Packages - Feature Request #3534

Enable checking sequence in PKGBUILD

2023-09-28 09:00 PM - nona

Status:	wont-fix	**% Done:**	0%
Priority:	feature		
Assignee:			
Category:			

Description

I recently had to create a custom PKGBUILD for a program with C++ & Python bindings. The Python bindings depend on the C++ compilation, and the only real ways to run the check() in the PKGBUILD were

1. Install the built C++ and Python objects in a temporary folder within build(), run the check() with that directory and reinstall in package() as if the first intallation did not exist. (Needless to say that the software is not prepared to make it easier).
2. Create a secondary package and rebuild the whole thing just to avoid creating the temporary installation
3. Create a split package wherein one of the package_*() is just a function to run the tests.

I think this is the situation for more than one program out there (having to install part of the software to run the tests--or not have tests at all). Thus, my suggestion is: allow for ordering of the check() and package() functions. If the package had been ready in ${pkgdir}, the whole thing could be avoided (check() would run after package()).

History

#1 - 2023-09-29 03:54 PM - bill-auger

you could do this in your copy fairly easily - makepkg and friends are all shell scripts

i suspect that your problem is due to something rather unconventional - in general, a program should not need to be installed in order to test it - it should be ready to test immediately after the build succeeds - that is why check() runs after the build() function

the very reason for a test suite is so that you can avoid installing a program which is not yet know to work properly - installing the program should always be optional, only if you actually want to use the program - if the tests can not run in the build tree, or at the least without su privileges, i consider that to be a bad design

generally, it doesnt make sense to modify the build tool to accommodate the quirks of one program or one language - i think that work-arounds for those quirks of foreign systems should be in the build recipe, or even prepared in advance as in your suggested option (2), but not codified in the tools - your suggested option (1) seems like the way to handle this - that may be not a work-around at all, but the proper procedure semantically

without inspecting the build, i can only speculate generally - the semantics of a failing check() function is: "the build is somehow broken or incomplete - the build() function did something wrong, or neglected to do something"

generally, the package() functions should copy files and nothing more - it should not need to do any work - so your suggested option (3) would not be sensible, unless there is some inappropriate work happening in the initial package() function - more likely, the significant thing which has not happened before check() runs now, is not actually "installation" (copying files), but the formatting or packaging of an "egg" or analogous what-not - that is: maybe the build is not actually complete (semantically) until the confusingly named `python setup install` or whatever generates the egg package structure, but the tests may not care where python lays that egg ("installs" the files), because they will be within it - in that case (if `python setup install` does more than copy files), the failing check() now is failing for the right reason: "the build is incomplete" - semantically, that entire pythonical procedure belongs in build(), then check() would work as expected, and package() could simply copy the fully-working program files, launch script, or what ever else into the distro package

#2 - 2023-09-29 08:25 PM - nona

I know, this is lame, but I really forgot that: it is possible to build one's own functions within build(), check() & package(). Thank you for shedding light--as always.

#3 - 2023-10-01 10:43 AM - bill-auger

- Status changed from open to wont-fix