1. Copyright.

2. *cweave_sdc* **grammar.**
Write out cweave subrule's syntax directed directives sentences.

3. Fsm Ccweave_sdc class.

4. Ccweave_sdc user-declaration directive.

⟨ Ccweave_sdc user-declaration directive 4 ⟩ ≡
```
public: std::ofstream * cweave_file_;
  T_subrule_def * subrule_def_;
  int subrule_no_;
  void initialize(std::ofstream * Cweave_file, T_subrule_def * Subrule_def, int Subrule_no);
  void output_sr_sdcode(const char *Directive);
  void output_sr_sdcode_title(const char *Directive);
  void wrt_directive(const char *Directive, T_syntax_code * Sdc);
  void deal_with_cweave_atsign(const char *Directive, T_syntax_code * Sdc);
```

5. Ccweave_sdc user-implementation directive.

⟨ Ccweave_sdc user-implementation directive 5 ⟩ ≡
```
void Ccweave_sdc::initialize(std::ofstream * Cweave_file, T_subrule_def * Subrule_def, int Subrule_no)
{
  cweave_file_ = Cweave_file;
  subrule_def_ = Subrule_def;
  subrule_no_ = Subrule_no;
}
```

6. *wrt_directive*.

```
⟨ More code 6 ⟩ ≡
  void Ccweave_sdc::wrt_directive(const char *Directive, T_syntax_code * Sdc)
  {
    if (Sdc ≡ 0) {
      output_sr_sdcode(Directive);
      (*cweave_file_) ≪ "/";
      (*cweave_file_) ≪ "/ no sdcode" ≪ endl;
      return;
    }
    if (Sdc→cweb_marker( ) ≠ 0) {
      WRT_CWEB_MARKER(cweave_file_, Sdc→cweb_marker( ));
    }
    output_sr_sdcode(Directive);
    string xlate;
    int len = Sdc→syntax_code( )→length( );
    string & sdc = *Sdc→syntax_code( );      /* prescan @ due to cweave reqmts */
    for (int x = 0; x < len; ++x) {
      char nc = sdc[x];
      if (nc ≡ '@') {      /* check next char for cweave type directives */
        char nnc = sdc[x + 1];
        if ((nnc ≡ '*') ∨ (nnc ≡ '<') ∨ (nnc ≡ '>')) {
          xlate += nc;
          continue;
        }
        else {
          xlate += "@@";
          continue;
        }
      }
      else {
        xlate += nc;
        continue;
      }
    }
    (*cweave_file_) ≪ xlate.c_str( ) ≪ endl;
  }
```

See also sections 7 and 8.

7. *output_sr_sdcode*.

```
⟨ More code 6 ⟩ +≡
  void Ccweave_sdc::output_sr_sdcode(const char *Directive)
  {
    char big_buf_[BIG_BUFFER_32K];
    char xa[Max_cweb_item_size];
    XLATE_SYMBOLS_FOR_cweave(subrule_def_→its_rule_def( )→rule_name( )→c_str( ), xa);
    KCHARP cweave_sentence = "@<%s subrule %i %s directive@>=\n";
    sprintf(big_buf_, cweave_sentence, xa, subrule_no_, Directive);
    (*cweave_file_) ≪ big_buf_;
  }
```

8. *output_sr_sdcode_title*.

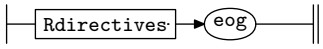
⟨ More code 6 ⟩ +≡
```
void Ccweave_sdc::output_sr_sdcode_title(const char *Directive)
{
  char big_buf_[BIG_BUFFER_32K];
  char xa[Max_cweb_item_size];
  XLATE_SYMBOLS_FOR_cweave(subrule_def_→its_rule_def()→rule_name()→c_str(), xa);
  KCHARP cweave_sentence = "@*4␣%s␣subrule␣%i␣%s␣directive.\n";
  sprintf(big_buf_, cweave_sentence, xa, subrule_no_, Directive);
  (*cweave_file_) ≪ big_buf_;
}
```

9. Ccweave_sdc user-prefix-declaration directive.

⟨ Ccweave_sdc user-prefix-declaration directive 9 ⟩ ≡
```
#include "o2_externs.h"
```

