1. **Prepare source and compiler tool**
2. Prepare toolchain

### 1.1 Download toolchain

**Vậy Toolchain cho Linux chung là gì?**

[Cross Compiler](https://en.wikipedia.org/wiki/Cross_compiler) hay còn gọi là Toolchain có thể được hiểu là một source code được viết trên máy tính chạy trên chip Intel, sau khi thông qua một cross compiler sẽ cho ra file nhị phân có khả năng chạy được trên một nền tảng chip khác là ARM. Một ví dụ cơ bản nhất là mình đã dùng một máy tính hệ điều hành Ubuntu để build ra một file [image có thể chạy trên Raspberry Pi](https://hocarm.org/huong-dan-tu-build-distro-cho-raspberrypi-voi-yocto/)

Nó ít nhất bao gồm những thành phần sau:

* binutils (công cụ xử lý liên quan đến mã nhị phân): GNU Assembler, Linker, etc.
* gcc: GNU C Compiler
* C library (libc): gồm các file header lẫn các file binary cho phép ứng dụng giao tiếp với hệ điều hành.
* gdb: Debugger

Nói chung, có 2 loại Toolchain là Native và Cross chia theo mối quan hệ giữa môi trường nó chạy và môi trường nó sinh mã nhị phân cho:

* Native : tức là chạy trên máy nào, sinh mã nhị phân chạy cho máy ấy luôn  
  Ví dụ các bản phân phối ta vẫn dùng có thể coi như sử dụng *Native Toolchain*
* Cross: tức là chạy trên một máy, nhưng sinh mã nhị phân cho máy khác.

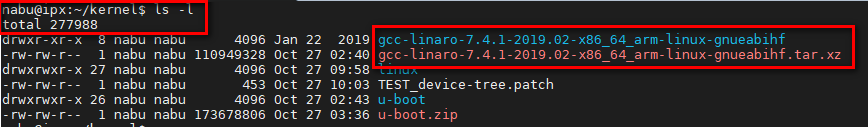
Khi phát triển Embedded Linux, hầu hết chúng ta sẽ sử dụng Cross Toolchain.  
Vì các thiết bị chạy thường tốc độ thấp, bộ nhớ ít nên không thích hợp để build source trên đó hay chạy Native Tool trên đó.

|  |
| --- |
| [gcc-linaro-7.4.1-2019.02-x86\_64\_arm-linux-gnueabihf.tar.xz](https://releases.linaro.org/components/toolchain/binaries/latest-7/arm-linux-gnueabihf/gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabihf.tar.xz) |
| <https://releases.linaro.org/components/toolchain/binaries/latest-7/arm-linux-gnueabihf/?fbclid=IwAR09Nz4m-aEgngNvM0rqLL0Xq-p2Pji7H__0wtoDoDpHI7OPbqAJD_nlOUA> |

### Setup environment variable

#### A, After downloaded, extra that file to a folder by “tar” command

|  |
| --- |
| tar xvf gcc-linaro-arm-linux-gnueabihf-4.8-2014.04\_linux.tar.xz –C ./ |



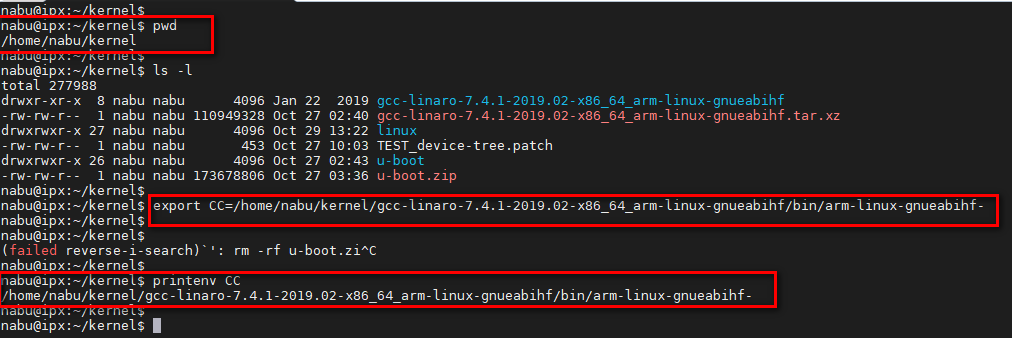
#### B, Setup a environment variable

|  |
| --- |
| export CC=./gcc-linaro-7.4.1-2019.02-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- |

