libretools - Feature Request #2403

[libremakepkg]: eradicate mksource() from abslibre

2019-07-26 10:10 PM - bill-auger

Status:	info needed	**% Done:**	100%
Priority:	discussion		
Assignee:			
Category:			

Description

related to forum thread https://labs.parabola.nu/boards/18/topics/125

IIRC when discussing the 'ruby' package, we concluded that it was probably intended per the FSDG, to provide complete and pre-cleaned source-balls to the distro users; which entails the devs to keep some of the libre-fications tools private

the FSDG does not require support for versions of 'foo' that were not included in any distro release however; but OTOH, this PKGBUILD contains instructions for downloading known non-free software

on the third hand (the philosophical hand), the 'gist' of the four freedoms is not about possessing or sharing non-free software; but that the user is able to do with it as they see fit - if the only thing that one wishes to do with that software is to merely posses it or share it, then anything redistributable would satisfy the 'gist' of the four freedoms for that person

probably the simplest way to resolve this class of issue, is to add a note to the PKGBULID with sed above every instance of 'mksource()', asking the reader to ignore it and ask a parabola dev to publish the intermediate source-ball, if they are building ahead of the current release; then to publish the missing source-ball for any enthusiastic user who asks for one - a better solution to this class of issue, would be for the PKGBUILD to explaining how to run mksource() and have the mksource() function create the source-ball with expected libre name in the sources() array - if that does not conflict with the FSDG - an even better solution to this class of issue, is to eliminate it by eradicating mksource() from abslibre somehow, even if that is simply moving the PKGBUILD to a private git branch

this incident is not a peculiar though - it is currently systematic AFAIK; so i consulted the FSDG:

```
"A free system distribution must not steer users towards obtaining any nonfree information
for practical use, or encourage them to do so. The system should have no repositories
for nonfree software and no specific recipes for installation of particular nonfree programs."
```

there is another clause that might suggest that this use case is within the guidelines; although it is in the "Documentation" section

```
"In general, something that helps people who already use nonfree software
to use the free software better with it is acceptable,
but something that encourages users of the free software to install nonfree software is not."
```

libre-fications would seem to fit into that documentation exception - id say the key point is that such scripts do not install the program "for practical use" - the use-case in which the user is interested, is to download a non-free tarball, and to liberate it, without viewing or executing any of the files within it; for the sole purpose of using the cleaned sources in freedom - it would seems better to instruct people how to liberate non-free software rather than impeding their curiosity or ambition; though it may snag upon some FSDG caveat

some options:

- should consult the FSDG mailing list about this grey area
- put these on the wiki as documentation, instead of in PKGBUILDs
- libretools could be made to download the file into memory instead of writing a hard copy
- hide all such scripts at the cost of impeding enthusiastic users who want to help

Subtasks:

Bug # 2141: libremakepkg failing to build package due to read-only startdir in recent l...	**fixed**

Related issues:

Related to Packages - Freedom Issue #1035: [your-system-sanity]: Non-Free Sof...	**in progress**
Related to libretools - Housekeeping #1973: [libremakepkg] Should extract the...	**open**
Related to libretools - Feature Request #760: [librefetch]/[libremakepkg] Run...	**open**

History

#1 - 2019-07-26 10:11 PM - bill-auger

- Description updated

#2 - 2019-07-26 10:14 PM - bill-auger

- Description updated

#3 - 2019-07-26 10:15 PM - bill-auger

- Description updated

#4 - 2019-07-26 10:36 PM - bill-auger

- Description updated

#5 - 2019-07-26 10:55 PM - freemor

I agree this should be automatic instead of the semi clunky mksource()
approach

The way I see it the issue is with Parabola trying to provide a source tarball
as that forces us to make a libre one as we can not provide the non-libre one.

The need to provide a libre tarball is unclear, if not false on its face as
building "version next" of something will always require pulling the non-free
source.

It might be better to forget about source tarballs. PKGBUILDS pull the source
they need. The PKGBUILDS in abslibre will always clean that source making it
libre,

As bill-auger pointed out above PKGBUILDS in abslibre in no way induce people
to install non-free. The do the opposite and provide the path to freedom.

Providing the "How" (as in how to clean that source) is always more instructive
then just giving someone pre-cleaned source.

