1. Copyright.

2. Error symbols vocabulary.
Error symbols for the telling.
Due to my verbosity and *cweb*'s (TEX) black slug output some of my symbol keys are more cryptic in description as i wanted to title them with their enumeration symbol. Now, the enumeration symbol is removed from the index so there is no blob of ... If the truth be told, these descriptions should be indirect key values that allow for multilingual translation. One step at a time...

Upon experimentation with error keys having a suffixed Exxx indicator in the key itself whereby the xxx is a number: 001 and associating a translation file with the error construct, i nixed the ideas. It is simply a post error evaluation on the given error keys to translate them using some form of associated map. The error enumeration key with a language indicator would suffice. A simple database could be used. For example T_Enum::T_LR1_err_nested_files_exceeded_ with a country code of "DE" for German would translate "nested files exceeded". Not finding a translation should default to the error token's literal key. See grammar "o2_err_hdr.lex" for a context sensitive approach to error reporting.

Ahh... grandeur. Now what about unicode?

3. # T in list not eq.
Enum: T_Err_no_of_native_Ts_in_list_not_equal_
Class: Err_no_of_native_Ts_in_list_not_equal AB: N AD: N

---

4. # T in list not eq user-declaration directive.

⟨ # T in list not eq user-declaration directive 4 ⟩ ≡
```
public: Err_no_of_native_Ts_in_list_not_equal(unsigned long Value);
  unsigned long no( ) const;
private: unsigned long no_;
```

5. # T in list not eq user-implementation directive.

⟨ # T in list not eq user-implementation directive 5 ⟩ ≡
```
Err_no_of_native_Ts_in_list_not_equal::Err_no_of_native_Ts_in_list_not_equal(unsigned long
        No)T_CTOR("no␣terminals␣in␣list␣not␣equal,␣chk␣items␣in␣list",
        T_Enum::T_Err_no_of_native_Ts_in_list_not_equal_, 0, false, false)
{
  no_ = No;
}
unsigned long Err_no_of_native_Ts_in_list_not_equal::no( ) const
{
  return no_;
}
```

6. # threads in list not eq.
Enum: T_Err_no_of_threads_in_list_not_equal_
Class: Err_no_of_threads_in_list_not_equal AB: N AD: N

---

7. # threads in list not eq user-declaration directive.

⟨ # threads in list not eq user-declaration directive 7 ⟩ ≡
```
public: Err_no_of_threads_in_list_not_equal(unsigned long No);
  unsigned long no( ) const;
private: unsigned long no_;
```

8. # threads in list not eq user-implementation directive.

```
⟨ # threads in list not eq user-implementation directive 8 ⟩ ≡
  Err_no_of_threads_in_list_not_equal::Err_no_of_threads_in_list_not_equal(unsigned long
          No)T_CTOR("no␣threads␣in␣list␣not␣equal,␣chk␣list",
          T_Enum::T_Err_no_of_threads_in_list_not_equal_, 0, false, false)
  {
    no_ = No;
  }
  unsigned long Err_no_of_threads_in_list_not_equal::no( ) const
  {
    return no_;
  }
```

9. ? ended subrule expr.
Enum: T_Err_not_T_or_R_or_eos_in_subrule_expr_
Class: Err_not_T_or_R_or_eos_in_subrule_expr AB: N AD: N

---

10. Empty file no grammar constructs present.
Enum: T_Err_empty_file_
Class: Err_empty_file AB: N AD: N

---

11. O2 epsilon badly gened: 0 items in fsc lists.
Enum: T_Err_epsilon_pass_thru_
Class: Err_epsilon_pass_thru AB: N AD: N
O_2 states that a thread has epsilon pass thru of thread calling. Example the called thread start state has an epsilon subrule. Sanity check or paranoia on my work.

---

12. O2 epsilon: T present, but no T list.
Enum: T_Err_epsilon_pass_thru_no_Ts_
Class: Err_epsilon_pass_thru_no_Ts AB: N AD: N
O_2 states that a thread has epsilon pass thru of thread calling. Example the called thread start state has an epsilon subrule. Sanity check or paranoia on my work.

---

13. T in list not defined in T-alphabet.
Enum: T_Err_bad_T_in_list_
Class: Err_bad_T_in_list AB: N AD: N
O_2^{linker}message. It indicates that the terminal is not defined in "T-alphabet" construct. This is the T vocabulary. If the terminal is not defined, it means that the grammar writer has not generated the vocabulary. So turn on the /t option and re-compile a grammar thus creating the updated list.

---

14. T not found in stbl.
Enum: T_Err_T_not_in_stbl_
Class: Err_T_not_in_stbl AB: N AD: N
Either the grammmar writer forgot to add a new T definition or its a mistype.

---

15. T not returned from a thread.
Enum: T_Err_not_T_for_rtned_token_from_th_
Class: Err_not_T_for_rtned_token_from_th AB: N AD: N

---

16. T-alphabet file does not exist.
Enum: T_Err_T_alphabet_file_does_not_exist_
Class: Err_T_alphabet_file_does_not_exist AB: N AD: N

---

17. T-alphabet file not present.
Enum: T_Err_T_alphabet_file_not_present_
Class: Err_T_alphabet_file_not_present AB: N AD: N

---

18. T-alphabet kw not present.
Enum: T_Err_T_alphabet_kw_not_present_
Class: Err_T_alphabet_kw_not_present AB: N AD: N

---

19. already defined AB tag.
Enum: T_Err_already_defined_AB_
Class: Err_already_defined_AB AB: N AD: N

---

20. already defined AD tag.
Enum: T_Err_already_defined_AD_
Class: Err_already_defined_AD AB: N AD: N

---

21. bad char.
Enum: T_Err_bad_char_
Class: Err_bad_char AB: N AD: N

---

22. bad char destructor directive.

⟨ bad char destructor directive 22 ⟩ ≡
```
if (R→bad_char_ ≠ 0) delete R→bad_char( );
```

23. bad char user-declaration directive.

