**Tails - Bug #15985**

Feature # 15292 (Resolved): Distribute a USB image
Feature # 15293 (Resolved): Creating & preparing the disk image

**Make the disk image reproducible / Make the image creation deterministic**

09/28/2018 10:19 AM - Anonymous

<table>
<thead>
<tr>
<th>Status:</th>
<th>Resolved</th>
<th>Start date:</th>
<th>09/28/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>High</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
<td>% Done:</td>
<td>100%</td>
</tr>
<tr>
<td>Category:</td>
<td></td>
<td>Estimated time:</td>
<td>4.00 hours</td>
</tr>
<tr>
<td>Target version:</td>
<td>Tails_3.12</td>
<td>Spent time:</td>
<td>0.00 hour</td>
</tr>
</tbody>
</table>
| Feature Branch: | feature/15985-reproducible-usb-image-in-
                   | Starter:     |             |
| Type of work:   | Code              |             |             |
| Blueprint:      |                   |             |             |

**Description**

**Related issues:**
Blocks Tails - Bug #16162: Test reproducibility of USB images for all branches

Resolved 11/28/2018

**Associated revisions**

Revision 1331aac7 - 11/22/2018 06:07 PM - segfault
Set a fixed partition GUID in .img (refs: #15985)

Revision b821d84f - 11/22/2018 06:07 PM - segfault
Also set a fixed FAT volume ID (refs: #15985)

Revision 3adb896d - 11/22/2018 07:56 PM - segfault
Also set a fixed FAT volume ID (refs: #15985)

Revision eb4a6a57 - 11/22/2018 07:56 PM - segfault
Reset timestamps on disk image (refs: #15985)

Revision 31cda63e - 11/23/2018 01:40 PM - intrigeri
Use mkdosfs ourselves to pass it the --invariant option (refs: #15985).

Revision 48a9700a - 11/23/2018 01:42 PM - intrigeri
Use mformat (refs: #15985).

Revision 9964d9e0 - 11/26/2018 11:51 PM - segfault
Use mlabel to set fixed UUID (refs: #15985)

Revision c6f806c9 - 11/26/2018 11:53 PM - segfault
Use mlabel to set fixed UUID (refs: #15985)

Revision efb5e912 - 11/27/2018 11:20 AM - segfault
Set a fixed partition GUID in .img (refs: #15985)
Revision 3bd14471 - 11/27/2018 11:20 AM - segfault
Also set a fixed FAT volume ID (refs: #15985)

Revision d8bc5eef - 11/27/2018 11:20 AM - segfault
Reset timestamps on disk image (refs: #15985)

Revision d4ba2480 - 11/27/2018 11:20 AM - segfault
Use mlabel to set fixed UUID (refs: #15985)

Revision cf72c0ff - 11/27/2018 02:35 PM - segfault
Use mkfs.fat to produce a deterministic FAT (refs: #15985)

Revision 45c131c6 - 11/27/2018 02:35 PM - segfault
Also reset timestamps of symlinks (refs: #15985)

Revision e2bade70 - 11/27/2018 03:09 PM - segfault
Also reset timestamps of symlinks (refs: #15985)

Revision 18babc39 - 11/27/2018 04:14 PM - segfault
Use mkfs.fat to produce a deterministic FAT (refs: #15985)

Revision f05a65d0 - 11/29/2018 09:19 AM - segfault
Use mkfs.fat to produce a deterministic FAT (refs: #15985)

Revision 3b3a04d2 - 12/15/2018 05:30 PM - intrigeri
Use mkfs.msdos ourselves to pass it the --invariant option (refs: #15985).

Revision acf876fc - 12/15/2018 05:30 PM - intrigeri
Don't use the kernel vfat driver when generating a USB image (refs: #15985)

It makes it hard to generate a reproducible FAT filesystem. Instead, use Mtools run under faketime, which has no such problem according to our preliminary tests.

To simplify the migration of the script to this new method, drop obsolete bootloader configuration mangling code:

- We don't ship EFI/BOOT/grub.conf. That must be a leftover of Fedora's liveusb-creator code.

