superuser

Copy entire file system hierarchy from one drive to another

Asked 13 years, 5 months ago Modified 9 months ago Viewed 304k times

168

I would like to copy the entire file system hierarchy from one drive to another..i.e contents of each directory as well as regular files in Linux platform. Would be gratefull to know the best way to do that with possibly Linuxes in-built functions. The file system is a ext family.

linux filesystems

Share Improve this question Follow

edited Dec 24, 2014 at 9:11
Chenmunka
3,226 13 30 38

asked Jul 7, 2011 at 3:30
Juggler
1,781 3 11 4

3 Umm... where is the love for `dd` ? `dd if=/dev/sda1 of=/dev/sdb1 bs=4096` – Joseph Mar 25, 2017 at 5:08

@juniorRubyist +1 for dd. I always use that. But which flags to use? I use `conv=notrunc,noerror,sync` . – BeniBela Aug 30, 2017 at 12:10

7 -1 for `dd` for 2 reasons: firstly it's a bad idea to perform a block-level copy of a filesystem that is mounted (which is the case for `/`) and secondly `dd` won't copy data from sources mounted within the filesystem like `/boot` and `/home` . – Eric Aug 16, 2019 at 10:37

8 -1 for dd also because whatever fragmentation has occurred on the source is copied as well; and a different sized destination isn't automatically handled – duanev Oct 30, 2019 at 17:53

1 I'm glad you mentioned that, @duanev . I was wondering whether this was happen when it's claimed that Linux file systems do not fragment. – Sridhar Sarnobat Mar 31, 2022 at 0:34

12 Answers

Sorted by: Highest score (default)

47

I often use

```
> cp -ax / /mnt
```

Presuming /mnt is the new disk mounted on /mnt and there are no other mounts on /.

the -x keeps it on the one filesystem.

This of course needs to be done as root or using sudo.

This link has some alternatives, including the one above

http://linuxdocs.org/HOWTOs/mini/Hard-Disk-Upgrade/copy.html

Share Improve this answer Follow

answered Jul 7, 2011 at 3:42
WolfmanJM
980 6 5

while this is dead old answer it is still worth noting that you usually do NOT want to copy all the stuff present in `/`, excluding i.e. `/dev`, `/sys`, `/proc` etc. Therefore before issuing that `cp` I suggest searching for better approaches (also using rsync) – Marcin Orlowski Jun 13, 2019 at 8:17

12 @MarcinOrlowski WolfJM's use of the `-x` flag means that the synthetic filesystems you mention will not be copied. – Jim L. Jul 31, 2019 at 21:37

312

What you want is rsync.

This command can be used to synchronize a folder, and also resume copying when it's aborted half way. The command to copy one disk is:

```
rsync -avxHAX --progress / /new-disk/
```

The options are:

```
-a  : all files, with permissions, etc..
-v  : verbose, mention files
-x  : stay on one file system
-H  : preserve hard links (not included with -a)
-A  : preserve ACLs/permissions (not included with -a)
-X  : preserve extended attributes (not included with -a)
```

To improve the copy speed, add `-W` (`--whole-file`), to avoid calculating deltas/diffs of the files. This is the default when both the source and destination are specified as local paths, since the real benefit of rsync's delta-transfer algorithm is reducing network usage.

Also consider adding `--numeric-ids` to avoid mapping uid/gid values by user/group name.

Share Improve this answer Follow

edited Jul 6, 2013 at 5:39
Raman
393 3 9

answered Jul 7, 2011 at 3:36
Michael Aaron Safyan
3,813 3 18 16

3 Genius, thank you. Btw, I ended up using `rsync -avxHAWX --numeric-ids --progress / mnt/` but I should have done `rsync -avxHAWX --numeric-ids --progress / mnt/ > ~/rsync.out`. I suspect pouring the output to the terminal slowed the process. :D – Krista K Jan 15, 2014 at 10:15

64 `--info=progress2` instead of `--progress` is useful for large transfers, as it gives overall progress, instead of (millions of lines for) individual files. – Florian Sep 27, 2015 at 8:43

4 I had to replace `X` and `A` with `E`, because extended attributes and ACLs are covered by `E` on my mac. Tested: `rsync  version 2.6.9  protocol version 29` – Jonathan Komar Aug 4, 2016 at

5:17

9 `-S, --sparse              handle sparse files efficiently` – Ken Sharp Mar 11, 2017 at 12:55

12 If you're copying from some other folder than `/`, note that having a trailing slash (or not) on the source directory makes a difference: `rsync source/ dest/` copies everything inside `source/` to `dest/`, while `rsync source dest/` copies the folder `source` and everything inside into `dest/`. – semi-extrinsic Mar 20, 2017 at 8:22

163

Michael Aaron Safyan's answer doesn't account for sparse files. `-S` option fixes that.

