About Maple Libraries, Paths, etc

Solomon R.C. Douglas and David M. Harrison,
Dept. of Physics, Univ. of Toronto, harrison@physics.utoronto.ca

January, 2004

Information about creating and maintaining libraries, and the ways in which *Maple* finds the libraries and their contents is scattered around many places in the documentation. This little document attempts to put the information all in one place: hopefully it will save others the ridiculously long time it took us to figure all this out.

A Sample File

We create a file in the `currentdir` named `source-code`. Here are its contents.

```
MyModule := module()

    export proc1, proc2;

    option package;

    proc1 := proc()
        print("we're in proc1");
        NULL
    end proc;

    proc2 := proc(a, b)
        a+b
    end proc;

end module:
```

We read the file into the *Maple* session.

```
>    read "source-code";
```

We can then execute the procedures.

```
>    MyModule:-proc1();

                         "we're in proc1"

>    MyModule:-proc2(3,4);

                                7
```

The Standard Library

The standard library for Maple consists of 2 files:

- `maple.ind` : an index of the contents of the library.
- `maple.lib` : the actual library

The directory where these files exist depends on the details of the installation. For a default installation the directory is given by the libname command.

```
>     libname;
```

```
"/usr/local/Maple9/lib"
```

We can list the contents of the directory.

```
FileTools[ListDirectory](libname);
```

```
["include","classic","maple.hdb","units.lib","units.ind","scandsea.lib","scandsea.ind",
"maplets.lib","maplets.ind","maple.lib","maple.ind","Student.lib","Student.ind",
"CodeGeneration.lib","CodeGeneration.ind","CodeGeneration.hdb","standard.hdb"]
```

You can see from the above that there are other libraries and directories, in addition to the standard Maple library.

Depending on the details of your installation, you may be able to add your own modules to the standard library, or create new libraries in the directory specified by libname. Both of these are bad ideas. Adding a broken module to the standard library can make Maple non-functioning. Adding a new library to the system area means that updating your version of Maple can cause you to lose your new library.

Creating a New Library Named maple.lib

As discussed, in general it is a bad idea to install your libraries in the system areas of Maple. We shall create a new library in the empty directory "/home/harrison/tmp/mylibs."

```
>     new_lib_dir := "/home/harrison/tmp/mylibs":
>     FileTools[ListDirectory](new_lib_dir);
```

```
[]
```

We create an empty library with slots for 10 archive members using the Maple archive program `march` .

```
>     march('create', new_lib_dir, 10);
```

This has created 2 files, `maple.ind` and `maple.lib`, in the directory.

```
>     FileTools[ListDirectory](new_lib_dir);
```

```
["maple.ind", "maple.lib"]
```

The library has no contents yet.

```
>   march('list', "/home/harrison/tmp/mylibs/maple.lib");
```

```
                                        []
```

Then we modify the `libname` variable to include the new directory.

```
>   libname := new_lib_dir, libname;
```

$$libname := \text{"/home/harrison/tmp/mylibs", "/usr/local/Maple9/lib"}$$

Maple will search the directories listed in `libname` in the order in which they are listed. Thus, `new_lib_dir` will be searched before the system directory.

Recall that in the first section we loaded the procedures in the `MyModule` module into Maple. We add the module to the library.

```
>   savelib('MyModule');
```

Now the module is in the library. In fact, it has been saved to the first library found in the first directory listed in `libname` .

```
>   march('list', "/home/harrison/tmp/mylibs/maple.lib");
```

```
[["MyModule.m", [2004, 1, 29, 11, 14, 3], 1024, 116],
[":-1.m", [2004, 1, 29, 11, 14, 3], 1140, 105],
[":-2.m", [2004, 1, 29, 11, 14, 3], 1245, 85]]
```

We can load the library into a fresh session and use its procedures, provided we tell Maple where to find it.

```
>   restart;
>   new_lib_dir := "/home/harrison/tmp/mylibs":
>   libname := new_lib_dir, libname;
```

$$libname := \text{"/home/harrison/tmp/mylibs", "/usr/local/Maple9/lib"}$$

```
>   with(MyModule);
```

$$[proc1, proc2]$$

```
>   proc1();
```

$$\text{"we're in proc1"}$$

```
>   proc2(3,4);
```

$$7$$

We can also use the procedures in the new library without using the `with` command.

```
>   restart;
>   new_lib_dir := "/home/harrison/tmp/mylibs":
>   libname := new_lib_dir, libname;
```

$$libname := \text{"/home/harrison/tmp/mylibs", "/usr/local/Maple9/lib"}$$

```
>    MyModule:-proc1();
```

```
                                "we're in proc1"
```

```
>    MyModule:-proc2(3,4);
```

```
                                      7
```

