[GUIDE]A Noob Guide On Building Your Own Custom Kernel on WIN10 (ARM & ARM64 & MTK)

By *3lambda*, Recognized Contributor on 9th April 2018, 03:08 PM

1,740 posts
Thanks: 1,548

Reply

1 2 3 › Last »

Search Thread

FORUMS

Android Software Development

Android General

Android Q&A, Help & Troubleshooting

Huawei Developers

Android Software and Hacking General [Developers Only]

[More]

1. INTRODUCTION:

This is a guide to build your own custom kernel. Although I'm still a "noob" at this,
I've struggled a lot to build one as all the guides which I followed were not very clear.
So I hope this will be clear enough and as noob friendly as possible!

You will learn how to:

- Build a kernel for arm and mediatek devices on windows 10
- Add feature
- Basic use of git

Prerequise :

- Updated windows 10 64bits (falls creators update)
- A decent computer with a decent internet speed
- Space on your HDD The minimum space for a kernel source (and its compiled code) is about 2 GB

- Minimal linux knowledge (Terminal, Commands etc)
- Your Brain
- And finally patience

2.Setting UP ENVIRONMENT:

Installing ubuntu :

1 - Go in Settings -> Update and Security -> For developers and turn on developers mode then
2 - go in Control Panel > Programs > Turn Windows Features On Or Off and enable Windows subsystem for linux
3 - Reboot your computer
4 - launch linux subsystem now and let it download all it need and set up your password (remember it ! you'll need this password later)
5 - Go in microsoft app store and download Ubunutu by canonical group limited
6 - Open ubuntu (a windows with your name and computer name wil appear), congrats you installed ubuntu on windows 10 !

Seting up you environment :

1 - Type "apt-get update" (will update all repo for apps and dependencies)

From here it is nearly the same as my previous guide, but be careful there is some little changes

2 - Type "sudo apt-get install -y build-essential kernel-package libncurses5-dev bzip2" (will install all dependencies to build kernel)
3 - Check if dependencies are correctly installed :

- Then type "gcc"

If "gcc" is already installed, you should see "gcc : fatal error : no input file"

- Then type "make"

If "make" is already installed, you should see "make: *** no target specified and no makefile found. stop."

- Then type "git"

If "git" is already installed, you should see bunch of basic git commands

Now you're almost ready to start building your kernel!

Toolchains:

There are several types of toolchains (GCC, Linaro and few custom made ones)
Warning : Not every single device kernel will boot (or even compiles) with older or newer GCC

- For ARM:

CLICK TO HIDE CONTENT

We'll be using GCC 4.7 in this tutorial (link : https://android.googlesource.com/pla.../arm-eabi-4.7/)

-Open terminal and type: "mkdir kernel"(Type the name you want, I used "kernel")

-Then type "cd kernel" (the name which you used above)

-Then type "git clone https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/arm/arm-eabi-4.7"

-Wait till it finishes.

- For ARM 64:

CLICK TO HIDE CONTENT

For ARM 64 you need a 64 bit kernel compiler (there's "aarch64" in the name for telling that's a 64 bit compiler)

Exemple : https://releases.linaro.org/archive/...1_linux.tar.xz

3.DOWNLOADING SOURCE FILES FOR YOUR DEVICE:

Now you have to find a github that contains your kernel source.
Search on Google or XDA to find a kernel github repo for your device.

A kernel github looks like this:
"https://github.com/atxoxx/android_ke...4/tree/xenomTW"

On the upper left side you can see branch: completed by a name
These are the different versions of the kernel/project (generally can be for testing, android version etc)

QUICK EXPLANATION OF FILES/FOLDERS:

- /arch/arm/configs : contains the config files for device (where you add option like new governors, features etc)
- /output/arch/arm/boot/ : Where zimage is stored (will explain that later)
- build.sh : Script to make the building much easier (will explain how it works later)
- /arm-cortex-linux-gnueabi-linaro_5.2-2015.11-2 : I put the toolchain in my kernel source making it easier to find (your kernel's toolchain name may be different)

If you don't have your kernel source yet, you need to download it.