Also this variant doesn't spam with the each file progressing and doesn't do delta syncing which kills performance in non-network cases.

Perfect for copying filesystem from one local drive to another local drive.

```
rsync -axHAWXS --numeric-ids --info=progress2
```

Share Improve this answer Follow

answered Mar 5, 2017 at 10:42
Ilia Sidorenko
1,731 1 10 6

1 Amazing. This is really doing a good job – Gildas Jul 26, 2018 at 8:13

3 This should be the accepted answer, works great. Example `55,431,669,792  57%   97.47MB/s 0:06:56  xfr#2888, ir-chk=5593/8534)` – Drew Aug 4, 2018 at 3:11

1 <3 this is perfect – Tim Strijdhorst Jan 21, 2019 at 15:10

3 WATCH OUT! What is not "perfect" is that this answer doesn't tell you that if you have two internal drives, subsequent reboots of a live install USB may switch "sda" and "sdb", and so this answer becomes, instead of a way to restore correct privileges from a back up, a way to instead overwrite your backup with the completely broken privileges of your 1st rysnc attempt, destroying the backup and leaving you with no good copy of the server you just spent 3 weeks setting up, like just happened to me. Then you get to spend hours reading all the answers on SE that tell you there's no way to fix it. – John Smith Nov 17, 2023 at 9:56

1 Great! But don't forget to put a trailing slash on both directories – nealmcb Dec 7, 2023 at 19:39

10

Like Michael Safyan suggests above, I've used `rsync` for this purpose. I suggest using some additional options to exclude directories that you probably don't want to copy.

This version is fairly specific to Gnome- and Debian/Ubuntu-based systems, since it includes subdirectories of users' home directories which are specific to Gnome, as well as the APT package cache.

The last line will exclude **any** directory named cache/Cache/.cache, which may be too aggressive for some uses:

```
rsync -WavxHAX --delete-excluded --progress \
  /mnt/from/ /mnt/to/
  --exclude='/home/*/.gvfs' \
  --exclude='/home/*/.local/share/Trash' \
  --exclude='/var/run/*' \
  --exclude='/var/lock/*' \
  --exclude='/lib/modules/*/volatile/.mounted' \
  --exclude='/var/cache/apt/archives/*' \
  --exclude='/home/*/.mozilla/firefox/*/Cache' \
  --exclude='/home/*/.cache/chromium'
  --exclude='home/*/.thumbnails' \
  --exclude=.cache --exclude Cache --exclude cache
```

Share Improve this answer Follow

answered Feb 6, 2014 at 6:11
Dan
424 4 10

7

As mentioned in the comments by juniorRubyist, the preferred approach here should be to use `dd`. The main reason is performance, it's a block-by-block copy instead of file-by-file.

Cloning a partition

```
# dd if=/dev/sda1 of=/dev/sdb1 bs=64K conv=noerror,sync status=progress
```

Cloning an entire disk

```
# dd if=/dev/sdX of=/dev/sdY bs=64K conv=noerror,sync status=progress
```

Clone a mounted writable partition

The key for cloning a partition that is mounted read-writable is to remount it as read-only. Then do the cloning and finally remount it again to read-writable.

```
# mount -o remount,ro /path/to/mount_point
# dd if=/dev/sda1 of=/dev/sdb1 bs=64K conv=noerror,sync status=progress
# mount -o remount,rw /path/to/mount_point
```

Note. Doing this may have some side-effects on running applications. E.g. if your system have applications that requires writing to this particular partition exactly when you need to clone it, these applications would need to be stopped while partition is cloned. Or if its your own application re-write it to handle this scenario.

