Yocto Project: conceptos básicos y flujo de trabajo

¿Qué es Yocto Project?

Yocto Project es un **framework de código abierto** que permite crear **distribuciones Linux a la medida para sistemas embebidos**. presenta un conjunto de herramientas que permiten crear imágenes de linux personalizadas a la medida de cada necesidad, incluyendo sólo lo que se necesita para la aplicación y el hardware específico. Esto permite ahorrar espacio, mejorar el rendimiento y tener control total sobre el sistema. dentro de sus características principales esta la portabilidad dado que se pueden generar imágenes para diferentes arquitecturas como x86, ARM,RISC entre otras, se pueden agregar solo las librerías, drivers y paquetes que se necesitan,tamse tiene control total de la configuración sel sistema operativo y sus capas se pueden combinar según las necesidades Dentro de los conceptos que se deben tener claros a la horta de trabajar se tienen:

- Portabilidad: puede generar imágenes para diferentes arquitecturas (x86, ARM, RISC-V, etc.).
- Modularidad: puedes agregar solo las librerías, drivers y paquetes que necesitas.
- Control total de la configuración del sistema operativo.
- Reutilización: se basa en capas (layers) que puedes combinar según tus necesidades.
- **BitBake:** Herramienta que interpreta recetas y construye paquetes e imágenes.
- **Recetas (.bb):** Archivos que indican cómo descargar, compilar e instalar un paquete.
- **Capas (layers):** Agrupan recetas y configuraciones relacionadas, facilitando la modularidad.
- **Imagen (image):** Sistema Linux completo generado por Yocto.
- **Target:** Hardware o arquitectura para la que se construye la imagen (por ejemplo, x86).

El flujo de trabajo para yocto project se puede definir en los siguientes 7 puntos:

1. **Preparar el entorno:** Instalar dependencias del host necesarias para ejecutar Yocto.
2. **Descargar Yocto y capas necesarias:** Por ejemplo, la capa base `poky` y `meta-openembedded` para paquetes adicionales.
3. **Configurar la build:** Inicializar el entorno y agregar las capas que se van a usar.
4. **Elegir imagen base:** Generalmente se parte de una imagen mínima (`core-image-minimal`).
5. **Agregar dependencias:** Modificar o crear recetas para incluir librerías necesarias, como GStreamer o OpenCV.
6. **Compilar la imagen:** Ejecutar `bitbake <imagen>` para generar la imagen final.
7. **Probar la imagen:** Cargarla en VirtualBox o QEMU y verificar que todo funcione correctamente.

pasos par la instalación de yocto project

1 instalar los paquetes necesarios.

$sudo apt-get update && sudo apt-get install -y \

build-essential chrpath cpio debianutils diffstat file gawk gcc git iputils-ping \

libacl1 liblz4-tool locales python3 python3-git python3-jinja2 python3-pexpect python3-pip \

python3-subunit socat texinfo unzip wget xz-utils zstd curl rsync ccache bc libssl-dev \

flex bison gperf libncurses5-dev libncursesw5-dev python3-distutils python3-cryptography \

python3-pyelftools libglib2.0-dev libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \

libx11-dev mesa-common-dev libglu1-mesa-dev cmake ninja-build pkg-config libjpeg-dev \

libpng-dev libtiff-dev libjasper-dev libgtk-3-dev libeigen3-dev btrfs-progs \

liblzma-dev libbz2-dev zlib1g-dev

- **Herramientas de compilación**: `build-essential gcc gawk make`

- **Yocto helpers**: `chrpath cpio diffstat texinfo rsync bc`
- **Python y librerías**: `python3, python3-pip, python3-jinja2, python3-cryptography, python3-distutils`
- **Compresión y archivos**: `xz-utils, zstd, liblz4-tool, libbz2-dev, liblzma-dev, unzip`
- **Bibliotecas multimedia**: `libglib2.0-dev, libgstreamer*, libgtk-3-dev, libx11-dev, mesa-common-dev, libglu1-mesa-dev`
- **OpenCV**: `cmake, ninja-build, pkg-config, libjpeg-dev, libpng-dev, libtiff-dev, libjasper-dev, libeigen3-dev`
- **Otras dependencias de bajo nivel**: `btrfs-progs, libacl1, libssl-dev, flex, bison, gperf, ncurses`