Open terminal and make sure that you are in "kernel" folder (the one you previously created)

Then type in terminal : "git clone "URL of the github kernel" -b "name of the branch" "

For Example : "git clone https://github.com/atxoxx/android_ke...amsung_msm8974 -b xenomTW"

Good! Now you have your kernel source!

4.BUILDING:

For an easier way you can go to the location using your file explorer to : "/home/"name of your session"/kernel"

You'll see two folders (The Toolchain and The Kernel Source)

Go into your kernel source folder.

- For ARM:

CLICK TO HIDE CONTENT

Copy paste this:

> Quote:
>
> #!/bin/bash

Fix the Razer Phone 2's camera viewfinder and EIS with this Magisk Module
July 11, 2020

Descendant X custom ROM for the Xiaomi Mi A1 and Mi 9 adds "Guardia," a background permission monitor
July 11, 2020

Unleash the true performance of the Red Magic 5G with this custom kernel
July 9, 2020

How to restore Widevine L1 DRM on the OPPO Find X2 Pro to stream Netflix and Amazon Prime Video in HD
July 7, 2020

/

```
export ARCH=arm
export CROSS_COMPILE=

mkdir output

make -C $(pwd) O=output "name of defconfig and variant if needed"
make -j4 -C $(pwd) O=output
```

Explaination:

> Quote:
>
> - #!/bin/bash: Tells the script to run in shell command
> - export ARCH=arm: Defining which kernel architecture type it is (For example arm64 etc)
> - export CROSS_COMPILE= : Locate where the toolchain is, it has to match the exact path to it and the dash ("-") in the end is really important ! (Almost everyone makes an error at this part!!!)
> - mkdir output: Create a directory for storing compiled zimage
> - make -C $(pwd) O=output : Defining defconfig for guiding kernel compilation (will explain later)
> - make -j4 -C $(pwd) O=output: where the building start, "-j4" is how fast it'll compile, you have to setup this number according to your CPU !
> - cp output/arch/arm/boot/Image $(pwd)/arch/arm/boot/zImage: This one is for moving image into the second path (thanks @Has.007 for this infromation)

Example :

> Quote:
>
> ```
> #!/bin/bash
>
> export ARCH=arm
> export CROSS_COMPILE=$(pwd)/arm-cortex-linux-gnueabi-linaro_5.2-2015.11-2/bin/arm-cortex-linux-gnueabi-
>
> mkdir output
>
> make -C $(pwd) O=output msm8974_sec_defconfig VARIANT_DEFCONFIG=msm8974_sec_ks01_skt_defconfig SELINUX_DEFCONFIG=selinux_defconfig
> make -j4 -C $(pwd) O=output
>
> cp output/arch/arm/boot/Image $(pwd)/arch/arm/boot/zImage
> ```

- For ARM 64:

CLICK TO HIDE CONTENT

Copy paste this:

> Quote:
>
> #!/bin/bash
>
> export ARCH=arm64
> export CROSS_COMPILE="path to your toolchain" (it have to end by something like "nameofarch-something-")
>
> mkdir output
>
> make -C $(pwd) O=output "name of defconfig and variant if needed"
> make -j4 -C $(pwd) O=output

Explaination:

> Quote:
>
> - #!/bin/bash: Tells the script to run in shell command
> - export ARCH=arm64: Defining which kernel architecture type it is (For example arm64 etc)
> - export CROSS_COMPILE= : Locate where the toolchain is, it has to match the exact path to it and the dash ("-") in the end is really important ! (Almost everyone makes an error at this part!!!)
> - mkdir output: Create a directory for storing compiled zimage
> - make -C $(pwd) O=output : Defining defconfig for guiding kernel compilation (will explain later)
> - make -j4 -C $(pwd) O=output: where the building start, "-j4" is how fast it'll compile, you have to setup this number according to your CPU !
> - cp output/arch/arm/boot/Image $(pwd)/arch/arm/boot/zImage: This one is for moving image into the second path (thanks @Has.007 for this infromation)

Example :