Clone a disk with one or more mounted writable partitions

The strategy and side-effects are the same as for *Clone a mounted writable partition* except this time the remount commands are repeated for each writable mount point.

```
# mount -o remount,ro /path/to/writeable_mount_point1
# mount -o remount,ro /path/to/writeable_mount_point..
# mount -o remount,ro /path/to/writeable_mount_pointN
# dd if=/dev/sdX of=/dev/sdY bs=64K conv=noerror,sync status=progress
# mount -o remount,rw /path/to/writeable_mount_point1
# mount -o remount,rw /path/to/writeable_mount_point..
# mount -o remount,rw /path/to/writeable_mount_pointN
```

Final notes

The preferred and recommended way of doing disk/partition cloning is to do so on a non-mounted system since that will not have any non-deterministic side effects. The same goes for systems built on the concept of read-only mounts.

References

1. https://wiki.archlinux.org/index.php/disk_cloning

Share Improve this answer Follow edited Dec 16, 2020 at 14:05 answered Jun 14, 2018 at 17:30

Rikard Söderström
231 2 5

3 `dd` is a very bad idea for 2 reasons: firstly performing a block-level copy of a filesystem that is mounted (which is the case for `/`) will more than likely result in target filesystem errors, and secondly `dd` won't copy data from sources mounted within the filesystem like `/boot` and `/home` . Your link is valid for disk cloning, not "file hierarchy" cloning – Eric Aug 16, 2019 at 10:42

@Eric I may have made a few assumptions here. Firstly I assumed "copy the entire file system hierarchy from one drive to another." to be the equivalence of "how do I clone my disk", since "file hierarchy cloning" only becomes useful when cloning a subset of the disk. Secondly I assumed this was a disk used for storage not for the system you are currently running. However, simply mounting the system does not imply filesystem errors. It depends on how its mounted, read-only mounts are perfectly fine to clone. – Rikard Söderström Dec 16, 2020 at 13:42

whatever float your boat – Eric Dec 17, 2020 at 14:22

6

For a one shot local copy from one drive to another, I guess **cp** suffices as described by Wolfmann here above.

For bigger works like local or remote backups for instance, the best is **rsync**.

Of course, rsync is significantly more complex to use.

Why rsync :

- this allows you to copy (synchronized copy) all or part of your drive A to drive B, with many options, like excluding some directories from the copy (for instance excluding /proc).
- Another big advantage is that this native tool monitors the file transfer: eg for massive transfers, if the connection is interrupted, it will continue from the breakpoint.
- And last but not least, rsync uses ssh connection, so this allow you to achive remote synchronized secured "copies". Have a look to the man page as well as here for some examples.

Share Improve this answer Follow

answered Jul 7, 2011 at 20:53
hornetbzz
211 3 13

5

rsync

> "This approach is considered to be better than disk cloning with dd since it allows for a different size, partition table and filesystem to be used, and better than copying with cp -a as well, because it allows greater control over file permissions, attributes, Access Control Lists (ACLs) and extended attributes."

From:

https://wiki.archlinux.org/index.php/Full_system_backup_with_rsync

Man Page Here

Share Improve this answer Follow

edited May 18, 2015 at 17:48
Francisco Tapia
2,654 4 25 43

answered May 18, 2015 at 15:45
Mauro Colella
151 1 3

5

I tried the rsync commands proposed here but eventually I got much cleaner and faster results with `partclone` . Unmount source and target partitions and then run the following:

```
partclone.ext4 -b -s /dev/sd(source) -o /dev/sd(target)
e2fsck -f /dev/sd(target)
resize2fs /dev/sd(target)
```

This performs the following steps:

1. Clone (only the used parts of) the partition
2. make sure the file system is o.k. (resize2fs enforces this step)
3. resize the partition to the new file system

The above works in case the target partition is the same size or larger than the source. If your target is smaller than the source (but fits all data) then do the following:

```
e2fsck -f /dev/sd(source)
resize2fs -M /dev/sd(source)
partclone.ext4 -b -s /dev/sd(source) -o /dev/sd(target)
resize2fs /dev/sd(target)
```

`resize2fs -M` shrinks the filesystem to the minimum size before cloning the data.

Note that `partclone` is not installed by default on most systems. Use a live distro like clonezilla or install partclone from your distros packet manager (`apt-get install partclone` on debian-based sytsems).

