



Manage Settings

rokytnji ▾

antiX-forum

Forum for users of antiX linux. "Mean and Lean"

☰ FORUMS

DISCUSSIONS

antiX-forum

General

Tips and Tricks ▾

Build your own!

8 posts • Page 1 of 1

Post Reply ↩



Search this topic...



anticapitalista

Posts 5,952

Site Admin

Joined: Tue Sep 11, 2007
9:55 am

🕒 Sun Jun 23, 2013 2:30 pm #1 ▾

antiX provides several ways for user customisation. You can make it as fat or thin as you like either installed to hard drive and then using the antixsnapshot application to make a live iso of your installed system, or running live on a stick or frugally on a hard drive and then using the persistence/remaster scripts to create a new live iso. There is also another way. That is building from scratch using our live build scripts.

This post is what I did earlier today, using the live persistence/remaster scripts, running frugally on a hard drive.

1. I booted the latest antiX-13.1-386-core-libre as a frugal install on an internal hard drive. Since this is X less, you boot into a terminal screen. Login as user (demo/demo). I checked the RAM use with inxi -F and it showed less than 20MB.

2. Set up root persistence.

```
sudo persist-makefs
```

Option 1 asks to set up rootfs. Choose yes and set the size. I chose 512MB and ext2.
reboot.

3. Boot again making sure you use the persistent root cheat. You will be prompted to change your root password and user (demo) password. Login as root and now we will install some apps.

4. For this test I wanted to get a low RAM desktop so I decided on JWM.

```
apt-get update
```

```
apt-get install xserver-xorg-core xserver-xorg-input-all xserver-xorg-video-vesa xserver-xorg-xinit (probably not needed) rxvt-unicode-256color leafpad rox-filer jwm iceape-browser slim
```

5. If you install slim, you should edit /etc/slim.conf to enable this top line (use mc editor or nano as root).

```
login_cmd exec /bin/bash -login /etc/X11/Xsession %session
```

```
# login_cmd exec /bin/bash -login ~/.xinitrc %session > ~/.xsession-errors$DISPLAY 2>&1
```

```
# login_cmd exec dbus-launch /bin/bash -login ~/.xinitrc %session > ~/.xsession-errors 2>&1
```

You may want to enable autologin as well. Edit the same slim.conf file.

exit and login as demo then type startx and you should login to a very basic jwm desktop.

Open htop to see the RAM use.

6. VIP before you reboot, you need to make sure the persistent changes are saved.

```
Ctrl Alt F1 to console screen
```

```
Login as demo
```

```
sudo persist-save
```

Once that is done, sudo reboot

7. Make sure rootfs is chosen again, you will see a message saying that that the persistent root changes are being copied and then you should boot to your jwm desktop. Check RAM via htop.

In my test, running live on real hardware it was 40MB, in VirtualBox it was less than 30MB.



Manage Settings

rokytnji ▾

Ctrl Alt F1 to console screen
Login as demo
sudo remaster-live

and follow the instructions.

Once finished, this creates a new linuxfs image that contains all your changes.

9. Reboot, this time do not choose persist option. It should boot very fast to a jwm desktop.
In my test in VB it booted in 8 seconds and RAM use was only 25MB! Remember we are still running live! Use the cli-installer if you want to install to hard drive.

10. Make an iso file. To do this we need to cd into where the linuxfs, initrd.gz and vmlinuz files are. Typically they will be in the iso directory you booted frugally from.
cd /media/sda3/frugal/iso
mkt (This bash alias makes a live iso called antiX-test.iso)

11. Enjoy!

Philosophers have interpreted the world in many ways; the point is to change it.

↩ Reply ✎ Edit

SamK

Posts 1,028

Joined: Sun Aug 21, 2011
3:59 am

🕒 Mon Jun 24, 2013 4:12 am #2



Very interesting indeed.

“ anticapitalista wrote:
...xserver-xorg-video-vesa...

Is the idea behind the inclusion of this to enable straight-from-the-box operation across the widest range of hardware?

“ anticapitalista wrote:
...xserver-xorg-input-all...

Abstract from

CODE: SELECT ALL

```
apt-cache show xserver-xorg-input-all
...
Description-en: X.Org X server -- input driver metapackage
This package depends on the full suite of input drivers for the X.Org X server
(Xorg). It does not provide any drivers itself, and may be removed if you wish to only have certain drivers installed.
...
```

Just musing aloud...