With $CC is environment variable. That dependency your decision (example: BB, asbd, kjd, blabla)

To test environment variable:

|  |
| --- |
| printenv [name of environment variable] |



Hardcode link to toolchain and save it to environment variable.

### Install support tool to build

Mkimage:

|  |
| --- |
| sudo apt install u-boot-tools |

Lzod:

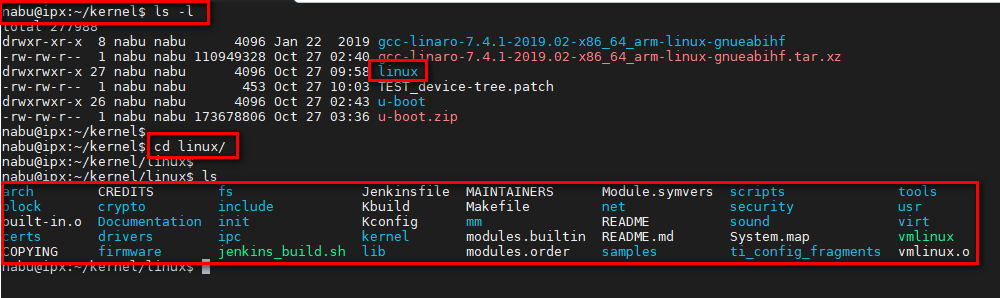
|  |
| --- |
| sudo apt-get install lzop |

1. Prepare kernel source

### 2.1 Download kernel source for beaglebone black

Git clone to download attach git file to easy for modify and commit

|  |
| --- |
| git clone https://github.com/beagleboard/linux.git |

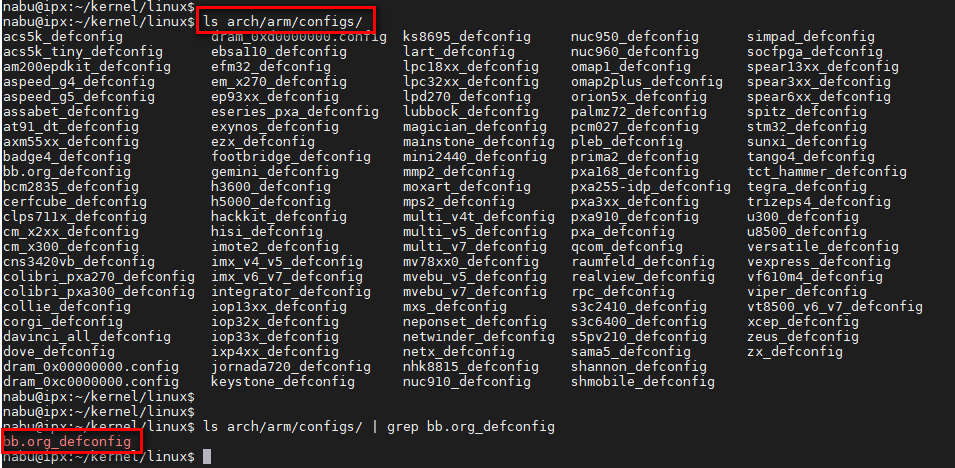


### Compile Linux kernel

#### A, Make defconfig to create a .config file

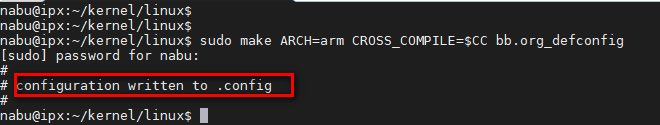
The defconfig file put in:

|  |
| --- |
| arch/arm/configs/bb.org\_defconfig |



|  |
| --- |
| sudo make ARCH=arm CROSS\_COMPILE=$CC bb.org\_defconfig |

With $CC is environment variable. That dependency your decision above

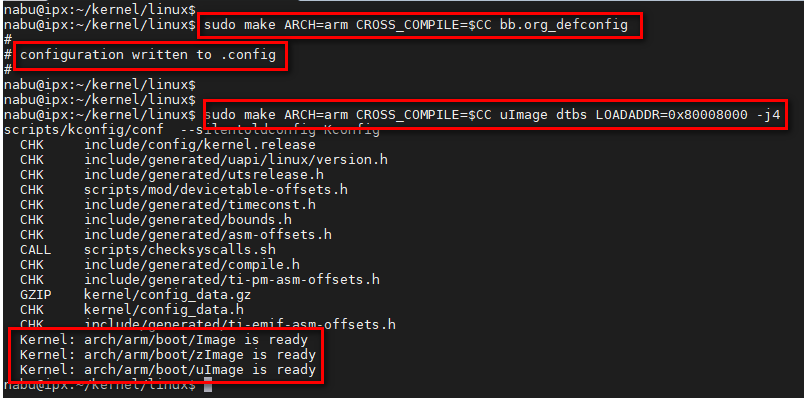


#### B, Compile the kernel.

|  |
| --- |
| sudo make ARCH=arm CROSS\_COMPILE=$CC uImage dtbs LOADADDR=0x80008000 -j4 |

With $CC is environment variable. That dependency your decision above. “-j4” is using how many processor for build. That dependency your PC. Largest number build will be faster.

The first times will take 30-40 minutes. But in second times that will be faster (1-2 minutes).



#### C, Result

Image:

|  |
| --- |
| arch/arm/boot/uImage |

Device tree:

|  |
| --- |
| arch/arm/boot/dts/am335x-boneblack.dtb |

Create a environment file setup: uEnv.txt

|  |
| --- |
| console=ttyS0,115200n8  netargs=setenv bootargs console=ttyO0,115200n8 root=/dev/mmcblk0p2 ro rootfstype=ext4 rootwait debug earlyprintk mem=512M  netboot=echo Booting from microSD ...; setenv autoload no ; load mmc 0:1 ${loadaddr} uImage ; load mmc 0:1 ${fdtaddr} am335x-boneblack.dtb ; run netargs ; bootm ${loadaddr} - ${fdtaddr}  uenvcmd=run netboot |

