The Package Builder

The package builder is used to customize a build of the Q'Nial V7 interpreter. It is written in Nial (*procpkg.ndf*) and so needs a Nial executable.

We have included two basic executables:

```
nial_osx
```

and,

```
nial_linux
```

in the the pkgblder directory to simplify the bootstrap process.

The package builder is driven by control files with extension ".txt" . Lines in a control file beginning with # are treated as comments.

Here is the control file for the simplest OSX or Linux version: simple.txt

```
-----
V7SIMPLE

# Standard Features
CORE coredefs.ndf
SYSTEM -
V6AT -
COMMAND -
TIMEHOOK -
DIRECTIO -
REGEXP posix_regexp.ndf
PROFILE -
GETENV -
CMDLINE -
------
```

There are two additional control files:

v7Qnialpkg.txt	files included in directory src
v7full.txt	generates files to add two experimental features

To use the package processor for an OSX build go into the pkgbuilder directory and enter the following command:

./nial_osx -defs procpkg

This initiates a Nial session and displays the following:

```
-------
Available Packages:
 0) simple.txt
 1) v7Qnialpkg.txt
 2) v7full.txt
Enter a single package number:
>
-----
```

Three package control files are available. If you reply to the prompt with 0, the program responds with:

```
-------
> 0
You have chosen:
simple.txt
-------
```

It does its work and results in the following files being written into the src parent directory:

```
basics.c
basics.h
defs.ndf
defstbl.h
pkgswchs.h
NialFeatures.txt
```

These files are as follows

- *basics.c* and *basics,h* create the switches that invoke primitive routines.
- *defs.ndf* and *defstbl.h* set up the predefined Nial semantic objects that are built into the initial workspace.
- *pkgswchs.h* contains a set of defines that are included in each C source file in the build directory. The defines serve as switches that include the
- C code needed to implement particular features.
- *NialFeatures.txt* documents which features have been included in the src directory and is used by *CMake* to add aditional headers or libraries

To then create an executable having the selected features you have to follow the steps to build Nial as described in the README file in directory QNial7.

The included execultables for OSX and Linux were built using the V7SIMPLE control file.

The Control File

The role of the control file is to provide a systematic way to include or exclude features in the particular Q'Nial version being built.

The first line of the control files gives the Version name of the package, in our case: V7SIMPLE.

The susbsequent non-comment lines consist of

```
<FeatureName> <NDFfilename>
```

where the latter is replaced by a "-" if there are no related definitions.

For each feature, its name is used to set up a define in *pkgswchs.h*, and its ndf file is added to *defs.ndf*. Then all entries in *allprims.nh* that have *Featurename* are processed to build up the entries in the dispatch tables of *basics.c* and external declarations in *basics.h* . The routine to initialize primitive names in the symbol table is also generated for each entry in *basics.c*.

An entry in *allprims.nh* has the following form:

```
Sw Prop Primname Ufname Bfname
```

where the last entry is omitted if there is not a separate binary function call.

The entries are

- *Sw*: the feature name, which becomes a defined name to include code
- *Prop*: the property of the feature being defined (explained below)
- *Primname* the name used in Nial for the functionality
- *Ufname*: the C function name for the function that implements the primitive
- *Bfname*: the C function name for the function that implements a binary use of the primitive.

The Prop field is

```
R - a unary reductive operation
C - a pervasive binary operation
B - a binary operation (argument must be a pair)
P - a pervasive unary operation
U - a unary operation
E - an expression
T - a transformer
```

The Prop field is inserted in a call to te function *init_primname()* that is placed in the routine *init_prims()* in *basics.c*.

To add new primitives to the system you do the following:

- Choose a Feature name:
- Add a line to allprims.nh for each new primitive you define.
- Create a .ndf file for the support definitions associated with the primitives and place it in pkgblder.
- Add a line in the control file to include the feature.