⟨ bad char user-declaration directive 23 ⟩ ≡
public: *Err_bad_char*(*CAbs_lr1_sym* ∗ *Err_char*);
yacco2 :: *CAbs_lr1_sym* ∗ *bad_char*( ) **const**;
void *zero_out_bad_char*( );
private: *yacco2* :: *CAbs_lr1_sym* ∗ *bad_char_*;

24. bad char user-implementation directive.

⟨ bad char user-implementation directive 24 ⟩ ≡
Err_bad_char :: *Err_bad_char*(*CAbs_lr1_sym* ∗ *Bad_char*)`T_CTOR("bad␣char"`, *T_Enum* :: *T_Err_bad_char_*, &*dtor_Err_bad_char*, *false*, *false*)
{
*bad_char_* = *Bad_char*;
}
yacco2 :: *CAbs_lr1_sym* ∗ *Err_bad_char* :: *bad_char*( ) **const**
{
return *bad_char_*;
}
void *Err_bad_char* :: *zero_out_bad_char*( )
{
*bad_char_* = 0;
}

25. bad cmd-opt.
Enum: T_Err_bad_cmd_lne_opt_
Class: Err_bad_cmd_lne_opt AB: N AD: N

---

26. bad directive.
Enum: T_Err_bad_directive_
Class: Err_bad_directive AB: N AD: N

---

27. bad eos.
Enum: T_Err_bad_eos_
Class: Err_bad_eos AB: N AD: N

---

28. bad esc.
Enum: T_Err_bad_esc_
Class: Err_bad_esc AB: N AD: N

---

29. bad filename.
Enum: T_Err_bad_filename_
Class: Err_bad_filename AB: N AD: N

---

30. bad filename user-declaration directive.

⟨ bad filename user-declaration directive 30 ⟩ ≡
```
public: Err_bad_filename(std::string & File_name);
  Err_bad_filename(const char *File_name);
  std::string * file_name( );
private: std::string file_name_;
```

31. bad filename user-implementation directive.

⟨ bad filename user-implementation directive 31 ⟩ ≡
```
  Err_bad_filename::Err_bad_filename(std::string & File_name)T_CTOR("bad␣filename",
          T_Enum::T_Err_bad_filename_, 0, false, false)
  {
    file_name_ += File_name.c_str( );
  }
  Err_bad_filename::Err_bad_filename(const char *File_name)T_CTOR("bad␣filename",
          T_Enum::T_Err_bad_filename_, 0, false, false)
  {
    file_name_ += File_name;
  }
  std::string * Err_bad_filename::file_name( )
  {
    return &file_name_;
  }
```

32. bad filename for Errors vocabulary header.

Enum: T_Err_bad_errors_hdrfilename_
Class: Err_bad_errors_hdrfilename AB: N AD: N

---

33. bad filename for Errors vocabulary header user-declaration directive.

⟨ bad filename for Errors vocabulary header user-declaration directive 33 ⟩ ≡
```
public: Err_bad_errors_hdrfilename(std::string & File_name);
  Err_bad_errors_hdrfilename(const char *File_name);
  std::string * file_name( );
private: std::string file_name_;
```

34. bad filename for Errors vocabulary header user-implementation directive.

⟨ bad filename for Errors vocabulary header user-implementation directive 34 ⟩ ≡

```
Err_bad_errors_hdrfilename::Err_bad_errors_hdrfilename(std::string &
        File_name)T_CTOR("bad␣filename␣for␣Errors␣vocabulary␣header",
        T_Enum::T_Err_bad_errors_hdrfilename_,0,false,false)
{
  file_name_ += File_name.c_str();
}
Err_bad_errors_hdrfilename::Err_bad_errors_hdrfilename(const char
        *File_name)T_CTOR("bad␣filename␣for␣Errors␣vocabulary␣header",
        T_Enum::T_Err_bad_errors_hdrfilename_,0,false,false)
{
  file_name_ += File_name;
}
std::string * Err_bad_errors_hdrfilename::file_name()
{
  return &file_name_;
}
```

35. bad filename for Errors vocabulary implementation.

Enum: T_Err_bad_errors_impfilename_

Class: Err_bad_errors_impfilename AB: N AD: N

---

36. bad filename for Errors vocabulary implementation user-declaration directive.

⟨ bad filename for Errors vocabulary implementation user-declaration directive 36 ⟩ ≡

```
public: Err_bad_errors_impfilename(std::string & File_name);
  Err_bad_errors_impfilename(const char *File_name);
  std::string * file_name();
private: std::string file_name_;
```

37. bad filename for Errors vocabulary implementation user-implementation directive.

⟨ bad filename for Errors vocabulary implementation user-implementation directive 37 ⟩ ≡

```
Err_bad_errors_impfilename::Err_bad_errors_impfilename(std::string &
        File_name)T_CTOR("bad␣filename␣for␣Errors␣vocabulary␣implementation",
        T_Enum::T_Err_bad_errors_impfilename_,0,false,false)
{
  file_name_ += File_name.c_str();
}
Err_bad_errors_impfilename::Err_bad_errors_impfilename(const char
        *File_name)T_CTOR("bad␣filename␣for␣Errors␣vocabulary␣implementation",
        T_Enum::T_Err_bad_errors_impfilename_,0,false,false)
{
  file_name_ += File_name;
}
std::string * Err_bad_errors_impfilename::file_name()
{
  return &file_name_;
}
```

38. bad filename to output cpp.
Enum: T_Err_bad_fsmcpp_filename_
Class: Err_bad_fsmcpp_filename AB: N AD: N

---

39. bad filename to output cpp user-declaration directive.

⟨ bad filename to output cpp user-declaration directive 39 ⟩ ≡
```
public: Err_bad_fsmcpp_filename(std::string & File_name);
  Err_bad_fsmcpp_filename(const char *File_name);
  std::string * file_name( );
private: std::string file_name_;
```

40. bad filename to output cpp user-implementation directive.

