Christopher Lum
Autonomous Flight Systems Laboratory
Updated: 12/09/05

Creating External Header Files for Functions in Mathematica

Introduction

In this document, we go over how to create complex functions (Modules) in Mathematica and then have them in an external file similar to a header file in C or C++.

Creating a Simple Function

Relevant Help Files

- `SetDelayed`
- `:=`

We can easily create a simple function. This can be defined as shown below in Figure 1.

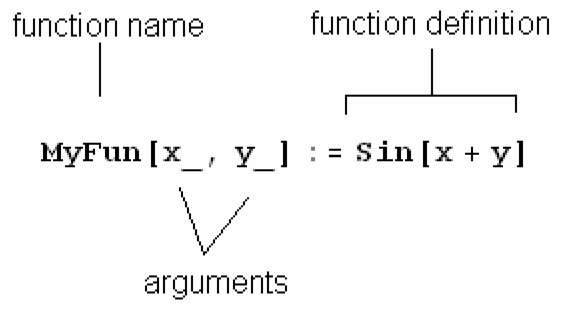


Figure 1: Creating a simple function

Items to notice

1. The function arguments are enclosed with square brackets, []
2. Each argument on the left side is followed by an underscore, _
3. The left side (function name and arguments) and the right side (function definition) is separated by a colon equal, :=. This is the `SetDelayed` concept (note that in many cases, a simple = is sufficient, but := and = mean different things. := is more versatile).

Creating a Module (Complex Function)

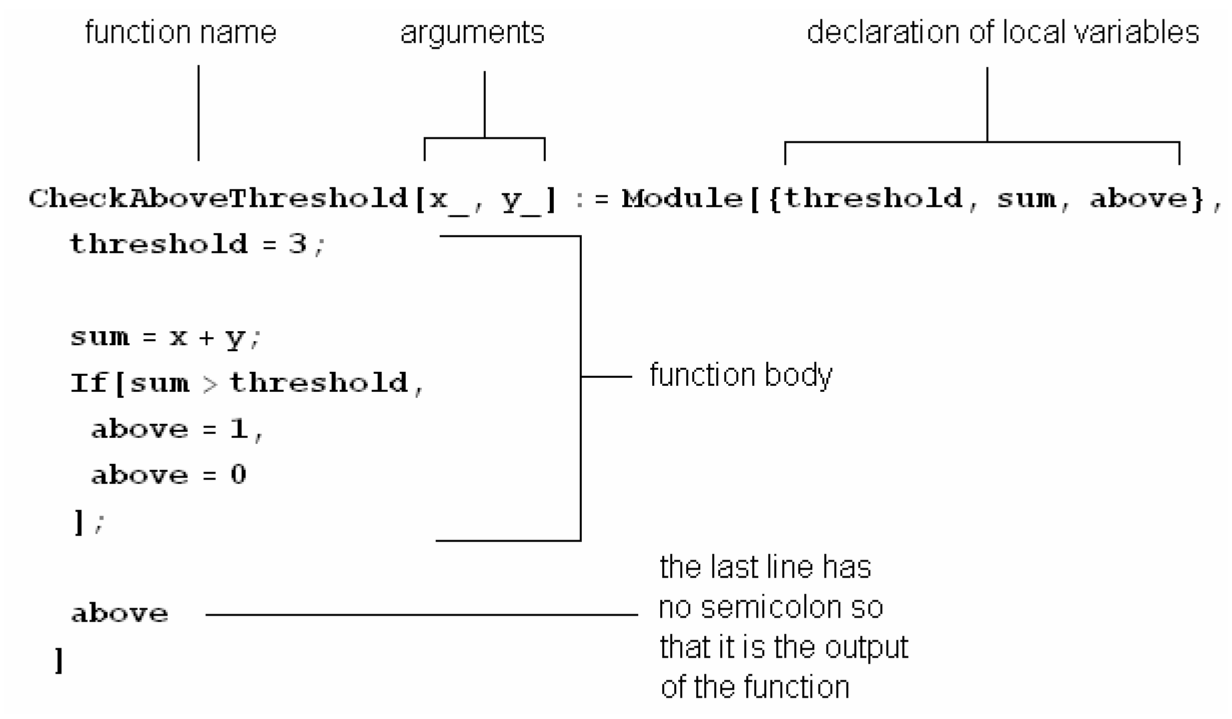
Relevant Help Files

- `Module`

The above method for defining a function is simple but falters if the function you want to define is complex (ie multiple steps). A more complex function can be can be generated using `Module` in Mathematica.

