NAME

ecpg - embedded SQL C preprocessor

SYNOPSIS

ecpg [*option*...] *file*...

DESCRIPTION

ecpg is the embedded SQL preprocessor for C programs. It converts C programs with embedded SQL statements to normal C code by replacing the SQL invocations with special function calls. The output files can then be processed with any C compiler tool chain.

ecpg will convert each input file given on the command line to the corresponding C output file. If an input file name does not have any extension, .pgc is assumed. The file’s extension will be replaced by .c to construct the output file name. But the output file name can be overridden using the **-o** option.

If an input file name is just **-**, **ecpg** reads the program from standard input (and writes to standard output, unless that is overridden with **-o**).

This reference page does not describe the embedded SQL language. See Chapter 36 for more information on that topic.

OPTIONS

ecpg accepts the following command-line arguments:

-c

Automatically generate certain C code from SQL code. Currently, this works for EXEC SQL TYPE.

-C *mode*

Set a compatibility mode. *mode* can be INFORMIX, INFORMIX_SE, or ORACLE.

-D *symbol*

Define a C preprocessor symbol.

-h

Process header files. When this option is specified, the output file extension becomes .h not .c, and the default input file extension is .pgh not .pgc. Also, the **-c** option is forced on.

-i

Parse system include files as well.

-I *directory*

Specify an additional include path, used to find files included via EXEC SQL INCLUDE. Defaults are . (current directory), /usr/local/include, the PostgreSQL include directory which is defined at compile time (default: /usr/local/pgsql/include), and /usr/include, in that order.

-o *filename*

Specifies that **ecpg** should write all its output to the given *filename*. Write -o - to send all output to standard output.

-r *option*

Selects run-time behavior. *Option* can be one of the following:

no_indicator

Do not use indicators but instead use special values to represent null values. Historically there have been databases using this approach.

prepare

Prepare all statements before using them. Libecpg will keep a cache of prepared statements and reuse a statement if it gets executed again. If the cache runs full, libecpg will free the least used statement.

questionmarks

Allow question mark as placeholder for compatibility reasons. This used to be the default long ago.

-t

Turn on autocommit of transactions. In this mode, each SQL command is automatically committed unless it is inside an explicit transaction block. In the default mode, commands are committed only when **EXEC SQL COMMIT** is issued.

-v

Print additional information including the version and the "include" path.

--version

Print the ecpg version and exit.

-?
--help

Show help about ecpg command line arguments, and exit.

NOTES

When compiling the preprocessed C code files, the compiler needs to be able to find the ECPG header files in the PostgreSQL include directory. Therefore, you might have to use the **-I** option when invoking the compiler (e.g., -I/usr/local/pgsql/include).

Programs using C code with embedded SQL have to be linked against the libecpg library, for example using the linker options -L/usr/local/pgsql/lib -lecpg.

The value of either of these directories that is appropriate for the installation can be found out using **pg_config**(1).

EXAMPLES

If you have an embedded SQL C source file named prog1.pgc, you can create an executable program using the following sequence of commands:

```
ecpg prog1.pgc
cc -I/usr/local/pgsql/include -c prog1.c
cc -o prog1 prog1.o -L/usr/local/pgsql/lib -lecpg
```