- Since lac41ac3af98f8dfaa29ed5882cf95e1cfff1b78 (Tails 3.3), we don't need to rename EFI/BOOT/isolinux.cfg to EFI/BOOT/syslinux.cfg anymore.

- Since 4ff8b67667fc3cc4d0f8f2f2f87ad37486504f72e (Tails 0.14), we don't need to adjust absolute paths to the "isolinux" directory in the syslinux configuration files anymore.

To simplify the migration of the script to this new method, also:

- Stop changing file permissions. I could not find what this code is for as it was part of the initial version of the script in tails.git and I doubt we need it in the context of the Tails ISO + USB image build.
Don't reset timestamps: this should not be needed anymore since all files on the FAT filesystem are created with Mtools + faketime, so their timestamps should be SOURCE_DATE_EPOCH.

Revision ab7c37de - 12/15/2018 11:52 PM - intrigeri
Use mtools from Buster to generate the USB image (refs: #15985)

Revision 6a5eefca - 12/15/2018 11:52 PM - intrigeri
Make mkfs.msdos verbose (refs: #15985)
Let's make it easier to spot issues, such as passing an invalid value to -i, which could make the USB image unreproducible.

Revision e6f470b8 - 12/15/2018 11:53 PM - intrigeri
Fill all 11 chars of the FAT volume label (refs: #15985)
Let's avoid a potential cause of build reproducibility trouble, by ensuring no uninitialized memory can sneaking in there.

Revision e218e29e - 12/15/2018 11:54 PM - intrigeri
Stop using faketime for Mtools (refs: #15985).
It's not needed anymore with mtools 4.0.18-2.1.

Revision 9672c466 - 12/15/2018 11:54 PM - intrigeri
Don't pass a volume ID (-i) to mkfs.msdos (refs: #15985).
We already pass the --invariant option with sets a fixed volume ID:

```c
    case OPT_INVARIANT:
      invariant = 1;
      volume_id = 0x1234abcd;
```

Revision 2071998f - 12/29/2018 10:01 AM - intrigeri
Merge branch 'feature/15985-reproducible-usb-image-intrigeri' into feature/15292-usb-image (Closes: #15985)

History

#1 - 09/28/2018 10:20 AM - Anonymous
- Estimated time set to 4.00 h

#2 - 09/28/2018 10:36 AM - Anonymous
- Blocked by Bug #15991: Code review & rubber-duck for USB Image added

#3 - 09/28/2018 01:33 PM - Anonymous
- Target version changed from Tails_3.11 to Tails_3.12

03/19/2020 3/16
- Parent task changed from #15293 to #15292

#4 - 09/28/2018 01:33 PM - Anonymous
- Target version changed from Tails_3.12 to Tails_3.11

Milestone 4

#5 - 10/09/2018 09:23 AM - intrigeri
- Blocked by deleted (Bug #15991: Code review & rubber-duck for USB Image)

#6 - 11/07/2018 10:09 AM - intrigeri

1. do the obvious test (build twice in a row on the same setup, compare) ASAP because better learn early if it breaks, as it may be costly to fix
2. consider adding build options about reproducibility for the 2nd build: look for "Variations useful for testing build reproducibility" on
https://tails.boum.org/contribute/build/; on Jenkins we use dateoffset=+8 cpus==$((($nproc) - 1)) cpumodel=qemu64
3. then mark the ticket as blocked by the CI adjustments one, and make the CI adjustment one blocked by #15990 (we can't adjust CI as long as
the branch breaks the CI, and I think breaking the build is #15990 material)

#7 - 11/22/2018 04:20 PM - segfault
The .img is currently not reproducible. I found that while we already set a fixed disk GUID, the partition GUID in the VBR was random. I fixed that now
and will test again.

#8 - 11/22/2018 04:31 PM - intrigeri

I fixed that now and will test again.

Woohoo! \o/

#9 - 11/22/2018 06:08 PM - segfault
- Status changed from Confirmed to In Progress

Applied in changeset tails|1331aac7d3a08c24f9fd40e68949ed38c7157c55.