⟨ bad filename to output cpp user-implementation directive 40 ⟩ ≡
```
  Err_bad_fsmcpp_filename::Err_bad_fsmcpp_filename(std::string &
          File_name)T_CTOR("bad filename to output cpp", T_Enum::T_Err_bad_fsmcpp_filename_, 0,
          false, false)
  {
    file_name_ += File_name.c_str( );
  }
  Err_bad_fsmcpp_filename::Err_bad_fsmcpp_filename(const char
          *File_name)T_CTOR("bad filename to output cpp", T_Enum::T_Err_bad_fsmcpp_filename_,
          0, false, false)
  {
    file_name_ += File_name;
  }
  std::string * Err_bad_fsmcpp_filename::file_name( )
  {
    return &file_name_;
  }
```

41. bad filename to output enumeration header.
Enum: T_Err_bad_enum_filename_
Class: Err_bad_enum_filename AB: N AD: N

---

42. bad filename to output enumeration header user-declaration directive.

⟨ bad filename to output enumeration header user-declaration directive 42 ⟩ ≡
```
public: Err_bad_enum_filename(std::string & File_name);
  Err_bad_enum_filename(const char *File_name);
  std::string * file_name( );
private: std::string file_name_;
```

43. bad filename to output enumeration header user-implementation directive.

```
⟨ bad filename to output enumeration header user-implementation directive 43 ⟩ ≡
  Err_bad_enum_filename::Err_bad_enum_filename(std::string &
          File_name)T_CTOR("bad␣filename␣to␣output␣enumeration␣header",
          T_Enum::T_Err_bad_enum_filename_,0,false,false)
  {
    file_name_ += File_name.c_str();
  }
  Err_bad_enum_filename::Err_bad_enum_filename(const char
          *File_name)T_CTOR("bad␣filename␣to␣output␣enumeration␣header",
          T_Enum::T_Err_bad_enum_filename_,0,false,false)
  {
    file_name_ += File_name;
  }
  std::string * Err_bad_enum_filename::file_name()
  {
    return &file_name_;
  }
```

44. bad filename to output grammar header.

Enum: T_Err_bad_fsmheader_filename_
Class: Err_bad_fsmheader_filename AB: N AD: N

---

45. bad filename to output grammar header user-declaration directive.

```
⟨ bad filename to output grammar header user-declaration directive 45 ⟩ ≡
public: Err_bad_fsmheader_filename(std::string & File_name);
  Err_bad_fsmheader_filename(const char *File_name);
  std::string * file_name();
private: std::string file_name_;
```

46. bad filename to output grammar header user-implementation directive.

```
⟨ bad filename to output grammar header user-implementation directive 46 ⟩ ≡
  Err_bad_fsmheader_filename::Err_bad_fsmheader_filename(std::string &
          File_name)T_CTOR("bad␣filename␣to␣output␣grammar␣header",
          T_Enum::T_Err_bad_fsmheader_filename_,0,false,false)
  {
    file_name_ += File_name.c_str();
  }
  Err_bad_fsmheader_filename::Err_bad_fsmheader_filename(const char
          *File_name)T_CTOR("bad␣filename␣to␣output␣grammar␣header",
          T_Enum::T_Err_bad_fsmheader_filename_,0,false,false)
  {
    file_name_ += File_name;
  }
  std::string * Err_bad_fsmheader_filename::file_name()
  {
    return &file_name_;
  }
```

47. bad filename to output sym.
Enum: T_Err_bad_fsmsym_filename_
Class: Err_bad_fsmsym_filename AB: N AD: N

---

48. bad filename to output sym user-declaration directive.

⟨ bad filename to output sym user-declaration directive 48 ⟩ ≡
```
public: Err_bad_fsmsym_filename(std::string & File_name);
  Err_bad_fsmsym_filename(const char *File_name);
  std::string * file_name( );
private: std::string file_name_;
```

49. bad filename to output sym user-implementation directive.

⟨ bad filename to output sym user-implementation directive 49 ⟩ ≡
```
  Err_bad_fsmsym_filename::Err_bad_fsmsym_filename(std::string &
          File_name)T_CTOR("bad␣filename␣to␣output␣sym", T_Enum::T_Err_bad_fsmsym_filename_, 0,
          false, false)
  {
    file_name_ += File_name.c_str( );
  }
  Err_bad_fsmsym_filename::Err_bad_fsmsym_filename(const char
          *File_name)T_CTOR("bad␣filename␣to␣output␣sym", T_Enum::T_Err_bad_fsmsym_filename_,
          0, false, false)
  {
    file_name_ += File_name;
  }
  std::string * Err_bad_fsmsym_filename::file_name( )
  {
    return &file_name_;
  }
```

50. bad filename to output tbl.
Enum: T_Err_bad_fsmtbl_filename_
Class: Err_bad_fsmtbl_filename AB: N AD: N

---

51. bad filename to output tbl user-declaration directive.

⟨ bad filename to output tbl user-declaration directive 51 ⟩ ≡
```
public: Err_bad_fsmtbl_filename(std::string & File_name);
  Err_bad_fsmtbl_filename(const char *File_name);
  std::string * file_name( );
private: std::string file_name_;
```

52. bad filename to output tbl user-implementation directive.

⟨ bad filename to output tbl user-implementation directive 52 ⟩ ≡

```
Err_bad_fsmtbl_filename::Err_bad_fsmtbl_filename(std::string &
        File_name)T_CTOR("bad␣filename␣to␣output␣tbl", T_Enum::T_Err_bad_fsmtbl_filename_, 0,
        false, false)
{
  file_name_ += File_name.c_str();
}
Err_bad_fsmtbl_filename::Err_bad_fsmtbl_filename(const char
        *File_name)T_CTOR("bad␣filename␣to␣output␣tbl", T_Enum::T_Err_bad_fsmtbl_filename_, 0,
        false, false)
{
  file_name_ += File_name;
}
std::string * Err_bad_fsmtbl_filename::file_name()
{
  return &file_name_;
}
```

53. bad int-no.
Enum: T_Err_bad_int_no_
Class: Err_bad_int_no AB: N AD: N

---

54. bad int-no range.
Enum: T_Err_bad_int_no_range_
Class: Err_bad_int_no_range AB: N AD: N

---

55. bad operator la expr: not a + or -.
Enum: T_Err_bad_operator_in_la_expr_
Class: Err_bad_operator_in_la_expr AB: N AD: N
Thread's lookahead expression malformed.

