[7.5] Considerations for referencing external programs

This is not so much as tip as it is some things to consider when you write a program that call routines which are external to that program. For example, suppose we have a program *A()* which calls program or function *B().* If both *A()* and *B()* are in the same folder, and that folder is the current folder, there is no problem. We have this:

```
A()                 Method 1
prgm
...
B()
...
endprgm
```

It is not so simple, however, when you consider some other likely cases. For example, B() might be a general-purpose utility program, located, for example, in the /utils folder. The most straightforward way to deal with this problem is to include the folder specification in the function call, like this:

```
A()                 Method 2
prgm
...
utils/B()
...
endprgm
```

This method has these disadvantages, because the folder name is hardcoded in program A():

- The *B()* utility must be in the */utils* folder, and it cannot be moved.
- The /utils folder cannot be renamed.
- An RAM or archive overhead is incurred each time *B()* is referenced in *A().*
- If you distribute your program to others, the */utils* folder must be created, and *B()* must be stored in it. Suppose that the user already uses that folder name, or already has a program of her own called *B()*? Then there is a conflict.

In many cases, the problem can be avoided by just keeping a copy of *B()* in the same folder as *A(),* and using method 1 above. This approach has these disadvantages:

- If *B()* is used my many programs, in many different folders, RAM or archive is wasted with each copy that is necessary.
- If changes are made to *B(),* you need to make sure that all of the copies are updated.

There is no clear solution to this problem. You need to weigh the advantages and disadvantages of each method. Most programmers use method 2, in spite of its disadvantages.