Creating a Library With an Arbitrary Name

In the previous section we created a library with the same name as the system library: `maple.lib` . Here we shall create a library named `ThisLibrary.lib` in the directory `/home/harrison/tmp/otherlibs` . Initially this directory is empty. We begin with a fresh Maple session and load the file containing the definition of the module that we want to place in the library. Our current directory contains only that source file.

```
>    restart;
>    currentdir();
```

```
                          "/home/harrison/tmp/maple"
```

```
>    FileTools[ListDirectory]( currentdir() );
```

```
                                ["source-code"]
```

```
>    read "source-code";
```

Now we set `libname` to include the directory where we wish to create the library and use `march` to create it.

```
>    new_lib_name := "/home/harrison/tmp/otherlibs":
>    libname := new_lib_name, libname;
```

```
          libname := "/home/harrison/tmp/otherlibs", "/usr/local/Maple9/lib"
```

```
>    FileTools[ListDirectory]( new_lib_name );
```

```
                                      []
```

```
>    march('create', cat( new_lib_name, "/ThisLibrary.lib"), 10);
>    FileTools[ListDirectory]( new_lib_name );
```

```
                      ["ThisLibrary.ind", "ThisLibrary.lib"]
```

The `savelib` command by default saves into the first library file that it finds in going through the directories specified in `libname`. Thus, we can save `MyModule` directly.

```
>     savelib('MyModule');
>     march('list', cat( new_lib_name, "/ThisLibrary.lib"));
```

```
[["MyModule.m", [2004, 1, 29, 13, 17, 29], 1024, 116],
[":-1.m", [2004, 1, 29, 13, 17, 29], 1140, 105],
[":-2.m", [2004, 1, 29, 13, 17, 29], 1245, 85]]
```

You may specify a specific library by setting the `savelibname` variable.

About the save Command

The `save` command saves names or symbols, by default into a file in Maple's internal format. For example, if our current directory contains only the file `source-code`, then we can create a file `try.m` containing a name definition.

```
>     restart;
>     currentdir();
```

```
                                   "/home/harrison/tmp/maple"
```

```
>     FileTools[ListDirectory]( currentdir() );
```

```
                                   ["source-code"]
```

```
>     foo := "hi sailor";
```

```
                                   foo := "hi sailor"
```

```
>     save foo, "try.m";
>     FileTools[ListDirectory]( currentdir() );
```

```
                                   ["source-code", "try.m"]
```

The definition can then later be read into a new Maple session.

```
>     restart;
>     foo;
```

```
                                   foo
```

```
>     read "try.m";
>     foo;
```

```
                                   "hi sailor"
```

However, if the `save` command is issued from a directory where there is a library, then the "file name" will be added to the library instead. We will illustrate with the `ThisLibrary.lib` that we created in the previous section.

```
>     restart;
>     currentdir("/home/harrison/tmp/otherlibs");
>     FileTools[ListDirectory]( currentdir() );
```

```
                    ["ThisLibrary.ind", "ThisLibrary.lib"]
```

```
>     march('list', "ThisLibrary.lib" );
```

```
[["MyModule.m", [2004, 1, 29, 13, 17, 29], 1024, 116],
[":-1.m", [2004, 1, 29, 13, 17, 29], 1140, 105],
[":-2.m", [2004, 1, 29, 13, 17, 29], 1245, 85]]
```

```
>     foo := "hi sailor";
```

```
                    foo := "hi sailor"
```

```
>     save( foo, "try.m" );
>     FileTools[ListDirectory]( currentdir() );
```

```
                    ["ThisLibrary.ind", "ThisLibrary.lib"]
```

```
>     march('list', "ThisLibrary.lib" );
```

```
[["MyModule.m", [2004, 1, 29, 13, 17, 29], 1024, 116],
[":-1.m", [2004, 1, 29, 13, 17, 29], 1140, 105],
["try.m", [2004, 1, 29, 13, 38, 32], 1356, 26],
[":-2.m", [ 2004, 1, 29, 13, 17, 29], 1245, 85]]
```

Then, we can read the definition of `foo` contained in `try.m` in a fresh Maple session. This time, the `read` command has dug the definition out of the library.

```
>     restart;
>     currentdir("/home/harrison/tmp/otherlibs");
>     foo;
```

```
                    foo
```

```
>     read "try.m";
>     foo;
```

```
                    "hi sailor"
```

Initialization

When you have libraries in non-system areas, it can be a pain to have to begin every Maple session by manipulating `libname`.

You can change the value of libname once and for all via an initialization file. The details of the name and location of this file depends on the operating system of the computer and on how Maple was installed by the administrator. Details may be found in the on-line Help Browser in *Getting Started* ... => *Configure Maple* ... => *initialization files* .

Regardless of the name and placement of the initialization file, putting the following line in it will set the value of `libname`:

```
libname := "/some/directory", libname:
```

Note the trailing colon in the above command. This keeps the new value of `libname` from being printed to your screen when Maple is invoked.