Report QEMU

Ở đây sử dụng buildroot:

• *Buildroot* là một tập các *Makefile* và các bản vá (patches) để nó có thể dễ dàng tạo ra một cross-compilation toolchain và hệ thống tập tin gốc (root filesystem) cho hệ thống Linux đích mà chúng ta muốn xây dựng. Cross-compilation toolchain sử dụng *uClibc*, một thư viện C rất nhỏ.

• Mục đích chủ yếu của *Buildroot* là được sử dụng để làm việc với các hệ thống nhỏ hoặc các hệ thống nhúng. Các hệ thống nhúng thường không sử dụng những bộ vi xử lí x86 như trong các máy PC bình thường, các bộ vi xử lí hay được sử dụng là PowerPC, MIPS, ARM,...

Một compilation toolchain là tập hợp các công cụ cho phép chúng ta có thể biên dịch mã nguồn của hệ thống muốn xây dựng. Nó gồm có một trình biên dịch (gcc), trình liên kết (binutils) và thư viện C (GNU Libc, uClibc hoặc dietlibc). Trong các hệ thống Linux được cài đặt thường có một compilation toolchain, chúng ta có thể sử dụng toolchain này để biên dịch các ứng dụng chạy trên hệ thống đang được dùng. Nếu đang sử dụng một máy PC có hệ điều hành Linux thì hệ thống này sẽ có một toolchain chạy trên bộ vi xử lí x86 và nó sẽ sinh ra mã cho bộ vi xử lí x86. Hầu hết các hệ thống Linux, thư viện C chuẩn được toolchain sử dụng là GNU libc. Compilation toolchain này được gọi là “compilation toolchain chủ” và hệ thống đang chạy, được sử dụng để làm việc được gọi là “hệ thống chủ”.

Như đã đề cập ở trên, toolchain chạy trên hệ thống chủ có bộ vi xử lí nào thì nó sẽ sinh ra mã cho bộ vi xử lí của hệ thống chủ đó. Vì thế, khi xây dựng hệ thống nhúng có bộ vi xử lí khác, chúng ta cần một cross-compilation toolchain: nó là compilation toolchain chạy trên hệ thống chủ nhưng phát sinh mã dùng cho hệ thống đích (là hệ thống có bộ vi xử lí mà chúng ta muốn xây dựng).

Ví dụ, nếu hệ thống chủ sử dụng x86 và hệ thống đích sử dụng bộ vi xử lí ARM, compilation toolchain bình thường của hệ thống chủ chạy trên x86 và phát sinh mã cho x86 trong khi cross-compilation toolchain chạy trên x86 nhưng phát sinh mã cho ARM.

Ngay cả khi hệ thống đích mà chúng ta muốn xây dựng sử dụng bộ vi xử lí x86 thì ta cũng nên sử dụng *Buildroot* vì hai lí do sau:

• Compilation toolchain của hệ thống chủ sử dụng GNU Libc nhưng thư viện C này quá lớn. Thay vì sử dụng GNU Libc trên hệ thống đích, chúng ta có thể sử dụng một thư viện C rất nhỏ là uClibc. Để sử dụng thư viện C này, chúng ta cần một cross-compilation toolchain để tạo ra những liên kết nhị phân với nó. Và *Buildroot* sẽ giúp chúng ta thực hiện công việc này.

• Buildroot sẽ tự động xây dựng một root filesystem chứa tất cả các công cụ cần thiết giống như *Busybox*. Thực hiện công việc này bằng *Buildroot* sẽ dễ dàng hơn rất nhiều so với khi chúng ta thực hiện bằng tay.

Chúng ta có thể biên dịch *gcc, binutils, uClibc* và các công cụ khác bằng tay nhưng việc xử lí các tùy chọn cấu hình, các vấn đề liên quan đến phiên bản của *gcc* hoặc *binutils* sẽ tốn rất nhiều thời gian và khá phức tạp. Vì thế, việc sử dụng *Buildroot* sẽ giúp chúng ta giải quyết được các vấn đề này.  
  
*Buildroot* sẽ thi hành quá trình này một cách tự động bằng việc sử dụng các *Makefile*, các bản vá của mỗi phiên bản *gcc* và *binutils* để làm cho chúng có thể hoạt động trên hầu hết các kiểu kiến trúc vi xử lí.

