Tcllib package indexing

=======================

This document describes the possibilities for using one or more

pkgIndex.tcl files in an installation of tcllib to provide the

information about all of its packages to a tcl interpreter, discusses

their pro and contra and makes a choice for Tcllib 1.4. A roadmap of

changes in the future is made available as an appendix.

Background under which to see the solutions:

There are three level of groupings:

- The tcllib project itself

- Modules in the project (== subdirectory of 'modules')

- Packages in a module.

Each module currently contains one package index file.

Some modules contain more than one package. They share the index.

Most packages require specific versions of the Tcl

interpreter. They perform the checks in their package index

file and do not register if the pre-requisites are not

fulfilled.

Other checks are possible, but currently not in use.

Note I:

Whether a solution is actually applicable depends on external

factors, like the chosen directory layout of an installed

tcllib.

Note II:

All solutions currently depend on the specific implementation

of [tclPkgUnknown] coming with the basic core, simply by the

fact that the files looked at are called 'pkgIndex.tcl'. This

is therefore no contra argument against any specific solution,

but against all. We ignore this as currently there is no

better replacement in existence.

Note III:

We have to support Tcl before 8.3. as some packages in tcllib

allow this.

[i1/ng] No global package index

-------------------------------

In this solution the module package indices are the only index

files present in an installation.

This solution is applicable if and only if one of the flat

directory layouts (L2/Fa or L2/Fb) has been chosen.

Pro:

Simple. No need for complex management.

[i2/ad] Global package index, auto\_path extension, direct

---------------------------------------------------------

A single global package index is present in the toplevel

directory of the installation.

This solution is applicable if and only if the deep directory

layout (L2/D) has been chosen.

The package index contains a series of statements extending

the auto\_path variable with all module directories. The list

of names of the module directories is hardcoded. In other

words, it is \_not\_ determined via [glob].

Example:

lappend auto\_path [file join $dir md4]

lappend auto\_path [file join $dir md5]

lappend auto\_path [file join $dir sha1]

...

Pro:

[[0]] Compared to [i3/ag] this should be bit faster

as glob'ing the directory tree of tcllib is

avoided. This performance-boost is not a big

pro according to the opinions below.

[[1]] Relies on the module package index files for

the actual registration of packages, thus

automatically inherits the correct constraints

on the registration of packages. No additional

complexities.

[[2]] Easier to generate than [i6/dr].

Contra:

[[3]] Hard coding the directory names implies that

adding modules to the installed tcllib is not

as easy as just creating a new directory for

the module/package. The global index has to be

updated too.

Contra-Contra:

<<Don: New, updated packages should be

installed separately, outside of

tcllib. The ticked version number

ensures that it is prefered over the

package in tcllib.>>

<<AK: Agree fully>>

[[4]] Extending the 'auto\_path' list causes the

package management of the tcl core to re-read

the list and glob through all of them for new

package indices. This has a high cost in terms

of filesystem access, i.e. is an issue of

performance.

Contra-Contra:

<<Don: IMHO, it's not really tcllib's

job to fix [tclPkgUnknown]'s

performance problems. If performance

is a problem, users could try the

patch with Tcl Feature Request 680169

and see if it helps.>>

<<AK: Not sure yet about this>>

[[5]] This enables auto-loading in each module

(according to any tclIndex file installed).

This should not be done by the package

indexer, but by the package itself. See

control for an example.

[[10]] Will not work with Tcl releases prior to

8.3.1. Only then was [tclPkgUnknown]

"enhanced" to deal with changing ::auto\_path

values. If tcllib 1.4 wishes to continue

supporting pre-8.3.1 Tcl, then this option has

to be supplemented with a fallback.

[i3/ag] Global package index, auto\_path extension, glob

-------------------------------------------------------

This is like [i2/ad], except that the list of sub directories

is not hardcoded into the index, but determined through glob.

Example:

foreach subdir [glob -nocomplain -type d $dir/\*] {

lappend auto\_path $subdir

}

Pro:

Anti-[[3]]

[[1]]

Contra:

All the contras of [i2/ad] and Anti-[[0]].

[i4/sd] Global package index, sourcing module indices, direct

-------------------------------------------------------------

A single global package index is present in the toplevel

directory of the installation.

This solution is applicable if and only if the deep directory

layout (L2/D) has been chosen.

The package index contains a series of statements source'ing

the package index files of the modules in tcllib. The list

of names of the module directories is hardcoded. In other

words, it is \_not\_ determined via [glob].

Example:

set main $dir

set dir [file join $main md4] ; source [file join $dir pkgIndex.tcl]

set dir [file join $main md5] ; source [file join $dir pkgIndex.tcl]

set dir [file join $main sha1] ; source [file join $dir pkgIndex.tcl]

...

Pro:

[[0]], but compared to [i5/sg].

[[1]]

[[2]]

[[6]] In contrast to [i2/ad] and [i3/ag] repeated

glob'ing for package index files is

avoided. This cuts down on costly FS accesses.

I.e. another perf. boost.

Contra:

[[3]]

[i5/sg] Global package index, sourcing module indices, glob

-----------------------------------------------------------

This is like [i4/sd], except that the list of package indices

to source is not hardcoded into the index, but determined

through glob.

Example:

foreach subdir [glob -nocomplain -type d $dir/\*] {

set dir $subdir

source [file join $dir pkgIndex.tcl]

}

Pro:

Anti-[[3]]

[[1]]

[[2]]

Contra:

All the contras of [i2/sd], and Anti-[[0]]

[i6/dr] Global package index, direct registration

-------------------------------------------------

A single global package index is present in the toplevel

directory of the installation.

This solution is applicable if and only if the deep directory

layout (L2/D) has been chosen.

