

This is Google's cache of <http://netlib.bell-labs.com/cm/cs/cbook/2ediffs.html>. It is a snapshot of the page as it appeared on 30 Aug 2008 13:13:03 GMT. The [current page](#) could have changed in the meantime. [Learn more](#)

These search terms are highlighted: **errata c programming language**

[Text-only version](#)

## Errata for *The C Programming Language, Second Edition*

This lists all known errors in *The C Programming Language, Second Edition*, by Brian Kernighan and Dennis Ritchie (Prentice-Hall, 1988).

The pagination of the book has changed only in minor ways across printings of the English-**language** edition. These **errata** now include section numbers, which are more likely to be preserved across translations. This might help identify errors that survive into translated editions.

### Changes between first and second printing:

The first printing of the book was made before the Standard was finalized; these copies say "Based on Draft-Proposed ANSI C" on the front cover. All subsequent printings are identified by a large red "ANSI C" on the right center of the cover. Fortunately, the changes are minor; some repair our bugs, a few account for last-minute changes in the draft standard. These changes were made so early that they probably do not apply to you.

Two or three sentences in the Preface and Introduction are updated to describe the state of the Standard.

`atof` is in `stdlib.h`, not `math.h`; this changes pages 71(§4.2), 76(§4.2), 82(§4.7), 121(§5.11).

On page 86(§4.9), error corrected: missing automatic initializers are zero too.

On page 168(§7.8.7): changed 1 to 1.0 in *frand* example to avoid potential overflow.

Minor typos are corrected on pages 87(§4.10), 89(§4.11.12), 164(§7.6), 165(§7.7), 168(§7.8.7), 180(§8.6).

In particular, the comment on p. 180(§8.6) attached to `st_ctime` in the `stat` structure is repaired: "ctime" stands for "change time" not "created time".

The inconspicuous references to `noalias` on pages 192(§A.2.4) and 211 or 212(§A8.2) are removed.

The following paragraph is added to the end of section A6.6 (page 199):

"A pointer may be converted to another pointer whose type is the same except for the addition or removal of qualifiers (A4.4, A8.2) of the object type to which the pointer refers. If qualifiers are added, the new pointer is equivalent to the old except for restrictions implied by the new qualifiers. If qualifiers are removed, operations on the underlying object remain subject to the qualifiers in its actual declaration."

On page 199, beginning of §A6.8, "Any pointer may be converted to type `void *`" is changed to "Any pointer *to an object* may be converted to type `void *`".

On page 204, §A7.4.4, "The operand of the unary `+` operator must have arithmetic or pointer type..." should read "must have arithmetic type...".

On page 206, §A7.9, about relational operators: ``Pointers to objects of the same type may be compared...'' is changed to ``Pointers to object of the same type (*ignoring any qualifiers*) may be compared...''.

The indented material on page 209 (§A7.17), ``According to the restrictions... relaxing it.'' is removed. [This is related to the paragraph added above. The wording of the penultimate draft made it useless to take an `(int *)` pointer, cast it to `(const int *)`, then cast it back to `(int *)`.]

On page 219 middle (§A8.7), initialization of structures, add ``Unnamed bit-field members are ignored, and are not initialized.''.

Appendix B changes:

p 242 (§B1.1): Add ```fflush(NULL)` flushes all output streams.'' to `fflush` description.

p 243 (§B1.1): Change to ``it must be called before reading, writing *or any other operation*'' in `setvbuf` description.

p 249 (§B.3): Add ``Comparison functions treat arguments as `unsigned char` arrays.'' to `string.h` description.

p 255 (§B10): Change range of `tm_sec` to (0,61) for leap seconds.

p 255 (§B.10): Change `CLK_TCK` to `CLOCKS_PER_SEC`.

p 257 (§B11): Drop U and L suffixes from `limits.h` constants. `tm_sec` range is (0,61) here too (§B10).

Appendix C change:

p 261 (§C): Change ``External declarations without any specifiers...'' to ``External *data* declarations without any specifiers...''.

The index has been reprinted to fix a couple of typos and account for motion within Appendix A; one page of the table of contents is changed.

A later printing in October, 1989, made minor changes on page 131 (§6.2) to add `&` to the last example (`struct rect r, *rp = &r;`), on page 208 (§A.17) to change ``equal'' to ``unequal'' in the description of logical OR, and on page 254 (§B.8) to clarify that for automatics variables, only those declared `volatile` are restored to their latest values after a `setjmp / longjmp` sequence.

### Errors not corrected in any printing:

The following errors have not yet been fixed in any printing.

41 (§2.6): The loop at the bottom of the page is functionally equivalent to the one on page 29 (§1.9), but not identical in form, as is implied by the text.

44 (§2.7): `1U` is an `unsigned int`, not an `int`.

49 (§2.9): In the discussion of shift operators, `which must be positive' should be `which must be non-negative'.

53 (§2.12): The note under the table should say `&` as well as `+ - *` has higher precedence as a unary operator.

102(§5.4): NULL is standardly defined in `<stddef.h>`, although it also appears in several other headers like `<stdio.h>` and `<stdlib.h>`.

111(§5.7): Indent is too large, and a bit of the `month_day` program text is cut off.

114(§5.9): quote missing in "Jan, near top of page.

117(§5.10): In the `find` example, the program increments `argv[0]`. This is not specifically forbidden, but not specifically allowed either.

119-121(§5.11): The `qsort` discussion needs recasting in several ways. First, `qsort` is a standard routine in ANSI/ISO C, so the rendition here should be given a different name, especially because the arguments to standard `qsort` are a bit different: the standard accepts a base pointer and a count, while this example uses a base pointer and two offsets.

