**BUILD KERNEL LINUX WITH YOCTO PROJECT FOR RASPBERRY PI 4 IN UBUNTU**

*Giới thiệu*

Raspberry pi là một board development rất phổ biến cho các dự án IoT và Công nghiệp. Nó có nhiều phiên bản khác nhau như Raspberry pi 5, Raspberry Pi 4 và Raspberry 3, ... Raspberry chạy trên hệ điều hành mã nguồn mở là Linux và nhà sản xuất có cung cấp một công cụ gọi là “Raspberry Pi Image” giúp người dùng dễ dàng cài đặt hệ điều hành và phần mềm cho các thiết bị Raspberry Pi được thiết kế để giúp người dùng dễ dàng cài đặt hệ điều hành và phần mềm cho các thiết bị Raspberry Pi. Tuy nhiên trong bài viết này sẽ không xử dụng công cụ đó mà sẽ sự dụng một công cụ khác đó là Yocto. Mục đích là để ta có thể tự xây dựng một kernel linux tùy chỉnh theo mục đích và điều kiện của mình.

**1) Yocto Project**

Yocto Project là một dự án mã nguồn mở được tạo ra để hỗ trợ việc phát triển và tạo ra các hệ điều hành tùy chỉnh dành cho các thiết bị nhúng và nhúng vào hệ thống. Dự án này cung cấp một khung làm việc linh hoạt và mạnh mẽ cho việc xây dựng các hệ điều hành nhúng theo yêu cầu.

Các điểm chính của Yocto Project bao gồm:

1. **BitBake**: BitBake là một công cụ xây dựng và tự động hóa quy trình xây dựng hệ điều hành. Nó sử dụng các tệp mô tả kiến trúc (recipes) để mô tả cách xây dựng và cấu hình các thành phần của hệ điều hành.
2. **OpenEmbedded Core (OE-Core)**: OE-Core là một tập hợp các công cụ, các tệp mô tả kiến trúc và các lớp mở rộng được sử dụng để xây dựng hệ điều hành nhúng. Yocto Project sử dụng OE-Core như là cơ sở để xây dựng các hệ điều hành.
3. **Metadata Layers**: Yocto Project sử dụng các "metadata layers" để tổ chức và quản lý các tệp mô tả kiến trúc, cấu hình và mã nguồn. Điều này cho phép người dùng mở rộng và tùy chỉnh các thành phần của hệ điều hành một cách linh hoạt.
4. **Cross-Compilation**: Yocto Project hỗ trợ việc biên dịch chéo (cross-compilation), cho phép bạn phát triển và xây dựng các ứng dụng và hệ điều hành trên một máy tính chạy một kiến trúc x86 hoặc x86-64, nhưng nhắm mục tiêu là chạy trên các thiết bị với kiến trúc khác như ARM.
5. **Tích hợp công cụ và thư viện**: Yocto Project cung cấp một loạt các công cụ và thư viện tiêu chuẩn, cũng như khả năng tích hợp các công cụ và thư viện bổ sung theo nhu cầu của dự án.

Với Yocto Project, người dùng có thể tạo ra các hệ điều hành nhúng tuỳ chỉnh, tối ưu hóa cho các yêu cầu cụ thể của họ, từ các thiết bị nhúng nhỏ gọn đến các hệ thống nhúng phức tạp.

**2) Build kernel**

**2.1) Yêu cầu hệ thống**

* Dung lượng ổ đĩa trống ít nhất là 90 Gbyte, tuy nhiên nhiều hơn nữa sẽ giúp chạy nhiều bản dựng và tăng hiệu suất bằng cách sử dụng lại các tạo phẩm bản dựng. Dung lượng ổ đĩa trống ít nhất là 90 Gbyte, tuy nhiên nhiều hơn nữa sẽ giúp chạy nhiều bản dựng và tăng hiệu suất bằng cách sử dụng lại các tạo phẩm bản dựng.
* Ít nhất 8 Gbyte RAM, mặc dù vậy, bạn nên sử dụng máy chủ xây dựng hiện đại có càng nhiều RAM và càng nhiều lõi CPU càng tốt để tối đa hóa hiệu suất xây dựng.
* Ubuntu 20.04 LTS
  + Git 1.8.3.1 trở lên
  + tar 1.28 trở lên
  + Python 3.8.0 trở lên
  + Python 3.8.0 trở lên
  + GNU make 4.0 trở lên

**2.2) Cài đặt các gói cần thiết**

Mở Terminal và từ giờ trở về sau ta sẽ dùng những command để làm việc. Sử dụng các command sau để cài đặt các gói cần thiết.

