IEEE Standards Interpretations for IEEE Std 1003.1™-2001 IEEE Standard for Information Technology - Portable Operating System Interface (POSIX®)

Interpretation Request #20
Topic: c99 **Relevant Sections:** Output Files

In earlier versions of this standard, it was generally assumed, but not explicitly specified that object files and executable files were regular files. With the changes that were made to XBD subclause 1.7.1.4 (File Read, Write, and Creation) in this revision and wording in c99 that doesn't quite match the templates in XBD 1.7.1.4, some people have interpreted the standard to require that c99 be able to write object files and executable files to files of type fattach()-ed STREAM, block special, character special, and FIFO special as well as to regular files. I do not believe that this was the intent when the c89 and c99 utilities were originally drafted. I do not object to implementations supporting all of these file types, but I see no reason for the standard to require that compilers work correctly with anything but regular files (and symlinks pointing to regular files).

Add a new sentence to the end of the paragraph in the OUTPUT FILES section on P214, L8400: If an existing file that does not resolve to a regular file matches the name of an object file being written or matches the name of an executable file being created by c99, it is unspecified whether c99 shall attempt to write the object file or create the executable file, or shall issue a diagnostic and exit with a non-zero exit status.

Add a new paragraph to the rationale after P218, L8602:
This standard specifies that the c99 utility must be able to use regular files for *.o files and for a.out files. Implementations are free to overwrite existing files of other types when attempting to create object files and executable files, but are not required to do so. If something other than a regular file is specified and using it fails for any reason, c99 is required to issue a diagnostic message and exit with a non-zero exit status. But for some file types, the problem may not be noticed for a long time. For example, if a FIFO named a.out exists in the current directory, c99 may attempt to open a.out and will

hang in the open() call until another process opens the FIFO for reading. Then c99 may write most of the a.out to the FIFO and fail when it tries to seek back close to the start of the file to insert a timestamp (FIFOs are not seekable files). The c99 utility is also allowed to issues a diagnostic immediately if it encounters an a.out or *.o file that is not a regular file. For portable use, applications should ensure that any a.out, -o option-argument, or *.o files corresponding to any *.c files do not conflict with names already in use that are not regular files or symbolic links that point to regular files.

Interpretation Response
The standards states the requirements for the c99 utility, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.

Rationale for Interpretation
None.

Notes to the Editor (not part of this interpretation)
A future revision should consider the following additions to c99: Add a new sentence to the end of the paragraph in the OUTPUT FILES section on P214, L8400: If an existing file that does not resolve to a regular file matches the name of an object file being written or matches the name of an executable file being created by c99, it is unspecified whether c99 shall attempt to write the object file or create the executable file, or shall issue a diagnostic and exit with a non-zero exit status.

Add a new paragraph to the rationale after P218, L8602: This standard specifies that the c99 utility must be able to use regular files for *.o files and for a.out files. Implementations are free to overwrite existing files of other types when attempting to create object files and executable files, but are not required to do so. If something other than a regular file is specified and using it fails for any reason, c99 is required to issue a diagnostic message and exit with a non-zero exit status. But for some file types, the problem may not be noticed for a long time. For example, if a FIFO named a.out exists in the current directory, c99 may attempt to open a.out and will hang in the open() call until another process opens the FIFO for reading. Then c99 may write most of the a.out to the FIFO and fail when it tries to seek back close to the start of the file to insert a timestamp (FIFOs are not seekable files). The c99 utility is also allowed to issues a diagnostic immediately if it encounters an a.out or *.o file that is not a regular file. For portable use, applications should ensure that any a.out, -o option-argument, or *.o files corresponding to any *.c files do not conflict with names already in use that are not regular files or symbolic links that point to regular files.