---

56. bad pos of t def.
Enum: T_Err_cannot_define_term_after_sufx_dir_
Class: Err_cannot_define_term_after_sufx_dir AB: N AD: N

---

57. bad pos of terminals-refs.
Enum: T_Err_terminals_refs_def_after_terminals_
Class: Err_terminals_refs_def_after_terminals AB: N AD: N

---

58. bad term in la expr.
Enum: T_Err_bad_term_in_la_expr_
Class: Err_bad_term_in_la_expr AB: N AD: N
Must be a Rule or a T term within the subrule expression.

---

59. bad term in subrule expr.
Enum: T_Err_not_T_or_R_in_subrule_expr_
Class: Err_not_T_or_R_in_subrule_expr AB: N AD: N

---

60. bad thread expr.
Enum: T_Err_bad_th_qualifier_
Class: Err_bad_th_qualifier AB: N AD: N

---

61. bad univ-seq.
Enum: T_Err_bad_univ_seq_
Class: Err_bad_univ_seq AB: N AD: N

---

62. bad univ-seq user-declaration directive.

⟨ bad univ-seq user-declaration directive 62 ⟩ ≡
public: *Err_bad_univ_seq*(**unsigned long** *Value*);
unsigned long *bad_univ_seq*( ) **const**;
private: **unsigned long** *bad_univ_seq_*;

63. bad univ-seq user-implementation directive.

⟨ bad univ-seq user-implementation directive 63 ⟩ ≡
Err_bad_univ_seq :: *Err_bad_univ_seq*(**unsigned long** *Bad_univ_seq*)T_CTOR("bad␣univ␣seq",
T_Enum :: *T_Err_bad_univ_seq_*, 0, *false*, *false*)
{
*bad_univ_seq_* = *Bad_univ_seq*;
}
unsigned long *Err_bad_univ_seq* :: *bad_univ_seq*( ) **const**
{
return *bad_univ_seq_*;
}

64. command line chaffe.
Enum: T_Err_cmd_line_chaffe_
Class: Err_cmd_line_chaffe AB: N AD: N

---

65. comment-overrun.
Enum: T_Err_comment_overrun_
Class: Err_comment_overrun AB: N AD: N

---

66. dup ? sufx directive.
Enum: T_Err_dup_sufx_directive_
Class: Err_dup_sufx_directive AB: N AD: N

---

67. dup-entry in sym-table.
Enum: T_Err_dup_entry_in_sym_table_
Class: Err_dup_entry_in_sym_table AB: N AD: N

---

68. duplicate T_enum phrase.
Enum: T_Err_already_processed_T_enum_phase_
Class: Err_already_processed_T_enum_phase AB: N AD: N

---

69. duplicate directive.
Enum: T_Err_duplicate_directive_
Class: Err_duplicate_directive AB: N AD: N

---

70. duplicate error phrase.
Enum: T_Err_already_processed_error_phase_
Class: Err_already_processed_error_phase AB: N AD: N

---

71. duplicate fsm phrase.
Enum: T_Err_already_processed_fsm_phase_
Class: Err_already_processed_fsm_phase AB: N AD: N

---

72. duplicate logical name.
Enum: T_Err_duplicate_logical_name_
Class: Err_duplicate_logical_name AB: N AD: N

---

73. duplicate lrk phrase.
Enum: T_Err_already_processed_lrk_phase_
Class: Err_already_processed_lrk_phase AB: N AD: N

---

74. duplicate parallel-parse phrase.
Enum: T_Err_already_processed_pp_phase_
Class: Err_already_processed_pp_phase AB: N AD: N

---

75. duplicate raw character phrase.
Enum: T_Err_already_processed_rc_phase_
Class: Err_already_processed_rc_phase AB: N AD: N

---

76. duplicate rule phrase.
Enum: T_Err_already_processed_rule_phase_
Class: Err_already_processed_rule_phase AB: N AD: N

---

77. duplicate terminal phrase.
Enum: T_Err_already_processed_T_phase_
Class: Err_already_processed_T_phase AB: N AD: N

---

78. duplicate-entry in alphabet.
Enum: T_Err_dup_entry_in_alphabet_
Class: Err_dup_entry_in_alphabet AB: N AD: N

---

79. emitfile file does not exist.
Enum: T_Err_emitfile_file_does_not_exist_
Class: Err_emitfile_file_does_not_exist AB: N AD: N

---

80. emitfile file not present.
Enum: T_Err_emitfile_file_not_present_
Class: Err_emitfile_file_not_present AB: N AD: N

---

81. emitfile kw not present.
Enum: T_Err_emitfile_kw_not_present_
Class: Err_emitfile_kw_not_present AB: N AD: N

---

82. file of T-alphabet not present.
Enum: T_Err_file_of_T_alphabet_not_present_
Class: Err_file_of_T_alphabet_not_present AB: N AD: N

---

83. file-name file does not exist.
Enum: T_Err_filename_file_does_not_exist_
Class: Err_filename_file_does_not_exist AB: N AD: N

---

84. fsc file does not exist.
Enum: T_Err_fsc_file_does_not_exist_
Class: Err_fsc_file_does_not_exist AB: N AD: N

---

85. fsc file not present.
Enum: T_Err_fsc_cntl_file_not_present_
Class: Err_fsc_cntl_file_not_present AB: N AD: N

---

86. improper closing of rules construct.
Enum: T_Err_improper_closing_of_rules_
Class: Err_improper_closing_of_rules AB: N AD: N

---

87. improper directive.
Enum: T_Err_improper_directive_
Class: Err_improper_directive AB: N AD: N

---

88. invalid fsm-debug value.
Enum: T_Err_fsm_debug_string_not_true_or_false_
Class: Err_fsm_debug_string_not_true_or_false AB: N AD: N

---

89. invalid logical name.
Enum: T_Err_invalid_logical_name_
Class: Err_invalid_logical_name AB: N AD: N

---

90. invalid logical value.
Enum: T_Err_invalid_logical_value_
Class: Err_invalid_logical_value AB: N AD: N

---

91. la expression calculates an empty set.
Enum: T_Err_la_expr_calc_empty_set_
Class: Err_la_expr_calc_empty_set AB: N AD: N