Must people that will be playing at the show me the source/pkgbuild level will
probably like myself see the libre package as an unnecessary hustle that I
never really have to deal with as all the work in liberating the source would
just move from prepare to mksource() but every version I'd end up pulling the
non-free, figuring out what needs fixing, and adjusting mksource. rolling a
tarball that no one will ever use, then building from the libre tarball instead
of just doing in all in prepare() and forgetting about creating a next to
unused libre source

#6 - 2019-07-26 11:12 PM - bill-auger

- Description updated

#7 - 2019-07-28 05:30 PM - theova

As an curious user interested in building/updating packages I support Freemor's point of view.

Altough I see the question of letting users download non-free software, I think it is important that the download is not "for practical use" since it is dirrectly removed afterwards.

I'm strongly opposed to making a private git branch. This would hide the work/information to liberate software and makes it only available for a selected circle.

```
"All information for practical use in a free distribution must be available in source form.
(“Source” means the form of the information that is preferred for making changes to it.)" [FSDG]
```

For me, updating packages is a very "practical use" and the source to do that are the PKGBUILDs. So the entire PKGBUILDs should be free (and available).

Other question: If the mksource() step is automated, what is then the benefit over doing the steps in prepare()?

#8 - 2019-07-29 07:17 PM - bill-auger

```
For me, updating packages is a very "practical use"
```

updating packages is not something that the distro user should need to do - the very reason why distros exist, is to provide curated software to the user that is ready to use OOTB - i see "practical use" in terms of the FSDG as very literally referring to "using" the program for its intended end-user use-case - liberating it, so that the user can use it freely "in practice", is something more "meta"; and the distro user should not need to do that - im quite certain that we should be hosting only pre-cleaned sources with all four freedoms already present in every byte - i think this is is an over-sight in the tooling; but that is a different issue than what the PKGBUILD downloads from external sources - i think we could make that work transparently with the end-result being a FSDG-compliant source-ball

```
If the mksource() step is automated, what is then the benefit over doing the steps in prepare()?
```

probably, if the mksource() step was automated, it would all be done in prepare() - the main difference is that the prepare() function often has to treat the sources differently depending on the arch

freemor verified today that the source-ball does get created before prepare() fires - that would explain the extra, non--standard, manual mksource() step - that is apparently the only way to publish pre-cleaned source-balls currently - in that case it is curious why there are so few PKGBUILDs that have mksource() - it seems to me that everything in [libre] would deserve that same treatment

ive been thinking we should make some changes to libremakpkg anyways - aside from the sources being pre-cleaned, ive noticed that any source-balls made from VCS sources end up with the entire VCS history inside; which makes them much much larger than necessary - theyre not versioned release tarballs at all, they are git-balls; which would still include the non-free bits even if they were made after the prepare() step

#9 - 2019-07-30 11:00 AM - freemor

Hmmm.. perhaps we are looking at this the wrong way.
The purpose of makepkg/libremakepkg is to build binaries from source. not to provide curated source.
We have all the patches. We know where the source lives.

what if we rolled a separate tool (libresource or some other nice name) That fetched the PKGBUILD and patches from abslibre and then just ran the equivalent of (or maybe actually) makepkg -o. resulting in a directory with the cleaned source.

advantages:

Makes management of libre source a separate step from makepkg/libremakepkg
It creates a unified easy too for anyone wanting to see the clean source: libresource libre/iceweasel
It is not complicated to build with standard tools (git, makepkg)
It (might maybe hopefully) could eliminate Parabola needing to store massive source tarballs
It feels like a better fit for the Arch way of doing things.
The only retooling of libremakepkg would be to remove the rolling/uploading of source.

Thoughts?