> Quote:
>
> #!/bin/bash
>
> export ARCH=arm64
> export CROSS_COMPILE=$(pwd)gcc-linaro-aarch64-linux-

gnu-4.8-2013.07-1_linux\bin\aarch64-linux-gnu-

mkdir output

make -C $(pwd) O=output msm8974_sec_defconfig
VARIANT_DEFCONFIG=msm8974_sec_ks01_skt_defconfig
SELINUX_DEFCONFIG=selinux_defconfig
make -j4 -C $(pwd) O=output

cp output/arch/arm/boot/Image
$(pwd)/arch/arm/boot/zImage

- For Mediatek:

CLICK TO HIDE CONTENT

Copy paste this:

Quote:

#!/bin/bash

export CROSS_COMPILE="path to your toolchain" (it have to end by something like "nameofarch-something-")
export ARCH=arm ARCH_MTK_PLATFORM=
make "name of defconfig and variant if needed"
make -j4

Explaination:

Quote:

- #!/bin/bash: Tells the script to run in shell command
- export CROSS_COMPILE= : Locate where the toolchain is, it has to match the exact path to it and the dash ("-") in the end is really important ! (Almost everyone makes an error at this part!!!)
- export ARCH=arm ARCH_MTK_PLATFORM=: Defining which kernel architecture type it is (For example arm64 etc) "ARCH_MTK_PLATFORM=" is for specifying which mediatek platform it is
- make _defconfig : Defining which defconfig to use (will explain later)
- make -j4: where the building starts, "-j4" is how fast it'll compile, you have to setup this number according to your CPU !

Example :

> Quote:
>
> #!/bin/bash
>
> export CROSS_COMPILE=$(pwd)/arm-eabi-4.8/bin/arm-eabi-
> export ARCH=arm ARCH_MTK_PLATFORM=mt6580
> make pixi4_4_8g1g_defconfig
> make -j4

When these step are done make sure you are in kernel folder in terminal and type "sudo bash build.sh" then type your password you set up in first launch of linux subsytem
(sudo is important, windows 10 ubuntu seems to handle permission differently than native ubuntu)

The compilation have started

If it compiles without any problems:

Wait till it finishes (it'll say something like "zimage is ready")

If you followed arm and arm64:

CLICK TO HIDE CONTENT

> Quote:
>
> Then go to "/Output/arch/arm/boot/" to find your zimage.

If you followed mediatek:

CLICK TO HIDE CONTENT

> Quote:
>
> Then go to "/arch/arm/boot/" to find your zimage.

Caution : Not all kernel build Zimage, it can build image or other compressed image

If in case you have any errors:

Check and see what it says, generally it'll tell you where the error is.

If the text is going too fast reduce the -j number as explained above.

For reference I compile with an AMD Phenom X4 3.4GHz,Samsung HDD and 8GB of RAM and it takes around 10min to build

It is recommanded to type in the terminal "make clean" and "make mrproper" before compiling again

5.MAKING THE KERNEL BOOT:

You have 2 solutions here:

1) You can use @osm0sis anykernel method, which is explainded here: "https://forum.xda-developers.com/sho....php?t=2670512" (A huge shoutout to him!)

OR

2) You can unpack the boot.img (from the same rom (CM, touchwizz,sense etc) and android version) and swap Zimage in it explained here : "https://forum.xda-developers.com/sho....php?t=2073775" (thanks again to @osm0sis !)

Before flashing the kernel which you've made, backup your "stock" boot.img and Then flash your kernel and see if it boots!

6.HOW TO ADD FEATURE TO KERNEL WORK:

Here starts the most interesting part! Now let's see how it works:

Basically you can add: Governors, IO Schedulers, Overclock the CPU & Many Tweaks...
Checkout the github section (Section 7) to see how to add them properly.

Here's an exemple for adding a governor (this one is called Intellimm) : https://github.com/gugu0das/android_...3be392d3186bb1

The text in the blue box is the commit description (generally tells you about the changelog, general information and who originally made the commit)

The other text boxes tell you about where and which files have been modified/changed.

Everything in green indicates what has been added.

Everything in red indicates what has been deleted.

We can see in the first 2 text boxes that in "arch/arm/configs/" "msm8974_sec_defconfig" and "cm_msm8974_sec_defconfig" have been modified.

