C107
WG14/N051
13.6.88

Additional Comments from the BSI C Panel

1. Comments on the signal() function

Consider the code fragment :

```
void (*old_handler)(), (*new_handler)();
old_handler = signal(SIGINT, new_handler);
```

Suppose that a SIGINT event occurs during the call on signal(); how is it guaranteed that the value of new_handler is not returned and placed in old_handler?

Also note that **longjump**() has to be coded in such a way that it can be interrupted by a SIGINT. If a conforming program is not allowed to return from SIGFPE and is not allowed to call a library routine other than signal(), how is control returned?

It is desirable to be able to **return** from a signal handler (particularly SIGFPE) with a guaranteed chance of interregating a **volatile** flag set by the handler.

Perhaps either the standard or the rationale could include an example of how a strictly conforming program should handle both SIGFPE and SIGINT.

Note

These comments refer to the draft dated 11th January 1988, in terms of both content and section and page numbers.

1) Calling Library Functions from Signal Handlers (4.7.1.1)

The draft indicates that calling library functions from inside a signal handler results in undefined behaviour (unless the *abort* or *raise* function). Hence if a program is to be portable (and the major aim of a standard is allow portabable pr to library functions in its signal handlers. Consider the following very simple signal handler, which has much in comm

```
catch(int sig)
{
  signal(sig, SIG_IGN);
  printf("Signal (%d) received\n", sig);
  exit(sig);
}
```

by the present draft it is triply unportable as every statement in it is a call to a library function.

Similarly a signal handler which sets a flag and calls the *longjmp* function is ruled out because it calls a library fun returns is not safe: the signal might be *SIGFPE* in which case returning results in undefined behaviour.

Clearly signal handling presents problems to some implementors; however the present proposal is unacceptable to portable interactive programs since it makes it impossible to write signal handlers which are both useful and portable.

Proposed Changes

The preferred change is to allow signal handlers to call at least a limited range of library functions (including at least *si* functions). It would be acceptable to limit the number of currently active signal handlers and/or input/output functions

A less acceptable, but still useful, change would be to replace the word "undefined" with "implementation defined" o provide the programmer with some documented behaviour on which to build.

2) Environmental Limits: BUFSIZ and Line Length (4.9.2)

Neither the macro *BUFSIZ* nor any of the other macros in the enviromental limits section give any clue to the maxi implementation (except that it be at least 254 characters). It is proposed to add an additional macro to <*stdio.h*> (c characters allowed. On systems with no upper limit this could take the same value as *INT_MAX* or even *ULONG_MA*

It could perhaps be argued that *LINELENGTH_MAX*, and perhaps also *BUFSIZ*, would be better located in <*limits.h*>.

3) String Termination with strncat and strncpy (4.11.2.4 & 4.11.3.2)

There is an inconsistency between these functions: *strncpy* copies at most *n* bytes, potentially leaving the new strin appends a null terminator, even if this mean adding *n+1* bytes. Even given the existing custom in this area the changing the behaviour of one of the functions (preferably *strncpy*), or by adding an extra function which would be a t

David Furber
King's College London
16th May 1988