

The Syntax of the SC-0/1 Language

March 24, 2008

1 External Declarations

translation-unit :
 external-declaration
 translation-unit external-declaration

external-declaration :
 declaration

2 Declarations

declaration-list :
 declaration
 declaration-list declaration

declaration :
 inlined-declaration
 (*identifier type-expression initializer_{opt}*) (SC-1 only)
 (*function-identifier (fn function-type-list*
 [*:attr function-attribute*]_{opt} *register-declarator_{opt} block-item-list_{opt}*) (SC-1 only)
 (*struct-or-union-specifier struct-declaration-list_{opt}*) (SC-1 only)
 (*enum-specifier enumerator-list*) (SC-1 only)

inlined-declaration-list :
 inlined-declaration
 inlined-declaration-list declaration

inlined-declaration :
 (*storage-class-specifier identifier type-expression initializer_{opt}*)
 (*storage-class-specifier function-identifier (fn function-type-list*
 [*:attr function-attribute*]_{opt} *register-declarator_{opt} block-item-list_{opt}*)
 (*def-or-decl struct-or-union-specifier struct-declaration-list_{opt}*)
 (*def enum-specifier enumerator-list*)
 (*compound-storage-class-specifier type-expression init-declarator-list*) (SC-1 only)
 (*deftype identifier type-expression*)
 (*deftype identifier struct-or-union struct-declaration-list_{opt}*)

(deftype identifier enum enumerator-list)

function-identifier :
 identifier
 (*identifier-list*)

def-or-decl :
 def
 decl

init-declarator-list :
 init-declarator
 init-declarator-list init-declarator

init-declarator :
 identifier
 (*identifier initializer*)

storage-class-specifier : one of
 def decl extern extern-def extern-decl
 static static-def auto auto-def register register-def

compound-storage-class-specifier : one of (SC-1 only)
 defs extern-defs static-defs auto-defs register-defs

function-attribute :
 inline

register-declarator :
 (**register** *identifier-list*)

struct-declaration-list :
 struct-declaration
 struct-declaration-list struct-declaration

struct-declaration :
 declaration [:**bit** *expression*]_{opt}

enumerator-list :
 enumerator
 enumerator-list enumerator

enumerator :
 enumeration-constant
 (*enumeration-constant expression*)

enumeration-constant :
 identifier

identifier-list :

identifier
identifier-list identifier

designator :
 (aref-this expression-list)
 (fref-this identifier-list)
 (aref designator expression-list)
 (fref designator identifier-list)

designated-initializer :
 initializer
 (*designator initializer*)

initializer-list :
 designated-initializer
 initializer-list designated-initializer

compound-initializer :
 (array initializer-list)
 (struct initializer-list)

initializer :
 expression
 compound-initializer

3 Type-expressions

type-expression :
 type-specifier
 (*type-qualifier-list type-expression*)
 (array type-expression array-subscription-list_{opt})
 (ptr type-expression)
 (fn function-type-list)

function-type-list :
 type-expression-list va-arg_{opt}

type-expression-list
 type-expression
 type-expression-list type-expression

type-specifier : one of
 void
 char signed-char unsigned-char short signed-short unsigned-short
 int signed-int unsigned-int long signed-long unsigned-long
 long-long signed-long-long unsigned-long-long
 float double long-double
 struct-or-union-specifier
 enum-specifier

typedef-name

array-subscription-list :
expression-list

struct-or-union-specifier :
 (*struct-or-union identifier*)

struct-or-union :
struct
union

enum-specifier :
 (**enum** *identifier*)

type-qualifier-list :
type-qualifier
type-qualifier-list type-qualifier

type-qualifier :
const
restrict
volatile

typedef-name :
identifier

4 Statements

statement :
compound-statement
expression-statement
selection-statement
iteration-statement
jump-statement
labeled-statement
 ()

compound-statement :
 (**begin** *block-item-list_{opt}*)
 (**let** (*declaration-list_{opt}*) *block-item-list_{opt}*) (SC-1 only)

block-item-list :
block-item
block-item-list block-item

block-item :
inlined-declaration

statement

labeled-statement :

(**label** *identifier* *statement*)
(**case** *expression*)
(**default**)

expression-statement :

expression

selection-statement :

(**if** *expression* *statement* *statement*_{opt})
(**switch** *expression* *block-item-list*_{opt})

iteration-statement :

(**while** *expression* *block-item-list*_{opt}) (SC-1 only)
(**do-while** *expression* *block-item-list*_{opt})
(**for** (*expression-list*_{opt} *expression* *expression*) *block-item-list*_{opt}) (SC-1 only)
(**for** (*inlined-declaration-list*_{opt} *expression* *expression*) *block-item-list*_{opt}) (SC-1 only)
(**loop** *block-item-list*_{opt}) (SC-1 only)

jump-statement :

(**goto** *identifier*)
(**continue**)
(**break**)
(**return** *expression*_{opt})

5 Expressions

expression :

identifier
constant
string-literal
compound-literal
(*expression-list*)
(**aref** *expression-list*)
(**fref** *expression* *field-identifier-list*)
(**inc** *expression*)
(**dec** *expression*)
(**++** *expression*)
(**--** *expression*)
(*unary-operator* *expression*)
(**sizeof** *expression*)
(**sizeof** *type-expression*)
(**cast** *type-expression* *expression*)
(*operator* *expression-list*)
(*comparator* *expression* *expression*)
(**if-exp** *expression* *expression* *expression*)
(*assignment-operator* *expression* *expression*)

(exprs *expression-list*)

compound-literal :

(init *type-expression* *compound-initializer*)

expression-list :

expression

expression-list *expression*

field-identifier-list :

field-identifier

field-identifier-list *field-identifier*

field-identifier :

identifier

-> *identifier*

(SC-1 only)

operator : one of

***** **/** **%** **+** **-** **<<** **>>** **bit-xor** **bit-and** **bit-or** **and** **or**

comparator : one of

< **>** **<=** **>=** **==** **!=**

assignment-operator : one of

= ***=** **/=** **%=** **+=** **-=** **<<=** **>>=** **bit-and=** **bit-xor=** **bit-or=**

unary-operator : one of

ptr **mref** **bit-not** **not**