---

92. linker's monolithic value not n or y.
Enum: T_Err_monolithic_value_bad_
Class: Err_monolithic_value_bad AB: N AD: N

---

93. linker's transitive value not n or y.
Enum: T_Err_transitive_value_bad_
Class: Err_transitive_value_bad AB: N AD: N

---

94. misplaced or missing T enumeration phrase.
Enum: T_ERR_no_T_enum_phrase_
Class: ERR_no_T_enum_phrase AB: N AD: N

---

95. misplaced or missing errors phrase.
Enum: T_ERR_no_errors_phrase_
Class: ERR_no_errors_phrase AB: N AD: N

---

96. misplaced or missing fsm phrase.
Enum: T_ERR_no_fsm_phrase_
Class: ERR_no_fsm_phrase AB: N AD: N

---

97. misplaced or missing lrk phrase.
Enum: T_ERR_no_lrk_phrase_
Class: ERR_no_lrk_phrase AB: N AD: N

---

98. misplaced or missing rc phrase.
Enum: T_ERR_no_rc_phrase_
Class: ERR_no_rc_phrase AB: N AD: N

---

99. misplaced or missing rules phrase.
Enum: T_ERR_no_rules_phrase_
Class: ERR_no_rules_phrase AB: N AD: N

---

100. misplaced or missing terminals phrase.
Enum: T_ERR_no_terminals_phrase_
Class: ERR_no_terminals_phrase AB: N AD: N

---

101. misplaced or misspelt Rule or T outside of Rules defs.
Enum: T_Err_misplaced_or_misspelt_Rule_or_T_
Class: Err_misplaced_or_misspelt_Rule_or_T AB: N AD: N

Thrown by *pass3.lex* grammar. It is outside of a proper grammar's construct and does not have a symbol table entry. So it could be a misspelt T or an orphanned rule outside of its defining Rules construct.

---

102. missing lrk-sufx kw.
Enum: T_Err_missing_lrk_sufx_kw_
Class: Err_missing_lrk_sufx_kw AB: N AD: N

---

103. missing terminals-refs kw.
Enum: T_Err_missing_terminals_refs_kw_
Class: Err_missing_terminals_refs_kw AB: N AD: N

---

104. missing terminals-sufx kw.
Enum: T_Err_missing_terminals_sufx_kw_
Class: Err_missing_terminals_sufx_kw AB: N AD: N

---

105. nested files exceeded.
Enum: T_Err_nested_files_exceeded_
Class: Err_nested_files_exceeded AB: N AD: N

---

106. nested files exceeded user-declaration directive.

⟨ nested files exceeded user-declaration directive 106 ⟩ ≡

```
public: Err_nested_files_exceeded(yacco2::INT Nested_file_cnt, std::string & File_name);
  yacco2::INT nested_cnt( )
  {
    return nested_cnt_;
  }
  ;
  std::string * file_exceeded( )
  {
    return &file_exceeded_;
  }
  ;
private: yacco2::INT nested_cnt_;
  std::string file_exceeded_;
```

107. nested files exceeded user-implementation directive.

```
⟨ nested files exceeded user-implementation directive 107 ⟩ ≡
  Err_nested_files_exceeded :: Err_nested_files_exceeded(yacco2 :: INT Nested_cnt,
          std :: string & File_name)T_CTOR("nested␣files␣exceeded",
          T_Enum :: T_Err_nested_files_exceeded_, 0, false, false)
  {
    nested_cnt_ = Nested_cnt;
    file_exceeded_ += File_name.c_str( );
  }
```

108. no *.**

Enum: T_Err_no_syntax_code_end_present_
Class: Err_no_syntax_code_end_present AB: N AD: N

---

109. no T in T-alphabet list.

Enum: T_Err_no_terminals_in_T_alphabet_list_
Class: Err_no_terminals_in_T_alphabet_list AB: N AD: N

---

110. no Ts in T-alphabet.

Enum: T_Err_no_terminals_present_in_T_alphabet_
Class: Err_no_terminals_present_in_T_alphabet AB: N AD: N

---

111. no close-brace.

Enum: T_Err_no_close_brace_
Class: Err_no_close_brace AB: N AD: N

---

112. no close-parenthesis.

Enum: T_Err_no_close_parenthesis_
Class: Err_no_close_parenthesis AB: N AD: N

---

113. no closing brace ending rules defs.

Enum: T_Err_no_close_brace_ending_rules_defs_
Class: Err_no_close_brace_ending_rules_defs AB: N AD: N

---

114. no cmd-lne-data.

Enum: T_Err_no_cmd_lne_data_
Class: Err_no_cmd_lne_data AB: N AD: N

---

115. no comma present.
Enum: T_Err_no_comma_present_
Class: Err_no_comma_present AB: N AD: N

---

116. no constant-defs keyword present.
Enum: T_Err_no_kdefs_kw_present_
Class: Err_no_kdefs_kw_present AB: N AD: N

---

117. no constant-defs-code present.
Enum: T_Err_no_kdefs_code_present_
Class: Err_no_kdefs_code_present AB: N AD: N

---

118. no constant-defs-directive present.
Enum: T_Err_no_constant_defs_present_
Class: Err_no_constant_defs_present AB: N AD: N

---

119. no directive present.
Enum: T_Err_no_directive_present_
Class: Err_no_directive_present AB: N AD: N

---

120. no end-T-alphabet present.
Enum: T_Err_end_T_alphabet_kw_not_present_
Class: Err_end_T_alphabet_kw_not_present AB: N AD: N

---

121. no end-list-of-native....
Enum: T_Err_end_list_native_T_kw_not_present_
Class: Err_end_list_native_T_kw_not_present AB: N AD: N

---

122. no end-list-of-trans....
Enum: T_Err_end_list_of_transitive_threads_kw_not_present_
Class: Err_end_list_of_transitive_threads_kw_not_present AB: N AD: N

---

123. no end-of-code.
Enum: T_Err_no_end_of_code_
Class: Err_no_end_of_code AB: N AD: N

---

124. no end-preamble present.
Enum: T_Err_end_preamble_kw_not_present_
Class: Err_end_preamble_kw_not_present AB: N AD: N

