Note on the Issue in compiling SPICE C Library in MacOS M1 Apple Silicon

R.Manikantan, ISTRAC/ISRO 7 Apr 2021

I was trying to compile the C version of the SPICE Library, version N0066 on my MacBook Air 2020 with M1 Apple Silicon chip, running macOS BigSur 11.2.3

There were issues in setting up the C compiler environment properly in this platform - just to even get gcc and getting the Hello-World program compiled properly. Those details are in the Appendix that follows, in case you need some help with that.

But this one is about the ERROR that I got even after having my Compiler environment setup OK:

```
Compiling:  inquire.c
inquire.c:24:7: error: implicit declaration of function
'access'
      [-Werror,-Wimplicit-function-declaration]
                x = access(buf,0) ? -1 : 0;

        ^
```

I made a quick search of which header-file gives the access() function declaration, and it was `unistd.h`

Just adding a #include line for this header-file in `inquire.c` solved the issue.

I untar-ed the C library `cspice.tar.Z` freshly, went to the `cspice/src/cspice` directory, added this `#include`.

The "`diff -c`" output of the old (`inquire.c.bkup`) and newer version (`inquire.c`) of this file look as follows:

```
*** inquire.c.bkup  Wed Apr  7 22:51:14 2021
--- inquire.c  Wed Apr  7 22:51:22 2021
***************
*** 1,6 ****
--- 1,7 ----
  #include "f2c.h"
  #include "fio.h"
  #include "string.h"
+ #include <unistd.h>
  #ifdef KR_headers
  integer f_inqu(a) inlist *a;
  #else
```

This was the only ERROR in the compilation process. However, there were several warnings the compiler throws out such as:

1.

```
ckr02.c:648:29: warning: operator '<<' has lower
precedence than '-'; '-' will be evaluated first
      [-Wshift-op-parentheses]
    n = beg + (skip + index - 1 << 3);
                ~~~~~~~~~~~~^~ ~~
ckr02.c:648:29: note: place parentheses around the '-'
expression to silence this warning
    n = beg + (skip + index - 1 << 3);
```

and

2.

```
        if(n=c_dfe(a))return(n);
```

```
            ~^~~~~~~~
dfe.c:97:6: note: place parentheses around the assignment
to silence this warning
        if(n=c_dfe(a))return(n);
             ^
            (          )
dfe.c:97:6: note: use '==' to turn this assignment into
an equality comparison
```

I think we can safely ignore these warnings as I am certain that the code-authors know C operator precedence and the behaviour with ‘=‘ in place of ‘==‘ etc., are exactly as intended only. It is just that the compiler on MacOS warns on this, just in case; but we know what we have done, and this can be ignored.

SUMMARY

Why we need to explicitly include unistd.h only in the MacOS Apple Silicon version of the library puzzles me, because, the `#include` for `unistd.h` is missing in MacOS on Intel Platform too, and the `access()` function is declared in the `unistd.h` file only, in Intel platform as well (I checked on my older Intel MacBook Pro). Somehow explicit inclusion of `unistd.h` seems to be mandatory on MacOS Apple Silicon, and doing this makes it all go through.

I have done some minimal testing with simple programs that generate AZEL angles, and calculate Light Time Solution using the SPICE library, and I get identical results (upto 6th decimal) with my programs on MacOS Intel and MacOX Apple Silicon.

APPENDIX:
C Compiler Environment on
MacOS M1 Apple Silicon / BigSur 11.2

The following steps were required to get the native C Compiler environment working on Apple Silicon. All these were learnt from several Google searches and helpful web-sites. Thanks to them!

1. Get Xcode installed.

You should be able to get it from Mac App Store. As on date, the version is 12.4

2. Install Command-line tools:

```
sudo xcode-select  —install
```

did it for me.

3. Set SDKROOT environment variable to point to the MacOSX SDK

Xcode comes with SDKs for MacOSX, iPhone, AppleTV, watchOS and all their suite. We need the MacOSX SDK here.

We may try to figure it out dynamically,

```
export SDKROOT=$(xcrun --sdk macosx —show-sdk-path)
```

or, as I have done, set it manually to the correct place (based on the output I saw with the above command):

```
export SDKROOT=/Applications/Xcode.app/Contents/
Developer/Platforms/MacOSX.platform/Developer/SDKs/
MacOSX.sdk
```

4. Set Developer directory environment variable. Again this may be done dynamically with `xcode-select` command, or statically (based on the dynamic output), as I have done here:

```
# Do, once only, manually for all users:
# sudo xcode-select -s /Applications/Xcode.app/Contents/
Developer
# or set it on per-user basis
export DEVELOPER_DIR=/Applications/Xcode.app/Contents/
Developer
```

5. At this point, you should have gcc going and should be able to test it with a quick Hello-World C program. Actually, it turns out that the C compiler installed here is the one called “clang”, and gcc, cc all seem to just be links to it. But I don’t know much more about it.

Note that we have not added any new entry to the PATH / LD_LIBRARY_PATH variable. I am using the macOS Terminal application, and it seems to figure out the necessary PATHs from just the SDKROOT and DEVELOPER_DIR environment variables.