

# How to Create Your Own Linux Distribution Using Yocto

Did you know that you can create your own Linux distro with a little coding experience? Building your own distro has the advantage that you can customize every aspect to suit your specific needs.

Check out the following instructions on how to create a Linux distribution using Yocto!

## Create a Linux distribution using Yocto

### 1. Required hardware and operating system

1. Minimum 4GB of RAM (Higher is better)
2. Latest Ubuntu OS (20.04 LTS) or any other Linux OS:
  1. Fedora
  2. openSUSE
  3. CentOS
  4. Debian
3. Hard drive with minimum 100GB free space (Larger size will ensure better performance). Yocto can be quite resource intensive, depending on your final product.

If you are a macOS or Windows user, use virtualization software like VMware or Virtualbox to run a Linux distribution. Alternatively, you can choose multiboot.

### 2. Host setup

First, install the required dependencies in the host system. For this article, the example is using the Ubuntu distribution. If you are running another distro, please check out the Yocto Project Quick Start tutorial and see what dependencies to install at:

<https://www.yoctoproject.org/docs/2.4/yocto-project-qs/yocto-project-qs.html>

Launch Terminal and execute the following commands:

```
sudo apt update sudo apt-get install wget git-core unzip make gcc g ++ build-ess
```

```
tuts@ubuntu:~$ sudo apt-get install wget git-core unzip make gcc g++ build-essential subversion sed autoconf automake texi2html texinfo coreutils diffstat python-pysqlite2 docbook-utils libstdc++2-dev libxml-parser-perl libgl1-mesa-dev libglu1-mesa-dev xsltproc desktop-file-utils chrpath groff libtool xterm gawk fop
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'git' instead of 'git-core'
coreutils is already the newest version (8.28-1ubuntu1).
coreutils set to manually installed.
diffstat is already the newest version (1.61-1build1).
diffstat set to manually installed.
libxml-parser-perl is already the newest version (2.44-2build3).
libxml-parser-perl set to manually installed.
sed is already the newest version (4.4-2).
sed set to manually installed.
unzip is already the newest version (6.0-21ubuntu1).
unzip set to manually installed.
desktop-file-utils is already the newest version (0.23-1ubuntu3.18.04.2).
desktop-file-utils set to manually installed.
wget is already the newest version (1.19.4-1ubuntu2.2).
wget set to manually installed.
The following additional packages will be installed:
  autotools-dev ca-certificates-java cpp-7 default-jre default-jre-headless
  docbook docbook-dsssl docbook-xml dpkg-dev fakeroot fonts-dejavu-extra
  fonts-lato fonts-lmodern fonts-texgyre g++-7 gcc-7 gcc-7-base gcc-8-base
  gir1.2-snapd-1 git-man icc-profiles-free java-common java-wrappers
  javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libapache2-oom-java libaori libaorutil libasan4
```

### 3. Copy Yocto Poky

With the dependencies installed, you can proceed to download Yocto. You will be copying the Yocto repository from the Yocto Project website. Execute the command below, this will download the latest release (branch 'sumo'). Let's create a directory in the Home directory to build the Yocto project in an easy to access and consistent way.

```
mkdir ~ / yocto mkdir ~ / yocto / Project-One / cd ~ / Yocto / Project-One / git
```

```
tuts@ubuntu:~$ mkdir ~/Yocto
tuts@ubuntu:~$ mkdir ~/Yocto/Project-One/
tuts@ubuntu:~$ cd ~/Yocto/Project-One/
tuts@ubuntu:~/Yocto/Project-One$ git clone -b sumo git://git.yoctoproject.org/poky.git
Cloning into 'poky'...
remote: Enumerating objects: 488453, done.
remote: Counting objects: 100% (488453/488453), done.
remote: Compressing objects: 100% (115850/115850), done.
remote: Total 488453 (delta 365795), reused 487325 (delta 364882)
Receiving objects: 100% (488453/488453), 168.25 MiB | 338.00 KiB/s, done.
Resolving deltas: 100% (365795/365795), done.
Checking out files: 100% (5239/5239), done.
tuts@ubuntu:~/Yocto/Project-One$
```

