1. Tổng quan về Yocto

Các thiết bị như Raspberry Pi, BeagleBone Black hay NXP i.MX thường bị giới hạn về bộ nhớ RAM, dung lượng lưu trữ và tài nguyên xử lý. Chúng không đủ mạnh để chạy các hệ điều hành lớn như Ubuntu. Do đó, cần một hệ điều hành nhẹ, phù hợp, được tinh chỉnh theo nhu cầu.

Để tạo được một hệ điều hành cho thiết bị nhúng thủ công gần như là "ác mộng":

- Cấu hình kernel bằng menuconfig hoặc chỉnh sửa .config thủ công
- Cấu hình bootloader(như u-boot) để thiết lập được môi trường load kernel.
- Thiết lập cross compiler và toolchain phù hợp với kiến trúc CPU và tài nguyên hệ thống.
- Thu thập source code từ nhiều nơi: kernel.org, github, busybox.net...
- Tự cross compile từng thành phần và đảm bảo mọi thứ tương thích với nhau
- Tạo root filesystem thủ công, tạo thư mục vào copy từng file vào /bin, /etc, /lib, /sbin ...
- Tạo file image và flash vào sd card hoặc eMMC

Tất cả các bước trên đòi hỏi kiến thức chuyên sâu, nhiều thời gian làm việc, và rất dễ sai sót nên việc maintain cực kỳ khó khăn dù chỉ là thay đổi nhỏ.

Vì vậy, Yocto ra đời để giải quyết tất cả những rắc rối trên bằng cách cung cấp một bộ công cụ hoàn chỉnh để:

- Tự động hóa quá trình build hệ điều hành từ A-Z.
- Tùy chỉnh mọi thứ từ kernel, filesystem, packages, đến startup script.
- Tái sử dụng các thành phần nhờ hệ thống layer.
- Dễ cập nhật, dễ bảo trì và tích hợp với CI/CD nếu cần.

2. Build image cho BeagleBone Black(BBB) bằng Yocto

Yêu cầu cấu hình tối thiểu để build image cho BeagleBone Black (BBB):

- Ổ cứng: Cần ít nhất 100GB dung lượng trống. Nên dùng SSD để tăng tốc độ build.
- RAM: Nên có trên 8GB. Nếu máy chỉ có 8GB RAM hoặc ít hơn, bạn nên tạo thêm swap để tránh lỗi trong quá trình build.

Thực hiện các commands sau để tạo swap 16GB:

```
sudo dd if=/dev/zero of=/swapfile bs=1M count=16384
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
sudo swapon --show # kiểm tra
```

2.1. Cài đặt môi trường và download Poky

Chúng ta sẽ sử dụng Ubuntu 20.04 làm hệ điều hành để chạy Yocto. Đầu tiênn chúng ta cài đặt các gói phần mềm cần thiết bằng commands sau để chuẩn bị cho môi trường build

```
sudo apt update
sudo apt install gawk wget git-core diffstat unzip texinfo gcc-multilib \
  build-essential chrpath socat cpio python3 python3-pip python3-pexpect \
  xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa \
  libsdl1.2-dev pylint xterm
```

Tiếp theo, chúng ta clone Poky bằng commands sau:

```
mkdir -p yocto
cd yocto
git clone git://git.yoctoproject.org/poky
git checkout -t origin/kirkstone -b my-kirkstone # kirkstone is LTS version
```

Yocto có nhiều phiên bản, nhưng chúng ta nên chọn bản LTS (Long Term Support) vì:

- Ổn định hơn
- Được cập nhật lâu dài
- Dễ tìm tài liệu và cách fix lỗi

2.2. Khởi tạo tạo môi trường build và cấu hình

Để build ra được image bước đầu tiên là thiết lập môi trường để build bằng command sau:

```
$ source poky/oe-init-build-env
```

Sau khi chạy lệnh ở trên sẽ thiết lập và tạo ra thư mục build mặc định, đồng thời Yocto sẽ in ra màn hình với các sự lựa chọn phổ biến để build image

```
...
You can now run 'bitbake <target>'

Common targets are:
    core-image-minimal
    core-image-full-cmdline
```

```
    core-image-sato
    core-image-weston
...
```

