How To Save (Backup or clone) Your ARTIK-520

Let's say, for argument's sake, that you have a Samsung ARTIK-520 (I assume this will work on other ARTIK modules aside from the ARTIK-0 ones) and you have it all configured the way you want, and don't want to lose anything. Or maybe you just want to replicate it a dozen times onto a dozen different ARTIK-520s. As it stands right now, apparently, the answer is "good luck with that" as there is no mechanism to actually dump a running ARTIK-520 to a Mini-SD Card that can then be used to install it again.

Until now. I have such an ARTIK-520, and I don't want to destroy it, but I want to be able to boot it to a different version of the OS to play around with some things, then be able to easily restore it back to this system later. After extensive searching and poking about, I could find no way to do that. So I invented one. Here's how to do it:

First, you will need the original ARTIK-520 image that you started with. Mine was still on the Mini-SD Card I installed from, so it was easy. If you don't have that, just re-download it to your card following the excellent instructions here.

Now, if you just put that card in the reader and look at what you've got:

```
[root@localhost mnt]# fdisk -l
Disk /dev/mmcblk0: 3.7 GiB, 3909091328 bytes, 7634944 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 00042021-0408-4601-9DCC-A8C51255994F

Device          Start     End Sectors  Size Type
/dev/mmcblk0p1   2048   67583   65536   32M Microsoft basic data
/dev/mmcblk0p2  67584  133119   65536   32M Microsoft basic data
/dev/mmcblk0p3 133120 7634910 7501791  3.6G Microsoft basic data
```

```
Disk /dev/mmcblk0boot1: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mmcblk0boot0: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mmcblk1: 7.4 GiB, 7892631552 bytes, 15415296 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xfd34147e

Device         Boot  Start     End Sectors  Size Id Type
/dev/mmcblk1p1        2048   67583   65536   32M 83 Linux
/dev/mmcblk1p2       67584  133119   65536   32M 83 Linux
/dev/mmcblk1p3      133120 1140735 1007616  492M 83 Linux
```

You'll see that last disk entry for

```
/dev/mmcblk1
```

that is partitioned into 3 parts. The last partition is the one you're interested in. Now, the SDCard I am using is an 8 GB card, but you'll notice that if you add up all those partitions, they don't add up to anywhere close to 8GB. That's because they didn't need to. They only needed to be big enough to hold the original OS, and in case you were only using a 1GB card it would still fit. But you're not using a 1GB card, are you? Of course not. So you need to grow the filesystem to hold the new OS you're actually running.

Here's how you grow that partition:

```
[root@localhost mnt]# fdisk /dev/mmcblk1
Welcome to fdisk (util-linux 2.26.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): p
Disk /dev/mmcblk1: 7.4 GiB, 7892631552 bytes, 15415296 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disklabel type: dos
Disk identifier: 0xfd34147e
Device         Boot  Start     End Sectors  Size Id Type
/dev/mmcblk1p1        2048   67583   65536   32M 83 Linux
/dev/mmcblk1p2       67584  133119   65536   32M 83 Linux
/dev/mmcblk1p3      133120 1140735 1007616  492M 83 Linux

Command (m for help): dPartition number (1-3, default 3): 3
Partition 3 has been deleted.

Command (m for help): n
Partition type   p   primary (2 primary, 0 extended, 2 free)   e   extended (contair
Select (default p):
Using default response p.
Partition number (3,4, default 3):
First sector (133120-15415295, default 133120):
Last sector, +sectors or +size{K,M,G,T,P} (133120-15415295, default 15415295):
Created a new partition 3 of type 'Linux' and of size 7.3 GiB.
Command (m for help): p
Disk /dev/mmcblk1: 7.4 GiB, 7892631552 bytes, 15415296 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dosDisk identifier: 0xfd34147e
Device         Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk1p1        2048    67583    65536   32M 83 Linux
/dev/mmcblk1p2       67584   133119    65536   32M 83 Linux
/dev/mmcblk1p3      133120 15415295 15282176  7.3G 83 Linux

Command (m for help): w
The partition table has been altered.Calling ioctl() to re-read partition table.
```

And you've just grown the partition table for that last section to fill the available space. But you're not done, because you also have to grow the file system to match!

Next you need to check the filesystem, then mount it, and then 'grow' it:

```
[root@localhost mnt]# e2fsck -f /dev/mmcblk1p3
e2fsck 1.42.12 (29-Aug-2014)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
rootfs: 20/125952 files (10.0% non-contiguous), 119676/125952 blocks
[root@localhost mnt]# mkdir /mnt/SD3
[root@localhost mnt]# mount /dev/mmcblk1p3 /mnt/SD3
```

```
[root@localhost mnt]# resize2fs /dev/mmcblk1p3
```

Great! You passed! Now you should have a very large 3rd partition on your SSD Card and we can start the process of dumping your running system as a backup.

Next, and I did this so you don't have to, I went through the filesystem and found all the directories you **don't** want to descend into while backing up, and then wrote a long but very dumb script to dump everything into a tar archive, and then gzip compress it (since you can't append to an already compressed archive). And here's an odd thing. GNU Tar says that to **not** add a directory recursively, you should run `tar cf foo.tar foo –no-recursion` but at least on the version of Fedora I'm running, which **claims** to have GNU tar installed, that failed to have any effect. In fact, what I ended up with, as you'll see in the script, is `tar cf foo.tar –no-recursion foo`. Apparently the –no-recursion has to come before the directory you don't want recursion on. Who knew. Run that script, and you'll end up with a new rootfs.tar.gz on your SD Card. This is what will be installed on a new system, thus making your new system a carbon-copy of the old one.

Note: as a precaution, you should rename the existing rootfs.tar.gz file to preserve it. If something goes wrong, you can then move it back into place and recover — though not recover the system you're trying to clone.

The script is available from my GitHub

UPDATE: Since this is now available via GitHub, I added some fancy to the script. It will safely calculate the size of the 3rd partition, and get it right. It will backup the existing rootfs.tar.gz file for you, it will ask you if there are any additional files or directories to include (first ones to include non-recursively, then the recursive includes), etc. It's a lot nicer now. Enjoy!

Here's how big the original was:

```
-rw-r--r-- 1 root root  410873697 Apr 15  2016 rootfs.tar.gz
```

my uncompressed tar file was YUGE by comparison:

```
-rw-r--r-- 1 root root 2893404160 Mar 11 14:06 rootfs.tar
```

And then, even compressed, it was a monster:

```
-rw-r--r-- 1 root root 1521728492 Mar 11 14:06 rootfs.tar.gz
```

So make sure that the SD Card you're using is large enough to hold the **uncompressed** tar file, since you'll need to store it as uncompressed until it's done, then compress it. I likely could have made mine considerably smaller had I removed a bunch of stuff I wasn't actually using, but given that I wanted to test this all out, I decided to make it a rigorous test.

Once you've got your full roots.tar.gz file, you can simply unmount the SD Card, put it in a new ARTIK-520, flip the boot switches, and boot your device. It will then install the cloned system, you shut it off, un-flip the boot switches, pull out the card and reboot and Voilà! You have a cloned system!

Don't be fooled though, this whole process will take you the better part of a day to complete. At least, that's how long it took me. But then again, I had to come up with it, and test it, as well.

ARTIK, ARTIK-520, IoT, Samsung