Command: **indent – reformat the layout of a program**
Syntax: **indent** *in_file* [*out_file*] [*options*]
Flags: (many)
Examples: indent –br –c25 prog.c # Indent *prog.c*
indent –npcs prog.c newprog.c # Put output on *newprog.c*

Indent reads a C program in, rearranges the layout, and outputs a new C program that will compile to the same executable binary as the original one. The difference between the input and output is that the output is in a standard layout determined by a large number of options. For most of the options there are two choices, one that enables it and one that disables it.

If *indent* is called with no file files, it operates as a filter. If called with one file name, that file is reformatted and the result replaces the original file. A backup is created, however, with the suffix *.BAK*. If it is called with two file names, the first one is the input file and the second one is the output file. Only one file can be reformatted at a time (e.g., one cannot call *indent* with *.c as argument; this is an error and will not work.).

Options

Many options are available. If you want to format a program to the ‘‘official’’ MINIX format, use *pretty*, which calls *indent* with the proper options and then post-processes the output. The options listed below control the formatting style.

OPTION: **–bad**, **–nbad**

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

OPTION: **–bap**, **–nbap**

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

OPTION: **–bbb**, **–nbbb**

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

OPTION: **–bc**, **–nbc**

If **–bc** is specified, then a newline is forced after each comma in a declaration. **–nbc** turns off this option. The default is **–nbc**.

OPTION: **–bl**, **–br**

Specifying **–bl** lines up compound statements like this:

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

```
}
```

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

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

OPTION: **–c***n*

The column in which comments on code start. The default is 33.

OPTION: **–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.

OPTION: **–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**.

OPTION: **–ce**, **–nce**

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

OPTION: **–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. **–ci** defaults to the same value as **–i**.

OPTION: **–cli***n*

Causes case labels to be indented *n* tab stops to the right of the containing switch statement. **–cli**0.5 causes case labels to be indented half a tab stop. The

default is **–cli**0. (This is the only option that takes a fractional argument.)

OPTION: **–d***n*

Controls the placement of comments which are not to the right of code. Specifying **–d**1 means that such comments are placed one indentation level to the left of code. The default **–d**0 lines up these comments with the code. See the section on comment indentation below.

OPTION: **–di***n*

Specifies the indentation, in character positions, from a declaration keyword to the following identifier. The default is **–di**16.

OPTION: **–dj**, **–ndj**

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

OPTION: **–ei**, **–nei**

Enables (disables) special else**–if** processing. If enabled, ifs following elses will have the same indentation as the preceding if statement. The default is **–ei**.

OPTION: **–fc**1, **–nfc**1

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, **–nfc**1 should be used. The default is **–fc**1.

OPTION: **–i***n*

The number of spaces for one indentation level. The default is 8.

OPTION: **–ip**, **–nip**

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

OPTION: **–ln**

Maximum length of an output line. The default is 78.

OPTION: **–lp**, **–nlp**

Lines up code surrounded by parenthesis in continuation lines. 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.

OPTION: **–npro**

Causes the profile files, *.indent.pro* in both the current directory and the user's home directory to be ignored.

OPTION: **–pcs**, **–npcs**

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

OPTION: **–ps**, **–nps**

If true (**–ps**) the pointer following operator ''–>'' will be surrounded by spaces on either side. The default is **–nps**.

OPTION: **–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**.

OPTION: **–sc**, **–nsc**

Enables (disables) the placement of asterisks (*) at the left edge of all comments. The default is **–sc**.

OPTION: **–sob**, **–nsob**

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

OPTION: **–st**

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

OPTION: **–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 #typedefs. 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 typedefs.

OPTION: **–troff**

Causes *indent* to format the program for processing by *troff*. It will produce a fancy listing in much the same spirit as *vgrind*. If the output file is not specified, the default is standard output, rather than formatting in place.

OPTION: **–v**, **–nv**

The **–v** flag 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**.

User Profiles

You may set up your own profile of defaults to *indent* by creating a file called *.indent.pro* in either your login directory and/or the current directory and including whatever switches you like. Switches in *.indent.pro* in the current directory override those in your login directory (with the exception of **–T** type definitions, which just accumulate). If *indent* is run and a profile file exists, then it is read to set up the program's defaults. The switches should be separated by spaces, tabs or newlines. Switches on the command line, however, override profile switches.

Comments

Indent 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.

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

If a comment is on a line with code it is started in the comment column, which is set by the **–cn** 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 **–dn** 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.

6

C Syntax

Indent 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 misformed syntax. In particular, the use of macros like:

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

is handled properly.