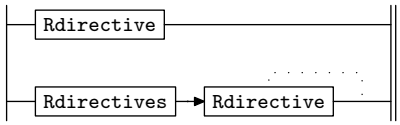
10. *Rweave_sdc* **rule.**

Rweave_sdc

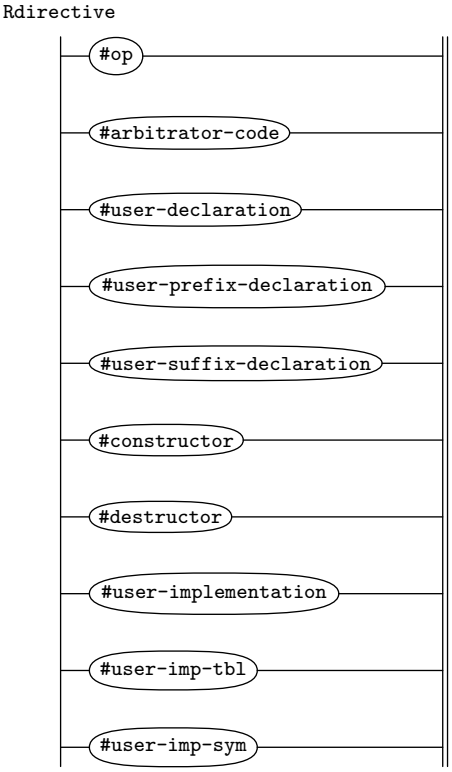


11. *Rdirectives* **rule.**

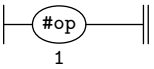
Rdirectives



12. *Rdirective* **rule.**

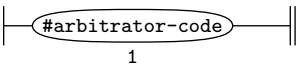


13. *Rdirective*'**s subrule 1.**



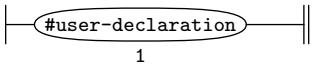
⟨ Rdirective subrule 1 op directive 13 ⟩ ≡
Ccweave_sdc ∗ *fsm* = (*Ccweave_sdc* ∗) *rule_info__.parser__→fsm_tbl__*;
KCHARP *sdc* = "op";
fsm→wrt_directive(*sdc*, *sf→p1__→syntax_code*( ));

14. *Rdirective*'**s subrule 2.**



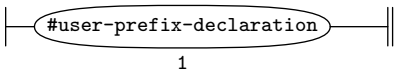
⟨ Rdirective subrule 2 op directive 14 ⟩ ≡
Ccweave_sdc ∗ *fsm* = (*Ccweave_sdc* ∗) *rule_info__.parser__→fsm_tbl__*;
KCHARP *sdc* = "arbitrator-code";
fsm→wrt_directive(*sdc*, *sf→p1__→syntax_code*( ));

15. *Rdirective*'s subrule 3.



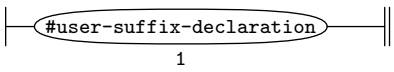
⟨ Rdirective subrule 3 op directive 15 ⟩ ≡
```
Ccweave_sdc * fsm = ( Ccweave_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "user-declaration";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

16. *Rdirective*'s subrule 4.



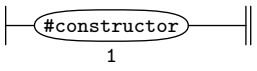
⟨ Rdirective subrule 4 op directive 16 ⟩ ≡
```
Ccweave_sdc * fsm = ( Ccweave_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "user-prefix-declaration";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

17. *Rdirective*'s subrule 5.



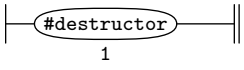
⟨ Rdirective subrule 5 op directive 17 ⟩ ≡
```
Ccweave_sdc * fsm = ( Ccweave_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "user-suffix-declaration";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

18. *Rdirective*'s subrule 6.



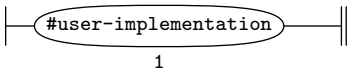
⟨ Rdirective subrule 6 op directive 18 ⟩ ≡
```
Ccweave_sdc * fsm = ( Ccweave_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "constructor";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

19. *Rdirective*'s subrule 7.



⟨ Rdirective subrule 7 op directive 19 ⟩ ≡
```
Ccweave_sdc * fsm = ( Ccweave_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "destructor";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

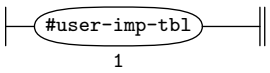
20. *Rdirective*'s subrule 8.



⟨ Rdirective subrule 8 op directive 20 ⟩ ≡

```
Ccweave_sdc * fsm = ( Ccweave_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "user-implementation";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

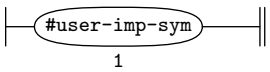
21. *Rdirective*'s subrule 9.



