NAME

indent - indent and format C program source

SYNOPSIS

indent [*input-file* [*output-file*]] [**-bacc** | **-nbacc**] [**-bad** | **-nbad**] [**-badp** | **-nbadp**] [**-bap** | **-nbap**] [**-bbb** | **-nbbb**] [**-bc** | **-nbc**] [**-bl** | **-br**] [**-bs** | **-nbs**] [**-c***n*] [**-cd***n*] [**-cdb** | **-ncdb**] [**-ce** | **-nce**] [**-ci***n*] [**-cli***n*] [**-cs** | **-ncs**] [**-d***n*] [**-di***n*] [**-dj** | **-ndj**] [**-ei** | **-nei**] [**-eei** | **-neei**] [**-fbs** | **-nfbs**] [**-fc1** | **-nfc1**] [**-fcb** | **-nfcb**] [**-i***n*] [**-ip** | **-nip**] [**-l***n*] [**-lc***n*] [**-ldi***n*] [**-lp** | **-nlp**] [**-lpl** | **-nlpl**] [**-npro**] [**-P***file*] [**-pcs** | **-npcs**] [**-ps** | **-nps**] [**-psl** | **-npsl**] [**-sc** | **-nsc**] [**-sob** | **-nsob**] [**-st**] [**-ta**] [**-T***typename*] [**-ts***n*] [**-U***file*] [**-ut** | **-nut**] [**-v** | **-nv**] [**--version**]

DESCRIPTION

The **indent** utility is a *C* program formatter. It reformats the *C* program in the *input-file* according to the switches. The switches which can be specified are described below. They may appear before or after the file names.

NOTE: If you only specify an *input-file*, the formatting is done ‘in-place’, that is, the formatted file is written back into *input-file* and a backup copy of *input-file* is written in the current directory. If *input-file* is named ‘*/blah/blah/file*’, the backup file is named ‘*file.BAK*’ by default. The extension used for the backup file may be overridden using the SIMPLE_BACKUP_SUFFIX environment variable.

If *output-file* is specified, **indent** checks to make sure that it is different from *input-file*.

The options listed below control the formatting style imposed by **indent**.

-bacc, **-nbacc** If **-bacc** is specified, a blank line is forced around every conditional compilation block. For example, in front of every #ifdef and after every #endif. Other blank lines surrounding such blocks will be swallowed. Default: **-nbacc**.

-bad, **-nbad** If **-bad** is specified, a blank line is forced after every block of declarations. Default: **-nbad**.

-badp, **-nbadp** This is vaguely similar to **-bad** except that it only applies to the first set of declarations in a procedure (just after the first ‘{’) and it causes a blank line to be generated even if there are no declarations. The default is **-nbadp**.

-bap, **-nbap** If **-bap** is specified, a blank line is forced after every procedure body. Default: **-nbap**.

-bbb, **-nbbb** If **-bbb** is specified, a blank line is forced before every block comment. Default: **-nbbb**.

-bc, **-nbc** If **-bc** is specified, then a newline is forced after each comma in a declaration. **-nbc** turns off this option. Default: **-nbc**.

-bl, **-br** Specifying **-bl** lines up compound statements like this:

```
if (...)
{
  code
}
```

Specifying **-br** (the default) makes them look like this:

```
if (...) {
  code
}
```

-bs, **-nbs** Whether a blank should always be inserted after sizeof. The default is **-nbs**.

-c*n* The column in which comments on code start. The default is 33.

-cd*n* The column in which comments on declarations start. The default is for these comments to start in the same column as those on code.

-cdb, **-ncdb** Enables (disables) the placement of comment delimiters on blank lines. With this option enabled, comments look like this:

```
/*
 * this is a comment
 */
```

Rather than like this:

```
/* this is a comment */
```

This only affects block comments, not comments to the right of code. The default is **-cdb**.