#10 - 11/22/2018 08:01 PM - segfault
I fixed other things which were not reproducible (FAT volume ID, timestamps of files created by our script and syslinux), but there are still some
differences which I couldn't figure out :(.

#11 - 11/23/2018 07:42 AM - intrigeri

I fixed other things which were not reproducible (FAT volume ID, timestamps of files created by our script and syslinux), but there are still some
differences which I couldn't figure out :(
I'll investigate a bit today. I plan to do it this way:

1. rake build an ISO + IMG from this branch
2. manually run the ISO→IMG script on the ISO to produce a 2nd IMG
3. diffoscope the 2 IMG:s
4. share my diffoscope setup and results with you

---

#12 - 11/23/2018 09:03 AM - intrigeri
- Parent task changed from #15292 to #15293

#13 - 11/23/2018 09:59 AM - segfault
intrigeri wrote:

I'll investigate a bit today.

Thanks.

I plan to do it this way:

1. rake build an ISO + IMG from this branch
2. manually run the ISO→IMG script on the ISO to produce a 2nd IMG
3. diffoscope the 2 IMG:s
4. share my diffoscope setup and results with you

That's similar too how I did it, except that I created both images via the script and used cmp -l *.img | gawk '{printf "%08X %02X %02X\n", $1-1, strtonum($2), strtonum($3)}' to compare them. When mounting the images, there are no differences in the file contents, so I don't see another way than byte-by-byte comparison.

---

#14 - 11/23/2018 10:12 AM - intrigeri
Using the rescue build option, I've built an ISO+IMG then rake vm:ssh, cd /tmp/tails-build.* && sudo ./auto/scripts/create-usb-image-from-iso /home/vagrant/amnesia/*.iso. Then I'm run cmp on the 2 IMG:s and it tells me differ: char 4243471, line 6163.

That's quite far in the file so I guess that's inside the FAT filesystem. A few semi-random ideas before I look deeper:
• mkdosfs has an --invariant option that could be useful: "Use constants for normally randomly generated or time based data such as volume ID and creation time. Multiple runs of mkfs.fat on the same device create identical results with this option." But to use it we might need to go low-level and bypass udisks there which will probably break the ability to run as non-root further (I think it's already broken by 3adb896dd75e3b9a50875dc3ea8ee33552cd52f5f).

• I'm not sure about the reset_timestamps implementation: if a file's timestamp is reset after its parent directory's is, this might set the parent directory's timestamp to an unwanted value; I'll check in the resulting filesystem if that worked as expected.

---


segfault wrote:

```
used cmp -l *.img | gawk '{printf "%08X %02X %02Xn", $1-1, strtonum($2), strtonum($3)}' to compare them
```

I'll try that (probably on the system partition instead of the whole image first) if diffoscope does not yield anything useful.

When mounting the images, there are no differences in the file contents, so I don't see another way than byte-by-byte comparison.

Indeed, I've already confirmed that at least the system partitions differ (below, /dev/mapper/loop[0,1]p1 are the system partition mappings kpartx set up for me):

```
$ sudo cmp /dev/mapper/loop[0,1]p1
/dev/mapper/loop0p1 /dev/mapper/loop1p1 differ: byte 3194895, line 6159
```

It would be interesting to compare the rest of the disk image so we can at least tell whether the only difference is inside the FAT filesystem or not.

#16 - 11/23/2018 10:58 AM - intrigeri

intrigeri wrote:

segfault wrote:

```
used cmp -l *.img | gawk '{printf "%08X %02X %02Xn", $1-1, strtonum($2), strtonum($3)}' to compare them
```

So the differences between the 2 system partitions (FAT) are all in concentrated in a few areas and the differing bytes look like repetitive patterns.
This smells like creation time to me. I'll give a try to --invariant.

We might be lucky here: looks like Qubes OS is working on making FAT reproducible too as they've added support for it in diffoscope a month ago. If we don't easily find solutions on our own, we could check what they're doing in their Git and/or ask Marek.