Between the lines 140 and 141 of this files this text has been added : "CONFIG_CPU_FREQ_GOV_INTELLIMM=y"
(This line is for enabling Intellimm when you're compiling your kernel)

Same technique applies to the other text boxes (what has been added and deleted and it's location)

Depending on the features you add, more or less files can be modified, added or deleted.

So to sum it up, a Commit let's you see all the changes which have been made and everything else!

7.GUIDE TO GITHUB:

For this, I'll direct you over to this awsome guide made by @eagleeyetom !

8.GPL (IMPORTANT !!!):

Quote:

CLICK TO HIDE CONTENT

The Rules as they apply on XDA

As XDA has no legal power to uphold the GPL (and frankly we want to stay as far away from doing so as possible), we can't force any of our users to abide by the GPL. However it is in XDA's interests as well as the interests of our developer-base to ensure all GPL-derived materials hosted or linked on XDA comply fully with the GPL.

GPL-derived materials that do not come with the complete sources used to compile the GPL components are considered warez, and will be treated as such under forum rule 6 and 9.
If you use GPL components, but do not make any modifications to them whatsoever, you should provide a link to the original source of your GPL code.
Sources accompanying a release should be complete, and contain all the necessary source code for any modules, scripts or definition files. Complete sources will be defined as those which compile correctly and completely against the platform for which the software is distributed, and which contain any and all modifications made to the released General Public Licenced code. The source code supplied should be the exact version for which the source code is being requested, complete with all modifications.

EXAMPLE: Here's a bit of code that could be used as a template to post your releases

<Kernel Or Author Name> <Kernel Nr>:
<Source>|<ReadMe>|<Credits>|<Other>

The Very Quick Summary of General Public License (GPL)

The text of the GPL Licence itself will be used to reach any final conclusion regarding any disputes over GPL Licenced materials. The above is a summary of what XDA expects of members using GPL code, and the complete text can be read at the GNU website.

The GPL states that anyone who modifies GPL licenced code is required to make available the sources used to compile it. This is to further improve and encourage collaborative work, as well as to ensure that the best code possible is produced, and to encourage peer-review of all work. This benefits both developers and end users in numerous ways, including:

Allowing anyone to verify the code they are trusting with their data, and its authenticity
Encouraging community collaboration to produce faster fixes and updates, and better code
Helping bring new developments from other devices and fields to your own, letting you benefit from new code that wouldn't have been available without this sharing.
The GPL imparts great freedom for GPL end users. It ensures innovation is never stifled and no project is dependent upon any single developer.

It is in everyone's interest for the GPL to be adhered to, as it gives us all better ROMs, better transparency, and a better atmosphere for developers to work together to make great code.

THANKS :

- @ravish_919 : For testing and correcting this guide :P
- @karkasss : As my friend and support
- @gugu0das : For helping me a lot when I tried to build my kernel
- @eagleeyetom : For his awsome github guide
- @osm0sis For his aswsome anykernel solution
- @kirito9 : Huge thanks to him for providing mediatek guide !
- @F4uzan : Huge thanks to him for giving me a lot of useful information to fill this guide !
- @sunilpaulmathew : For providing an advanced method to rename your kernel ! (again)
- @nathanchance : For a proper kernel naming method
- @RendyAK and @DroidThug : For correcting me about "#!/bin/bash"
- @ahmed.ismael : For helping me, giving feedback and his huge support !
- Microsoft and canonical for the windows linux subsystem documentation
- All the developers for their hard work !
- XDA and The Community!

3lambda
OP Recognized Contributor

Thanks: 1,548

9th April 2018, 03:09 PM | #2

TIPS AND TRICKS

1. You can use a copy of a defconfig file with different setup :

Usage : Use a "stock" one and use another one with esperimental feature for testing without altering original defconfig

Exemple : copy "stock" defconfig and in copied one add a governor see if it compile and work

How to do : Create a second build.sh with modified defconfig name !