2. Copiar el repositorio de yocto

    $ git clone git://git.yoctoproject.org/poky

3. Revisar la rama en la que se quiere trabajar

    $ cd poky

    $ git branch -a

    $ git checkout -t origin/kirkstone -b my-kirkstone

4. descarga meta-openembedded esta la carpeta poky

$ git clone -b kirkstone git://git.openembedded.org/meta-openembedded

5. abrir … poky/build/conf y copiar los archivos local.conf y bsplayer del repositorio

6.copiar carpeta mata kernel en la carpeta poky

7. ir a poky y correr oe-init-build-env

$ source oe-init-build-env

8. Ya modificado todo lo que ocupamos se procede a crear la imagen

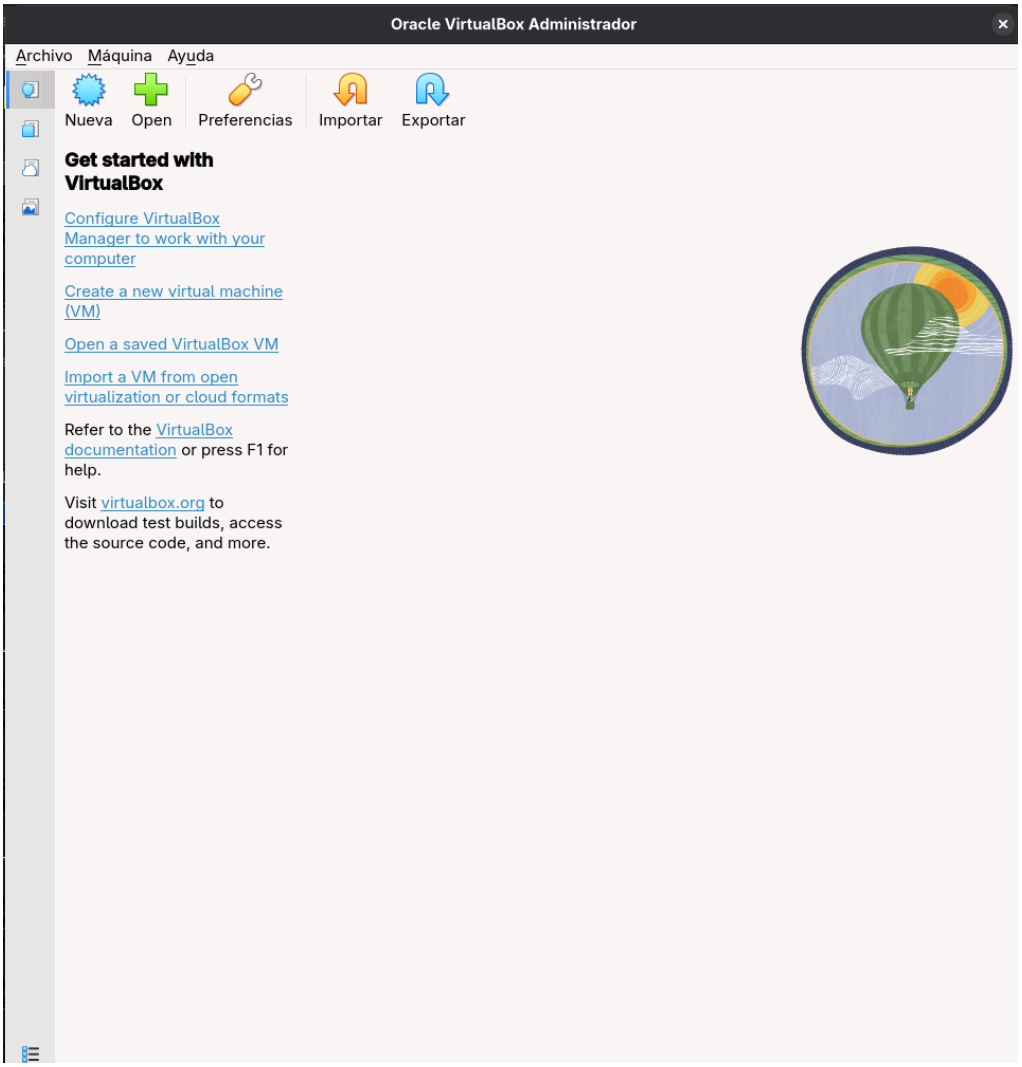
$ bitbake core-image-minimal

9. ya creada la imagen sin problemas se procede a cambiar el formato a .vdi

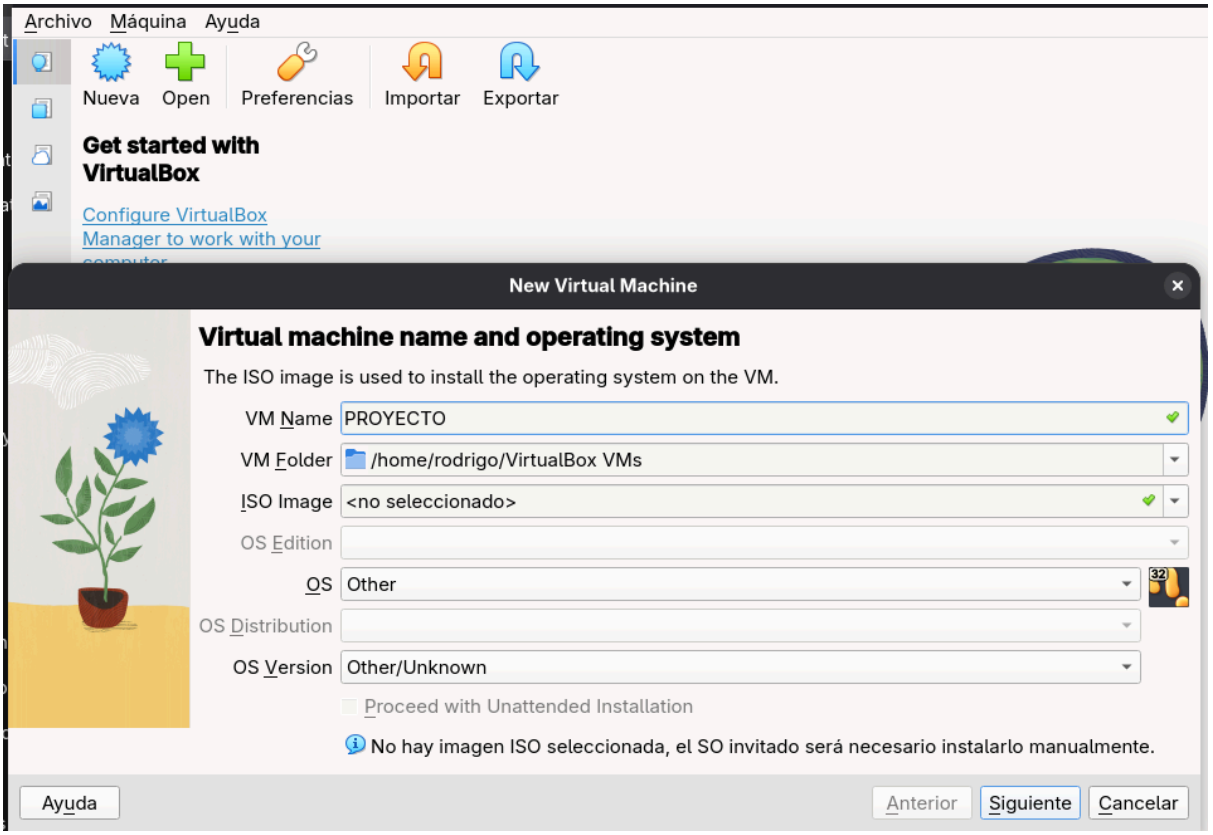
$ cd /directorio-poky/poky/build/tmp/deploy/images/qemux86

$VBoxManage clonemedium disk core-image-minimal.rootfs.wic.vmdk linux_minimal.vdi --format VDI

10. Ya teniendo la imagen se procede a abrir la aplicación virtualbox



11. Selecciona nueva, se coloca el nombre de la máquina que se va a crear, en os selecciona otro,en os version selecciona other/unknown



12. selecciona las especificaciones de hardware de la máquina