- Maybe this means it can be omitted?
- I wonder if leaving it out has any noticeable affect on the amount of RAM used?
- Perhaps the value of including it is as preparation for the user to install other drivers

If I understand correctly the build produces a system

- Capable of booting from a USB flash drive
- Contains a user specified application set
- Is able to be used with non-persistent / and /home

This is the way I use a different distro to build live systems dedicated to a narrow range of tasks rather than general purpose desktop duties. To do this and have access to the Debian stable repository is very appealing.



Manage Settings

rokytnji ▾

- The new linuxfs image that contains all your changes
- The ISO file created from the new linuxfs image

A major benefit of non-persistent / and /home is that the system is highly resistant to corruption, malicious tampering and infection. At bootup a pristine environment is always created. At power-down this environment is destroyed together with all its contents. One consequence of this is that changes made to the configuration of an app during a live session are not preserved across a reboot. Live persistence might be enabled for /home, however this means that a pristine environment is no longer created following a reboot.

One way of maximizing the pristine condition and enabling the preserving of configuration changes made during a live session might be similar to the following method.

During a live session (non-persistent / and /home), the user specifies the app configuration to be preserved. This is stored in a compressed preserved.tgz format. Preserved.tgz is stored in the persistent storage area that holds the linuxfs bootable image. Preserved.tgz may be manually created by the user at any time during the live session, and/or automatically at log-out or reboot. At bootup, the startup sequence automatically restores preserved.tgz.

The outcome is

- A pristine environment is produced for / and /home
- The user can change the configuration of an app during a live session
- At the user's discretion, the configuration is preserved across a reboot

Do you see this as complementing and compatible with your promising idea? If so, perhaps selecting a file to be preserved might be done via ROX-Filer "send to". Alternatively a method independent of ROX might be to use Dialog to present a "GUI" selection list that will also work in a CLI system (does Yad work in CLI or only GUI ?).

↩ Reply ✎ Edit

anticapitalista

Posts 5,952

Site Admin

Joined: Tue Sep 11, 2007
9:55 am

🕒 Mon Jun 24, 2013 6:08 am #3 ▾

Is the idea behind the inclusion of this to enable straight-from-the-box operation across the widest range of hardware?

It was only to use this driver as it is supposed to work on all hardware and with low RAM.

xserver-xorg-input-all

This detects devices like mouse, keyboards etc.

If I understand correctly the build produces a system

Capable of booting from a USB flash drive
Contains a user specified application set
Is able to be used with non-persistent / and /home

That is correct.

What are the storage sizes of the build as described for

The new linuxfs image that contains all your changes
The ISO file created from the new linuxfs image

New linuxfs is 215MB and iso is 221MB

About persistence. Once user is satisfied with persistence changes, user should then remaster to create a new, pristine linuxfs file and then run without persistence or only run with home persistence and have the home data encrypted.

Philosophers have interpreted the world in many ways; the point is to change it.

↩ Reply ✎ Edit



Manage Settings

rokytnji ▾

SamK

Posts 1,028Joined: Sun Aug 21, 2011
3:59 am

🕒 Mon Jun 24, 2013 8:29 am #4 ▾

“ anticapitalista wrote:
About persistence.

I was not really aiming at persistence, but at a way of obtaining a pristine environment at each bootup, particularly the user's home directory.

Perhaps a simplified example may help. Your build on a bootable USB flash drive is used to boot the system with persistent / and non-persistent user home. The user now installs an additional app that looks to place its configuration file in the user home area.

“ anticapitalista wrote:
Once user is satisfied with persistence changes, user should then remaster to create a new, pristine linuxfs file...

This will capture the application executables but not its configuration file.

“ anticapitalista wrote:
...and then run without persistence...

The app configuration will not exist in this circumstance.

“ anticapitalista wrote:
...or only run with home persistence...

In which case there is no longer a pristine user home.

“ anticapitalista wrote:
...and have the home data encrypted.

In order to read and write to the encrypted user home, the encryption must first be "unlocked". Once this has been done, the system has transparent access to the user home. In this condition the user home is persistent and non-pristine. It is also non-selective. At shutdown everything in the user home area is saved and encrypted. The encryption will help prevent unauthorized access to the data in a persistent user home area when the USB stick is attached to a machine that was not booted from the stick.