1. **Install to board**

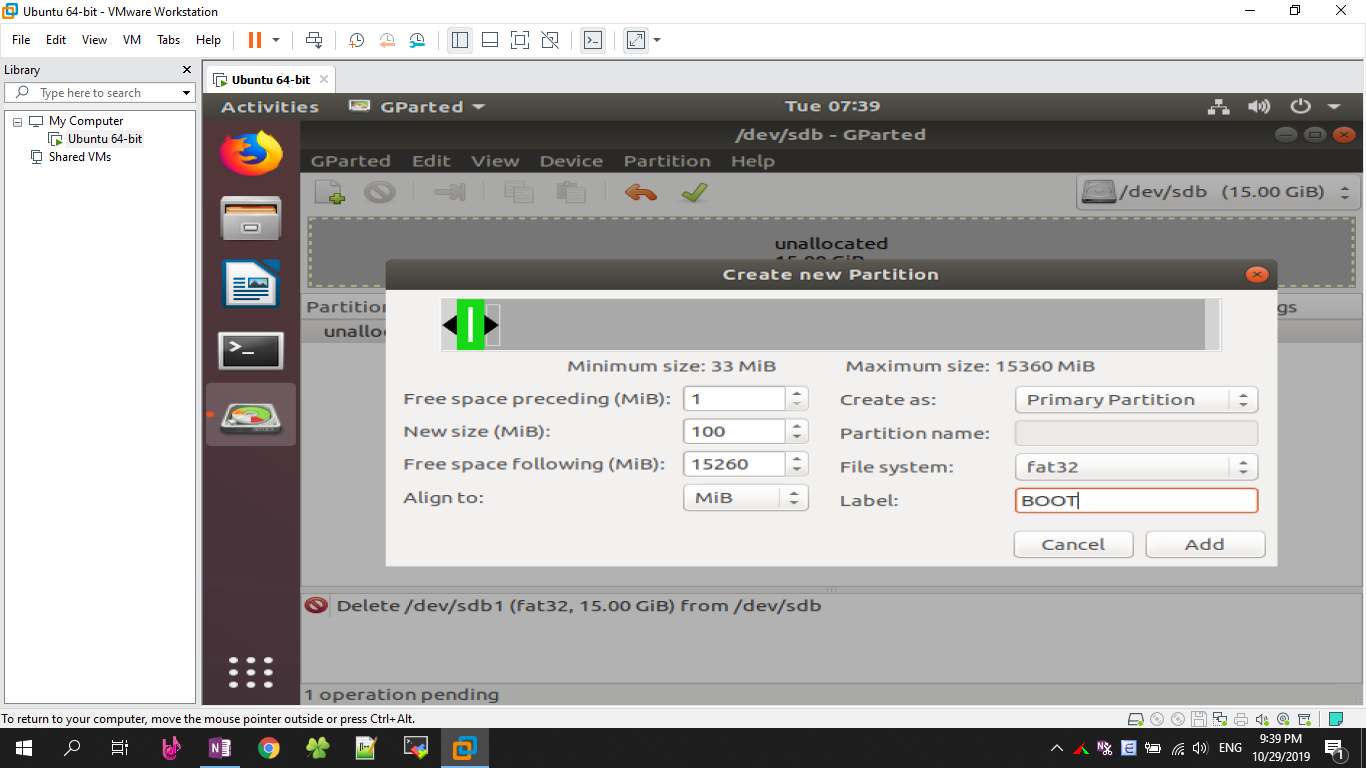
## 1. Create a patition for Boot and rootfs in SDcard