2. Change kernel name and version :

Simple method :

Edit this line "CONFIG_LOCALVERSION="-" after - in your defconfig

Exemple : CONFIG_LOCALVERSION="-XenomTW-3.2.6"

Advanced methods :

Method 1 :

> Quote:
>
> 1. Go in Makefile in the root folder of your kernel source
> 2. Add
>
> > Quote:
> >
> > CONFIG_LOCALVERSION="nameofyourkernel"
> > LOCALVERSION="versionofyourkernel"
>
> Exemple :
>
> > Quote:
> >
> > VERSION = 4
> > PATCHLEVEL = 4
> > SUBLEVEL = 127
> > EXTRAVERSION =
> >
> > CONFIG_LOCALVERSION="-FlashKernel"
> >
> > export LOCALVERSION="-v1.00"
>
> Caution ! Never touch or edit VERSION, PATCHLEVEL, SUBLEVEL, and EXTRAVERSION !

Method 2 :

> Quote:
>
> 1. Go in "scripts/mkcompile_h"
> 2. Add
> LINUX_COMPILE_BY="nameofyourchoice"
> LINUX_COMPILE_HOST="nameofyourchoice"
>
> Exemple

3. Solve problem with PATH :

If you encounter "IS YOUR PATH CORRECT" problem try in terminal :

"export PATH="pathtotoolchainlocation"/bin:$PATH"

Exemple : export PATH=/home/3lambda/kernel/M8_Kernel/arm-eabi-4.7/bin:$PATH

4. Access ubuntu folders :

Path location to ubuntu folder is :
C:\Users"NAME"\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_79rhkp1fndgsc\LocalState \rootfs\home

Caution ! Editing files here directly from windows may break permission, you'll have to fix them if so (look on google on how to)

More to come...

The Following 16 Users Say Thank You to 3lambda For This Useful Post: [View]

Gift 3lambda Ad-Free

3lambda
OP Recognized Contributor

Thanks: 1,548

9th April 2018, 03:12 PM | #3

Edit and update are coming, I may have forgot things let me know

Feed back also appreciated

The Following 10 Users Say Thank You to 3lambda For This Useful Post: [View]

Gift 3lambda Ad-Free

ahmed.ismael
Senior Member

Thanks: 1,269

9th April 2018, 03:30 PM | #4

would result in the following output:

Code:

```
4.4.127-FlashKernel-v1.00
```

The Following 6 Users Say Thank You to nathanchance For This Useful Post: [View]

3lambda
OP Recognized Contributor

Thanks: 1,548

9th April 2018, 08:07 PM | #6

Quote:
Originally Posted by **nathanchance**

Good guide although I would say the advanced method for changing the kernel name is making totally unnecessary edits to the source code. There is already a framework in place for configuring the version string however you want. Editing EXTRAVERSION could result in conflicts during stable merges. The version gets generated in the following order:

VERSION, PATCHLEVEL, SUBLEVEL, and EXTRAVERSION are set in the main Makefile, the first three should never be touched. CONFIG_LOCALVERSION and LOCALVERSION should be what the user sets.

For example:

would result in the following output:

Thanks for pointing this out
I'll take a look when I'll have time

The Following 3 Users Say Thank You to 3lambda For This Useful Post: [View]

CrazyBenjy9
Senior Member

Thanks: 86

9th April 2018, 09:01 PM | #7

Nice! Been looking for something like this. XDA feed brought me here.

highly appreciated
thank you

The Following User Says Thank You to ahmed.ismael For This Useful Post: [View]

Gift ahmed.ismael Ad-Free

nathanchance
Senior Recognized Developer / Recogn...

Thanks: 50,864

9th April 2018, 07:59 PM | #5

Good guide although I would say the advanced method for changing the kernel name is making totally unnecessary edits to the source code. There is already a framework in place for configuring the version string however you want. Editing EXTRAVERSION could result in conflicts during stable merges. The version gets generated in the following order:

Code:

```
$(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)$(CON
```

VERSION, PATCHLEVEL, SUBLEVEL, and EXTRAVERSION are set in the main Makefile, the first three should never be touched. CONFIG_LOCALVERSION and LOCALVERSION should be what the user sets.