-ce, **-nce** Enables (disables) forcing of ‘else’s to cuddle up to the immediately preceding ‘}’. The default is **-ce**.

-ci*n* Sets the continuation indent to be *n*. Continuation lines will be indented that far from the beginning of the first line of the statement. Parenthesized expressions have extra indentation added to indicate the nesting, unless **-lp** is in effect or the continuation indent is exactly half of the main indent. **-ci** defaults to the same value as **-i**.

-cli*n* Causes case labels to be indented *n* tab stops to the right of the containing **switch** statement. **-cli0.5** causes case labels to be indented half a tab stop. The default is **-cli0**.

-cs, **-ncs** Control whether parenthesized type names in casts are followed by a space or not. The default is **-ncs**.

-d*n* Controls the placement of comments which are not to the right of code. For example, **-d1** means that such comments are placed one indentation level to the left of code. Specifying the default **-d0** lines up these comments with the code. See the section on comment indentation below.

-di*n* Specifies the indentation, in character positions, of global variable names and all struct/union member names relative to the beginning of their type declaration. The default is **-di16**.

-dj, **-ndj** **-dj** left justifies declarations. **-ndj** indents declarations the same as code. The default is **-ndj**.

-ei, **-nei** Enables (disables) special **else-if** processing. If it is enabled, an **if** following an **else** will have the same indentation as the preceding **if** statement. The default is **-ei**.

-eei, **-neei** Enables (disables) extra indentation on continuation lines of the expression part of **if** and **while** statements. These continuation lines will be indented one extra level. The default is **-neei**.

-fbs, **-nfbs** Enables (disables) splitting the function declaration and opening brace across two lines. The default is **-fbs**.

-fc1, **-nfc1** Enables (disables) the formatting of comments that start in column 1. Often, comments whose leading ‘/’ is in column 1 have been carefully hand formatted by the programmer. In such cases, **-nfc1** should be used. The default is **-fc1**.

-fcb, **-nfcb** Enables (disables) the formatting of block comments (ones that begin with ‘/*\n’). Often, block comments have been not so carefully hand formatted by the

programmer, but reformatting that would just change the line breaks is not wanted. In such cases, **-nfcb** should be used. Block comments are then handled like box comments. The default is **-fcb**.

-i*n* The number of columns for one indentation level. The default is 8.

-ip, **-nip** Enables (disables) the indentation of parameter declarations from the left margin. The default is **-ip**.

-l*n* Maximum length of an output line. The default is 78.

-lc*n* Maximum length of an output line in a block comment. The default is 0, which means to limit block comment lines in accordance with **-l**.

-ldi*n* Specifies the indentation, in character positions, of local variable names relative to the beginning of their type declaration. The default is for local variable names to be indented by the same amount as global ones.

-lp, **-nlp** Lines up code surrounded by parentheses in continuation lines. With **-lp**, if a line has a left paren which is not closed on that line, then continuation lines will be lined up to start at the character position just after the left paren. For example, here is how a piece of continued code looks with **-nlp** in effect:

```
p1 = first_procedure(second_procedure(p2, p3),
  third_procedure(p4, p5));
```

With **-lp** in effect (the default) the code looks somewhat clearer:

```
p1 = first_procedure(second_procedure(p2, p3),
                     third_procedure(p4, p5));
```

Inserting two more newlines we get:

```
p1 = first_procedure(second_procedure(p2,
                                      p3),
                     third_procedure(p4,
                                     p5));
```

-lpl, **-nlpl** With **-lpl**, code surrounded by parentheses in continuation lines is lined up even if it would extend past the right margin. With **-nlpl** (the default), such a line that would

extend past the right margin is moved left to keep it within the margin, if that does not require placing it to the left of the prevailing indentation level. These switches have no effect if **-nlp** is selected.

-npro Causes the profile files, '*./.indent.pro*' and '*~/.indent.pro*', to be ignored.

-P*file* Read profile from *file*.