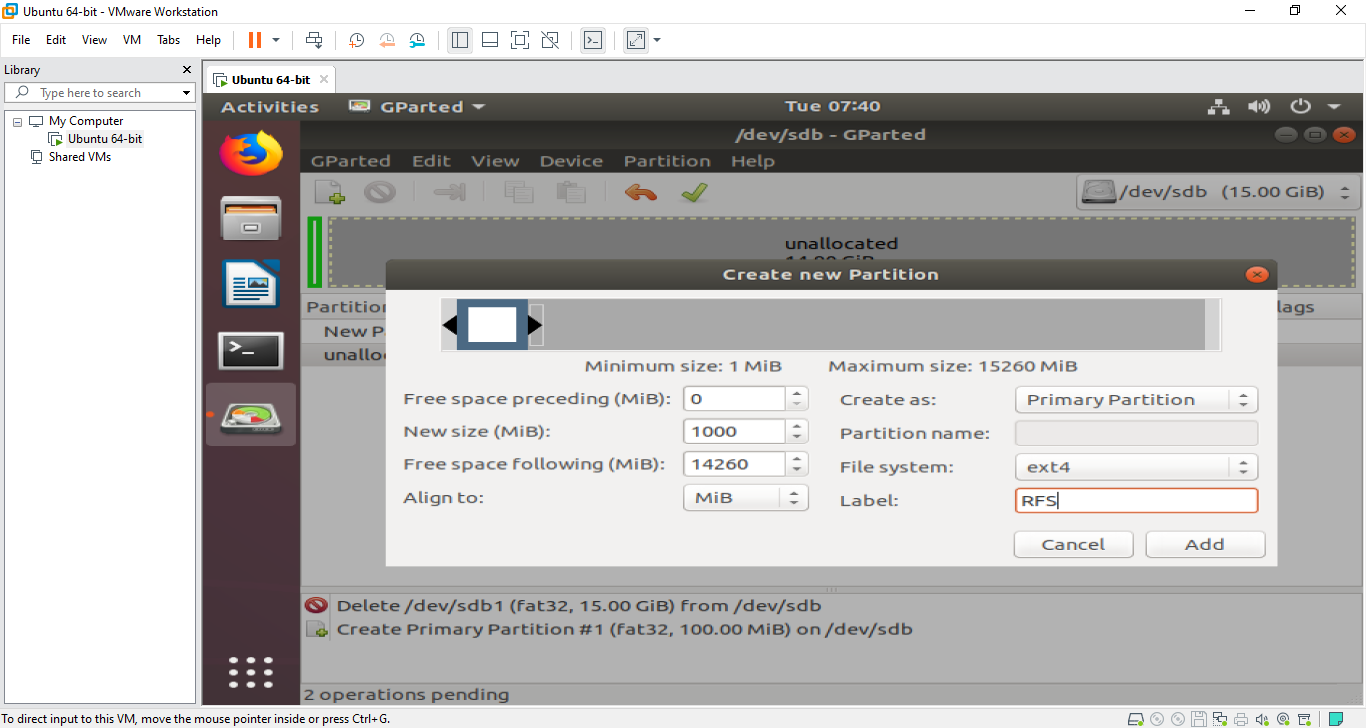
### 1.1 Install “Gparted” tool

|  |
| --- |
| sudo apt-get install gparted |

### 1.2 Create 2 patition

|  |
| --- |
| * Label: BOOT   +) New size: 50MBembed journal  +) File System: FAT32   * Label: RFS   +) New Size: 1000MB  +) File System: EXT3 |

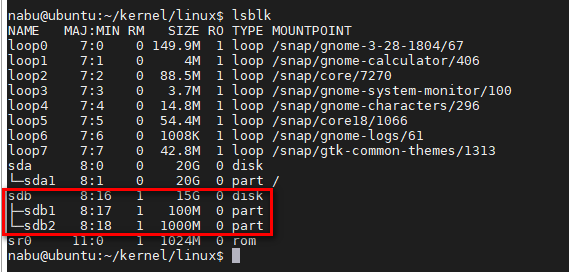




* 1. Mount to add file

Show all storage device:

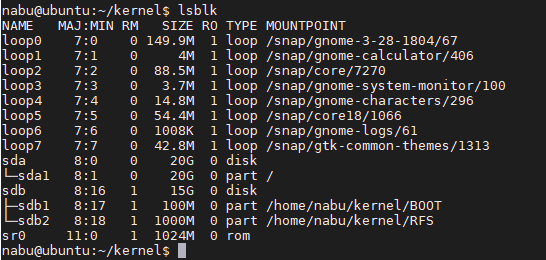
|  |
| --- |
| lsblk |



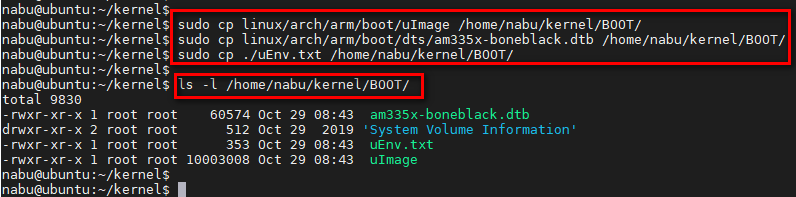
Mount to actual file to modify:

|  |
| --- |
| sudo mount /dev/sdb1 [patch]  sudo mount /dev/sdb2 [patch] |



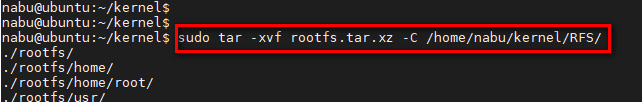


Now, copy 3 files: uImage, am335x-boneblack.dtb and uEnv.txt to mount file boot



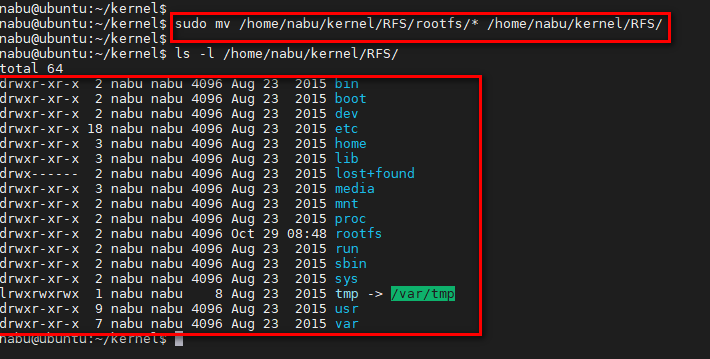
Copy rootfs file to mount file rootfs

|  |
| --- |
| sudo tar -xvf rootfs.tar.xz –C [patch mounted RFS] |



Move all file in rootfs to RFS file

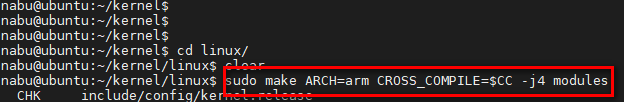
|  |
| --- |
| sudo mv /home/nabu/kernel/RFS/rootfs/\* /home/nabu/kernel/RFS/ |



## Install Kernel Modules

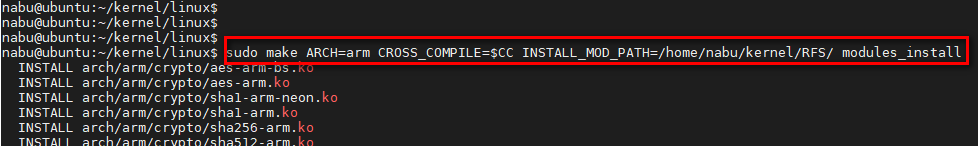
Make kernel module

|  |
| --- |
| sudo make ARCH=arm CROSS\_COMPILE=$CC -j4 modules |



Install kernel modules to rootfs

|  |
| --- |
| sudo make ARCH=arm CROSS\_COMPILE=$CC INSTALL\_MOD\_PATH=[patch] modules\_install |



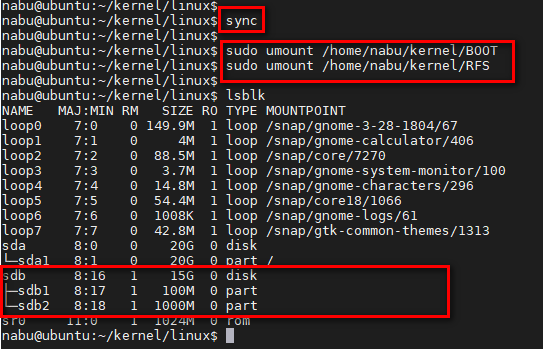
* 1. Unmount file

Sync to write data to file

|  |
| --- |
| sync |

Unmount file

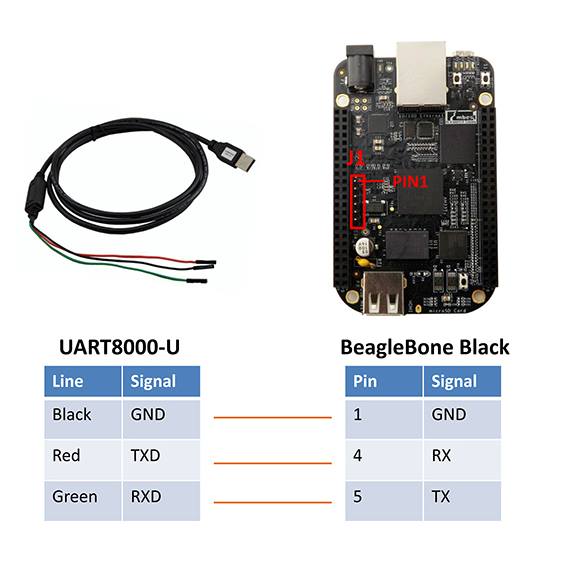
|  |
| --- |
| sudo umount [patch mount file] |



