C99 Parser User's Guide

rough and incomplete

Matt Wette

The C99 parsers can use "include helpers". This allows files to be parsed without reading full include files. The user provides typenames (types defined using `typedef`) and defines. The syntax for the include-helper optional argument to the parsers is

1 Introduction

This is a manual for ...

Note on CPP replacement text: IIRC, C99 will remove comments from CPP statements before processing. I preserve this and remove inside the CPP parser.

1.1 Include Helpers

The C99 parsers can use "include helpers". This allows files to be parsed without reading full include files. The user provides typenames (types defined using `typedef`) and defines. The syntax for the include-helper optional argument to the parsers is

```
(define my-inc-helper
 '(("foo.h" "foo_t" "ABC=123" "SUM(X,Y)=((X)+(Y))")
   ("bar.h" "bar_t" "DEF=456" "MAX(X,Y)=((X)>(Y)?(X):(Y))")))
```

The C99 `parser` and `xparser` modules export `c99-std-help`.

1.2 Misc Items

The special symbol `C99_ANY` can be used for symbols which you don't want to define. In the parser will handle this as XXX

2 The Unit Parser

TALK ABOUT `fixed-width-int-names`

TALK ABOUT `c99-std-help`

TALK ABOUT `stripdown`

`parse-c99` [*#:cpp-defs def-a-list*] [*#:inc-dirs dir-list*] [*#:mode* (*'code*|*'file*)] [*#:debug bool*] [Procedure]

This needs to be explained in some detail.

Default mode is `'code`.

```
(with-input-from-file "abc.c"
  (parse-c #:cpp-defs '("ABC=123"))
           #:inc-dirs (append '("./incs") c99-std-dict)
           #:inc-help '(("myinc.h" "foo_t" "bar_t"))
           #:mode 'file))
```

2.0.1 Modes

There are several modes for parsing which affect the way the C preprocessor statements are handled, and how the parse tree is generated. The following list explains the intent behind these parsing modes. Later we mention some fine points.

- *code* mode is the default. In this mode, the proprocess works like a normal C compiler. The preprocessor statements are evaluated as they are read and macros in the code are expanded as they are read.
- *decl* mode is intended to be used for tools which want to extract the declarations and definitions which are explicit in a file, but allow access to declarations and definitions in included files.
- *file* mode is intended to be used for tools which want to transform C files somehow. For example, one could parse a file and remove all comments. This will keep the CPP structure at the top level. Preprocessor statements at the top level are not evaluted.

Note:
There is a change in versions starting with 0.77.0.
In these all defines required for evaluating CPP expressions in if-then have to be resolved.

Options are as follows

`name` *mode* => *#t*|*#f* [xdef?]

Given string *name* and *mode* indicate whether the parser should expand using CPP defines. The default is `(lambda(name mode) (eqv? mode 'code))`.

3 Expression Parser

stuff

4 Copying