Functions:	*Module*
Note:	*Module[{a,b,…},procedure] defines a procedure with local variables a, b,…*

For example, let's create a function which sums the two arguments and then checks if this sum is above a certain threshold. This is shown below in Figure 2.



```
CheckAboveThreshold[x_, y_] := Module[{threshold, sum, above},
  threshold = 3;

  sum = x + y;
  If[sum > threshold,
   above = 1,
   above = 0
  ];

  above
 ]
```

function name
arguments
declaration of local variables
function body
the last line has no semicolon so that it is the output of the function

Figure 2: Example use of Module to create a complex function

Items to notice

1. _ and `SetDelay` := used as before

2. Use of returns and spacing to make overall function more readable
3. Only 1 line of the function has no semi-colon (this is the desired output of the function). Every other line (including `for` loops and `if` statements) have semi-colons.

In this example, `threshold`, `sum`, and `above` are all local variables. That is, they are not defined outside the scope of the function.

It may be helpful for debugging purposes have some variables not be defined as local functions. This way, we may view the value of the variable once the function has been run.

Creating a Library and Header File

Relevant Help Files

- `Off`
- `SetDirectory`
- `Put`
- `Save`
- `<<`

If there are a lot of custom functions that we would like to create or if they are exceedingly complex, we would like to be able to define them in a separate file (known as a library) and then simply include this library into other notebooks.

The procedure for this is

1. Define functions in library file (.nb file)
2. From library file, save the desired functions to a header file (.nbd file)
3. Include the header file (.nbd file) into other notebooks

