NAME

fmtcheck - sanitizes user-supplied printf(3)-style format string

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <stdio.h>

*const char **
fmtcheck(*const char *fmt_suspect*, *const char *fmt_default*);

DESCRIPTION

The **fmtcheck**() scans *fmt_suspect* and *fmt_default* to determine if *fmt_suspect* will consume the same argument types as *fmt_default* and to ensure that *fmt_suspect* is a valid format string.

The printf(3) family of functions cannot verify the types of arguments that they are passed at run-time. In some cases, like catgets(3), it is useful or necessary to use a user-supplied format string with no guarantee that the format string matches the specified arguments.

The **fmtcheck**() was designed to be used in these cases, as in:

```
printf(fmtcheck(user_format, standard_format), arg1, arg2);
```

In the check, field widths, fillers, precisions, etc. are ignored (unless the field width or precision is an asterisk ‘*’ instead of a digit string). Also, any text other than the format specifiers is completely ignored.

RETURN VALUES

If *fmt_suspect* is a valid format and consumes the same argument types as *fmt_default*, then the **fmtcheck**() will return *fmt_suspect*. Otherwise, it will return *fmt_default*.

SEE ALSO

printf(3)

BUGS

The **fmtcheck**() function does not recognize positional parameters.

SECURITY CONSIDERATIONS

Note that the formats may be quite different as long as they accept the same arguments. For example,

"%p %o %30s %#llx %-10.*e %n" is compatible with "This number %lu %d%% and string %s has %qd numbers and %.*g floats (%n)". However, "%o" is not equivalent to "%lx" because the first requires an integer and the second requires a long.