It would be interesting to compare the rest of the disk image so we can at least tell whether the only difference is inside the FAT filesystem or not.

Done: according to dmsetup table the system partition starts at sector 2048. So I've extracted the first 2048 sectors (1.0MB) of the 2 IMG files and they are identical => we shall focus on the FAT filesystem.

---

#17 - 11/23/2018 12:09 PM - intrigeri

intrigeri wrote:

So the differences between the 2 system partitions (FAT) are all in concentrated in a few areas and the differing bytes look like repetitive patterns. This smells like creation time to me. I'll give a try to --invariant.

I've tried using mkfs.vfat --invariant -n Tails123456 + a fixed -i value derived from $SOURCE_DATE_EPOCH (yeah, there's some overlap and that's probably overkill) but that was not enough to get a reproducible system partition.

Next step: compare the filesystem immediately after mkfs.vfat, before mounting and copying files, to determine at what point which difference is introduced.

We might be lucky here: looks like Qubes OS is working on making FAT reproducible too as they've added support for it in diffoscope a month ago. If we don't easily find solutions on our own, we could check what they're doing in their Git and/or ask Marek.

We're not that lucky but still, it helps. They don't seem to do much yet the corresponding PR acknowledges the issue, explains a little bit where it comes from (our current approach will generate differing inode numbers), and suggests some workarounds.

---

#18 - 11/23/2018 01:04 PM - intrigeri
I'm now testing on my own (sid) system instead of inside the Vagrant builder. I hope this does not taint my results.

intrigeri wrote:

Next step: compare the filesystem immediately after mkfs.vfat, before mounting and copying files, to determine at what point which difference is introduced.

Done:

- The resulting FS is already different at that point: 80 lines diff using segfault's cmp -l | gawk command, i.e. a small subset of the differences we see after copying the files in there, but still. The amount of difference is equivalent regardless of whether I pass --invariant or fixed -i + -n values.
- Creating the FS with mformat -F -N 5b9009a9 -v Tails123456 -t 50484 -h 255 -s 63 -i /dev/loop0p1 :: yields a similar amount of differences. That's strange because Marek said the opposite, so perhaps I'm not doing it right.
- Prefixing these mformat and mkfs.vfat commands with faketime '1980-01-01 00:00:00' does not help.

Happy to share the corresponding code if you want to reproduce (it's a little bit more complicated than it looks like because I had to patch mount_partition too so it waits for the filesystem to appear).

I'm afraid I can't think of other workarounds so perhaps it's time to take a step back and brainstorm other options (e.g. is there any other type of FS that would work here? any other implementation of mkfs.vfat?).

And if we can't find simpler options:

- To solve the first problem (formatted partition differs), I think someone needs to prepare a simple reproducer (--invariant does not do what it's supposed to do), report a bug upstream, and quite possibly dive into the mkfs.fat source code to fix that. If we can't do the latter ourselves, we should consider hiring lamby who has lots of experience in this area.
- We need to work on the 2nd problem, i.e. copying the files to an initialized FS does not produce a reproducible image. This can be done in parallel: give some test script an already formatted partition as input, mount it, extract the ISO, do it again with the same input, compare. There, mtools might help as suggested on the Qubes OS pull request.

All this seems non-trivial and could take some time so unless there's progress really soon (as in, by the end of the month) that makes us confident we'll complete this by Dec 15, we need to discuss strategy & timeline.

#19 - 11/24/2018 06:27 AM - intrigeri

I've taken a step back and dreamt awake a little bit, assuming we really need FAT. All the deterministic FS creation processes I'm aware of implement both steps (creating a new FS with data in it) in one single tool, that takes a directory with the data as input and provides some way to use fixed timestamps and metadata: xorriso, mksquashfs. I bet this makes is much easier to ensure the resulting filesystem is reproducible, compared to using
a FS kernel driver to copy the data there; and there's no need to duplicate code (that calls mcopy over all the files in stable order) in every OS image creation script. If such a tool existed for FAT, I bet it would be used by a number of operating systems and tools: Tails, Qubes OS, Debian, debian/build-efi-images in src:grub, grub-mkrescue, etc. I optimistically assume that such a tool would need to support only a rather small subset of the FAT functionality; its output would need to conform to UEFI's specific version of the FAT file system. I'm wondering if it would be cheaper to write such a tool than to try to make the system partition reproducible with the approach we've had so far. In any case, unless we find a cheap solution soon, we'll probably need to reconsider the budget and timeline for this project.