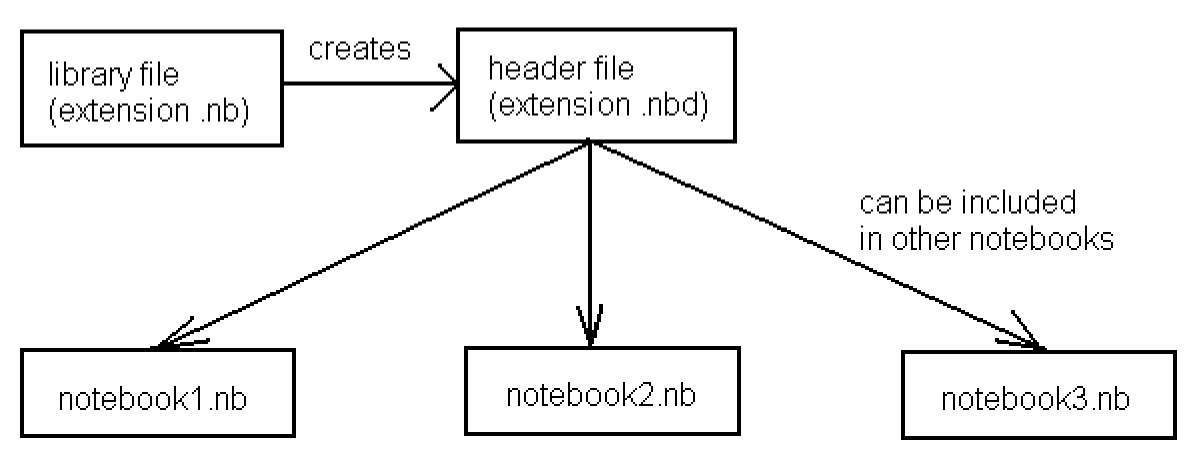
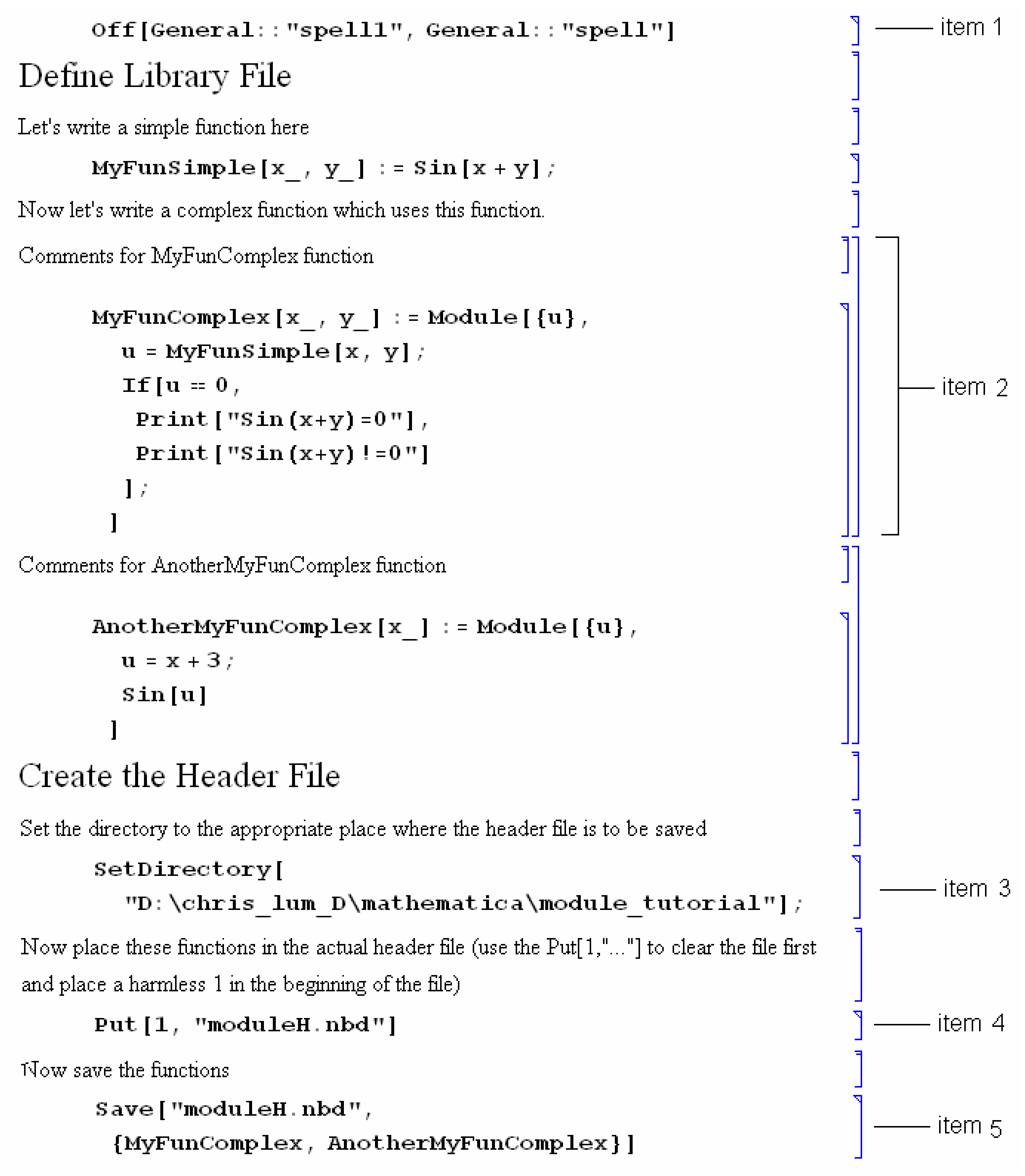

library file (extension .nb)
creates
header file (extension .nbd)
can be included in other notebooks
notebook1.nb
notebook2.nb
notebook3.nb

Figure 3: Diagram showning how different files are created/used

Step 1: Define functions in library file (.nb file)

We first define the library file. This file has all the comments about the function and the function definition. An example of a library file is shown below in Figure 4.



```
Off[General::"spell1", General::"spell"]
```

Define Library File

Let's write a simple function here

```
MyFunSimple[x_, y_] := Sin[x + y];
```

Now let's write a complex function which uses this function.

