Defect Report DIN-001

Submission Date: 12 Jun 95
Submittor: DIN
Source: Jutta Degener*

Question 1

According to the current standard, programs containing

```
char array[] = "Hello, World";
```

are not strictly conforming.

A Constraint in Subclause 6.5.7, "Initialization", demands that

> All the expressions in an initializer for an object that has static storage duration or in an initializer list for an object that has aggregate or union type shall be constant expressions.

Subclause 6.7, "Constant expressions", defines various kinds of constant expression; in its **Semantics** it states that a constant expression in an initializer evaluates to one of the following:

— an arithmetic constant expression
— a null pointer constant,
— an address constant, or
— an address constant for an object type plus or minus an integral constant expression.

String literals are neither. (A string literal used to initialize a character array does not decay to a pointer to its first element, according to Subclause 6.2.2.1.

> Except when it is the operand of the **sizeof** operator or the unary **&** operator, or is a character string literal used to initialize an array of character type, or is a wide string literal used to initialize an array compatible with **wchar_t**, an lvalue that has type "array of *type*" is converted to an expression that has type "pointer to *type*" that points to the initial element of the array object and is not an lvalue.

and hence is *not* an address constant.)

Suggested Technical Corrigendum

In Subclause 6.5.7, change

> All the expressions in an initializer for an object that has static storage duration or in an initializer list for an object that has aggregate or union type shall be constant expressions.

to

> All the expressions in an initializer for an object that has static storage duration or in an initializer list for an object that has aggregate or union type shall be constant expressions **or string literals**.

---

* This Defect Report was prepared with considerable help from Mark Brader, Clive Feather, Ronald Guilmette, and a person whose employment conditions require anonymity.

Defect Report DIN-002

Submission Date: 12 Jun 95
Submittor: DIN
Source: Jutta Degener*

Question 1

The standard's specification of what

```
printf("%#.0o", 0);
```

outputs is ambiguous, and compiler vendors have indeed interpreted it differently.

For a zero integer value, the descriptions of the **#** flag and the **0** precision in Subclause 7.9.6.1 contradict each other. The **#** demands that the first digit be zero;

The result is to be converted to an "alternate form." For **o** conversion, it increases the precision, if and only if necessary, to force the first digit of the result to be a zero.

but a precision of **0** demands that nothing at all be printed.

o,u,x,X [...] The result of converting a zero value with a precision of zero is no characters.

In the hexadecimal case, the description of the **#** flag's effects has been worded such that the conflict is avoided:

[...] For **x** (or **X**) conversion, a nonzero result will have **0x** (or **0X**) prefixed to it.

If it was intended that the octal case, too, should print nothing, this crucial "nonzero" should be introduced into its description as well.

Suggested Technical Corrigendum

In 7.9.6.1, replace

For **o** conversion, it increases the precision, if and only if necessary, to force the first digit of **the result** to be a zero.

by

For **o** conversion, it increases the precision, if and only if necessary, to force the first digit of **a nonzero result** to be a zero.

---

* This Defect Report was prompted by articles posted to comp.std.c by Bakul Shah, Rex Jaeschke, and Soenke Behrens.

Defect Report DIN-003

Submission Date: 12 Jun 95
Submittor: DIN
Source: Jutta Degener*

Question 1

Can **longjmp** be used to return from a signal handler invoked other than through **abort** or **raise**? The descriptions of **signal** and **longjmp** contradict each other.

According to Subclause 7.7.1.1, "The **signal** function,"

> If the signal occurs other than as the result of calling the **abort** or **raise** function, the behavior is undefined if the signal handler calls any function in the standard library other than the **signal** function itself (with a first argument of the signal number corresponding to the signal that caused the invocation of the handler) or refers to any object with static storage duration other than by assigning a value to a static storage duration variable of type **volatile sig_atomic_t**.

Since **longjmp** is a function, it cannot be called.

But Subclause 7.6.2.1, "The **longjmp** function," broadly guarantees the opposite.

> As it bypasses the usual function call and return mechanisms, the **longjmp** function shall execute correctly in contexts of interrupts, signals and any of their associated functions.

Suggested Technical Corrigendum

If the intent is to *allow* calls to **exit** and **longjmp** from signal handlers not invoked through calls to **raise** or **abort**, replace in 7.6.2.1

> ... other than the **signal** function itself ...

by

> ... other than **longjmp**, **exit**, or the **signal** function itself ...

Alternatively, if the intent is to *disallow* calls to **longjmp** from signal handlers not invoked through calls to **raise** or **abort**, replace in 7.6.2.1

> As it bypasses the usual function call and return mechanisms, the **longjmp** function shall execute correctly in contexts of interrupts, signals and any of their associated functions. However, if the **longjmp** function is invoked from a nested signal handler (that is, from a function invoked as a result of a signal raised during the handling of another signal), the behavior is undefined.

by

> If the **longjmp** function is invoked from a nested signal handler (that is, from a function invoked as a result of a signal raised during the handling of another signal), the behavior is undefined.

---

* This Defect Report was prepared with considerable help from Mark Brader, Clive Feather, Ronald Guilmette, and a person whose employment conditions require anonymity.