#20 - 11/24/2018 06:49 AM - intrigeri
I've emailed Marek Cc'ing segfault. In passing, fatcat (in Debian) can be useful to explore the low-level details of a FAT filesystem; I suspect it'll give us clearer info about the differences we see than byte-by-byte diff'ing :)

#21 - 11/24/2018 06:52 AM - intrigeri
Also, it might be worth giving FATSort (in Debian too) a quick try. Who knows, perhaps it'll make some internal FAT data, such as the cluster numbers, stable.

#22 - 11/26/2018 10:19 AM - intrigeri
- Target version changed from Tails_3.11 to Tails_3.12

Taking a step back, it's good that we've done the initial analysis and we now have a good understanding of the problem. But fixing this problem is not the critical path of this project at the moment: we can very well have a beta version ready for testing by Dec 15, merged into devel (without the corresponding doc update though), and integrated with the rest of our stuff, with the caveat that generating the USB image is not reproducible. So we should put this on the backburner for now and instead focus on the remaining sibling tickets, that are in the critical path for the Dec 15 milestone.

During our next team meeting, we can discuss how we'll tackle this (e.g. get bonus budget to hire someone to fix the tools we use?) and whether/how it impacts the timeline of the overall project.

I'll initiate discussions on the reproducible FAT topic at the reproducible builds summit on Dec 11-13.

#23 - 11/26/2018 09:48 PM - segfault
Thanks for your work on this. Because of #15988 I have to use a different tool for formatting anyway, udisks doesn't support setting the UUID - so I will try using mformat as suggested by Marek in his reply to your email.

#24 - 11/26/2018 11:39 PM - segfault
segfault wrote:

Because of #15988 I have to use a different tool for formatting anyway, udisks doesn't support setting the UUID

Ignore that, it's nonsense.

so I will try using mformat as suggested by Marek in his reply to your email.

I tried it and it still produces non-deterministic results very similar to the ones I saw before. This is the command I used to produce a filesystem similar to the one produced by create-usb-image-from-iso:

mformat -i PATH_TO_IMAGE -F -h 255 -s 63 -t 197 -H 0 -I 0 -m f8 -v Tails -N a69020d2
While working on the above, I realized that we can use mlabel to set the fixed UUID instead of patching the VBR with dd.

By the way, there is also minfo, which is similar to fatcat, but more useful to find out values to use for mformat. And then there is also parse_vbr.py from tails-verifier, which prints all fields in the VBR.

By the way, there is also minfo, which is similar to fatcat, but more useful to find out values to use for mformat. And then there is also parse_vbr.py from tails-verifier, which prints all fields in the VBR.

Nice! I'm starting to think we should start collecting this info on a wiki page, ideally a cross-distro one. Depending on our progress I'll come back to this topic around the reproducible builds summit.

BTW, lamby did manage to build FAT filesystems reproducibly: https://salsa.debian.org/installer-team/debian-installer/merge_requests/3. So, once the more pressing sibling tickets are done, before dropping the ball and deciding we need external help, I'd like us to give a quick try to the approach used in that code :)

This requires mtools from testing/sid, which fixes some reproducibility issues (https://bugs.debian.org/900409 and https://bugs.debian.org/900410).

And if we end up having to use mcopy, in order to avoid increasing resources requirements on builders, instead of extracting files from the ISO we should probably copy them directly from the binary directory (that's used to create the ISO).

intrigeri wrote:

By the way, there is also minfo, which is similar to fatcat, but more useful to find out values to use for mformat. And then there is also parse_vbr.py from tails-verifier, which prints all fields in the VBR.

