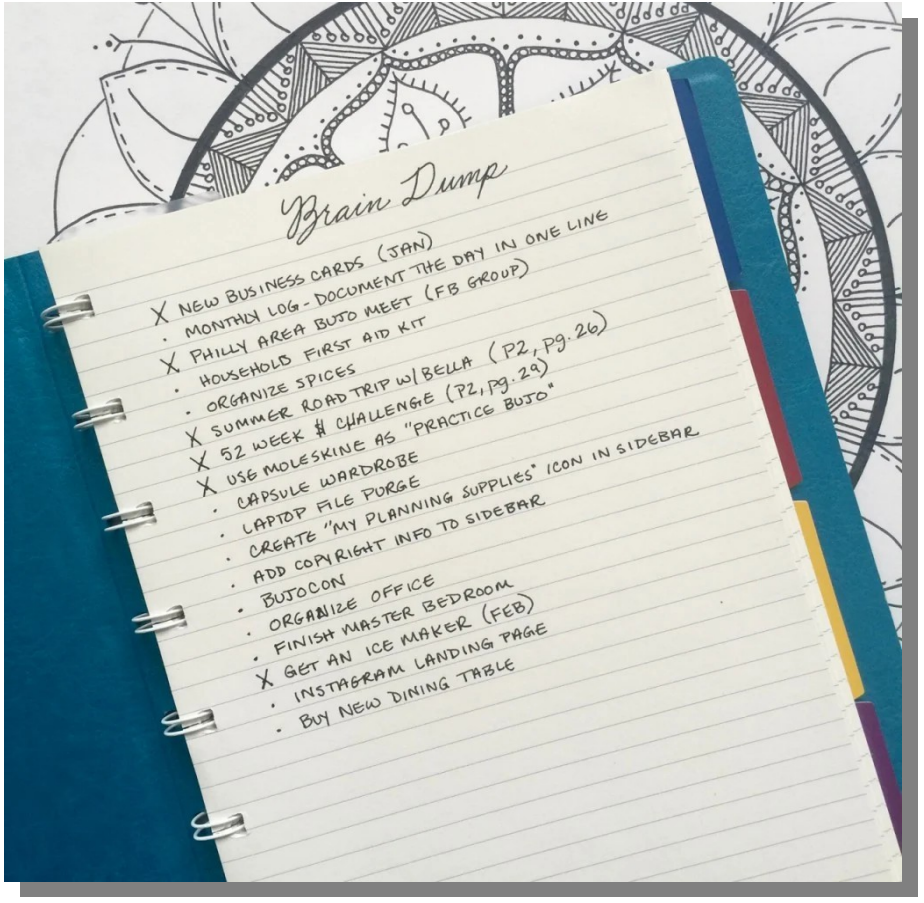
Real Life Lists

Real Life Lists

Real Life Lists



Real Life Lists



Real Life Lists

Would you like this one ?

```
Groceries List
==============

Apples, milk, rice, bananas, pizza, coffee, chewing gum,
                          tomatoes, pasta, bread, oranges (5),
                  potatoes, chips, Belgian beer, water
```

So

- Why on earth do we think the following are good lists
  - Yes, a **function** has a **LIST of parameters**

Gcc source file

```
84  extern int filename_cmp (const char *s1, const char *s2);
85  #define FILENAME_CMP(s1, s2)»   filename_cmp(s1, s2)
86
87  extern int filename_ncmp (const char *s1, const char *s2,
88  »   »   »     size_t n);
89
90  extern hashval_t filename_hash (const void *s);
91
92  extern int filename_eq (const void *s1, const void *s2);
93
94  extern int canonical_filename_eq (const char *a, const char *b);
95
```

Gcc source file

```
1146
1147  const char *
1148  _cpp_find_header_unit (cpp_reader *pfile, const char *name, bool angle,
1149  »   »           location_t loc)
1150  {
1151    if (_cpp_file *file = test_header_unit (pfile, name, angle, loc))
1152      {
1153        if (file->fd > 0)
1154  »     {
1155  »       /* Don't leave it open.  */
```

Fmt library source file

```
374
375  namespace detail {
376
377  const char* parse_scan_specs(const char* begin, const char* end,
378                               format_specs& specs, scan_type) {
379    while (begin != end) {
380      switch (to_ascii(*begin)) {
381      // TODO: parse more scan format specifiers
382      case 'x':
383        specs.set_type(presentation_type::hex);
```

Poco source file

```
77
78      pos_type seekoff(off_type off, std::ios_base::seekdir way, std::ios_base::openmode which = std::ios_base::in | std::ios_base::out) override
79      {
80          const pos_type fail = off_type(-1);
81          off_type newoff = off_type(-1);
82
83          if ((which & std::ios_base::in) != 0)
84          {
85              if (this->gptr() == nullptr)
86                  return fail;
87
88              if (way == std::ios_base::beg)
89              {
90                  newoff = 0;
91              }
92              else if (way == std::ios_base::cur)
```

Boost source file – web doc

async_connect (3 of 4 overloads)