For example:

Code:

```
VERSION = 4
PATCHLEVEL = 4
SUBLEVEL = 127
EXTRAVERSION =

CONFIG_LOCALVERSION="-FlashKernel"
```

The Following User Says Thank You to CrazyBenjy9 For This Useful Post: [View]

Gift CrazyBenjy9 Ad-Free

tobarreh
Junior Member

Thanks: 0

10th April 2018, 01:25 AM | #8

is it necessary do in win10? or you can simply do it in linux pc without WIN10?

thanks for the post!

ahmed.ismael
Senior Member

Thanks: 1,269

10th April 2018, 03:37 AM | #9

> Quote:
> Originally Posted by **tobarreh**
>
> is it necessary do in win10? or you can simply do it in linux pc without WIN10?
>
> thanks for the post!

it should be done on linux
but windows can handle linux as a subsystem now so the guide for people like me who is dumb enough to use windows

seriously skip the ubuntu installation steps on sindows and start building your kernel by following the other steps 👆

Sent from my OnePlus 5 using XDA Labs

The Following 2 Users Say Thank You to ahmed.ismael For This Useful Post: [View]

Gift ahmed.ismael Ad-Free

sunilpaulmathew
Recognized Developer

Thanks: 5,577

10th April 2018, 06:16 AM | **#10**

Quote:
Originally Posted by **nathanchance**

Good guide although I would say the advanced method for changing the kernel name is making totally unnecessary edits to the source code. There is already a framework in place for configuring the version string however you want. Editing EXTRAVERSION could result in conflicts during stable merges. The version gets generated in the following order:

Code:

```
$(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)$(CON
```

VERSION, PATCHLEVEL, SUBLEVEL, and EXTRAVERSION are set in the main Makefile, the first three should never be touched. CONFIG_LOCALVERSION and LOCALVERSION should be what the user sets.

For example:

Code:

```
VERSION = 4
PATCHLEVEL = 4
SUBLEVEL = 127
EXTRAVERSION =

CONFIG_LOCALVERSION="-FlashKernel"

export LOCALVERSION="-v1.00"
```

would result in the following output:

Code:

```
4.4.127-FlashKernel-v1.00
```

I personally like to define "LINUX_COMPILE_BY" and "LINUX_COMPILE_HOST" in "scripts/mkcompile_h" just like in this commit

by @franciscofranco. By adding this

Code:

```
LINUX_COMPILE_BY="francisco"
LINUX_COMPILE_HOST="franco"
```

would display "francisco@franco"

> Quote:
> Originally Posted by **3lambda**
>
> Thanks for pointing this out
> I'll take a look when I'll have time

The Following 3 Users Say Thank You to sunilpaulmathew For This Useful Post: [View]

3lambda
OP Recognized Contributor

Thanks: 1,548

10th April 2018, 06:56 AM | **#11**

Added new kernel naming method by @nathanchance and @sunilpaulmathew, huge thanks to them !
I'll need feedback see if some of these steps aren't clear or if I forgot things
I may also add a video soon and maybe screenshots
Stay tuned

The Following User Says Thank You to 3lambda For This Useful Post: [View]

Gift 3lambda Ad-Free

Reply | Subscribe to Thread

Guest Quick Reply (no urls or BBcode)

Message:

Post Quick Reply | Go Advanced

Previous Thread | Next Thread

Top Threads in Android Software Development by ThreadRank

OnePlus Parallel App Installer — 17th May 2020
Android Software Development

[APP][5.0+]Android VoIP Call Recording - Record calls from WhatsApp, Vibe... — 16th March 2020
Android Apps and Games

[App][9.0+] Wavelet: Headphone specific equalization — 11th May 2020
Android Apps and Games

[APP][5.0+] Volume Styles - Customize your volume panel (iOS, MIUI, OneUI ... — 4th April 2020
Android Apps and Games

Gms vs hms — 20th March 2020
Huawei Developers

[ROM][unlocked][aquaris_m8] LineageOS 14.1 [9 MAR 2020] — 9th March 2020
Android Software Development

[LWP][7.0+] Metropolis 3D City Live Wallpaper (similar to Google Pixel & Xia... — 1st May 2020
Android Apps and Games

[COLLECTION] Swiftkey Themes port — 21st February 2020
Android Themes