Comments for MyFunComplex function

```
MyFunComplex[x_, y_] := Module[{u},
  u = MyFunSimple[x, y];
  If[u == 0,
   Print["Sin(x+y)=0"],
   Print["Sin(x+y)!=0"]
  ];
 ]
```

Comments for AnotherMyFunComplex function

```
AnotherMyFunComplex[x_] := Module[{u},
  u = x + 3;
  Sin[u]
 ]
```

Create the Header File

Set the directory to the appropriate place where the header file is to be saved

```
SetDirectory[
  "D:\chris_lum_D\mathematica\module_tutorial"];
```

Now place these functions in the actual header file (use the Put[1,"..."] to clear the file first and place a harmless 1 in the beginning of the file)

```
Put[1, "moduleH.nbd"]
```

Now save the functions

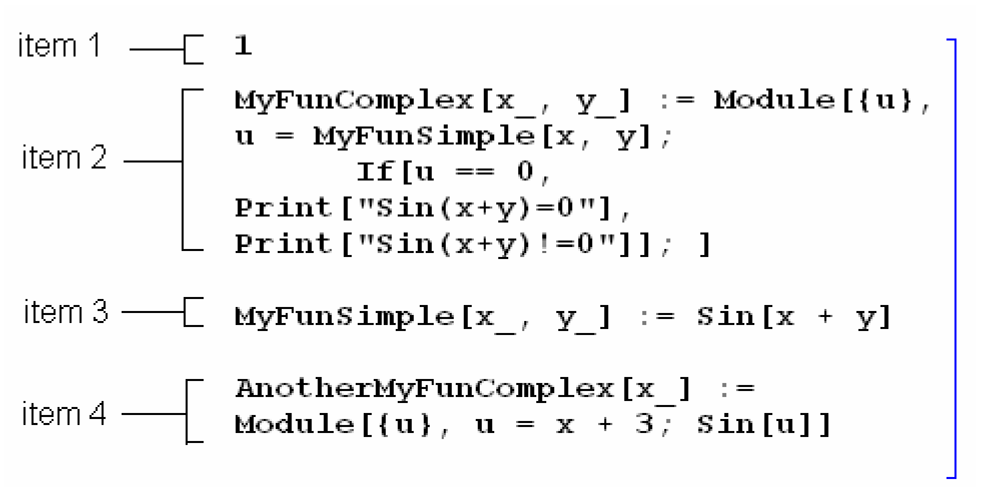
```
Save["moduleH.nbd",
 {MyFunComplex, AnotherMyFunComplex}]
```

Figure 4: Sample library file (moduleH.nb)

Items to notice
1. The `Off["…"]` is helpful for turning off similar spelling warnings
2. It may be helpful to group the comments for a function and the function definition together.
3. Set the directory to where you want to save the header file (.nbd)
4. `Put[1,"moduleH.nbd"]` creates a file moduleH.nbd and places a 1 at the very top of the file if it does not exists already. If the file already exists, then this simply deletes the old file and then puts a 1 at the top of the file. The 1 is arbitrary, we simply use it so that the old file is deleted and we start from scratch.
5. `Save["moduleH.nbd",{MyFunComplex,AnotherMyFunComplex}]` appends the functions to the file moduleH.nbd. Note that `MyFunComplex` requires the definition of `MyFunSimple`, but `MyFunSimple` is not explicitly saved. This is automatically saved by Mathematica.

Step 2: From library file, save the desired functions to a header file (.nbd file)

We already saved the desired functions to the header file (.nbd file). The file moduleH.nbd is shown below in Figure 5.