#10 - 2019-07-31 05:15 PM - bill-auger

are you suggesting not to host any sources, or to continue
hosting the pristine upstream sources warts and all? - that is
the most important concern, regardless of what the tools do

probably the original intention was that users should never need
to download known non-free software, and to avoid publishing any
tools that automatically do so - on the one hand, that would be
even worse than what TPPMs do, in terms of suggesting and leading
users toward non-free software - on the other hand, i think we
agree that this particular use-case is not "for practical use" ;
so it may not be a strict FSDG requirement - if the parabola
source packages are not clean though, as they will be. if created
before prepare() runs, then PKGBUILD with the mksource() function
are the only ones that produce pre-cleaned sources - everything
in [libre] should have had that same treatment; and perhaps that
was the goal - if nothing else in [libre] gets that treatment,
then those few are just a drop in the bucket; and moving the
mksource() procedure into prepare() for those few package which
have it, would be barely significant to the big picture - that
would be the easiest way to resolve the present issue; but its
not a very satisfying solution

if the suggestion was not to host any sources, the biggest
concern i have with that is the obvious one - im not sure if the
FSDG requires that; but it does require that we ensure all source
code to available for the current release; and the only way to
ensure that is to self-host it - that would make source code
storage to be essential infrastructure; and it is always unwise
to rely on third-parties to maintain your essential
infrastructure , especially with no contract - regardless of the
FSDG, the GPL alone extends that requirement to three years
beyond the current release, for GPL programs - the only
third-party code host that i would be comfortable using for that

purpose would be the software heritage repos - i think they are
a legitimate charity, so they do have some obligation to the
public - that can not be said of most individual projects or any
commercial freebie hosts; but it would be a lot of work to
change every PKGBUILD to fetch from that host - also, i dont
think they prioritize tracking the bleeding edge of changes; so
that cuod be a no-starter for parabola

aside from ensuring that sources are somehow available, there is
the FSDG "self-hosting" requirement; according to which, one
should be able to compile vN+1 of the distro using only vN of
the distro and tools that are provided by the distro - i dont
think that ubiquitous networking should be taken for granted as
an excuse to avoid supplying everything required to build the
distro - software curation is the one defining factor of what a
distro is and what distro devs do - ideally, i would like to
take that a step further and publish a sources ISO for
offline/clean-room use - currently, a user would need to do an
unreasonable amount of prep work, if they wanted to accomplish
that; but we already have most of the tooling to make that a
possibility

im pretty sure that is why parabola mirrors everything from
arch, rather than having users rely solely on the arch repos -
some users are very concerned about to which servers they expose
their IP - i would like to take that concern a step further and
modify the redirector to always use a parabola mirror - if
winston had a gigabit pipe with unlimited traffic, we could
remove the redirector entirely - the redirector is a necessary
compromise due to circumstances, in a similar way as relying on
the upstreams to feed the user facing PKGBUILDs; but we do have
the resources to avoid the latter because of the lesser demand
for source-ball

i think that complete self-reliance was a goal of the project
that is still incomplete; so if we need to modify libretools, i
would rather continue in that direction rather than removing
relevant functionality - i like the idea of downloading from the
upstream and running prepare() in memory, then creating the
source-ball after prepare() has done its work; resulting in a
cleaned tarball on the local filesystem - librestage and
librerelease already are expecting that libremakepkg has created
one - the only real problem with the current workflow is that the
source-balls are not clean and those sourced from VCS are overweight

an alternative idea, which i like better, would be to put those
pre-cleaned source-balls into use by modifying makepkg to
recognize, according to $pkgname-$pkgver-$pkgrel, that a
pre-cleaned source-ball exists on the parabola server and to
pull that source instead of the upstream's, bypassing the
prepare() function - if no pre-cleaned source-ball exists for
that version, then that is unsupported software, just like AUR,
and falling back on the upstream sources is the reasonable thing
to do - as with the AUR, we could suggest against doing so "for
practical use"; but we could endorse it for the use-case of
helping parabola devs liberate it "for practical use" by others,
or for taming the next version N+1

#11 - 2019-07-31 06:14 PM - eschwartz

The concept of "libre" tarballs is essentially the same concept as Debian's "orig" tarballs, where an upstream source release artifact is taken, but the distro decides it isn't "good enough" and determines to apply certain changes. In debian's case, that sometimes includes deleting lots of files that maybe aren't used by the package, for example if software contains a vendored copy of a library, with a configure switch to either use the vendored copy or a system copy... Debian might delete that. Debian might delete minified js from the source tarball. Debian might rename the directory it is stored in, Debian might just re-compress the tarball so its checksums don't match but it takes less space on their mirror network.
https://www.debian.org/doc/manuals/developers-reference/ch06.html#repackagedorigtargz

The question is why this is necessary. Debian certainly seems to think it is so, but then, Debian hosts the sources for every package, no matter what. They also seem to be going to some lengths merely for the sake of prettiness.