The idea of using preserved.tgz is to capture app config files *without enabling home persistence*. It would also be able to capture similar files in /. It is very selective. Only the configuration files explicitly specified by the user are retained in preserved.tgz, all other files in the user home area are destroyed once the system is powered off. In so doing a fresh / and a pristine user home are always created at each boot up. The outcome is a self-protected system which is insulated from unwanted changes and maximizes reliability.

If preserved.tgz cannot restore the configs to the USB live file system then the idea is redundant. It is also redundant if the user home must be persistent. I am looking for information on these points but have not found any up to now. Perhaps **BitJam** might be willing to offer a view on these particular aspects? If they are not "blockers", then it is likely that it can be done without needing changes to the way antiX live USB works. The idea is as yet untested in antiX and I won't be able to test it for a while.

Note:

I found this file on the local system /usr/share/antiX/FAQ/persistence.html. It is referring to antix2usb and indicates

Persistence. There are four options here.

- No persistence - just leave as default.
- home persistence - check home and select size
- root persistence - check root and select size
- Both home and root persistence - check both home and root and select sizes.

If this is correct then LiveUSB can be run with non-persistent user home



Manage Settings

rokytnji ▾

rmcellig

Posts 177

Joined: Tue Mar 04, 2014
12:37 pm

🕒 Wed Mar 12, 2014 3:43 pm #5 ▾

Am I right to assume that this is similar to Puppy Linux in regard to save files and remastering in Puppy Linux? Can I have multiple persistent files like I can have multiple save files in Puppy?

Does the Antix way of doing this offer a better solution than Puppy? One thing I like about Antix is that it is connected to the Debian repos so I don't have to go searching for .pet files.

I think I will try out your above steps and see what happens.

I love a minimalist approach to computing, and this is from a Mac user who didn't want to deal with bloat anymore. I'm hoping Antix will meet my needs. At the moment, my distro is Crunchbang 11. I love it!

↩ Reply ✎ Edit

skidoo

Posts 1,441

Joined: Wed Feb 08, 2012
11:29 pm

🕒 Wed Mar 12, 2014 11:03 pm #6 ▾

I think antix provides more flexibility in choosing what to persist (does puppyPrecise offer a choice of root vs home persistence?)

Using the antix "semi-automatic" root persistence, I'm able to update the savefile at will, on demand (vs only at shutdown).
I don't recall whether pPrecise supports on-demand savefile update.
In puppy FatDog64 and some other puppy derivatives, on-demand savefile update is available via bullseye icon on the desktop.

Multiple savefiles? How awkward is it to manage multiple savefiles (*.2fs) in puppyPrecise?
(I'm uncertain which exact puppy derivative you're using)
Managing multiples in antix would be equally (managed manually) awkward, and a bit more "dicey".
Due to the antix rolling release model, and the frequency of Debian's package updates, mix-n-match root persistence files invites breakage.

Last edited by skidoo on Fri Mar 14, 2014 3:16 am, edited 1 time in total.

↩ Reply ✎ Edit

BitJam

Posts 1,308

Joined: Mon Aug 31, 2009
5:48 pm

🕒 Thu Mar 13, 2014 3:39 am #7 ▾



Manage Settings

rokytnji ▼

Am I right to assume that this is similar to Puppy Linux in regard to save files and remastering in Puppy Linux?

I'm not really familiar with how Puppy does it so I don't know how similar it is. I imagine it is basically the same idea of adding persistence and remastering to a live system. If you know Puppy, maybe you could write a post or an article comparing the two systems after you get some more experience with ours. Perhaps that would provide some useful cross fertilization.

Can I have multiple persistent files like I can have multiple save files in Puppy?

