# 6 The C Preprocessor

The iWarp C language preprocessor is invoked as the first pass of any iWarp processor C compilation using the iwcc command. The output of the preprocessor is designed to be in a form acceptable as input to the next pass of the iWarp C compiler.

## Invoking the Preprocessor

The preprocessor is invoked automatically when you invoke the iWarp C compiler using the iwcc command. To invoke the preprocessor separately, use the iwcpp command as follows:

`iwcpp [`*`option ...`*`] [`*`infile`*`] [`*`outfile`*`]`

*infile* is an input file. If *infile* is not specified, input is taken from standard input.

*outfile* is the output file. If *outfile* is not specified, output is written to standard out.

*option* is any of the options listed in the following section.

### Preprocessor Options

| | |
|---|---|
| `-C` | retain comments in output |
| `-D`*name* | define *name* as “1” |
| `-D`*name*`=`*def* | define *name* as *def* |
| `-H` | print the pathnames of files as included, one per line, on the standard error output |
| `-I`*directory* | add *directory* to search list for #include files |
| `-P` | do *not* include cpp-type line number information in the output |
| `-R` | allow recursive macros |
| `-U`*name* | undefine *name* |

## Directory Search Order for #include

The preprocessor searches directories for #includes in the following order:

1. The directory of the file that contains the #include request, if the file name is contained in double quotes instead of angle brackets. (e.g. #include is relative to the file being scanned when the request is made).
2. The directories specified by -I, in left-to-right order.
3. /iwarp/include

## Multiple Line Definitions (#define)

A line ending in '\' (and followed immediately by the end of the line [*newline*]) may be used in the body of a '#define' statement to continue the definition onto the next line. The escaped linefeed is converted into a single blank in the macro body.

## Macro Expansion

Macro formal parameters are recognized in '#define' bodies even inside character constants and quoted strings. The output from the following would be the six characters `'\bar'`:

```
#define foo(a) '\a'
foo(bar)
```

Macro names are not recognized inside character constants or quoted strings during the regular scan as shown by:

```
#define foo bar
printf("foo");
```

The above does not expand 'foo' in the second line, because it is inside a quoted string which is not part of a '#define' macro definition.

Macros are not expanded while processing a '#define', '#undef', '#ifdef' or '#ifndef'. Thus, the following produces 'foo':

```
#define foo bletch
#define bar foo
#undef foo
bar
```

Any kind and amount of white space (space, tab, linefeed, vertical tab, formfeed, carriage return) is allowed between a macro name and the left parenthesis which introduces its actual parameters.

**NOTE**

Mismatch between the number of formal and actual parameters in a macro call produces only a warning, and not an error. Excess actual parameters are ignored; missing actual parameters are turned into null strings.

### Recursive Macros

If the -R flag is not specified then the invocation of some recursive macros is trapped and the recursion is forcibly terminated with an error message. The recursions that are trapped are the ones in which the nesting level is non-decreasing from some point on. For example, the following will be detected:

```
#define a a
a
```

The following recursion will not be detected because the nesting level decreases after each expansion of "c".

```
#define a c b
#define b c a
#define c foo
a
```

The -R flag specifically allows recursive macros and recursion will be strictly obeyed (to the extent that space is available).

### Comments

Comments are removed unless the -C option is used. They are also ignored, except to terminate a token. Thus “foo/* la di da */bar” may expand ‘foo’ and ‘bar’ but will never expand ‘foobar’. If neither ‘foo’ nor ‘bar’ is a macro then the output is ‘foobar’, even if ‘foobar’ is defined as something else.

## Files

The following files are associated with the preprocessor:

| | |
|---|---|
| *file*.c | input file (source file) |
| *file*.i | preprocessor output |
| *file*.w | inline file (list of functions to inline) |
| *file*.x | noinline file (list of functions not to inline) |
| /iwarp/lib/iwcpp | C preprocessor |