Does the FSDG truly require you to not make non-compliant source code available at all? If all you are doing is providing a persistent mirror of the upstream tarball, but the canonical "help" that Parabola offers, guiding people towards the act of building and installing the software, is the PKGBUILD, then the PKGBUILD -- regardless of the state of the sources themselves -- does not encourage using non-FSDG software. The PKGBUILD will, when left to its own devices, always build a FSDG-compliant binary product, and extracting the build information but removing the FSDG bits is not a supported or recommended route.

From my outside perspective, it would seem like Parabola should have the option to just upload mirrors of the tarball, but have the PKGBUILD perform all FSDG compliance modifications in prepare() -- with suitable comments saying "# these changes, in accordance with the FSDG, protect the user by removing nonfree software".

The alternative is to make it difficult to use the PKGBUILD at all, causing users who want to use the PKGBUILD as a base for updating the version to give up and download something from Github or another upstream website and run ./configure; make; sudo make install without the benefit of a PKGBUILD or FSDG modifications.

#12 - 2019-08-01 02:25 AM - bill-auger

- Related to Freedom Issue #1035: [your-system-sanity]: Non-Free Software From Third-party Package Managers (TPPM) added

#13 - 2019-08-02 01:50 AM - bill-auger

- Related to Housekeeping #1973: [libremakepkg] Should extract the sourceball into the chroot, instead of a tmpdir added

#14 - 2019-08-02 01:51 AM - bill-auger

i just found a open issue (#1973) that suggests unpacking sources into memory that could tie into this issue

#15 - 2019-09-08 08:56 PM - bill-auger

i just learned some important information about mksource() from discussing this issue with emulatorman - mksource() is intended not to be a manual step, as we have been assuming it was - there is another tool named librefetch that uses it to create the pre-cleaned tarball - the librefetch documentation suggests that it can (or will be) run automatically whenever the intermediate tarball is not found on the server - it seems to be pretty much what we have been discussing to implement - we just need to learn how to use it properly - hyperbola started publishing their pre-cleaned source-balls recently and that is they do it

https://git.parabola.nu/packages/libretools.git/tree/src/librefetch/librefetch.8.ronn

#16 - 2019-09-09 11:44 AM - freemor

Interesting, and good researching of the subject. But To my minimalist mind
that still leaves us with mksource() in the PKGBUILDs and yet another tool all
to accomplish what: makepkg -o does without these things. I still feel that
having libretools roll the source tarball post prepare() would be the cleanest
way to go.

Going the other way would require us to strip steps out of prepare to move them
into mksource() for every package we build. This creates a glaring divide
between Parabola packages and Arch packages and also add a lot of un-necessary
work.

Since we probably don't want to re-write makepkg, I'd suggest taking the
rolling of the source tarball out of libremakepkg and moving it into librestage
which could just run makepkg -o, roll up the tarball and move it to staging. Of
course the problem with this is we then have a clean source that the PKGBUILD
may fail to build from depending on the steps in prepare(). Which is what
mksource is trying to fix I guess.

My concern is the further we get from standard Arch tools the larger the
learning curve for new people coming to Parabola. People coming from Arch are
going to be confused if the usual pull the PKGBUILD, run makepkg approach
fails.

I understand that if Parabola hosts the source tarballs they should be clean
ones, But the ugly truth is that the actual process is: pull unclean source,
make it libre, build it. In such a case I see hosting clean source files as
slightly disingenuous. The PKGBUILD is always going to have the location of the
original source so it's not like we are able to hide that, nor should we.
Trying to debug from pre-cleaned sources could be problematic too as if one of
our patches breaks something is a weird way it becomes very difficult to see
what was supposed to happen short of going and downloading the un-patched
source.

