# 一、YOCTO基础知识

## 1.可参考已有工程

RISCV平头哥gitee地址：https://gitee.com/thead-yocto
参考瑞芯微的meta-rockchip: https://github.com/rockchip-linux/meta-rockchip
参考赛灵思的meta-petalinux: https://github.com/Xilinx/meta-petalinux
参考angstrom的meta-angstrom: https://github.com/Angstrom-distribution/meta-angstrom

## 2.yocto常用调试命令

bitbake常用命令：bitbake -c clean
bitbake -e 显示当前的执行环境，常用于查找当前bitbake的包的源路径和目标路径。
查找包的原路径 bitbake -e hello | grep ^SRC_URI
查找包的安装路径 bitbake -e hello | grep ^S=
bitbake -s 用于显示所有可以bitbake的包
例如如果自己在一个Layer下面安装了一个hello.bb,在build目录下面，
用 bitbake -s | grep hello 查看hello这个package能否被bitbake.
bitbake -c 用于执行一个特定的命令
bitbake -g 用于显示一个包在bitbake的时候于其他包的依赖关系
bitbake -v 显示执行过程。
bitbake -b 后面加上.bb文件的路径，可以用bitbake直接执行这个.bb文件。

# 二、编译环境构建

编译环境构建参考：https://gitee.com/thead-yocto/Misc/blob/master/riscv%20yocto%20%E4%BB%93%E5%BA%93%E6%9E%84%E5%BB%BA%E8%AF%B4%E6%98%8E.md

## 1.安装repo

```
下载：
curl https://mirrors.tuna.tsinghua.edu.cn/git/git-repo -o repo
chmod +x repo
更新：
vim ~/.bashrc
```

```
export REPO_URL='https://mirrors.tuna.tsinghua.edu.cn/git/git-repo'
```

## 2.拉取代码：

```
mkdir riscv_yocto && cd riscv_yocto
repo init -u git@gitee.com:thead-yocto/manifests.git -m riscv-yocto.xml
repo sync -c -d -j16
```

## 3.安装工具链：

```
# 将百度云下载的 host.gcc-8.1.0.tar.gz 解压到 ~/.thead
mkdir ~/.thead
tar zxf host.gcc-8.1.0.tar.gz -C ~/.thead
# 也可随意放置工具链，然后修改 meta-riscv/conf/machine/include/thead-base.inc
```

## 4.打入润和给的path

```
root@ubuntu:/home/run/code/riscv_yocto_new# repo status

... A new version of repo (2.11) is available.
... You should upgrade soon:
    cp /home/run/code/riscv_yocto_new/.repo/repo/repo /root/bin/repo

project meta-riscv/                             (*** NO BRANCH ***)
 -m     conf/machine/include/thead-base.inc
project meta-openembedded/                      (*** NO BRANCH ***)
 --     meta-xfce/recipes-core/images/core-image-xfce.bb
 --     meta-xfce/recipes-core/images/core-image-xfce.bb_ok （备份）
project openembedded-core/                      (*** NO BRANCH ***)
 -m     meta/classes/sanity.bbclass
 -m     meta/recipes-devtools/apt/apt_1.8.2.1.bb
 --     meta/recipes-devtools/apt/apt_1.8.2.1.bb_adduser_apt （备份）
 --     meta/recipes-devtools/gcc/gcc-10.2.inc_1        (暂不修改)
 --     meta/recipes-extended/man-db/man-db_2.9.3.bb_1 (暂不修改)
project thead-build/                            (*** NO BRANCH ***)
 --     ice-base/conf/bblayers.conf_1   (不用修改)
 -m     ice-xfce/conf/local.conf
```

```
制作成自动打patch的脚本：
vim do_patch.sh
 cp ./thead-base.inc ../meta-riscv/conf/machine/include/
 cp ./core-image-xfce.bb ../meta-openembedded/meta-xfce/recipes-core/images/
 cp ./apt_1.8.2.1.bb ../openembedded-core/meta/recipes-devtools/apt/
 cp ./local.conf ../thead-build/ice-xfce/conf/
 cp ./man-db_2.9.3.bb_1 ../openembedded-core/meta/recipes-extended/man-db/man-db_2.9.3.bb
 cp ./gcc-10.2.inc_1 ../openembedded-core/meta/recipes-devtools/gcc/gcc-10.2.inc
```

## 5.编译ice-xfce

```
# 在 yocto 工程根目录 riscv_yocto/ 下执行：
source openembedded-core/oe-init-build-env thead-build/ice-base # 将 build 目录指定为 thead-build/ice-base

# 编译：thead-build/ice-base/conf/auto.conf 中默认设置为 MACHINE ?= "ice"
bitbake core-image-minimal

# sstate-cache/ 会放在 thead-build/ 下，不同配置可以共享使用
# 镜像文件
./build/tmp-glibc/deploy/images/ice
```

设置不编译某个包：`IMAGE_INSTALL_remove += "chromium webkitgtk"`

## 6. 烧录镜像

```
# **三、映像烧录**
1、单板进入`u-boot cmd`模式，设置环境变量：
setenv ipaddr 192.168.1.110 ## BOARD IP
setenv serverip 192.168.1.114 ## PC IP
setenv gatewayip 192.168.1.1

输入`fastboot`命令，进入烧录模式
fastboot udp

 2、PC
```

```
在`cmd`里输入：`fastboot -s udp:192.168.1.110 flash root debian-rootfs-buildroot.ext4`

烧录完成后，重启单板即可。

参考：ice_evb_book_20210110_v0.1.pdf
```

# 三、指定编译生成deb包的方法

## 1.修改方法

vim riscv_yocto/build/conf/local.conf +74

```
# Package Management configuration
#
# This variable lists which packaging formats to enable. Multiple package backends
# can be enabled at once and the first item listed in the variable will be used
# to generate the root filesystems.
# Options are:
#  - 'package_deb' for debian style deb files
#  - 'package_ipk' for ipk files are used by opkg (a debian style embedded package manager)
#  - 'package_rpm' for rpm style packages
# E.g.: PACKAGE_CLASSES ?= "package_rpm package_deb package_ipk"
# We default to ipk:
PACKAGE_CLASSES ?= "package_ipk"
```

将 PACKAGE_CLASSES ?= "package_ipk"

改成 PACKAGE_CLASSES ?= "package_deb" 即可配置生成deb包。

编译生成的deb包存放位置：build/tmp-glibc/deploy/deb/qemuriscv64/

## 2.相关文档

官方文档：The package_ipk class uses the DEPLOY_DIR_IPK variable to make sure the do_package_write_ipk task writes IPK packages into the appropriate folder. For more information on how packaging works, see the "Package Feeds" section in the Yocto Project Overview and Concepts Manual. The BitBake configuration file initially defines this variable as a sub-folder of DEPLOY_DIR:

vim riscv_yocto/openembedded-core/meta/conf/bitbake.conf +409

DEPLOY_DIR_IPK = "${DEPLOY_DIR}/ipk"

DEPLOY_DIR_RPM = "${DEPLOY_DIR}/rpm"

DEPLOY_DIR_DEB = "${DEPLOY_DIR}/deb"