---

125. no file-name kw present.
Enum: T_Err_filename_kw_not_present_
Class: Err_filename_kw_not_present AB: N AD: N

---

126. no file-name present.
Enum: T_Err_no_filename_present_
Class: Err_no_filename_present AB: N AD: N

---

127. no file-name value.
Enum: T_Err_filename_value_not_present_
Class: Err_filename_value_not_present AB: N AD: N

---

128. no file-name-id present.
Enum: T_Err_no_filename_id_present_
Class: Err_no_filename_id_present AB: N AD: N

---

129. no file-of-T-alphabet.
Enum: T_Err_file_of_T_alphabet_kw_not_present_
Class: Err_file_of_T_alphabet_kw_not_present AB: N AD: N

---

130. no filename.
Enum: T_Err_no_filename_
Class: Err_no_filename AB: N AD: N

---

131. no fsm-class present.
Enum: T_Err_no_fsm_class_present_
Class: Err_no_fsm_class_present AB: N AD: N

---

132. no fsm-comments present.
Enum: T_Err_no_fsm_comments_present_
Class: Err_no_fsm_comments_present AB: N AD: N

---

133. no fsm-comments string present.
Enum: T_Err_no_fsm_comments_string_
Class: Err_no_fsm_comments_string AB: N AD: N

---

134. no fsm-date present.
Enum: T_Err_no_fsm_date_present_
Class: Err_no_fsm_date_present AB: N AD: N

---

135. no fsm-date string present.
Enum: T_Err_no_fsm_date_string_
Class: Err_no_fsm_date_string AB: N AD: N

---

136. no fsm-debug present.
Enum: T_Err_no_fsm_debug_present_
Class: Err_no_fsm_debug_present AB: N AD: N

---

137. no fsm-debug string present.
Enum: T_Err_no_fsm_debug_string_
Class: Err_no_fsm_debug_string AB: N AD: N

---

138. no fsm-filename id present.
Enum: T_Err_no_fsm_filename_id_present_
Class: Err_no_fsm_filename_id_present AB: N AD: N

---

139. no fsm-filename present.
Enum: T_Err_no_fsm_filename_present_
Class: Err_no_fsm_filename_present AB: N AD: N

---

140. no fsm-id-present.
Enum: T_Err_no_fsm_id_present_
Class: Err_no_fsm_id_present AB: N AD: N

---

141. no fsm-id-string present.
Enum: T_Err_no_fsm_id_string_
Class: Err_no_fsm_id_string AB: N AD: N

---

142. no fsm-namespace id present.
Enum: T_Err_no_fsm_namespace_id_present_
Class: Err_no_fsm_namespace_id_present AB: N AD: N

---

143. no fsm-namespace present.
Enum: T_Err_no_fsm_namespace_present_
Class: Err_no_fsm_namespace_present AB: N AD: N

---

144. no fsm-version present.
Enum: T_Err_no_fsm_version_present_
Class: Err_no_fsm_version_present AB: N AD: N

---

145. no fsm-version string present.
Enum: T_Err_no_fsm_version_string_
Class: Err_no_fsm_version_string AB: N AD: N

---

146. no grammar-name present.
Enum: T_Err_grammar_name_kw_not_present_
Class: Err_grammar_name_kw_not_present AB: N AD: N

---

147. no grammar-name value.
Enum: T_Err_grammar_name_value_not_present_
Class: Err_grammar_name_value_not_present AB: N AD: N

---

148. no identifier present.
Enum: T_Err_no_identifier_present_
Class: Err_no_identifier_present AB: N AD: N

---

149. no int present.
Enum: T_Err_no_int_present_
Class: Err_no_int_present AB: N AD: N

---

150. no key-value present in definition.
Enum: T_Err_no_terminal_key_present_
Class: Err_no_terminal_key_present AB: N AD: N

---

151. no list-of-native-terminals.
Enum: T_Err_list_of_terminals_kw_not_present_
Class: Err_list_of_terminals_kw_not_present AB: N AD: N

---

152. no list-of-transit....
Enum: T_Err_no_list_of_trans_threads_kw_
Class: Err_no_list_of_trans_threads_kw AB: N AD: N

---

153. no monolithic present.
Enum: T_Err_monolithic_kw_not_present_
Class: Err_monolithic_kw_not_present AB: N AD: N

---

154. no name-space.
Enum: T_Err_namespace_kw_not_present_
Class: Err_namespace_kw_not_present AB: N AD: N

---

155. no name-space present.
Enum: T_Err_no_namespace_present_
Class: Err_no_namespace_present AB: N AD: N

---

156. no name-space value.
Enum: T_Err_namespace_value_not_present_
Class: Err_namespace_value_not_present AB: N AD: N

---

157. no name-space-id present.
Enum: T_Err_no_namespace_id_present_
Class: Err_no_namespace_id_present AB: N AD: N

---

158. no no-of-T present.
Enum: T_Err_no_of_T_kw_not_present_
Class: Err_no_of_T_kw_not_present AB: N AD: N

---

159. no open-brace.
Enum: T_Err_no_open_brace_
Class: Err_no_open_brace AB: N AD: N

---

160. no open-parenthesis.
Enum: T_Err_no_open_parenthesis_
Class: Err_no_open_parenthesis AB: N AD: N

---

161. no parallel thread function.
Enum: T_Err_no_pp_fnct_id_present_
Class: Err_no_pp_fnct_id_present AB: N AD: N

---

162. no parallel-code.
Enum: T_Err_no_pp_code_present_
Class: Err_no_pp_code_present AB: N AD: N

---

163. no parallel-code-syntax-code.
Enum: T_Err_no_pp_code_stc_present_
Class: Err_no_pp_code_stc_present AB: N AD: N

---

164. no parallel-control-monitor.
Enum: T_Err_no_pp_ctrl_mntor_kw_present_
Class: Err_no_pp_ctrl_mntor_kw_present AB: N AD: N

---

165. no parallel-la-bndary expr.
Enum: T_Err_pp_la_boundary_attribute_not_fnd_
Class: Err_pp_la_boundary_attribute_not_fnd AB: N AD: N