The package index contains a series of statements which

directly register all the tcllib packages.

Example:

if {[constraint]} {return}

package ifneeded md4 [list source [file join $dir md4 md4.tcl]]

package ifneeded md5 [list source [file join $dir md4 md4.tcl]]

package ifneeded sha1 [list source [file join $dir md4 md4.tcl]]

... more constraints ... package ifneeded

Pro:

[[7]] This is the fasted solution as the number of

accesses to the filesystem is minimal.

Contra:

[[[3]]

Anti-[[1]] Care has to be taken to ensure that

the constraints the module indices

place on the registration of packages

are replicated in the global

index. All other solutions simply used

the module indices and thus got it

right automatically. Now supporting

code is required to detect such

constraints and then to properly

recreate them globally.

= High complexity for the maintainer.

[i7/ad] Global package index, auto\_path extension, direct

---------------------------------------------------------

A single global package index is present in the toplevel

directory of the installation.

This solution is applicable if and only if the deep directory

layout (L2/D) has been chosen.

The package index contains a single statement extending the

auto\_path variable with the tcllib main directory. The

standard package management will then find all module sub

directories and the package indices in them.

Example:

lappend auto\_path $dir

Pro:

[[1]]

[[8]] This is the easiest solution by far in terms

of code to write, and complexities to solve

(none).

[[9]] <<Don: I believe this is the only proposal listed

that actually fixes tcllib Bug 720318

(successful [package require] of packages

within a SafeBase) because it is the only one

that changes the value of ::auto\_path.>>

<<AK: This is true, yet brittle. It depends on

when the SafeBase sees the auto\_path. If it

happens to be before a [package require

something] forced the reading of all package

indices (and thus the extension of

'auto\_path') we are still SOL.>>

Contra: [[4]]

[[10]]

[i8/pm] Global package index, pkg\_mkIndex

-----------------------------------------

Just use [pkg\_mkIndex modules \*/\*.tcl] to generate the master index.

Pro:

Easy to do.

Contra:

Does not handle constraints in subordinate package

indices, simply because they are actually ignored

during processing.

Adding code to handle constraints evolves this into

[i6/dr].

Note: The contra is hard enough IMHO to make this solution not

applicable for 1.4, which does have constraints, and handling

them wrong (not at all) is a bug.

General discussion

------------------

Given that a deep directory layout was chosen [i1/ng] is not

applicable and therefore dropped from the discussion.

In the pro and contra arguments listed above three independent axes of

reasoning emerged:

a) Performance of the solution, with the number of accesses to

filesystem the main factor determining it.

b) Complexity/difficulty of the solution with regard to

adding/updating packages.

c) Complexity of generating the master index.

Axis (b) has essentially been thrown out. Trying to modify the

installation of tcllib itself is bad practice. Install new/updated

packages separately. The version numbering takes care of the rest,

i.e. usage of the new over the older version found in tcllib.

With respect to axis (c), complexity of generation, [i7/ad] is the

definite winner, with the other \*d solutions close behind (all use

fixed scripts, I7/ad wins on size). This is followed by the \*g

solutions as they require actual dynamic generation of code. And at

the bottom of the ladder is [i6/dr] with its need for close inspection

of the sub-ordinate indices to get everything right.

Now axis (a), performance, [i6/dr] is most likely the winner as it

causes only one index to be read and nothing else. This is followed by

the all \*d solutions, which read the subordinate indices, but do not

need much glob'ing. The actual order in this group is difficult to

determine. I guess that the auto\_path extending methods are slower

than the sourcing methods, and the adding of one directory faster than

the adding of all, as the latter looks for much more subdirectories.

The next group are the \*g solutions as they perform their own glob'ing too

beyond that done by the package mgmt.

Two final rankings

(c), then (a) (a), then (c)

------------- -------------

[i7/ad] [i6/dr]

[i4/sd] [i4/sd]

[i2/ad] [i7/ad]

[i5/sg] [i2/ad]

[i3/ag] [i5/sg]

[i6/dr] [i3/ag]

------------- -------------

[i4/sd] seems to be a good compromise solution between performance and

complexity of generation, but [i7/ad] is not too bad either.

[i4/sd] reminder:

set main $dir

set dir [file join $main md4] ; source [file join $dir pkgIndex.tcl]

set dir [file join $main md5] ; source [file join $dir pkgIndex.tcl]

set dir [file join $main sha1] ; source [file join $dir pkgIndex.tcl]

...

[i7/ad] reminder:

lappend auto\_path $dir

Other opinions:

Don Porter prefers [i7/ad], and [i6/dr] as second choice. Also

as [i7/ad] fallback for older Tcl before 8.3.1

Joe English strictly opposes any solution modifying the

auto\_path, violating his opinion that index scripts should

have no side-effects beyond registering a package.

Chosen solution for Tcllib 1.4

------------------------------

After comparing the code for the combination of [i7/ad] and [i6/dr] as

submitted by Don Porter, and for [i4/sd] as submitted by myself

(Andreas), and a small discussion on the Tcl'ers chat between Don and

me, we took [i4/sd] for the main body of the index, and the header of

Don's code. Basically the chosen package index is a combination of

[i7/id] and of [i4/sd] as fallback.

This is still as easy to generate as [4/sd], the index is also only a

bit more complex, and speed should be okay too.

Don convinced me that while extending auto\_path is definitely bad in

the long-term it is still okay for the short-term and release 1.4.

Roadmap

-------

After Tcllib has been driven into the state of one package per module

directory, and switched to a flat directory layout for its

installation we switch to [i1/ng] for the indexing structure.

-----------------------------------

This document is in the public domain.

Andreas Kupries <andreas\_kupries@users.sf.net>