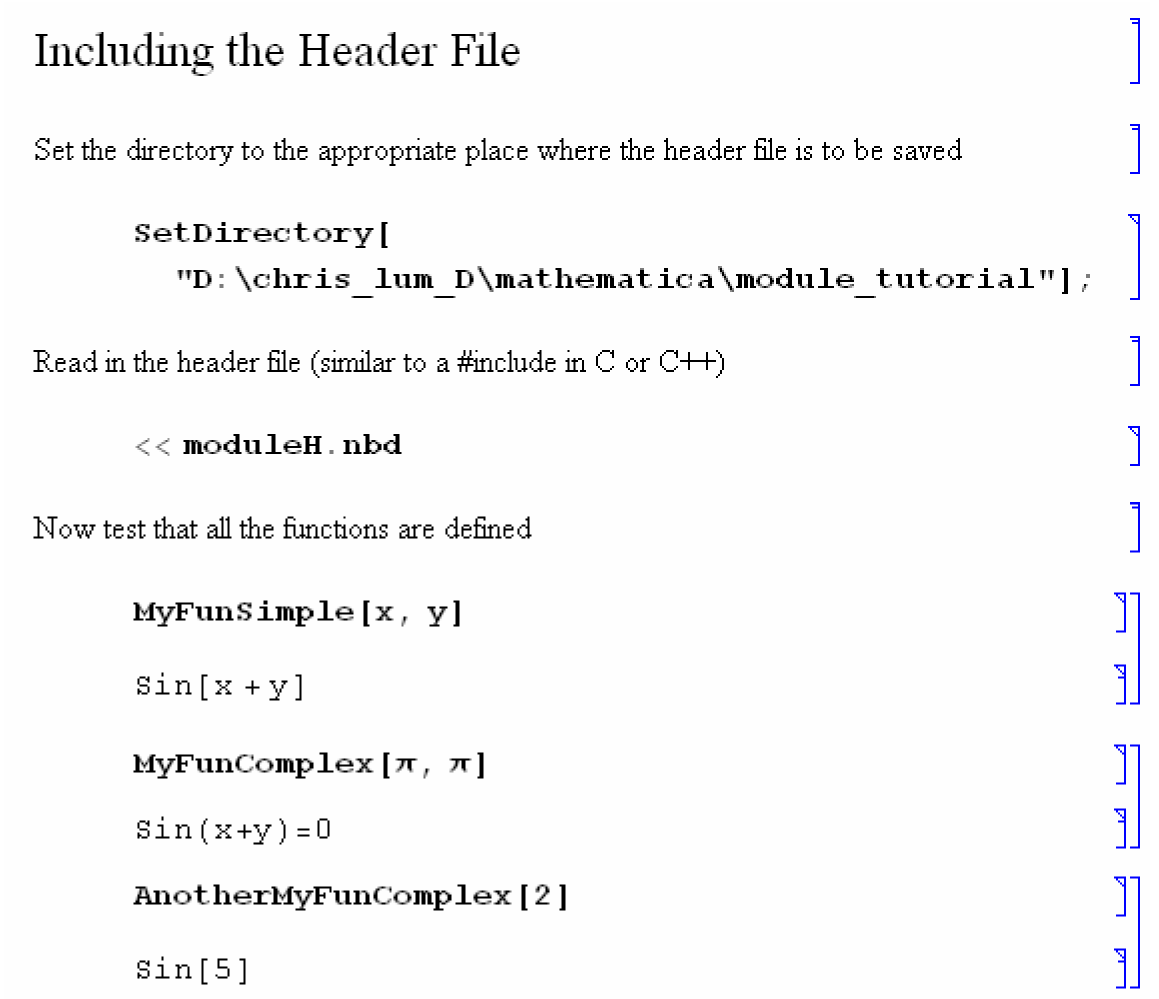
```
1

MyFunComplex[x_, y_] := Module[{u},
u = MyFunSimple[x, y];
      If[u == 0,
Print["Sin(x+y)=0"],
Print["Sin(x+y)!=0"]]; ]

MyFunSimple[x_, y_] := Sin[x + y]

AnotherMyFunComplex[x_] :=
Module[{u}, u = x + 3; Sin[u]]
```

Figure 5: Header file (moduleH.nbd)

Items to notice
1. This is the 1 that was placed using `Put[1,"moduleH.nbd"]`
2. Function definition for `MyFunComplex`
3. Although `MyFunSimple` was not saved explicitly in library file, it is saved automatically by Mathematica because `MyFunComplex` requires its definition.
4. Function definition for `AnotherMyFunComplex`

Step 3: Include the header file (.nbd file) into other notebooks

Now that we have a header file (.nbd extension), it can be included in other notebooks using the << operation as shown below in Figure 6.



Including the Header File

Set the directory to the appropriate place where the header file is to be saved

```
SetDirectory[
  "D:\chris_lum_D\mathematica\module_tutorial"];
```

Read in the header file (similar to a #include in C or C++)

```
<< moduleH.nbd
```

Now test that all the functions are defined

```
MyFunSimple[x, y]

Sin[x + y]

MyFunComplex[π, π]

Sin(x+y)=0

AnotherMyFunComplex[2]

Sin[5]
```

Figure 6: Example of including the header file in a notebook

Version History:	11/23/05: Created:	
	12/01/05: Updated:	Made this so it fits the template of all how-to documents
	12/09/05: Updated:	Made changes to layout and added footer.