Edit: thanks @jbroome for pointing out an error in the second code block

Share Improve this answer Follow edited Mar 12 at 14:08 answered Jul 31, 2019 at 21:30
Thawn
326 2 6

1 Hi from 2024. Did you mean for the first two lines in the "target is smaller" code block to be run against the SOURCE, not the target? – jbroome Mar 7 at 2:56

@jbroome yes you are correct. Thanks for spotting this :+1: – Thawn Mar 12 at 14:06

For this you need to unmount an entire running system, and introduce a third one (a live USB or something). Who would want that? Just use rsync and change boot-drives in BIOS. – Julius Sep 9 at 12:47

@Julius that is only true, if you are cloning the system drive. And even if you are cloning the system drive, partclone is just much faster than rsync. Furthermore, I have had problems with /dev, /sys and /proc when cloning a running system. Therefore, I would always recommend to clone the system disk from a live USB stick. – Thawn Sep 16 at 8:26

I've done this using rsync hundreds of times in my life. Never had any issues with /dev, /sys and /proc to be honest. Copying from them goes fine, and a booting linux corrects what may be wrong in those anyway. – Julius Dec 12 at 13:33

2

Adding two useful bits to the thread re rsync: changing cypher, and using `--update` :

As per Wolfman's post, `cp -ax` is elegant, and cool for local stuff.

However, `rsync` is awesome also. Further to Michael's answer re `-W`, *changing the cypher* can also speed things up (read up on any security implications though).

```
rsync --progress --rsh="ssh -c blowfish" / /mnt/dest -auvx
```

There is some discussion (and benchmarks) around the place about a slow CPU being the actual bottleneck, but it does seem to help me when machine is loaded up doing other concurrent things.

One of the other big reasons for using rsync in a large, recursive copy like this is because of the ***-u*** switch (or --*update*). If there is a problem during the copy, you can fix it up, and rsync will pick up where it left off (I don't think scp has this). Doing it locally, cp also has a -u switch.

(I'm not certain what the implications of --update and --whole-file together are, but they always seem to work sensibly for me in this type of task)

I realise this isn't a thread about rsync's features, but some of the most common I use for this are:

- *--delete-after* etc (as Michael mentioned in follow-up), if you want to sync the new system back to the original place or something like that. And,
- *--exclude* - for skipping directories/files, for instances like copying/creating a new system to a new place whilst skipping user home directories etc (either you are mounting homes from somewhere else, or creating new users etc).

Incidentally, if I ever have to use windows, I use rsync from cygwin to do large recursive copies, because of explorer's slightly brain-dead wanting to start from the beginning (although I find Finder is OS X even worse)

Share Improve this answer Follow

edited Dec 2, 2012 at 18:13
cdeszaq
113 5

answered Mar 19, 2012 at 23:53
herdingofthecats
353 2 6

1

'dd' is awesome, but ddrescue (apt install gddrescue) is even better. If dd gets interrupted, there is no way to restart (another good reason to use rsync). When you use ddrescue with a logfile, it keeps track of which blocks have been copied.

When backing up a dual boot Windows/Linux system, I use ntfsclone for the Windows partitions and ddrescue for the Linux partition and dd for the MBR. (I haven't tried to back up a dual boot system using GPT/UEFI.)

What I'd love to see is a ddrescue tool that can create files like ntfsclone where unallocated space is marked with control characters. This makes the image not directly mountable, but allows it to be only as big as the contained data.

Someone please come up with the ntfsclone "special image format" for ddrescue...

Share Improve this answer Follow

answered Jan 27, 2019 at 21:53
ECJB
21 3

0

`rsync` is the perfect solution as explained above.

I'd just add `-S` to "*handle sparse files efficiently*" in case there is a docker devicemapper volume or similar to be copied.

Share Improve this answer Follow

edited Aug 7, 2015 at 22:59
Francisco Tapia
2,654 4 25 43

answered Aug 7, 2015 at 18:04
Jürgen Weigert
111 2

0

Before you start copying large amounts of data, you may wish to remount your new drive with some more efficient parameters.

For example, I just remounted my new SSD like this so that the rsync copying my `/home` to the new drive would go faster:

```
mount -oremount,defaults,noiversion,auto_da_alloc,noatime,errors=remount-
ro,inode_readahead_blks=32,discard /dev/sdc1 /run/media/turgut/ssd1t/
```

Note that this prevents the update of access times, so if you need to re-run the rsync, it might attempt to copy ALL files again. To prevent that, add --ignore-times and --checksum options to your rsync line, so that it only copies the files that have been changed since your first attempt.

Share Improve this answer Follow edited Jan 18, 2023 at 5:20 answered Jan 17, 2023 at 15:28

Turgut Kalfaoglu
31 3