Trong buildroot có 1 số thành phần giúp ta chạy thiết lập 1 OS ( Operating System) Hệ Điều Hành

• Makefile : Chứa sẵn các script để build các OS với các config khác nhau

• Config.in : Chứa các option giúp ta config một các dễ dàng

• .config : Giống như file log của hệ thống, các option cài đặt được lưu lại vào đây

Diagram

Description automatically generated

Sau khi build, ta có được đầy đủ các thành phần cơ bản của 1 Hệ điều hành:

• Kernel

• Root filesystem

• Bootloader

• Các thành phần phụ trợ khác ( toolchain)

Cách cài QEMU ( Quick Emulator):

<https://www.qemu.org/download/#linux>

sudo apt-get install qemu qemu-system-arm

Dowload buildroot at:

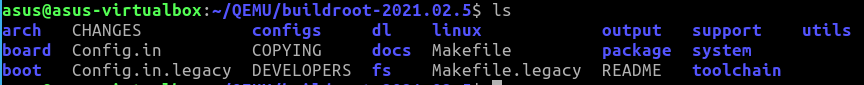
<https://buildroot.org/download.html>

tar -xvzf buildroot-2021.02.5.tar.gz

or :

git clone git://git.buildroot.net/buildroot





First we need to create a configuration file suitable to our board:

+ create a new configuration file suitable to our board for this we have to configure kernel cross compiler and root file system.

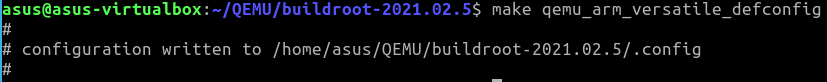
+ make list-defconfigs to check available configuration type:



Or 

A picture containing text, plaque

Description automatically generated



Then command make to build configuration

Type make to start building:

• Dowload source files (as required)

• Configure, build and install the cross-compilation toolchain, or simply import an external toolchain

• Configure, build and install selected target packages

• Build a kernel image

• Build a bootloader image

• Create a root filesystem

Then make complete, the buildroot output is stored in a single directory, output/.







Host side: output/host



+ Cross compiler name: “arm-buildroot-linux-uclibcgnueabi-gcc”

+ Cross Compiler binaries: output/host/bin

Graphical user interface, text, application, chat or text message

Description automatically generated

+ Sysroot: output/host/arm-buildroot-linux-uclibcgnueabi/sysroot



Target side: output/images



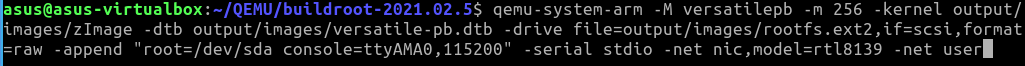
+ Root file system: rootfs.ext2

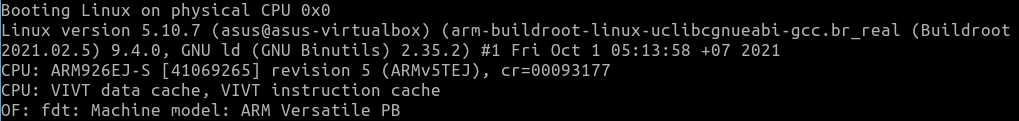
+ Device Tree: Versatile-pb.dtb

+ Kernel: zImage



Booting linux in Qemu:





A screenshot of a computer

Description automatically generated with medium confidence

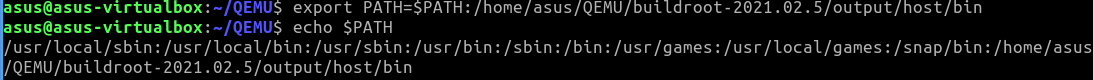
To use toolchain for target device:

+ Cross compiler is located in “output/host/bin”

+ Cross compiler name is: “ arm-buildroot-linux-uclibgnueabi-gcc”

+ Set enviroment variable path to “output/host/bin”:

export PATH=$PATH:/home/asus/QEMU/buildroot-2021.02.5/output/host/bin



Text

Description automatically generated

Build a C program:

Text

Description automatically generated



To connect VM with Qemu:

+ Mount command to mount root file system to try directory

+ *ls try* and here is the all directories of root file system. Copy file “hello” into root directory in try directory.

Graphical user interface, text

Description automatically generated

Then let umount try

Graphical user interface, text

Description automatically generated

Booting again

Text

Description automatically generated