So it really ends up feeling that all hosting clean sources does is:

```
a.) complicate things.
   b.) kick the unclean source a step or two further down the road.
```

#17 - 2019-09-09 04:05 PM - bill-auger

freemor wrote:

> Going the other way would require us to strip steps out of

> prepare to move them into mksource() for every package we build.

well not every package; but every one that replaces something in arch - the parity with the arch PKGBUILDs could still be maintained that way - only the "librefication" commands would need to be in mksource() - doing so could actually make the diffs cleaner than they are now

freemor wrote:

> Since we probably don't want to re-write makepkg, the problem with this is we then have a clean source that the PKGBUILD may fail to build

yea thats my thinking - any changes from the current workflow would entail some modification to makepkg - at least we understand now why mksource() exists - it just was not fully utilized for every package; probably because it is such a huge take

freemor wrote:

> People coming from Arch are going to be confused if the usual pull the PKGBUILD, run makepkg approach fails.

makepkg would not fail with the current workflow because the cleaned tarballs are published at the location where the PKGBUILD expects them to be - as i understand, the mksource() function only runs if that tarball is not found on the server

freemor wrote:

> Trying to debug from pre-cleaned
> sources could be problematic too

thats certainly true; but simply changing the URL of the expected cleaned source-ball in the sources array would force mksource() to run again - AFAIK the newly generated source-ball only gets published upon librerelease

freemor wrote:

> a.) complicate things.
> b.) kick the unclean source further down the road.

i dont think mksource+librefetch: would be "complicated" - it would just be a shit-ton of work to sort the librefication commands out of prepare() and into mksource() for every libre package; but presumably the tools already have that functionality to handle it

it would be more complex to modify makepkg to create the tarball after prepare() has run, to detect the presence of the pre-cleaned source-ball on the parabola server and download that instead of from the upstream , and to skip over prepare() if pre-cleaned source-balls are detected - if we were writing makepkg from scratch, i think that would be most reasonable from the engineering perspective - it is a significant deviation from arch; but thats going to be the case no matter how this is done

although it would make the tools more comple, itx would surely be less work; but rescuing mksource() (and fully applying it) has the added advantage of making the diffs against arch much cleaner for evermore into the future

if librefetch does its magic transparently, without manual steps, it probably is the best approach as a long-term investment

- the deviation from arch is constrained to libretools

- the tooling already exists and is working well for hyperbola
- cleaner diffs

freemor wrote:

> I'd suggest taking the
> rolling of the source tarball out of libremakepkg and moving it into librestage
> which could just run makepkg -o

we should probably avoid that, because the build process could modify the sources; and it would rely on a comprehensive `make clean` target to actually produce the same source-ball, somewhat deterministically

#18 - 2019-09-11 10:21 PM - bill-auger

rescuing mksource() is not without is cons also - ive been mulling them over:

- something like 800 PKGBUILDs would need the mksource() treatment - ouch!
- some source-balls end up duplicated in the repo under other/ and sources/
- its kinda goofy

so i am coming around to the merits of plan B (re-tooling) - the most attractive property of that approach is that it could be accomplished in a reasonably finite amount of time - it would also "just work" for any future packages without remembering to write the non-standard mksource() function; plus it would allow the PKGBUILDs to behave the same, backward compatible with other arch-like distros - i think it would entail the following:

- modify libremakepkg to:
    - create the source-ball not before the prepare() function; but before build()
- modify makepkg to:
    - always attempt to download the pre-cleaned source-ball from the sources repo
    - if the pre-cleaned source-ball was found
        - ignore the sources() array and skip prepare()
        - ignore the checksums array and verify the source-balls with GPG
    - if the pre-cleaned source-ball was not found
        - fallback to normal behavior

i have had another feature request in mind recently regarding these source packages, that does not have a redmine issue - namely that any VCS sources could have the VCS history pruned out, reducing the size of the source-ball considerably - the 'librefetch' script apparently does that; but it would be good to do for for every source-ball

#19 - 2019-09-11 10:57 PM - bill-auger

poking around old issues, i just stumbled across the undocumented libremakepkg -S flag - the plot thickens

https://labs.parabola.nu/issues/1972

#20 - 2019-09-11 11:24 PM - bill-auger

- Related to Feature Request #760: [librefetch]/[libremakepkg] Run mksource() in the chroot added