⟨ Rdirective subrule 9 op directive 21 ⟩ ≡

```
Ccweave_sdc * fsm = ( Ccweave_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "user-imp-tbl";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

22. *Rdirective*'s subrule 10.



⟨ Rdirective subrule 10 op directive 22 ⟩ ≡

```
Ccweave_sdc * fsm = ( Ccweave_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "user-imp-sym";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

23. First Set Language for O_2^{linker}.

```
/*
 File: cweave_sdc.fsc
 Date and Time: Fri Jan  2 15:33:30 2015
*/
transitive   n
grammar-name "cweave_sdc"
name-space   "NS_cweave_sdc"
thread-name  "Ccweave_sdc"
monolithic   y
file-name    "cweave_sdc.fsc"
no-of-T      569
list-of-native-first-set-terminals 10
   T_arbitrator_code
   T_user_declaration
   T_user_prefix_declaration
   T_user_suffix_declaration
   T_constructor
   T_destructor
   T_op
   T_user_implementation
   T_user_imp_tbl
   T_user_imp_sym
end-list-of-native-first-set-terminals
list-of-transitive-threads 0
end-list-of-transitive-threads
list-of-used-threads 0
end-list-of-used-threads
fsm-comments
"Write out cweave subrule's sdc irectives sentences"
```

24. Lr1 State Network.

⇒ State: 1 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
c	Rdirective		3	2	1	#	arbitrator-code		1	2	2	
c	Rdirective		3	3	1	#	user-declaration		1	3	3	
c	Rdirective		3	4	1	#	user-prefix-declaration		1	4	4	
c	Rdirective		3	5	1	#	user-suffix-declaration		1	5	5	
c	Rdirective		3	6	1	#	constructor		1	6	6	
c	Rdirective		3	7	1	#	destructor		1	7	7	
c	Rdirective		3	1	1	#	op		1	8	8	
c	Rdirective		3	8	1	#	user-implementation		1	9	9	
c	Rdirective		3	9	1	#	user-imp-tbl		1	10	10	
c	Rdirective		3	10	1	#	user-imp-sym		1	11	11	
c	Rweave_sdc		1	1	1	Rdirectives	*eog*		1	12	13	
c	Rdirectives		2	2	1	Rdirectives	*Rdirective*		1	12	14	
c	Rdirectives		2	1	1	Rdirective			1	15	15	

⇒*#arbitrator−code* State: 2 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	2	2				1	0	2	1

⇒*#user−declaration* State: 3 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	3	2				1	0	3	1

⇒*#user−prefix−declaration* State: 4 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	4	2				1	0	4	1

⇒*#user−suffix−declaration* State: 5 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	5	2				1	0	5	1

⇒*#constructor* State: 6 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	6	2				1	0	6	1

⇒*#destructor* State: 7 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	7	2				1	0	7	1

⇒*#op* State: 8 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	1	2				1	0	8	1

⇒*#user−implementation* State: 9 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	8	2				1	0	9	1

⇒*#user−imp−tbl* State: 10 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA

t	Rdirective	3	9	2			1	0	10	1

⇒*#user−imp−sym* State: 11 state type: *r*

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	10	2				1	0	11	1

⇒*Rdirectives* State: 12 state type: *s*

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rweave_sdc		1	1	2	eog			1	13	13	
c	Rdirective		3	2	1	# arbitrator-code			12	2	2	
c	Rdirective		3	3	1	# user-declaration			12	3	3	
c	Rdirective		3	4	1	# user-prefix-declaration			12	4	4	
c	Rdirective		3	5	1	# user-suffix-declaration			12	5	5	
c	Rdirective		3	6	1	# constructor			12	6	6	
c	Rdirective		3	7	1	# destructor			12	7	7	
c	Rdirective		3	1	1	# op			12	8	8	
c	Rdirective		3	8	1	# user-implementation			12	9	9	
c	Rdirective		3	9	1	# user-imp-tbl			12	10	10	
c	Rdirective		3	10	1	# user-imp-sym			12	11	11	
t	Rdirectives		2	2	2	Rdirective			1	14	14	

⇒*eog* State: 13 state type: *r*

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rweave_sdc		1	1	3				1	0	13	2

⇒*Rdirective* State: 14 state type: *r*

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirectives		2	2	3				1	0	14	1

⇒*Rdirective* State: 15 state type: *r*

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirectives		2	1	2				1	0	15	1

25. Index.

cweave_sdc Grammar

Date: January 2, 2015 at 15:34

File: cweave_sdc.lex

Ns: NS_cweave_sdc

Version: 1.0 Debug: false

Grammar Comments: Type: Monolithic

Write out cweave subrule's sdc irectives sentences