Asynchronously establishes a socket connection by trying each endpoint in a sequence.

```
template<
    typename Protocol,
    typename Executor,
    typename EndpointSequence,
    typename ConnectCondition,
    typename RangeConnectToken = default_completion_token_t<Executor>>
DEDUCED async_connect(
    basic_socket< Protocol, Executor > & s,
    const EndpointSequence & endpoints,
    ConnectCondition connect_condition,
    RangeConnectToken && token = default_completion_token_t< Executor >(),
    constraint_t< is_endpoint_sequence< EndpointSequence >::value >  = 0,
    constraint_t< is_connect_condition< ConnectCondition, decltype(declval< const EndpointSequence & >().begin())>::value >= 0);
```

Boost source file - reality

```
849   */
850   template <typename Protocol, typename Executor,
851       typename EndpointSequence, typename ConnectCondition,
852       BOOST_ASIO_COMPLETION_TOKEN_FOR(void (boost::system::error_code,
853         typename Protocol::endpoint)) RangeConnectToken
854           = default_completion_token_t<Executor>>
855   inline auto async_connect(basic_socket<Protocol, Executor>& s,
856       const EndpointSequence& endpoints, ConnectCondition connect_condition,
857       RangeConnectToken&& token = default_completion_token_t<Executor>(),
858       constraint_t<
859         is_endpoint_sequence<EndpointSequence>::value
860       > = 0,
861       constraint_t<
862         is_connect_condition<ConnectCondition,
863           decltype(declval<const EndpointSequence&>().begin())>::value
864       > = 0)
865     -> decltype(
866       async_initiate<RangeConnectToken,
867         void (boost::system::error_code, typename Protocol::endpoint)>(
868           declval<detail::initiate_async_range_connect<Protocol, Executor>>(),
869           token, endpoints, connect_condition))
```

Well, you disagree ?

- Then I assume the following is also perfectly nice code
- A struct/class has a **LIST of members**
  - And you just allowed a list to be horribly formatted

Well, you disagree ?

Start here | *SQLParserResult.h

```
1  #ifndef SQLPARSER_SQLPARSER_RESULT_H
2  #define SQLPARSER_SQLPARSER_RESULT_H
3
4  #include "sqlparser_win.h"
5  #include "sql/SQLStatement.h"
6
7  namespace hsql {
8  // Represents the result of the SQLParser.
9  // If parsing was successful it contains a list of SQLStatement.
10 class SQLParser_API SQLParserResult {
11 /// ...
12
13  private:
14
15   mutable std::vector<SQLStatement*>* statements_; bool isValid_; char* errorMsg_; int errorLine_;
16                                int errorColumn_; std::vector<Expr*>* parameters_;
17 };
18
19 }  // namespace hsql
20
21 #endif  // SQLPARSER_SQLPARSER_RESULT_H
```

Well, you disagree ?

```
#ifndef BOOST_REDIS_CONFIG_HPP
#define BOOST_REDIS_CONFIG_HPP

#include <boost/redis/request.hpp>

#include <chrono>
#include <limits>
#include <optional>
#include <string>

namespace boost::redis {

/// Address of a Redis server.
struct address {
   /// Redis host.
   std::string host = "127.0.0.1";
   /// Redis port.
   std::string port = "6379";
};

/// Configure parameters used by the connection classes.
struct config {
   bool use_ssl = false; address addr = address{"127.0.0.1", "6379"}; std::string unix_socket; std::string username = "default"; std::string password;
   std::string clientname = "Boost.Redis"; std::optional<int> database_index = 0;std::string health_check_id = "Boost.Redis"; std::string log_prefix = "(Boost.Redis) ";
   std::chrono::steady_clock::duration resolve_timeout = std::chrono::seconds{10}; std::chrono::steady_clock::duration connect_timeout = std::chrono::seconds{10};
   std::chrono::steady_clock::duration ssl_handshake_timeout = std::chrono::seconds{10}; std::chrono::steady_clock::duration health_check_interval = std::chrono::seconds{2};
   std::chrono::steady_clock::duration reconnect_wait_interval = std::chrono::seconds{1}; std::size_t max_read_size = (std::numeric_limits<std::size_t>::max)();
   std::size_t read_buffer_append_size = 4096; bool use_setup = false; request setup = detail::make_hello_request();
};

}  // namespace boost::redis

#endif  // BOOST_REDIS_CONFIG_HPP
```

1 declaration on a line

- int x =0;
- double y = 0.0;
- The same applies to the declaration of a function, and it's arguments
- void foo(
  int x,
  int y);
- And to template parameters

We do not want to play

- The ‘find the comma game’
- The ‘find the semicolon game’

- Note: when you age, this game becomes even harder ;-)

Grumpy old man complains

- Lists are lists
- Human readable lists
- Stay on the left
  - Diff friendly
  - No sore neck from a tennis game in code (left, right, left, right)
- 1 item per line
  - Human readable list
  - Diff friendly

Kevlin’s talk

- **DevTernity Conference, December 2016, Riga**
- Seven Ineffective Coding Habits of Many Programmers
- Excellent talk, watch it completely please
- Argument Lists: around min. 20.00 – min 26.30
- **https://www.youtube.com/watch?v=SUIUZ09mnwM**
  - Or from DevWeek 2014: https://www.youtube.com/watch?v=168qgkBm8c0&t=2281s

QUESTIONS