FAQ: What is 'wrong' with IBM iconv?

Contributed by Michael Felt

Is IBM iconv really wrong - or is it something else?

Short answer: Nothing is 'wrong' with the implementation according to the standards document. GNU autotools reports 'difference' as failed.

ICONV and OSS

I have had to load GNU iconv (better libiconv) almost as often as I load GNU gettext. Many OSS (based on GNU autotools) need codeset conversion and a test for a iconv() is performed. The IBM AIX implenentation 'fails' the test. Why?

The GNU autotools always fail an iconv() test on IBM AIX. What is wrong with IBM's implementation of iconv?

The answer is simple.

There is nothing 'wrong' with IBM's implementation!

IBM's implementation is just different - and GNU autotools would be more accurate if they reported a warning (i.e., there is a difference) rather than calling it an error.

From the opengroups documentation you can find: If iconv() encounters a character in the input buffer that is valid, but for which an identical character does not exist
in the target codeset, iconv() shall perform an implementation-defined conversion on this character.

The test that GNU fails is one that is trying to convert a valid input character to a codeset that does not have a direct (or valid?) conversion. IBM's implementation converts that character to a 'space' character (aka ASCII 32) while GNU returns an error. The test that GNU is failing is that the return value from iconv() is 0 when IBM iconv() is used - and it wants -1 (i.e., the GNU result).

The test is misleading

The opengroup standards commission gave implementors freedom of choice. It is misleading (of GNU autotools) to say IBM AIX is 'wrong' or 'failed' when it is only different. Maybe this would impact an application- that is going to rely on knowing there is not a complete literal translation (effect is that the translated code cannot be used to regenerate the original input). So, a warning that translations may not always be di-directional is warrented. But calling an implementation 'failed' is not accurate.

My two cents!

Is there a simple solution?

Probably. The solution to this has been to go to the IBM AIX Toolbox or to Michael Perzl's alturnate Toolbox and download one or two RPM's (one for 32-bit, one for 64-bit, still more for 'development').

I have a new solution. Like both of the toolbox approaches I extract the IBM original archive members from /usr/lib/libiconv.a and add them to the new archive (/opt/lib/libiconv.a) and replace the orignal archive with a symbolic link tothe new archive.

Why bother with adding the IBM members?

A rather frequent issue is a program is links with the GNU libiconv but later calls another library that linked with the IBM implementation. The linker has remembered /opt/lib/libiconv.a as the library with iconv_open() so it never resolves /usr/lib/libiconv.a for the later routine looking for the other member object (e.g., shr4_64.o). Nasty. So, the simple solution is to add those members to the archive and they are found as needed. The member libiconv.so.2 for the GNU approach and one of the members shr*.o for the IBM approach.

Sound interesting - a unified approach?

After years of doing this manually I decided to automate the packaging of GNU libiconv-1.14. If you are interested in trying this just go to http://www.aixtools.net/index.php/libiconv for my project page. I am working hard to make installing AIXTOOLS a 1-2-3 process:

- Download
- Unpack (if needed)
- Install (installp -d . all)

Post on http://forums.rootvg.net/aixtools if you have comments or need help!