parse.y 抜粋

```
enum lex_state_e {
    EXPR_BEG,             /* ignore newline, +/- is a sign. */
    EXPR_END,             /* newline significant, +/- is an operator. */
    EXPR_ENDARG,          /* ditto, and unbound braces. */
    EXPR_ARG,             /* newline significant, +/- is an operator. */
    EXPR_CMDARG,          /* newline significant, +/- is an operator. */
    EXPR_MID,             /* newline significant, +/- is an operator. */
    EXPR_FNAME,           /* ignore newline, no reserved words. */
    EXPR_DOT,             /* right after `.' or `::', no reserved words. */
    EXPR_CLASS,           /* immediate after `class', no here document. */
    EXPR_VALUE            /* alike EXPR_BEG but label is disallowed. */
};
--------------------------------------------------------------------------------
%token <id>   tIDENTIFIER tFID tGVAR tIVAR tCONSTANT tCVAR tLABEL
%token <node> tINTEGER tFLOAT tSTRING_CONTENT tCHAR
%token <node> tNTH_REF tBACK_REF
%token <num>  tREGEXP_END

%type <node> singleton strings string string1 xstring regexp
%type <node> string_contents xstring_contents string_content
%type <node> words qwords word_list qword_list word
%type <node> literal numeric dsym cpath
%type <node> bodystmt compstmt opt_stmts stmts stmt expr arg primary command com-
mand_call method_call
%type <node> expr_value arg_value primary_value
%type <node> if_tail opt_else case_body cases opt_rescue exc_list exc_var opt_en-
sure
%type <node> args call_args opt_call_args
%type <node> paren_args opt_paren_args
%type <node> command_args aref_args opt_block_arg block_arg var_ref var_lhs
%type <node> mrhs superclass block_call block_command
%type <node> f_block_optarg f_block_opt
%type <node> f_arglist f_args f_arg f_arg_item f_optarg f_marg f_marg_list f_margs
%type <node> inbrace assocs assoc undef_list backref string_dvar for_var
%type <node> block_param opt_block_param block_param_def f_opt
%type <node> bv_decls opt_bv_decl bvar
%type <node> lambda f_larglist lambda_body
%type <node> brace_block cmd_brace_block do_block lhs none fitem
%type <node> mlhs mlhs_head mlhs_basic mlhs_item mlhs_node mlhs_post mlhs_inner
%type <id>   fsym variable sym symbol operation operation2 operation3
%type <id>   cname fname op f_rest_arg f_block_arg opt_f_block_arg f_norm_arg
f_bad_arg
/*%%%*/
/*%
%type <val> program reswords then do dot_or_colon
%*/
%token tUPLUS       /* unary+ */
%token tUMINUS      /* unary- */
%token tPOW         /* ** */
%token tCMP         /* <=> */
%token tEQ          /* == */
%token tEQQ         /* === */
%token tNEQ         /* != */
%token tGEQ         /* >= */
%token tLEQ         /* <= */
%token tANDOP tOROP /* && and || */
%token tMATCH tNMATCH      /* =~ and !~ */
%token tDOT2 tDOT3  /* .. and ... */
%token tAREF tASET  /* [] and []= */
%token tLSHFT tRSHFT       /* << and >> */
%token tCOLON2             /* :: */
%token tCOLON3             /* :: at EXPR_BEG */
%token <id> tOP_ASGN       /* +=, -=  etc. */
%token tASSOC       /* => */
%token tLPAREN      /* ( */
%token tLPAREN_ARG  /* ( */
%token tRPAREN      /* ) */
```

```
%token tLBRACK      /* [ */
%token tLBRACE      /* { */
%token tLBRACE_ARG  /* { */
%token tSTAR        /* * */
%token tAMPER       /* & */
%token tLAMBDA      /* -> */
%token tSYMBEG tSTRING_BEG tXSTRING_BEG tREGEXP_BEG tWORDS_BEG tQWORDS_BEG
%token tSTRING_DBEG tSTRING_DVAR tSTRING_END tLAMBEG

%%
------------------------------------------------------------------------------
      case ':':
       c = nextc();
       if (c == ':') {
           if (IS_BEG() ||
              lex_state == EXPR_CLASS || (IS_ARG() && space_seen)) {
              lex_state = EXPR_BEG;
              return tCOLON3;
           }
           lex_state = EXPR_DOT;
           return tCOLON2;
       }
       if (lex_state == EXPR_END || lex_state == EXPR_ENDARG || (c != -1 &&
ISSPACE(c))) {
           pushback(c);
           lex_state = EXPR_BEG;
           return ':';
       }
       switch (c) {
         case '\'':
           lex_strterm = NEW_STRTERM(str_ssym, c, 0);
           break;
         case '"':
           lex_strterm = NEW_STRTERM(str_dsym, c, 0);
           break;
         default:
           pushback(c);
           break;
       }
       lex_state = EXPR_FNAME;
       return tSYMBEG;
------------------------------------------------------------------------------
var_ref             : variable
                    {
                    /*%%%*/
                       if (!($$ = gettable($1))) $$ = NEW_BEGIN(0);
                    /*%
                       $$ = dispatch1(var_ref, $1);
                    %*/
                    }
+             | tINCR variable
+                   {
+                   /*%%%*/
+                      $$ = assignable($2, 0);
+                      $$->nd_value = NEW_CALL(gettable($$->nd_vid), rb_in-
tern("succ"), 0);
+                   /*%
+                      $$ = dispatch2(unary, ripper_intern("++@"), $2);
+                   %*/
+                   }
              ;
```