Also, the comparison-routine argument is not treated well. The call shown on p 119, with an argument

```
(int (*)(void*,void*)) (numeric? numcmp : strcmp)
```

is not only complicated, but only barely passes muster. Both `numcmp` and `strcmp` take `char *` arguments, but this expression casts pointers to these functions to a function pointer that takes `void *` arguments. The standard does say that `void *` and `char *` have the same representation, so the example will almost certainly work in practice, and is at least defensible under the standard. There are too many lessons in these pages.

136(§6.3) The `getword` function can overflow word by one character. A sufficient fix is to test `'--lim>1'` instead of `'>0'` in the for loop.

142(§6.5, toward the end): The remark about casting the return value of `malloc` ("the proper method is to declare ... then explicitly coerce") needs to be rewritten. The example is correct and works, but the advice is debatable in the context of the 1988-1989 ANSI/ISO standards. It's not necessary (given that coercion of `void *` to `ALMOSTANYTYPE *` is automatic), and possibly harmful if `malloc`, or a proxy for it, fails to be declared as returning `void *`. The explicit cast can cover up an unintended error. On the other hand, pre-ANSI, the cast was necessary, and it is in C++ also.

143(§6.5): `strdup` is not indexed.

154(§7.2) table 7-1: arguments corresponding to `o`, `x`, `X`, `u` should be described as `unsigned int`.

158(§7.4) table 7-2: type of arguments corresponding to `o`, `x` should be `unsigned int *`.

164,165(§7.7): (text and code) `fputs` returns EOF on error, non-negative for OK.

165(§7.7): The comment at the beginning of the `fgets` example, "get at most n chars" would be better written "get at most n-1 chars, plus a null".

167(§7.8.5): [the return value of `malloc` or `calloc`] "must be cast into the appropriate type" is incorrect as stated. See the remarks just above for p. 142.

176-178(§8.5): `OPEN_MAX` is used to define the largest number of simultaneously open files in the example `stdio.h`; the ANSI/ISO name for this is instead `FOPEN_MAX`.

187-188(§8.7): `malloc` and the accompanying routine `morecore` are not presented as a whole program, and the routines should be OK, but moving the declaration  

```
static Header *morecore(unsigned);
```

to file scope (instead of inside `malloc`) would be a better way to make sure scope and linkage issues are handled properly.

193(§A.2.5.1): If an integer constant is suffixed with `UL`, it is `unsigned long`.

195(§A4): To the list at the start of the section, "Identifiers, or names, refer to a variety of things...", add labels as well.

195(§A4.1) The first few sentences might be reworded a bit to emphasize that there is a distinction between storage duration and scope, though both are influenced by explicit and implicit storage-class specifiers.

197(§A5): "An object is a named region of storage..." should read "named or pointed-to region..."

200(§ A7.1): The actual list of operators that prevent the conversion of arrays into pointers is just `&` and `sizeof`, not the various assignment operators. In practice this is not important, because arrays (converted or not) cannot be assigned to for other reasons.

207 (§A7.10): add to first paragraph: equality operators on pointers may also be applied to pointers to different objects.

229(§A12.3), 239(§A13): Syntax: the argument list is optional (can be empty) in macro definitions with parentheses.

231(§A12.3): Extra right parenthesis in nested call to `cat` macro.

232(§A12.5): The result of the `defined` operator is not replaced literally by `0L` or `1L`, nor are undefined names literally by `0L`, but just by plain `0` or `1`. However, the constant expression is nevertheless evaluated as if these and other constants appearing have long or unsigned long type.

232(§A12.5), likewise on p. 239(§A13), in the sections on the pseudo-grammar of the preprocessor, the productions

```
preprocessor-conditional:
  if-line text elif-parts else-part_opt # endif
```

```
elif-parts:
  elif-line text
  elif-parts_opt
```

should be written

```
preprocessor-conditional:
  if-line text elif-parts_opt else-part_opt # endif
```

```
elif-parts:
  elif-line text elif-parts_opt
```

244(§B1.2): In table B-1, the argument corresponding to `o`, `u`, `x`, `X` is `unsigned int`.

245(§B1.2): `vprintf`, `vfprintf`, `vsprintf` return `int`.

245(§B1.3, and also at p. 157 §7.4): The `scanf` functions do not ignore white space in formats; if white space occurs at a place in the format, any white space in the corresponding input is skipped.

246(§B1.3): In table B-2, arguments corresponding to `o`, `x`, `X`, `u` are `unsigned int *`.

246(§B1.3): First argument of `sscanf` should have type `const char *`.

249(§B3): In the description of `strncpy`, `t` should be `ct`.

210(§A8.1): It is said that a register object cannot have the `&` operator applied 'explicitly or implicitly,' which is true, but needs an amplifying clause to show that the implicit `&` comes from mention of an array name, so declaring an array as a register is fruitless.

There is no mention of the `offsetof` macro in §B.

257,258(§B11): In the definitions of both `FLT_MAX_EXP` and `DBL_MAX_EXP`, the expression `FLT_RADIX^n - 1` should be written `FLT_RADIX^(n-1)`.

Index: `stddef.h` is listed but not summarized in the text; It needs a section in Appendix B.

`char`, `double`, `long`, `short` are on page 9, not page 10. `sizeof` is on 242, not 247. "byte" is not indexed. `size_t` does occur on p. 247 in declarations.

*Updated Oct 23 2006 with program bug on line 136*

[Copyright © 2002 Lucent Technologies](#). All rights reserved.