IJG JPEG LIBRARY: CODING RULES

Copyright (C) 1991-1996, Thomas G. Lane.

This file is part of the Independent JPEG Group's software.

For conditions of distribution and use, see the accompanying README file.

Since numerous people will be contributing code and bug fixes, it's important

to establish a common coding style. The goal of using similar coding styles

is much more important than the details of just what that style is.

In general we follow the recommendations of "Recommended C Style and Coding

Standards" revision 6.1 (Cannon et al. as modified by Spencer, Keppel and

Brader). This document is available in the IJG FTP archive (see

jpeg/doc/cstyle.ms.tbl.Z, or cstyle.txt.Z for those without nroff/tbl).

Block comments should be laid out thusly:

/\*

\* Block comments in this style.

\*/

We indent statements in K&R style, e.g.,

if (test) {

then-part;

} else {

else-part;

}

with two spaces per indentation level. (This indentation convention is

handled automatically by GNU Emacs and many other text editors.)

Multi-word names should be written in lower case with underscores, e.g.,

multi\_word\_name (not multiWordName). Preprocessor symbols and enum constants

are similar but upper case (MULTI\_WORD\_NAME). Names should be unique within

the first fifteen characters. (On some older systems, global names must be

unique within six characters. We accommodate this without cluttering the

source code by using macros to substitute shorter names.)

We use function prototypes everywhere; we rely on automatic source code

transformation to feed prototype-less C compilers. Transformation is done

by the simple and portable tool 'ansi2knr.c' (courtesy of Ghostscript).

ansi2knr is not very bright, so it imposes a format requirement on function

declarations: the function name MUST BEGIN IN COLUMN 1. Thus all functions

should be written in the following style:

LOCAL(int \*)

function\_name (int a, char \*b)

{

code...

}

Note that each function definition must begin with GLOBAL(type), LOCAL(type),

or METHODDEF(type). These macros expand to "static type" or just "type" as

appropriate. They provide a readable indication of the routine's usage and

can readily be changed for special needs. (For instance, special linkage

keywords can be inserted for use in Windows DLLs.)

ansi2knr does not transform method declarations (function pointers in

structs). We handle these with a macro JMETHOD, defined as

#ifdef HAVE\_PROTOTYPES

#define JMETHOD(type,methodname,arglist) type (\*methodname) arglist

#else

#define JMETHOD(type,methodname,arglist) type (\*methodname) ()

#endif

which is used like this:

struct function\_pointers {

JMETHOD(void, init\_entropy\_encoder, (int somearg, jparms \*jp));

JMETHOD(void, term\_entropy\_encoder, (void));

};

Note the set of parentheses surrounding the parameter list.

A similar solution is used for forward and external function declarations

(see the EXTERN and JPP macros).

If the code is to work on non-ANSI compilers, we cannot rely on a prototype

declaration to coerce actual parameters into the right types. Therefore, use

explicit casts on actual parameters whenever the actual parameter type is not

identical to the formal parameter. Beware of implicit conversions to "int".

It seems there are some non-ANSI compilers in which the sizeof() operator

is defined to return int, yet size\_t is defined as long. Needless to say,

this is brain-damaged. Always use the SIZEOF() macro in place of sizeof(),

so that the result is guaranteed to be of type size\_t.

The JPEG library is intended to be used within larger programs. Furthermore,

we want it to be reentrant so that it can be used by applications that process

multiple images concurrently. The following rules support these requirements:

1. Avoid direct use of file I/O, "malloc", error report printouts, etc;

pass these through the common routines provided.

2. Minimize global namespace pollution. Functions should be declared static

wherever possible. (Note that our method-based calling conventions help this

a lot: in many modules only the initialization function will ever need to be

called directly, so only that function need be externally visible.) All

global function names should begin with "jpeg\_", and should have an

abbreviated name (unique in the first six characters) substituted by macro

when NEED\_SHORT\_EXTERNAL\_NAMES is set.

3. Don't use global variables; anything that must be used in another module

should be in the common data structures.

4. Don't use static variables except for read-only constant tables. Variables

that should be private to a module can be placed into private structures (see

the system architecture document, structure.doc).

5. Source file names should begin with "j" for files that are part of the

library proper; source files that are not part of the library, such as cjpeg.c

and djpeg.c, do not begin with "j". Keep source file names to eight

characters (plus ".c" or ".h", etc) to make life easy for MS-DOSers. Keep

compression and decompression code in separate source files --- some

applications may want only one half of the library.

Note: these rules (particularly #4) are not followed religiously in the

modules that are used in cjpeg/djpeg but are not part of the JPEG library

proper. Those modules are not really intended to be used in other

applications.