1. **Install uart to screen console**
2. Setup driver for PL2303

|  |
| --- |
| https://hocarm.org/huong-dan-cai-driver-pl2303-voi-win10/?fbclid=IwAR1rjKc5F3bWPvoch8RmNuoH1fmPzfCKX-7e\_tVLvCS7V5fcQ-6Hdgg9s1w |

1. Connect with board



1. **CROSS\_COMPILE**

Example:

|  |
| --- |
| CROSS=/home/nabu/kernel/gcc-linaro-7.4.1-2019.02-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-  PWD := $(shell pwd)  KERNEL=/home/nabu/kernel/linux  obj-m += led\_device\_tree.o  all:  make ARCH=arm CROSS\_COMPILE=$(CROSS) -C $(KERNEL) SUBDIRS=$(PWD) modules  clean:  make -C $(KERNEL) SUBDIRS=$(PWD) clean |

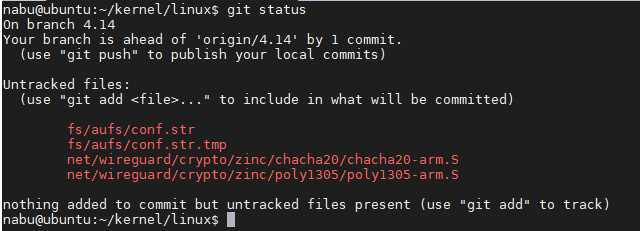
1. **How to use git**
2. Config user name and email

|  |
| --- |
| git config --global user.name "John Doe"  git config --global user.email johndoe@example.com |

1. Git status

To see what file you modified?

|  |
| --- |
| git status |



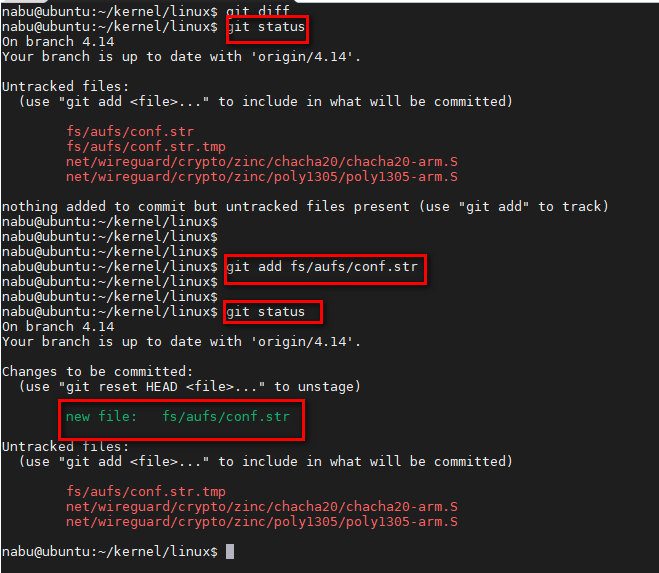
Red is file you modified and not commit.

1. Git add

To add any file you modified to prepare for commit

|  |
| --- |
| git add [file] |

File you added will change to green



1. Git commit

To commit to a commit in log with what file you added.

|  |
| --- |
| git commit –m “notice”  Or  git commit –s (that way will be professional) |

1. Git diff

To show what different from current code and nearest commit

|  |
| --- |
| git diff |

You can save it to a file to can apply in future

|  |
| --- |
| git diff > test.patch |

1. Git am / git apply

To apply a new changed of somebody in type is diff file (.patch)

|  |
| --- |
| git am [link to patch file] |