---

166. no parallel-la-boundary.
Enum: T_Err_no_pp_bndry_present_
Class: Err_no_pp_bndry_present AB: N AD: N

---

167. no parallel-la-boundary-expr.
Enum: T_Err_no_pp_la_bndary_expr_present_
Class: Err_no_pp_la_bndary_expr_present AB: N AD: N

---

168. no parallel-thread-function.
Enum: T_Err_no_pp_thread_function_present_
Class: Err_no_pp_thread_function_present AB: N AD: N

---

169. no preamble source code.
Enum: T_Err_preamble_srce_code_not_present_
Class: Err_preamble_srce_code_not_present AB: N AD: N

---

170. no rule name present.
Enum: T_Err_no_rule_name_present_
Class: Err_no_rule_name_present AB: N AD: N

---

171. no sub rule present.
Enum: T_Err_no_sub_rule_present_
Class: Err_no_sub_rule_present AB: N AD: N

---

172. no sym-class id present.
Enum: T_Err_no_sym_class_id_present_
Class: Err_no_sym_class_id_present AB: N AD: N

---

173. no sym-class present.
Enum: T_Err_no_sym_class_present_
Class: Err_no_sym_class_present AB: N AD: N

---

174. no symbol definition present.
Enum: T_Err_no_sym_defs_present_
Class: Err_no_sym_defs_present AB: N AD: N

---

175. no syntax-code present.
Enum: T_Err_no_syntax_code_present_
Class: Err_no_syntax_code_present AB: N AD: N

---

176. no terminal-def-code present.
Enum: T_Err_no_tdef_code_present_
Class: Err_no_tdef_code_present AB: N AD: N

---

177. no thread-name present.
Enum: T_Err_threadname_kw_not_present_
Class: Err_threadname_kw_not_present AB: N AD: N

---

178. no thread-name value.
Enum: T_Err_threadname_value_not_present_
Class: Err_threadname_value_not_present AB: N AD: N

---

179. no transitive present.
Enum: T_Err_transitive_kw_not_present_
Class: Err_transitive_kw_not_present AB: N AD: N

---

180. no # in list-of-native-term....
Enum: T_Err_no_of_terminals_not_present_
Class: Err_no_of_terminals_not_present AB: N AD: N

---

181. no # in list-of-trans....
Enum: T_Err_no_of_threads_not_present_
Class: Err_no_of_threads_not_present AB: N AD: N
Cross check against the O_2 linker "fsc" info. The "list-of-transitive-threads" construct also states the number of entries in its list. Some how they don't jive. i think thar's something rotten in the fingers and diletants playing with the file?

---

182. no-of-T value not present.
Enum: T_Err_no_of_T_value_not_present_
Class: Err_no_of_T_value_not_present AB: N AD: N

---

183. not :: in thread expr.
Enum: T_Err_not_dbl_colon_in_th_stmt_
Class: Err_not_dbl_colon_in_th_stmt AB: N AD: N

---

184. not a Rule in chained dispatcher expr.
Enum: T_Err_not_a_Rule_
Class: Err_not_a_Rule AB: N AD: N
Thrown by *subrule_def.lex* grammar supporting the new back-to-back thread call construct whereby the following procedure call is chained to the pushed "return T" of the 1st thread call. The term following the 1st thread call expression must be a rule whose contents use the TRAshift operator.

---

185. not a kw to start the top/down parse construct.
Enum: T_Err_not_kw_defining_grammar_construct_
Class: Err_not_kw_defining_grammar_construct AB: N AD: N

Thrown by *pass3.lex* grammar. Caused by an identifier that has been misplaced. Could be a typo where the item should be within the defining Rules Vocabulary or a premature ending of the Rules construct by an extra close brace whereby it is trying to define a rule.

---

186. not a lhs kw.
Enum: T_Err_not_a_lhs_kw_
Class: Err_not_a_lhs_kw AB: N AD: N

---

187. not a namespace id in thread expr.
Enum: T_Err_not_id_for_ns_in_th_stmt_
Class: Err_not_id_for_ns_in_th_stmt AB: N AD: N

---

188. not a terminal definition.
Enum: T_Err_not_a_terminal_definition_
Class: Err_not_a_terminal_definition AB: N AD: N

---

189. not a thread name id in expr.
Enum: T_Err_not_id_for_th_name_in_th_stmt_
Class: Err_not_id_for_th_name_in_th_stmt AB: N AD: N

---

190. not an arbitration-code keyword.
Enum: T_Err_not_arbitration_code_kw_
Class: Err_not_arbitration_code_kw AB: N AD: N

---

191. not an eosr in subrule expr.
Enum: T_Err_not_eos_in_subrule_expr_
Class: Err_not_eos_in_subrule_expr AB: N AD: N

---

192. preamble kw not present.
Enum: T_Err_preamble_kw_not_present_
Class: Err_preamble_kw_not_present AB: N AD: N

---

193. re-compile grammar: bad T-alphabet.
Enum: T_Err_bad_T_alphabet_
Class: Err_bad_T_alphabet AB: N AD: N
Mismatch with the contents of the "T-alphabet" construct and the number of terminals stated in each individual grammar "fsc" file. Usually means that terminals of some ilk have changed the population and requires a recompilation per grammar to match the vocabulary numbers. That is regen grammars with the "-t" and "-err" switches. This gens the eumeration file that "o2linker" depends on.

---

194. removal of term against empty set in la expr.
Enum: T_Err_empty_set_removal_in_la_expr_
Class: Err_empty_set_removal_in_la_expr AB: N AD: N
Bad expression where a removal expression is against an empty set. For example, a - eolr - b is invalid. The eolr empties the set. a - eolr + b is a round about way of having just b in the set.

---

195. report-card-ptr-0.
Enum: T_Err_report_card_ptr_0_
Class: Err_report_card_ptr_0 AB: N AD: N

---

196. rotten chr in T-alphabet.
Enum: T_Err_rotten_chr_in_T_alphabet_
Class: Err_rotten_chr_in_T_alphabet AB: N AD: N

---

197. rule already defined.
Enum: T_Err_rule_already_defined_
Class: Err_rule_already_defined AB: N AD: N

---

198. rule does not gen T strings - sick grammar.
Enum: T_ERR_sick_grammar_
Class: ERR_sick_grammar AB: N AD: N

---

199. rule not found in stbl.
Enum: T_Err_rule_not_in_stbl_
Class: Err_rule_not_in_stbl AB: N AD: N
Post process the grammar's vocabulary. A rule can be referenced before its definition. Just help the grammar writer of misselling or forgetfulness.