Nice! I'm starting to think we should start collecting this info on a wiki page, ideally a cross-distro one. Depending on our progress I'll come back to this topic around the reproducible builds summit.

OK.

BTW, lamby did manage to build FAT filesystems reproducibly: https://salsa.debian.org/installer-team/debian-installer/merge_requests/3.
Interesting. I only saw https://bugs.debian.org/900409 before (from Marek's email), which is about mtools indeed (i.e. mformat). But in the merge request lamby uses mkfs.msdos (from dosfstools I assume).

So, once the more pressing sibling tickets are done, before dropping the ball and deciding we need external help, I'd like us to give a quick try to the approach used in that code :)

I have to wait for a build anyway right now, so I will see if I can create a deterministic FAT with mkfs.msdos as used in lamby's merge request.

segfault wrote:

I tried it and it still produces non-deterministic results very similar to the ones I saw before

FTR, when only executing the mformat command I pasted above (and not installing syslinux or extracting the ISO) there is only one difference, two bytes at offset 0x40742E and 0x407436.

#29 - 11/27/2018 01:11 PM - segfault
segfault wrote:

FTR, when only executing the mformat command I pasted above ... 

that is mformat from mtools 4.0.18-2.1, i.e. with the two reproducibility fixes.

#30 - 11/27/2018 02:29 PM - segfault
I was able to produce a deterministic FAT with this command:

```
mkfs.msdos --invariant -v -i 1234ACAB /dev/loop1p1
```

But when files are created on the filesystem, the "Change" or "ctime" timestamp is set, and this can't be easily changed via touch or similar. I wonder how lamby's debian installer patch works - it uses touch to change the mtime, but that doesn't fix the modified ctime.
I was able to produce a deterministic FAT with this command:

Great! I did not reach that point the other day and I wonder why. Anyways, good news :)

Regarding `cf72c0ff58477ec36cba166d95348cddcdefb885f`, in my experience this won't work reliably: as said above, `self.partition.props.filesystem.call_mount_sync` will fail occasionally. See wip/feature/15292-usb-image-mdosfs for how I made it robust. So for now I won't cherry-pick that commit on my rebased branches because I don't want to make all branches build fragile. I'm also surprised this works without root. Once I've completed #16154, WIP on this ticket should probably live in a dedicated topic branch (even more so when they're untested). I'll keep you updated.

---

intrigeri wrote:

Regarding `cf72c0ff58477ec36cba166d95348cddcdefb885f`, in my experience this won't work reliably: as said above, `self.partition.props.filesystem.call_mount_sync` will fail occasionally.


---

The Qubes OS folks and I came up with a PoC that works fine today. I'll implement it on a topic branch right now.

---

Pushed a PoC! On my sid system, this consistently succeeds:
for i in 1 2 ; do sudo SOURCE_DATE_EPOCH=1544628570 ~/tails/git/auto/scripts/create-usb-image-from-iso tails-amd64-3.11.iso && mv tails-amd64-3.11.img tails-amd64-3.11.img.$i ; done && cmp tails-amd64-3.11.img.1 tails-amd64-3.11.img.2

I'll try this in a Tails build with Vagrant now but I bet that the good results I'm seeing are partly thanks to mtools 4.0.18-2.1, which we'll probably need to bring into the Vagrant VM.

#37 - 12/15/2018 06:39 PM - intrigeri
- QA Check set to Ready for QA

(Force Jenkins to test reproducibility.)

#38 - 12/15/2018 07:02 PM - segfault
- % Done changed from 20 to 0
- QA Check deleted (Ready for QA)

Awesome! I looked through the code and it looks good - didn't test it yet though. I pushed a commit which fixes two minor style issues.

#39 - 12/15/2018 07:03 PM - segfault
- % Done changed from 0 to 20
- QA Check set to Ready for QA

#40 - 12/15/2018 08:55 PM - intrigeri
BTW I've tried to drop faketime (hoping that lamby's patches would be sufficient given I export SOURCE_DATE_EPOCH) and on my sid system, with the same test procedure as above, I also get identical USB images.

