

**1. License.**

Generated date: January 30, 2015

Copyright © 1998-2014 Dave Bone

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <http://mozilla.org/MPL/2.0/>.

## 2. Testdriver introduction.

Testout the lexical part of  $O_2$ . This is a quick and dirty program originally c++ but now literate? So don't critique me on its sparseness. Its header files etc are cloned from  $O_2$ 's implementation. Its logic is the same pattern as  $O_2$  program. Same notes and chords to parsing.

The driver uses the **test\_components.lex** grammar to parse and test out the data files inputs. It is equivalent to  $O_2$ 's **pass3.lex** token pass without the other lexical components. In `/usr/local/yacco2/qa`, there are various dat files acting as exercisers for testdriver. `/usr/local/yacco2/qa/o2testdriver_tests.bat` bash script runs the various tests.

The claim-to-fame for **test\_components.lex** is how the **arbitrator-code** is placed within a rule that has direct thread calls. Rule **Rtoken** closures these other rules that directly call threads. Due to current  $O_2$  implementation, this is a requirement to gen the arbitrator procedure. Why not have a read: `/usr/local/yacco2/docs/test_components.pdf` with its enlightened comments.

## 3. Append header files for testdriver environment.

```
< accrue testdriver code 3 > ≡
#include "globals.h"
#include "yacco2_stbl.h"
    using namespace yacco2_stbl;
#include "test_components.h"
#include "o2_extrns.h"
```

See also sections 4 and 5.

This code is used in section 6.

**4. Define variables.**

```
( accrue testdriver code 3 ) +≡  
int RECURSION_INDEX__(0);  
YACCO2_define_trace_variables();  
yacco2 :: TOKEN_GAGGLEP3_tokens;  
yacco2 :: TOKEN_GAGGLEError_queue;  
yacco2 :: TOKEN_GAGGLEtc_recycle_bin;  
T_sym_tbl_report_card report_card;  
std :: string tc_file_to_compile;  
STBL_T_ITEMS_type STBL_T_ITEMS;  
yacco2 :: CHART_SW('n');  
yacco2 :: CHARERR_SW('n');  
yacco2 :: CHARPRT_SW('n');  
STATES_type LR1_STATES;  
LR1_STATES_type LR1_COMMON_STATES;  
bool LR1_HEALTH(LR1_COMPATIBLE);  
int NO_LR1_STATES(0);  
STATES_SET_type VISITED_MERGE_STATES_IN_LA_CALC;  
CYCLIC_USE_TBL_type CYCLIC_USE_TABLE;
```

## 5. Mainline.

```

⟨accrue testdriver code 3⟩ +=
int main(int argc, char *argv[])
{
    cout << "Testdriver_start" << endl;
    yacco2::lrclog << "Testdriver_start" << endl;
    LOAD_YACCO2_KEYWORDS_INTO_STBL();
    GET_CMD_LINE(argc, argv, Yacco2_holding_file, Error_queue);
    if (Error_queue.empty() ≠ true) {
        DUMP_ERROR_QUEUE(Error_queue);
        return -1;
    }
    YACCO2_PARSE_CMD_LINE(T_SW, ERR_SW, PRT_SW, tc_file_to_compile, Error_queue);
    if (Error_queue.empty() ≠ true) {
        DUMP_ERROR_QUEUE(Error_queue);
        return -1;
    }
    tok_can < std::ifstream > cmd_line(tc_file_to_compile.c_str());
    if (cmd_line.file_ok() ≡ NO) {
        cout << "Error_occurred_file_does_not_exist:" << tc_file_to_compile.c_str() << endl;
        return 1;
    }
    /* yacco2::YACCO2_TH__ = 1; //watch test_component.lex parse */
    /* yacco2::YACCO2_T__ = 1; //watch terminals as fetched */
    yacco2::YACCO2_AR__ = 1; /* watch arbitration */
    /* yacco2::YACCO2_MSG__ = 1; //watch communication between threads */
    using namespace NS_test_components;

    Ctest_components tc_fsm;
    Parser tc(tc_fsm, &cmd_line, &P3_tokens, 0, &Error_queue, &tc_recycle_bin, 0);
    tc.parse();
    yacco2::TOKEN_GAGGLE::iterator i = P3_tokens.begin();
    yacco2::TOKEN_GAGGLE::iterator ie = P3_tokens.end();
    cout << "Dump_of_P3_tokons" << endl;
    for (int yyy = 1; i ≠ ie; ++i) {
        CAbs_lr1_sym * sym = *i;
        if (sym ≡ yacco2::PTR_LR1_eog_) continue;
        cout << yyy << "::-:" << sym-id() << "_file_no:" << sym-external_file_id() << "_line_no:" <<
            sym-line_no() << "_pos:" << sym-pos_in_line() << endl;
        yacco2::lrclog << yyy << "::-:" << sym-id() << "_file_no:" << sym-external_file_id() <<
            "_line_no:" << sym-line_no() << "_pos:" << sym-pos_in_line() << endl;
        ++yyy;
    }
    if (Error_queue.empty() ≠ true) {
        DUMP_ERROR_QUEUE(Error_queue);
        cout << "Testdriver_exit" << endl;
        yacco2::lrclog << "Testdriver_exit" << endl;
        return -1;
    }
    /* yacco2::Delete_tokens(Error_queue, true); */
    exit: cout << "Testdriver_exit" << endl;
    yacco2::lrclog << "Testdriver_exit" << endl;
    return 0;
}

