\* Enhancing tcllib with CriTcl

I have been looking into using critcl [http://wiki.tcl.tk.critcl]

to enhance some of the computationally intensive packages in

tcllib with compiled functions. The intention here is to provide

an alternative implementation for the 'hot-spots' in a package. In

the case of the uuencode package, for instance, we can provide

just the function that encodes or decodes a chunk of data. The

remainder of the package can be left in Tcl - which importantly

means that the public interface to the package remains identical.

There is already some precedent for enhancing tcllib in this

way. Trf is used in the base64, md5 and sha1 packages to

optionally provide a compiled implementation of the

package. Importantly we still have a pure-tcl implementation.

Provided that we accept that this hot-spot compilation strategy is

a good idea there is an issue to do with how the compiled code is

bundled. Critcl can build libraries or packages. Packages are the

same as libraries except that the library is placed into a

platform dependent subdirectory and a suitable pkgIndex file is

created.

There are four main possibilities:

1) library-per-package

2) package-per-package

3) package-per-module

4) tcllib-c-package

\*\* library-per-package

a library is built for each critcl-enabled tcllib package. For

instance, 'critcl -lib uuencode.dll base64/uuencode.tcl'. This

doesn't generate any package loading tcl code and so doesn't

require a new package name. Instead the calling code will have to

handle loading the correct library. Once loaded the compiled

commands are available in the tcl namespace. For example,

' load uuencode.tcl ; uuencode::CEncode abc '

\*\* package-per-package

a compiled package is built for each critcl-enabled package. For

instance, 'critcl -pkg uuencode\_c base64/uuencode.tcl'. This

builds the same library and for library-per-package but also

generates the package loading code. This requires a unique package

name (such as ${package}\_c).

\*\* package-per-module

a compiled package is built for each tcllib module. For instance,

'critcl -pkg base64c base64/base64c.tcl base64/uuencode.tcl

base64/yencode.tcl'. This collects all the critcl sections for a

module (which may contain a number of packages) into one library

and then creates the package loading code for this library.

\*\* tcllib-c-package

a compiled package is built for tcllib. This combines all the

critcl sections for all tcllib packages together in one

library. An advantage to this is that there is only one package

name and only one library.

\* Examples

\*\* library-per-package

critcl -lib yencode.dll base64\yencode.tcl

critcl -lib uuencode.dll base64\uuencode.tcl

critcl -lib md4c.dll md4\md4c.tcl

critcl -lib md4c.dll md4\md4c.tcl

critcl -lib sum.dll crc\sum.tcl

This gives us the named dll's in the current directory. For this case

I get (under Windows):

md4c.dll - 15,360

md5c.dll - 15,360

sum.dll - 12,800

uuencode.dll - 13,312

yencode.dll - 12,800

\*\* package-per-package

This creates libraries of the same size as library-per-package.

\*\* package-per-module

critcl -libdir . -pkg base64c base64c\base64c.tcl base64\uuencode.tcl

base64\yencode.tcl

critcl -libdir . -pkg md4c md4\md4c.tcl

critcl -libdir . -pkg md5c md5\md5c.tcl

critcl -libdir . -pkg crc crc\crcc.tcl crc\sum.tcl

This gives:

base64c.dll - 14,848

crcc.dll - 12,800

md4c.dll - 15,360

md5c.dll - 15,360

\*\* tcllib-c-package

critcl -libdir . -pkg tcllibc <all the above>

tcllibc.dll - 19,968

\* Conclusion

We can see that the overhead of producing a dynamic library is

quite significant - at least for Windows. Building a library for

all 5 packages is only around 4000 bytes larger than the library

for a single package. There is also a cost involved in loading

each dynamic library as they have to be loaded from disk and

relocated in memory. Other concerns are the package namespace

pollution - by which I mean the proliferation of package names

caused by creating compiled packages for each tcllib package. Also

ease of loading - I don't really think abandoning the tcl package

mechanism is a good idea.

Pat Thoyts