Technical manual of `MMAlpha`[*]

Patrice Quinton

May 2005

1 Introduction

This document presents the organisation of the LipForge distribution of `MMAlpha`. This new archive is being built by Tanguy Risset and Patrice Quinton.

2 Organization

The distribution contains the following directories:

1. `bin.cygwin`, `bin.darwin`, `bin.linux`, etc. are the directories where the binary files for various systems are put. These binary files are the result of compiling source files.
2. `config` contains various files needed to configure the installation.
3. `demos` contains notebooks for various demonstrations of `MMAlpha`.
4. `doc` contains the documentation.
5. `lib` contains the mma packages.
6. `lib.cygwin`, `lib.darwin`, etc. contain some library files obtained as byproducts of compiling source files.
7. `sources` contains the source files.
8. `tests` contains test files.

[*]Version of January 6, 2007

In addition, the main directory contains a `readme` file, and several files for the `html` documentation.

The main directory also contains a file `CopyMMA.m`. This is a `Mathematica` program that can be used to create a distribution. The documentation of this program is in directory `doc/Distribution`.

3 The `sources` directory

3.1 Content

The `sources` directory contains one directory for each `C` program used by `MMAlpha`. As any other directory, it contains a `readme` file, `html` files.

The directories are:

1. `Code_gen`,
2. `Domlib`,
3. `Makeinclude`,
4. `Mathlink` (obsolete),
5. `Pip`,
6. `Polylib`,
7. `Pretty`
8. `Read_Alpha`,
9. `Write_Alpha`.

It contains also a sed directory, which is obsolete. The `Mathlink` directory is also obsolete.

It contains a make file, named `Makefile`.

3.2 Compiling source files

It is possible to recompile the source files. The method is as follows.

In the `sources` directory, type

```
make all
```

The `Makefile` is then executed. If the `OSTYPE` variable is set, the corresponding binary files are produced and put in the corresponding `bin` directory. In case of problem, check the value of `OSTYPE` and of `MMALPHA`.

3.3 How it works

The `Makefile` in `sources` contains a few variable definitions and a few rules. The `DIR` variable defines the list of directories where files should be compiled. The `all` rule, defines what should be done for all element of `DIR`: namely, go in the corresponding directory, and run make. The `all` rule depends on any file contained in the bin or lib directories.

The `clean` rule defines what should be done to clean directories. Again, it consists in going in the corresponding directory, and running

```
make -f Makefile clean
```

Finally, two rules help create the lib and bin directories, if they are missing.

Once a directory is entered, say `Pretty`, this directory contains a make file named `Makefile`, some source files, some Object directories, a `readme` file and `html` files.

The organization of this `Makefile` depends on the considered source directory.

3.3.1 The `Pretty` directory

For the `Pretty` directory, it is very simple, and is explained in the corresponding `Makefile`. This directory contains some source files which are needed for `Read_Alpha` and `Write_Alpha`.

3.3.2 The `Code_gen` directory

For the `Code_gen` directory, the structure of the make file is much more complicated.

First, the `Makefile` calls the file `Makefile.config` which is situated in the `Makeinclude` directory, where some variables will be set. This make file also calls system dependent make file, say `Makefile.darwin` for the `darwin` system, where some system dependent definitions are set.

Then, back to the local make file, some other variables are defined:

- `CPPFLAGS`: include paths for the compiler,
- `LDFLAGS`: library paths for the loader (where libraries should be looked for),
- `LOADLIBES`: libraries for the loader,

- `DUMMY`: list of files to be removed when cleaning
- `NAME`: the name of the binary file to be created
- `OBJS`: the list of all object files
- `LIBS`: the list of libraries needed. This variable is used for completing make dependencies.

After these definition, one additional rule (which I do not understand).

And finally, call to a generic `Makefile.rules` (which is barely understandable...).

3.3.3 The `Read_Alpha` and `Write_Alpha` directories

These directories are defined in much the same way as the `Code_gen` directory.

3.3.4 The `Domlib` directory

The `Domlib` directory is different, after a deep modification that was recently done, in order to become less dependent on the `Polylib` which is distributed within `MMAlpha`.

- First, the main `Makefile` file contains a switch depending on the system type. If the OS is Cygwin, the Visual C++ environment is called, otherwise, an appropriate make file is called.
- I only define here what a make file for a Unix-like system looks like. It contains the definition of a few variables.
  - `CC`: which compiler to use (could be removed)
  - `POLYINCLUDE`: the directory where the include files for the `Polylib` are. In the current version, this directory is where the `Polylib` installation puts these files. (A companion document explains how to install the `Polylib`, from the Web distribution. In the installation procedure, one must run a command

    ```
    ./configure.in --prefix="Path"
    make
    make install
    ```

    where `Path` is the directory where the `include` and `lib` directories of `Polylib` are put. Say for example, you install `Polylib` with

```
./configure.in --prefix="$MMALPHA/sources/Poly/$OSTYPE"
make
make install
```

then (provided `MMALPHA` and `OSTYPE` are properly set), you will find in `sources/Poly/OSTYPE` the include and lib files.

- `BINDIR`: where the binary files will be put, enventually,
- `OBJDIR`: where the object files will be put, during compilation,
- `EXTRA_FLAGS`: additional flags for the compiler. Uses some variables that are set by `MakeIncludes/Makefile.$OSTYPE`,