Tiếp theo, chúng ta cấu hình hệ thống bằng cách chỉnh sửa file conf/local.conf

```
### vim conf/local.conf
# cấu hình machine
MACHINE ?= "beaglebone-yocto"
# cấu hình số lượng thread sử dụng để build
BB_NUMBER_THREADS = "12"
PARALLEL_MAKE = "-j12"

# enable ssh server
IMAGE_INSTALL:append = " openssh"
```

Chi tiết về một số images:

- core-image-minimal: Đây là image với cấu hình nhẹ, đủ để khởi động thiết bị. Nó không bao gồm giao diện GUI hoặc bất cứ tiện ích nào. Phù hợp cho người bắt đầu để học tập hoặc các dự án cơ bản sử dụng các board cấu hình khiêm tốn.
- core-image-full-cmdline: Một bản image chỉ có giao diện dòng lệnh console-only nhưng được cài đặt thêm các chức năng hệ thống Linux đầy đủ hơn.
- core-image-sato: Image này đầy đủ hơn và có cả giao diện GUI, được thiết kế cho các thiết bị di động và nhúng. Đây là sự lựa chọn lý tưởng cho những ai sử dụng có giao diện tương tác nhưng vẫn đảm bảo tính nhẹ và tối ưu.
- core-image-weston: là một image cơ bản cho Wayland, chỉ có terminal và các thư viện cần thiết cho Wayland và Weston, giúp quản lý giao diện. Để tìm hiểu thêm, bạn có thể tham khảo tài liệu của Yocto Project.

Ở đây chúng ta sẽ build với option core-image-full-cmdline với bằng command:

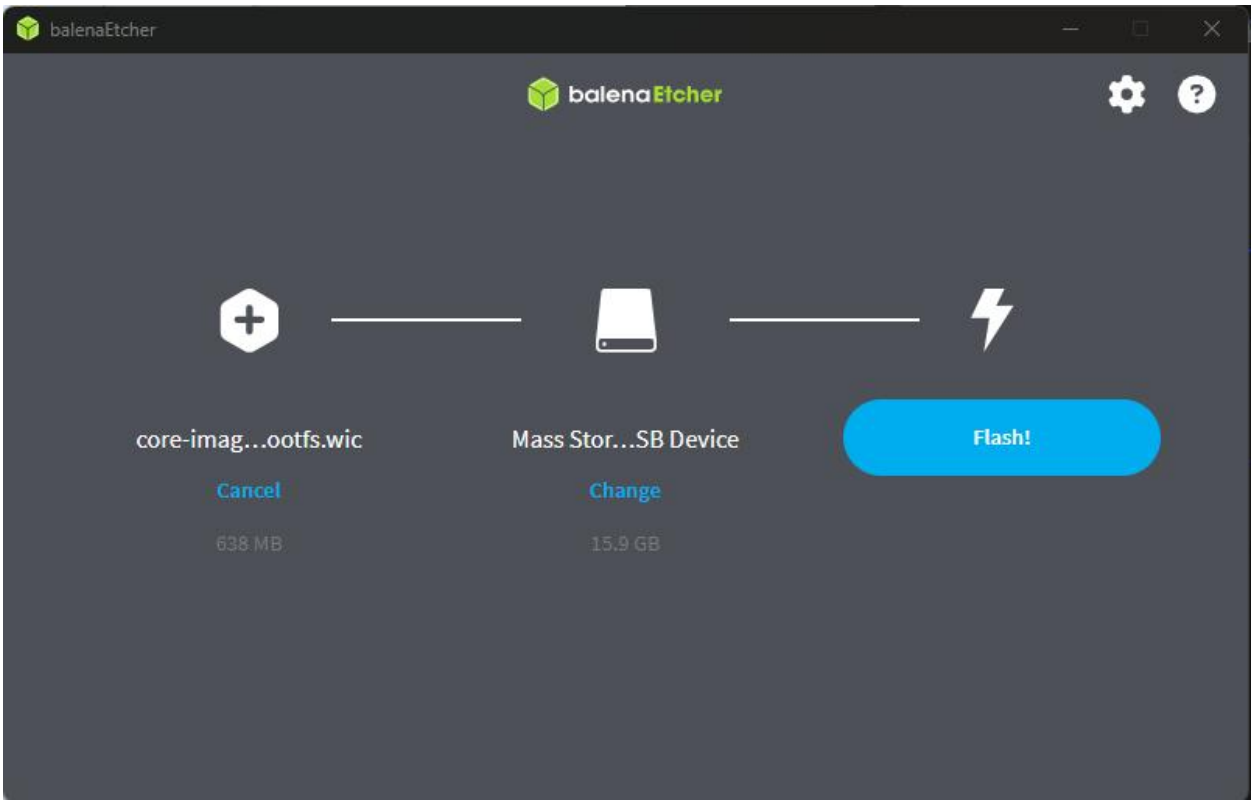
```
$ bitbake core-image-full-cmdline
```