This is possible but we have not set it up for this to be streamlined and easy. You just need to put homefs and rootfs files in different directories and then specify the directory as a boot option "pdir=xxxx". The default pdir is the boot directory, the one that contains the linuxfs file. So if you want to have different remaster versions just put the remastered linuxfs files in different directories and use the "bdir=xxxx" option. This way each remastered linuxfs file can have its own rootfs persistence. All of this makes it easy to have several distros or several versions of the same distro on one LiveUSB. Here is an example of the file layout for booting 2 different systems:

CODE: SELECT ALL

```
antix/
  vmlinuz
  initrd.gz
  linuxfs
  rootfs

antix-64/
  vmlinuz
  initrd.gz
  linuxfs
  rootfs
```

And so on. The default boot-directory is "antix" so the 64-bit system would need a "bdir=antix-64" boot parameter while the system in antix/ would not need any. Legacy Grub entries would look like:

CODE: SELECT ALL

```
Title antiX 32-bit
kernel /antix/vmlinuz [etc]
initrd /antix/initrd.gz

title antiX 64-bit
kernel /antix-64/vmlinuz bdir=antix-64 [etc]
initrd /antix-64/initrd.gz
```

We've made it easy for users to modify bootloader menus so if you just want to change the rootfs file, you could do that with a custom bootloader menu that sets "pdir=xxx". Also, you should check out the "F7 Save" feature in our Live bootloader. This is only enabled on LiveUSBs. It lets you set the defaults in the bootloader itself. Just set whatever menus or custom boot parameters you want (language, timezone, persistence option, etc) and then select "F7 Save" --> "both" and those settings should become the defaults the next time you boot. This might be the easiest way possible to customize a LiveUSB.

Does the Antix way of doing this offer a better solution than Puppy? One thing I like about Antix is that it is connected to the Debian repos so I don't have to go searching for .pet files.

I'm not familiar with the Puppy system so I can't tell you about better or worse. Each system probably has its own strengths and weaknesses. If you are looking for something that is like Puppy + Debian then antiX is probably a good choice.

One hurdle we got over was being about to do a dist-upgrade on the Live system. First we needed to make the Live file system mimic the installed filesystem better. Next we added static root persistence so RAM would no longer be a limiting factor for how much you can install on the root filesystem. AFAIK we are the only ones offering the choice at boot time of static and dynamic root persistence. Another thing we are very careful about is cleanly unmounting everything at shutdown. I think our Live shutdown procedures might be better than those on most Linux distros. We can give you shell access even after all the filesystems have been unmounted and all process but init have been killed. This means it should be safe to put static root persistence files on an ntfs file system. The normal Debian shutdown procedure does not handle this situation well. To get shell access during shutdown use the boot parameter "ubp=a". This will set a series of breakpoints during the shutdown process before and after processes get killed off and filesystems get unmounted. You can also get our shutdown process to print more with the "uverb=X" parameter. Use "uverb=8" or so. You can get more debugging output with "ushow=X". Here are what various ushow values will give you:

CODE: SELECT ALL

```
1) ps
2) free -m
3) lsof
4) df -h
5) df -ha
11) ps
```



Manage Settings

rokytnji ▾

Science is the belief in the ignorance of experts.
 -- Richard Feynman

↩ Reply ✎ Edit

skidoo

Posts 1,441

Joined: Wed Feb 08, 2012
 11:29 pm

🕒 Mon Feb 13, 2017 5:04 pm #8 ▾

posting to this pinned topic to mention that although instruction 2 is still "valid"
 nowadays setup will be performed automatically if persistence option is selected in the (antiX or MX 'legacy boot') setup menu
 and
 can nowadays setup can be invoked via ControlCenter (vs terminal "sudo persist-makefs")

Regarding Instruction3, it might still be effective, but nowadays hand-typing (use the 'persistent root cheat') isn't necessary ~~ a user can choose a
 bootmenu screen option (dropdown item) instead.

2. Set up root persistence.

`sudo persist-makefs`

Option 1 asks to set up rootfs. Choose yes and set the size. I chose 512MB and ext2.
 reboot.

3. Boot again making sure you use the persistent root cheat. You will be prompted to change your root password and user (demo)
 password. Login as root and now we will install some apps.

↩ Reply ✎ Edit

Submit

Full Editor & Preview

Post Reply ↩ ✎ | 📄 | 🔄 |

8 posts • Page 1 of 1

< Return to "Tips and Tricks"

Jump to ▾

Back to top


[Privacy Policy](#)
[Terms of Use](#)
[Code of Conduct](#)
[End-User License Agreement](#)
[Site Owner License Agreement](#)
[Tapatalk API Terms of Service](#)
[© 2017 Tapatalk Inc](#)



tapatalk

Manage Settings

rokytnji ▾