$ sudo apt install gawk wget git diffstat unzip texinfo gcc build-essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev python3-subunit mesa-common-dev zstd liblz4-tool file locales libacl1

$ sudo locale-gen en\_US.UTF-8

$ sudo apt install git make inkscape texlive-latex-extra

$ sudo apt install sphinx python3-saneyaml python3-sphinx-rtd-theme

**2.3) Lấy các lớp meta (Getting the meta layers)**

Đầu tiên ta cần tạo một thư mục **yocto** để làm việc và di chuyển tới thư mục đó bằng câu lệnh sau

$ mkdir yocto && cd yocto

* Tải xuống Poky

Poky là bản phân phối tham chiếu Yocto. Nó chứa tất cả các lớp meta cơ bản, trong bài viết này ta sử dụng phiên bản Nanbield của poky ( có thể tham khảo các bản khác tại link sau: <https://wiki.yoctoproject.org/wiki/Releases>)

$ git clone git://git.yoctoproject.org/poky -b nanbield

* Tải xuống lớp meta Raspberry pi

Bạn sẽ cần một lớp BSP để hỗ trợ các bo mạch Raspberry Pi. vì vậy Dự án Yocto đang cung cấp lớp meta-raspberrypi chứa thông tin liên quan đến bảng pi mâm xôi được yêu cầu trong quá trình xây dựng.

$ cd poky

$ git clone git://git.yoctoproject.org/meta-raspberrypi -b nanbield

* Tải xuống meta openembedded

Các lớp meta luôn nêu rõ chúng phụ thuộc vào lớp meta nào. Readme của meta-raspberrypi nói rằng chúng ta cần meta-openembedded. Bản thân siêu lớp này được chia thành nhiều lớp. Chúng ta nhận được tất cả bằng command này.

$ git clone https://git.openembedded.org/meta-openembedded -b nanbield

**2.4) Cài đặt môi trường**

Khởi tạo tập lệnh oe-init-build-env bằng command bên dưới và nó sẽ tự động tạo thư mục build.

$ source oe-init-build-env

**2.5) Đặt tên máy trong local.conf và thêm lớp Raspberry Pi trong bblayer.conf**

* Local.conf

Tệp local.conf trong thư mục conf được sử dụng cho cấu hình người dùng, quan trọng nhất đối với chúng ta là tên của máy cần xây dựng. Nếu chúng ta muốn xây dựng cho Raspberry Pi 4, chúng ta sẽ sử dụng raspberrypi4, cho Raspberry Pi 3 thì sử dụng raspberrypi3

$ nano conf/local.conf

[...]

MACHINE ?= "raspberrypi4" #thêm dòng này vào trong file local.conf, có thể sử dụng nhiều trình soạn thảo khác nhau như nano hoặc vi hoặc text editor đều được

LICENSE\_FLAGS\_ACCEPTED = "synaptics-killswitch"

ENABLE\_UART = "1"

INIT\_MANAGER = "systemd"

# thêm SSH và WiFI

IMAGE\_INSTALL:append = " openssh-sftp-server sudo python3 python3-pip rpi-gpio raspi-gpio"

IMAGE\_FEATURES:append = " ssh-server-openssh"

MACHINE\_FEATURES:append = "wifi"

#

IMAGE\_FSTYPES = "wic.bz2 rpi-sdimg"

# For TeamViewer Embedded Agent

DISTRO\_FEATURES:append = " X11"

DISTRO\_FEATURES:append = "wayland"

CORE\_IMAGE\_EXTRA\_INSTALL += "wayland weston"

[...]

* bblayers.conf

Bây giờ, chúng ta phải thêm lớp meta-raspberrypi trong tệp build/conf/bblayers.conf.

Tệp bblayers.conf trong thư mục conf chứa tất cả đường dẫn đến các lớp meta mà chúng ta sẽ sử dụng. Lưu ý xxx là username của máy bạn

/home/xxx/yocto/poky/meta-raspberrypi \

/home/xxx/yocto/poky/meta-openembedded/meta-oe \

/home/xxx/yocto/poky/meta-openembedded/meta-multimedia \

/home/xxx/yocto/poky/meta-openembedded/meta-networking \

/home/xxx/yocto/poky/meta-openembedded/meta-python \

A screenshot of a computer

Description automatically generated

**2.6) Chạy bitbake để build file imager**

Sau khi hoàn thành tất cả các bước trên, đã đến lúc bắt đầu quá trình xây dựng thực sự. Taget core-minimal-image cung cấp file imager có khả năng khởi động tối thiểu cho bo mạch Raspberry pi. Ngoài ra còn nhiều targets khác như core-image-full-cmdline, core-image-sato, core-image-weston, meta-toolchain, meta-ide-support và bạn có thể tham khảo chức năng của những tagets này trên internet.

Ở đây ta sẽ xây dựng bản kernel có GUI (giao diện người dùng) là core-image-weston

Bitbake sẽ mất vài giờ để hoàn tất quá trình xây dựng. nó phụ thuộc vào tốc độ internet và hiệu suất CPU của bạn.

$ bitbake core-image-weston

Nếu không có lỗi màn hình sẽ có dạng như hình dưới đây

A screenshot of a computer program

Description automatically generated

**2.7) Flash SD card**

File sau khi build: core-image-weston-raspberrypi4-64.rootfs.wic.bz2nằm ở thưc mục : /home/xxx/yocto/poky/build/tmp/deploy/images/raspberrypi4-64

Di chuyển tới mục này và giải nén file

$ cd /home/xxx/yocto/poky/build/tmp/deploy/images/raspberrypi4-64

$ bzip2 -d -f core-image-weston-raspberrypi4-64.rootfs.wic.bz2

Cắm thẻ nhớ SD card vào máy và nạp file image đã tạo

$ sudo dd bs=4M if=core-image-weston-raspberrypi4-64.rootfs.wic of=/dev/sdb status=progress conv=fsync

The end