2.3. Kiểm tra kết quả build và flash image

Sau khi build xong, output files sẽ ở trong folder `tmp/deploy/images/beaglebone-yocto`:

- .ext4, .wic: Hệ thống file
- .dtb: Device Tree
- .uImage, .zImage: Kernel
- .tar.bz2, .hddimg,...

Chúng ta sẽ sử dụng file `core-image-full-cmdline-beaglebone-yocto.wic` với balenaEtcher để flash image vào sdcard.



3. Cross-compile chương trình cho BeagleBone Black(BBB) bằng Yocto

Trong quá trình phát triển phần mềm cho các thiết bị nhúng chạy Linux, việc cross compile là một bước quan trọng giúp tăng tốc độ phát triển và kiểm thử ứng dụng trên PC trước khi triển khai lên thiết bị thật. Để hỗ trợ điều này, Yocto Project cung cấp khả năng tạo ra bộ SDK (Software Development Kit) phù hợp cho thiết bị Linux Embedded

3.1. Build SDK cho BBB và install SDK

Việc tạo SDK được thực hiện thông qua lệnh sau:

```
$ bitbake -c populate_sdk core-image-full-cmdline
```

Sau khi quá trình build hoàn tất, một bash script cài đặt SDK sẽ được tạo ra trong thư mục:

tmp/deploy/sdk/poky-glibc-x86_64-core-image-full-cmdline-cortexa8hf-neon-beaglebone-yocto-toolchain-4.0.26.sh

Chạy script trên để thực hiện cài đặt, lúc này script sẽ yêu cầu chọn thư mục install:

- Nếu bạn đồng ý với thư mục mặc định(/opt/poky/4.0.26), chỉ cần nhấn Enter hai lần.
- Nếu muốn cài đặt vào vị trí tùy chỉnh, bạn có thể nhập đường dẫn, ví dụ: /opt/workspace/bbb/qt-sdk. Sau đó nhấn Enter để tiếp tục và hoàn tất quá trình cài đặt.

Trước khi cross-compile cho BBB chúng ta cần thiết lập môi trường bằng cách chạy

```
$ source /opt/poky/4.0.26/environment-setup-cortexa8hf-neon-poky-linux-gnueabi
```

Lệnh này sẽ export các biến môi trường cần thiết như:

- CC, CXX, LD (trỏ tới trình biên dịch cross-compile)
- PKG_CONFIG_PATH, CFLAGS, LDFLAGS, v.v…

3.2. Kiểm tra SDK

Để kiểm tra SDK đã được thiết lập đúng, thử biên dịch một file đơn giản:

```
// helloworld.cc
#include <iostream>
int main() {
    std::cout << "Hello from SDK!" << std::endl;
    return 0;
}
```

Thực hiện biên dịch bằng command sau:

```
$CXX -o hello hello.cc
```

Chúng ta có thể copy chương trình hello vào BBB và thực hiện chạy nó