

# Wrapping routines

---

*Source:* <http://coding.derkeiler.com/Archive/Fortran/comp.lang.fortran/2007-11/msg00419.html>

---

- *From:* Arjen Markus <[arjen.markus@xxxxxxxxxxx](mailto:arjen.markus@xxxxxxxxxxx)>
  - *Date:* Mon, 12 Nov 2007 12:42:07 -0800
- 

Hello,

My question requires a rather lengthy preparation:

I have been experimenting a bit with wrapping a library, that is, I wanted to see if it is possible to place some other code between the library routines and the calling program without actually changing the source code of either.

For instance:

My program uses a routine `sub_a`. I want to see how `sub_a` is actually called (with what parameters) and what the results are. But I do not have the source code for it or I do not want to mess with it.

The strategy I follow is:

- routine `sub_a` is part of a module `mod_a`.
- I write a routine `sub_b` (part of a module `mod_b`) that calls `sub_a`.
- `sub_b` and `sub_a` have exactly the same interface, but `sub_b` prints some extra information.
- I now have a third module – with the same name "`mod_a`" – that renames the routines in `sub_b`, thus hiding the fact that there is a module `mod_b`.
- By storing all these modules in separate directories, (at least) the compiler does not see any conflicts.

Schematically:

```
testwrap:  
use mod_a (alternative module, gives access to mod_b)  
|  
+--> use mod_b (calls the original routines)  
|  
+--> mod_a (contains the original routines)
```

I have tried this on three different compilers (and associated linkers,

## Wrapping routines

g95, gfortran and CVF) and the result was what I hoped:

```
Wrapping sub_a ...  
Original sub_a: x = 1  
Original sub_ab: x = 1  
Calling sub_ab:  
Wrapping sub_ab ...  
Original sub_ab: x = 1
```

So the original routines are nicely hidden from the original program and the wrapping versions are called.

My question is:

Is this a mere coincidence? Or is this actually supposed to work this way?

The reason I ask is that all compiler I know of prepend the internal name of the routines with the name of the module. So I would have thought there would be a conflict between the routines from the two modules "mod\_a", but there seems to be none.

Below you will find the code.

Let me add that this is merely an experiment – I was curious whether this would work. I came across something like this in a completely different context.

Regards,

Arjen

---

The main program looks like this:

```
! testwrap.f90 --  
! Test whether wrapping existing subroutines works  
!  
program testwrap  
  
use mod_a  
  
integer :: x  
  
x = 1  
call sub_a( x )  
write(*,*) 'Calling sub_ab:'  
call sub_ab( x )  
end program testwrap
```

The original module "mod\_a" is (in subdirectory mod\_a):

## Wrapping routines

```
! mod_a.f90 --
! Original module
!
module mod_a

contains
subroutine sub_a( x )
integer :: x

write(*,*) 'Original sub_a: x = ', x
call sub_ab( x )
end subroutine sub_a
subroutine sub_ab( x )
integer :: x

write(*,*) 'Original sub_ab: x = ', x
end subroutine sub_ab
end module mod_a
```

Module "mod\_b" (in subdirectory mod\_b) wraps the two routines:

```
! mod_b.f90 --
! MModule wrapping mod_a
!
module mod_b
use mod_a
private
public :: sub_b
public :: sub_bb

contains
subroutine sub_b( x )
integer :: x

write(*,*) 'Wrapping sub_a ...'
call sub_a( x )
end subroutine sub_b
subroutine sub_bb( x )
integer :: x

write(*,*) 'Wrapping sub_ab ...'
call sub_ab( x )
end subroutine sub_bb
end module mod_b
```

And finally, the alternative module "mod\_a" renames these routines and makes them available in essentially the same way as the original ones – with the same module name:

```
! mod_c.f90 --
```

Wrapping routines

## Wrapping routines

```
! Rename mod_b to mod_a, hiding the wrapping
!  
module mod_a  
use mod_b, sub_a => sub_b, sub_ab => sub_bb  
private  
public sub_a  
public sub_ab  
end module mod_a
```

To create the program I used this shell script:

```
#!/usr/bin/sh  
#  
# Script to build the whole thing:  
# We juggle a lot with the directories  
#  
cd mod_a  
gfortran -c mod_a.f90  
  
cd ../mod_b  
gfortran -c mod_b.f90 -I../mod_a  
ar r modules.a mod_b.o ../mod_a/mod_a.o  
  
cd ../mod_c  
gfortran -c mod_c.f90 -I../mod_b  
  
cd ..  
gfortran -o testwrap testwrap.f90 -Imod_c mod_c/mod_c.o mod_b/modules.a  
  
.
```