# Kernel zImage To Fastboot BootImage

By

Regards

Ashish Tanwer

Masters, Computer Science, Stony Brook University

1063 morse avenue, apt 3-200, Sunnyvale, California-94089

[ashishtanwer@gmail.com](mailto:ashishtanwer@gmail.com), +1-[(631) 551-3060](tel:%28631%29%20551-3060)

<http://ashishtanwer.info>, <http://in.linkedin.com/in/ashishtanwer>

<http://www.youtube.com/user/ashishtanwer> <https://github.com/ashishtanwer>

Adopted From XDA-Forum Discussions

**Requirements**:

1. **Linux** (any form will do, for example virtual box etc)

2. **Perl** (already included in Ubuntu, *me thinks*)

3. The [**Xperia**](http://www.ebay.com/sch/i.html?_from=R40&_trksid=p2050601.m570.l1313.TR12.TRC2.A0.H0.Xsony+xperia&_nkw=sony+xperia&_sacat=0) **Boot Tools** package attached in this post

4. The **mkbootimg** binary (I already compiled it for Linux x86 and included it in the attachment)

**Step 1: The mkbootimg binary**

Let's install mkbootimg (we need it in order to repack the boot.img).

In a terminal window, cd to the directory where you extracted the mkbootimg file and type:

Code:

***sudo cp mkbootimg /bin/***

And then

Code:

***sudo chmod 755 /bin/mkbootimg***

Done!

\* After first command, terminal will ask for your user password, type it, press enter and the command will be executed immediately.

**Step 2: Splitting the boot.img**

At this point, we need a boot.img file to play with.

Pick your favorite custom kernel and get the .img.

Place it in a folder along with the 2 perl scripts from the attached package.

In a terminal window, cd to the above-mentioned folder.

Next, type this command:

Code:

***perl split\_bootimg.pl boot.img***

The command will return something like this:

*Page size: 2048 (0x00000800)*

*Kernel size: 3132176 (0x002fcb10)*

*Ramdisk size: 3484496 (0x00352b50)*

*Second size: 0 (0x00000000)*

*Board name:*

*Command line:*

*Writing boot.img-kernel ... complete.*

*Writing boot.img-ramdisk.gz ... complete.*

Now the kernel is successfully extracted to your current directory.

It consists of 2 files: **boot.img-kernel** and **boot.img-ramdisk.gz**.

**Step 3: Unpacking ramdisk**

So we unvealed the ramdisk in our previous step but it is still of no use to us, so we need to un-gzip and then un-cpio it.

In the same terminal window as before, type:

Code:

***mkdir ramdisk***

*Code:*

***cd ramdisk***

*Code:*

***gzip -dc ../boot.img-ramdisk.gz | cpio -i***

Last command will return something like:

*10503 blocks*

Our ramdisk is unpacked now in the newly created /ramdisk folder.

**Step 4: Editing ramdisk files**

Now you can edit the contents of the ramdisk.

BE EXTREMELY CAREFUL AND NEVER EVER DO ANYTHING *"JUST 'CAUSE"*...

You need to be fully aware of what you're doing.

Good news is, changing the logo.rle (the static boot logo image) is fairly easy.

You just need to replace it with your .rle file.

I am not going to explain how to make your own .rle files right now.

Feel free to have a look at this [**thread**](http://forum.xda-developers.com/showthread.php?t=1140406) for more info.

However, I attach the stock SE boot logo in [case](http://viglink.pgpartner.com/rd.php?r=5346&m=1653431694&q=n&rdgt=1419171054&it=1419603054&et=1419775854&priceret=79.99&pg=%7E%7E3&k=8d388fbdea9737af9801de08441c7b3f&source=feed&url=http%3A%2F%2Fcabelas%2Erdr%2Echannelintelligence%2Ecom%2Fgo%2Easp%3FfVhzOC8fBggESSNiW1NTRxJpET92Z1EABEwBb2VbUBdDXSJXNzAjFx0fR0AofAwCCRZNP0wKXEsuayNfXVRFWyt9Iiw7O2gbCQkRGm53HWB4HBlCMCo%2AchASBBd8HAsFAAVrYUEWHwAeLk1LW1EgMAVMQFlRWysbC15fQxZtHXZydV5AU1NZOzdSAAIafmsIe3BrHAQWU0cxMQAHDk4dPVNNXE8qIHwTFGdkFishHQBWMHIbAhQQfUBRb1ldYiIdCggWRixeISY2XxRnZR4lDQIANBpFbwIUCxR7cGAaBxZCXDtvXVM%3D%26nAID%3D5784816&st=feed&mt=%7E%7E%7E%7E%7E%7E%7Eviglink_54785_1635526%7En%7E%7E%7E) anyone misses it when he's on custom kernel. http://cdn3.xda-developers.com/images/smilies/wink.gif

IMPORTANT: Don't accidentally add irrelevant files to the ramdisk directory as cpio will include them too and your new boot.img will be useless.

Also, bear in mind:

Quote:

Originally Posted by **DooMLoRD**

there is a bug in the semc S1 bootloader... sometimes if the boot.img isnt of a correct size then the device will not boot... the workaround is to add a "filler" file (which is say 512KB file with junk data) to the ramdisk so as to increase the size of the final boot.img file...

thanks to jerpelea for this incredible tip... i must have wasted atleast 10-15 hrs trying to figure out why my kernel failed to boot :P

**Step 5: Repack the ramdisk**

Now that's all done, we need to pack the kernel back up into a flashable img file.

First, we will pack the ramdisk back to it's original state.

In the same terminal window as before, type:

Code:

***find . | cpio -o -H newc | gzip > ../newramdisk.cpio.gz***

**Step 6: Repack the boot.img**

In our last step, back into the terminal we go to use the "repack-bootimg" perl script that will give us our brand new boot.img.

Make sure to cd to the directory that the perl scripts are, much like in Step 1.

Type this command:

Code:

***perl repack-bootimg.pl '/.../.../.../boot.img-kernel' '/.../.../.../ramdisk' newboot.img***

Of course, replace /.../.../ with the path to the boot.img-kernel file and the ramdisk directory respectively.

All done! Your new img is newboot.img and is ready to be flashed! http://cdn3.xda-developers.com/images/smilies/smile.gif

**ADVICE**: Use **fastboot boot** and not **~~fastboot flash boot~~** the first time, to test that the new kernel boots. If all is well, proceed with the actual flashing!