---

200. rule used but undefined.
Enum: T_Err_used_rule_but_undefined_
Class: Err_used_rule_but_undefined AB: N AD: N

---

201. stbl char-pool full.
Enum: T_Err_sym_tbl_char_pool_full_
Class: Err_sym_tbl_char_pool_full AB: N AD: N

---

202. stbl context-buf-overflow.
Enum: T_Err_sym_tbl_context_buf_overflow_
Class: Err_sym_tbl_context_buf_overflow AB: N AD: N

---

203. stbl full.
Enum: T_Err_sym_tbl_full_
Class: Err_sym_tbl_full AB: N AD: N

---

204. stbl has entry but not a rule.
Enum: T_Err_stble_has_entry_but_not_a_rule_
Class: Err_stble_has_entry_but_not_a_rule AB: N AD: N

---

205. stbl scope-stk overflow.
Enum: T_Err_sym_tbl_nested_scope_stk_overflow_
Class: Err_sym_tbl_nested_scope_stk_overflow AB: N AD: N

---

206. stbl scope-stk underflow.
Enum: T_Err_sym_tbl_nested_scope_stk_underflow_
Class: Err_sym_tbl_nested_scope_stk_underflow AB: N AD: N

---

207. subrule overrun.
Enum: T_Err_subrule_overrun_
Class: Err_subrule_overrun AB: N AD: N

---

208. subscript out-of-range.
Enum: T_Err_subscript_out_of_range_
Class: Err_subscript_out_of_range AB: N AD: N

---

209. term not a lhs or parallel-control-monitor kw.
Enum: T_Err_not_lhs_pcnrl_mntr_
Class: Err_not_lhs_pcnrl_mntr AB: N AD: N

---

210. terminals-refs duplicate.
Enum: T_Err_terminals_refs_dup_def_
Class: Err_terminals_refs_dup_def AB: N AD: N

---

211. thread defined by another fsc file.
Enum: T_Err_already_defined_in_fsc_file_
Class: Err_already_defined_in_fsc_file AB: N AD: N

Each individual thread's fsc file contains its bio. This error indicates that the grammar writer probably cloned off a grammar but did not edit properly the vital statistics. So go edit the erroneous grammar and correct its thread name, namespace etc.

---

212. thread in stbl but subscript badly set.
Enum: T_Err_bad_thread_subscript_
Class: Err_bad_thread_subscript AB: N AD: N

More of a sanity check in the symbol table. If this occurs i'm really in trouble.

---

213. thread xrefed, but not defed.
Enum: T_Err_thread_refed_but_not_defined_
Class: Err_thread_refed_but_not_defined AB: N AD: N

---

214. thread xrefed, but not defed user-declaration directive.

⟨ thread xrefed, but not defed user-declaration directive 214 ⟩ ≡

```
public: Err_thread_refed_but_not_defined(CAbs_lr1_sym * Th_id);
  CAbs_lr1_sym * th_id( );
private: CAbs_lr1_sym * th_id_;
```

215. thread xrefed, but not defed user-implementation directive.

⟨ thread xrefed, but not defed user-implementation directive 215 ⟩ ≡

```
Err_thread_refed_but_not_defined::Err_thread_refed_but_not_defined(CAbs_lr1_sym *
        Th_id)T_CTOR("thread␣xrefed,␣but␣not␣defed",
        T_Enum::T_Err_thread_refed_but_not_defined_, 0, false, false)
{
  th_id_ = Th_id;
}
yacco2::CAbs_lr1_sym * Err_thread_refed_but_not_defined::th_id( )
{
  return th_id_;
}
;
```

216. token found in stbl but not a kw.
Enum: T_Err_not_a_keyword_
Class: Err_not_a_keyword AB: N AD: N
The symbol is not a keyword. Could be an initialization problem with my O_2 setup.

---

217. transitive list thread not defined by fsc files.
Enum: T_Err_bad_th_in_list_
Class: Err_bad_th_in_list AB: N AD: N
The hand coded "???.fsc" control file provides all the threads' "fsc" files has thread entries missing. Typically the grammar writer created a new thread but forgot the include it in the global "fsc" file. So go add the missing thread entries in the global "fsc" file.

---

218. undefined terminal in subrule.
Enum: T_Err_subrule_use_undefined_T_
Class: Err_subrule_use_undefined_T AB: N AD: N

---

219. unknown symbol type in stbl.
Enum: T_Err_stbl_entry_unknown_
Class: Err_stbl_entry_unknown AB: N AD: N

---

220. use of Non-terminal (rule) outside Rules's construct.
Enum: T_Err_use_of_N_outside_Rules_construct_
Class: Err_use_of_N_outside_Rules_construct AB: N AD: N
Thrown by *pass3.lex* grammar. The rule reference has already been seen by the parsed Rules construct and stored in the symbol table. The symbol table returns the found item as a rule. Cause is a reference to the rule outside of the Rules construct. Probably a typo by the grammar writer caused by the premature parsing of the Rules Vocabulary.

---

221. use of T outside Rules's construct.
Enum: T_Err_use_of_T_outside_Rules_construct_
Class: Err_use_of_T_outside_Rules_construct AB: N AD: N
Thrown by *pass3.lex* grammar. The terminal definitions have already been parsed and stored in the symbol table. The symbol table returns the found item as a terminal rather than the unfound identifier. Cause is reference of a terminal outside of the Rules construct. Probably a typo by the grammar writer.

---

222. zero len symbol.
Enum: T_Err_zero_len_sym_
Class: Err_zero_len_sym AB: N AD: N

---

223. Index.

Error Vocabulary

Date: January 2, 2015 at 16:29

File: yacco2_err_symbols Namespace: NS_yacco2_err_symbols

Number of terminals: 191