#41 - 12/15/2018 11:58 PM - intrigeri
Unfortunately, USB images are still not reproducible: one built on my laptop does not match one built on my local Jenkins. I'm trying a few more things. If they fail, next step: check whether the difference appears at mkfs time already or only later.

#42 - 12/15/2018 11:59 PM - intrigeri
- QA Check deleted (Ready for QA)

(Added the branch to the list of those where reproducibility will always be tested on Jenkins.)

#43 - 12/16/2018 12:23 AM - intrigeri
intrigeri wrote:

Unfortunately, USB images are still not reproducible: one built on my laptop does not match one built on my local Jenkins. I'm trying a few more things.

Well, one of these things (not sure which one) worked! I now have two matching USB images, one built on my laptop does not match one built on my local Jenkins :) Also, I'm not using faketime anymore, which is great (the mere mention of this tool makes reproducible builds people unhappy because it has much potential for breaking stuff randomly). If all goes well, https://jenkins.tails.boum.org/view/Tails_ISO/job/reproducibly_build_Tails_ISO_feature-15985-reproducible-usb-image-intrigeri/5/ should succeed.
Five builds at 8d8969c513f78ad15eb419ca3d0cf37baac1b71 produced identical USB images: one on my laptop, 1 on my local Jenkins + the corresponding 2nd build in a different environment with variations, 1 on our shared Jenkins + the corresponding 2nd build in a different environment with variations.

Please review and if happy, merge into feature/15292-usb-image and delete this topic branch :)

Here are the notes from my review:

- ab7c37dead9d75ba32af8b78ae45306258a2a305: Is it on purpose that you overwrite stretch-updates.list instead of appending?
- e6f470b8e87e572af29106259caf624887d10de1: I would prefer filling the label with spaces instead of numbers, i.e. 'TAILS' + 6 * '

I will now test if I can create reproducible images from this branch.

I successfully built two images with identical SHA hashes.

Thanks!

ab7c37dead9d75ba32af8b78ae45306258a2a305: Is it on purpose that you overwrite stretch-updates.list instead of appending?

Good catch! Copy'n'paste error. Fixed.
Done. Let's see if that still works (I don't see why not, but well).

---

#49 - 12/19/2018 07:15 PM - segfault
- Assignee changed from segfault to intrigeri
- QA Check changed from Ready for QA to Pass

LGTMD. Should I merge this into feature/15292-generate-usb-image -> master -> stable -> devel?

---

#50 - 12/19/2018 08:07 PM - intrigeri
- Assignee changed from intrigeri to segfault

LGTMD. Should I merge this into feature/15292-generate-usb-image -> master -> stable -> devel?

---

Sadly, I've based my branch on one that's based on devel, so please merge only in our integration branch (feature/15292-usb-image).

---

#51 - 12/27/2018 09:45 AM - Anonymous
- Priority changed from Normal to High

@segfault: we planned to release a beta before the end of the year, that has basically 4 more days. Could you please merge this as described above? Or if not available, please let me know so I can find another solution. Thank you.

---

#52 - 12/27/2018 09:45 AM - Anonymous
- QA Check changed from Pass to Dev Needed

---

#53 - 12/28/2018 04:53 PM - segfault
- Assignee changed from segfault to intrigeri

Sorry, I won't be able to merge it in the next 3 days. intrigeri, could you do this?

---

#54 - 12/28/2018 05:00 PM - intrigeri

Sorry, I won't be able to merge it in the next 3 days. intrigeri, could you do this?

Sure.
#55 - 12/29/2018 10:03 AM - intrigeri
- Status changed from In Progress to Resolved
- % Done changed from 50 to 100

Applied in changeset tails|2071998f3e3d5ae90f04c60903dbd21f4f1fbc83.

#56 - 12/29/2018 10:08 AM - intrigeri
- Assignee deleted (intrigeri)
- QA Check changed from Dev Needed to Pass