```

```
}
```

**6. Testdriver implementation.**

Output program to *testdriver.cpp*.

```
<testdriver.cpp 6> ≡  
  <accrue testdriver code 3>;
```

**7. Index.**

*argc*: [5](#).  
*argv*: [5](#).  
*begin*: [5](#).  
*c\_str*: [5](#).  
*CAbs\_lr1\_sym*: [5](#).  
**CHAR**: [4](#).  
*cmd\_line*: [5](#).  
*cout*: [5](#).  
*cpp*: [6](#).  
*Ctest\_components*: [5](#).  
**CYCLIC\_USE\_TABLE**: [4](#).  
*CYCLIC\_USE\_TBL\_type*: [4](#).  
*Delete\_tokens*: [5](#).  
**DUMP\_ERROR\_QUEUE**: [5](#).  
*empty*: [5](#).  
*end*: [5](#).  
*endl*: [5](#).  
**ERR\_SW**: [4](#), [5](#).  
*Error\_queue*: [4](#), [5](#).  
*exit*: [5](#).  
*external\_file\_id*: [5](#).  
*file\_ok*: [5](#).  
**GET\_CMD\_LINE**: [5](#).  
*id*: [5](#).  
*ie*: [5](#).  
*ifstream*: [5](#).  
*iterator*: [5](#).  
*line\_no*: [5](#).  
**LOAD\_YACCO2\_KEYWORDS\_INTO\_STBL**: [5](#).  
*lrlog*: [5](#).  
**LR1\_COMMON\_STATES**: [4](#).  
**LR1\_COMPATIBLE**: [4](#).  
**LR1\_HEALTH**: [4](#).  
**LR1\_STATES**: [4](#).  
*LR1\_STATES\_type*: [4](#).  
*main*: [5](#).  
**NO**: [5](#).  
**NO\_LR1\_STATES**: [4](#).  
**NS\_test\_components**: [5](#).  
*parse*: [5](#).  
*Parser*: [5](#).  
*pos\_in\_line*: [5](#).  
**PRT\_SW**: [4](#), [5](#).  
*PTR\_LR1\_eog\_*: [5](#).  
*P3\_tokens*: [4](#), [5](#).  
**RECURSION\_INDEX\_**: [4](#).  
*report\_card*: [4](#).  
*STATES\_SET\_type*: [4](#).  
*STATES\_type*: [4](#).  
**STBL\_T\_ITEMS**: [4](#).  
*STBL\_T\_ITEMS\_type*: [4](#).  
*std*: [4](#), [5](#).  
*string*: [4](#).  
*sym*: [5](#).  
**T\_SW**: [4](#), [5](#).  
*T\_sym\_tbl\_report\_card*: [4](#).  
*tc*: [5](#).  
*tc\_file\_to\_compile*: [4](#), [5](#).  
*tc\_fsm*: [5](#).  
*tc\_recycle\_bin*: [4](#), [5](#).  
*testdriver*: [6](#).  
*tok\_can*: [5](#).  
**TOKEN\_GAGGLE**: [4](#), [5](#).  
*true*: [5](#).  
**VISITED\_MERGE\_STATES\_IN\_LA\_CALC**: [4](#).  
*yacco2*: [4](#), [5](#).  
**YACCO2\_AR\_**: [5](#).  
*YACCO2\_define\_trace\_variables*: [4](#).  
*Yacco2\_holding\_file*: [5](#).  
**YACCO2\_MSG\_**: [5](#).  
**YACCO2\_PARSE\_CMD\_LINE**: [5](#).  
**yacco2\_stbl**: [3](#).  
**YACCO2\_T\_**: [5](#).  
**YACCO2\_TH\_**: [5](#).  
*yyy*: [5](#).

⟨accrue testdriver code 3, 4, 5⟩ Used in section 6.  
⟨testdriver.cpp 6⟩

# TESTDRIVER

	Section	Page
<b>License</b> .....	<a href="#">1</a>	1
<b>Testdriver introduction</b> .....	<a href="#">2</a>	2
Append header files for testdriver environment .....	<a href="#">3</a>	2
<b>Define variables</b> .....	<a href="#">4</a>	3
<b>Mainline</b> .....	<a href="#">5</a>	4
Testdriver implementation .....	<a href="#">6</a>	5
<b>Index</b> .....	<a href="#">7</a>	6