-pcs, **-npcs** If true (**-pcs**) all procedure calls will have a space inserted between the name and the '('. The default is **-npcs**.

-ps, **-nps** If true (**-ps**) the pointer dereference operator ('->') is treated like any other binary operator. The default is **-nps**.

-psl, **-npsl** If true (**-psl**) the names of procedures being defined are placed in column 1 - their types, if any, will be left on the previous lines. The default is **-psl**.

-sc, **-nsc** Enables (disables) the placement of asterisks ('*'s) at the left edge of all comments. The default is **-sc**.

-sob, **-nsob** If **-sob** is specified, indent will swallow optional blank lines. You can use this to get rid of blank lines after declarations. Default: **-nsob**.

-st Causes **indent** to take its input from stdin and put its output to stdout.

-ta Automatically add all identifiers ending in "_t" to the list of type keywords.

-T*typename* Adds *typename* to the list of type keywords. Names accumulate: **-T** can be specified more than once. You need to specify all the typenames that appear in your program that are defined by **typedef** - nothing will be harmed if you miss a few, but the program will not be formatted as nicely as it should. This sounds like a painful thing to have to do, but it is really a symptom of a problem in C: **typedef** causes a syntactic change in the language and **indent** cannot find all instances of **typedef**.

-ts*n* Assumed distance between tab stops. The default is 8.

-U*file* Adds type names from *file* to the list of type keywords.

-ut, **-nut** Enables (disables) the use of tab characters in the output. The default is **-ut**.

-v, **-nv** **-v** turns on ‘verbose’ mode; **-nv** turns it off. When in verbose mode, **indent** reports when it splits one line of input into two or more lines of output, and gives some size statistics at completion. The default is **-nv**.

--version Causes **indent** to print its version number and exit.

You may set up your own ‘profile’ of defaults to **indent** by creating a file called *.indent.pro* in your login directory and/or the current directory and including whatever switches you like. A ‘.indent.pro’ in the current directory takes precedence over the one in your login directory. If **indent** is run and a profile file exists, then it is read to set up the program’s defaults. Switches on the command line, though, always override profile switches. The switches should be separated by spaces, tabs or newlines.

Comments

‘*Box*’ *comments*. The **indent** utility assumes that any comment with a dash or star immediately after the start of comment (that is, ‘/*-’ or ‘/**’) is a comment surrounded by a box of stars. Each line of such a comment is left unchanged, except that its indentation may be adjusted to account for the change in indentation of the first line of the comment.

Straight text. All other comments are treated as straight text. The **indent** utility fits as many words (separated by blanks, tabs, or newlines) on a line as possible. Blank lines break paragraphs.

Comment indentation

If a comment is on a line with code it is started in the ‘comment column’, which is set by the **-c***n* command line parameter. Otherwise, the comment is started at *n* indentation levels less than where code is currently being placed, where *n* is specified by the **-d***n* command line parameter. If the code on a line extends past the comment column, the comment starts further to the right, and the right margin may be automatically extended in extreme cases.

Preprocessor lines

In general, **indent** leaves preprocessor lines alone. The only reformatting that it will do is to straighten up trailing comments. It leaves embedded comments alone. Conditional compilation (**#ifdef...#endif**) is recognized and **indent** attempts to correctly compensate for the syntactic peculiarities introduced.

C syntax

The **indent** utility understands a substantial amount about the syntax of C, but it has a ‘forgiving’ parser. It attempts to cope with the usual sorts of incomplete and malformed syntax. In particular, the use of macros like:

```
#define forever for(;;)
```

is handled properly.

ENVIRONMENT

The **indent** utility uses the HOME environment variable.

FILES

./.indent.pro profile file
~/.indent.pro
profile file

HISTORY

The **indent** command appeared in 4.2BSD.

BUGS

The **indent** utility has even more switches than ls(1).

A common mistake is to try to indent all the *C* programs in a directory by typing:

```
indent *.c
```

This is probably a bug, not a feature.