#21 - 2019-10-27 03:04 AM - bill-auger

i added and stub section about mksource() to the wiki - we should elaborate it once we get this sorted out

https://wiki.parabola.nu/Package_maintainer_guide#Uploading_packages

#22 - 2020-05-13 11:10 AM - bill-auger

i came across an interesting wiki article that i was not aware of - it describes the original implementation of mksource() as being primarily for packages that will not build without an active network connection

https://wiki.parabola.nu/Packages_that_need_network_in_build

#23 - 2020-05-23 01:12 PM - bill-auger

just to log a related discussion on the mailing list https://lists.parabola.nu/pipermail/dev/2019-November/007547.html

#24 - 2020-08-19 08:15 PM - bill-auger

as an update: i believe that i have fixed the problem preventing the mksource mechanism from working automatically, as intended - i still think this feature should be re-vamped, to eliminate mksource() and roll the source-balls after prepare(); but for now, i have created a wiki article based on my "adventures in legacy-land" notes, to serve as a guide for future debugging and development, or if only to guide in it's removal - for now, the wiki article will serve as a guide for debugging, or if only to assist with it's removal

https://wiki.parabola.nu/Hacking:mksource()

#25 - 2020-09-16 03:26 PM - GNUtoo

I've discussed with bill-auger on #parabola on Freenode, and as I understand here's the current status:

- In Parabola, the only way to clean up source code is to use mksource().
- mksource() is currently broken.
- The 8e18d6da36399f4a26e6ffb4910d40f729e7fbae commit fixes it, but it's not in libretools master yet.
- We need to test libretools for a bit of time before making a release.

So to use mksource() right now the way to go is to package an experimental version of libretools in another package like libretools-dev and install that.

#26 - 2020-09-16 05:02 PM - bill-auger

GNUtoo wrote:

> - In Parabola, the only way to clean up source code is to use mksource().

to be precise, its the way to publish pre-cleaned source-balls, and allows users to build from those, instead of downloading the upstream sources and patching them in-place

FWIW, there has been a lot of grumbling about this recently - not that it is a new revelation, but because RMS has taken issue with the way that guix builds chromium and linux-libre, using cleaning scripts to patch the upstream sources in-place - if the FSDG work-group gets back on track, this will probably be one of the first item discussed; and we may need to ensure that every package in [libre] uses the mksource mechanism, or whatever we decide to repalce it with

#27 - 2020-09-16 05:12 PM - oaken-source

I have yet to hear a convincing rationale as to why in the heavens this should be necessary.

#28 - 2020-09-16 06:00 PM - bill-auger

i have been arguing that the objection is rather trite; and that the FSDG would need to be modified to make the requirement explicit - we have been discussing it as a nice-to-have bonus feature; but not a requirement - it has become more clear to me recently, that we have been misunderstanding what the objections is

the main sticking point is that the FSDG requires that distros do not steer users toward non-free software - but in reality, it is probably 1% of the software that parabola patches, which has any non-free software in the upstream sources - the patching is predominantly to meet the sub-set of FSDG prescriptions, which are in addition to the four freedoms

for anything that is clearly non-free or impossible to build from source, we would simply delete the entire package - there are a small few that have some non-free bits in them; and we could probably delete them all, without losing anything important - as the FSDG is written now, i think this is relatively insignificant issue for parabola

but i think that is not what the objection is really about - the main objection is not so much about publishing the cleaning recipes which allow users to clean their own sources - that was what we thought the objection was about, when this came up last year; and several people argued that we absolutely do want to publish the cleaning sciprts

as i understand it better now (i think), the objection is more about the way that the guix build recipes, do not have the option of downloading pre-cleaned sources - users running the guix build recipes, necessarily must download the upstream sources, and clean them on the local machine - most of the hubbub recently, has been about the way that the guix recipe downloads the upstream source from linux, and applies the linux-libre de-blob scripts, rather than using the sources published by linux-libre

i also learned recently, that this the same way the guix recipe builds icecat - i have pointed out before that the icecat cleaning script would be unfit for a FSDG distro, if people apply this rationale strictly - that would be somewhat ironic; but that is the essence of the argument - just as with linux-libre, the primary reason why no one has ever objected to icecat in FSDG distros, is because distros and users do not need to run makeicecat.sh; because gnuzilla publishes pre-cleaned source-balls

if this comes to public discussion, i think the consensus will be that it is not a problem to publish the recipes, as long as the standard tools build from pre-cleaned sources by default - that is exactly what mksource does for parabola, and now it is working again; so i think we are already in good shape, even if people push the issue