If you get an error like 'git command not found', it means you don't have git installed in your system. Execute the command below to install it.

```
sudo apt install git
```

### 4. Initialize the build environment

To get started with Yocto, you need to initialize the 'build environment'. Execute the commands below. The first one will change the directory to the directory you just copied. The second command will initialize the build environment.

```
cd ~ / Yocto / Project-One / poky source oe-init-build-env build
```

```
tuts@fossilinux:~$ cd ~/Yocto/Project-One/poky/
tuts@fossilinux:~/Yocto/Project-One/poky$ source oe-init-build-env build
You had no conf/local.conf file. This configuration file has therefore been
created for you with some default values. You may wish to edit it to, for
example, select a different MACHINE (target hardware). See conf/local.conf
for more information as common configuration options are commented.

You had no conf/bblayers.conf file. This configuration file has therefore been
created for you with some default values. To add additional metadata layers
into your configuration please add entries to conf/bblayers.conf.

The Yocto Project has extensive documentation about OE including a reference
manual which can be found at:
  http://yoctoproject.org/documentation

For more information about OpenEmbedded see their website:
  http://www.openembedded.org/

### Shell environment set up for builds. ###

You can now run 'bitbake <target>'

Common targets are:
  core-image-minimal
  core-image-sato
  meta-toolchain
  meta-ide-support

You can also run generated qemu images with a command like 'runqemu qemu86'
tuts@fossilinux:~/Yocto/Project-One/poky/build$
```

When the initialization is complete, you will have a build directory and a configuration file. The build directory is where all system builds take place and stores the image files after the process is completed. In fact, after initializing, Terminal will automatically point to the build directory. You can see that in the image above.

## 5. Configuration

When you execute the ls command in the / build directory, you will see a / conf directory containing all the configuration files. Navigate into this directory with the command below:

```
$ cd ~ / Yocto / Project-One / poky / build / conf / $ ls
```

```
File Edit View Search Terminal Help
tuts@fossilinux:~/Yocto/Project-One/poky/build/conf$ ls
bblayers.conf local.conf templateconf.cfg
tuts@fossilinux:~/Yocto/Project-One/poky/build/conf$
```

By executing the ls command on the conf directory, you will see the local.conf file. This file specifies the details of the target machine and the SDK for the desired target architecture.

Open this file for editing with the following command:

```
$ sudo nano local.conf
```

From the image below, the target build is 'qemux86-64'.

```
GNU nano 2.9.3 local.conf
# Lines starting with the '#' character are commented out and in some cases the
# default values are provided as comments to show people example syntax. Enabling
# the option is a question of removing the # character and making any change to the
# variable as required.
#
# Machine Selection
#
# You need to select a specific machine to target the build with. There are a selection
# of emulated machines available which can boot and run in the QEMU emulator:
#
#MACHINE ?= "qemuarm"
#MACHINE ?= "qemuarm64"
#MACHINE ?= "qemumips"
#MACHINE ?= "qemumips64"
#MACHINE ?= "qemuppc"
#MACHINE ?= "qemu86"
#MACHINE ?= "qemu86-64"
#
# There are also the following hardware board target machines included for
# demonstration purposes:
```

Now proceed to uncomment the following lines (uncomment the # sign).

```
DL_DIR? = "$ {TOPDIR} / downloads" SSTATE_DIR? = "$ {TOPDIR} / sstate-cache" TMPDIR? = "$ {TOPDIR} / tmp"
```

```
GNU nano 2.9.3 local.conf Modified
# BitBake has the capability to accelerate builds based on previously built output.
# This is done using "shared state" files which can be thought of as cache objects
# and this option determines where those files are placed.
#
# You can wipe out TMPDIR leaving this directory intact and the build would regenerate
# from these files if no changes were made to the configuration. If changes were made
# to the configuration, only shared state files where the state was still valid would
# be used (done using checksums).
#
# The default is a sstate-cache directory under TOPDIR.
#
SSTATE_DIR ?= "${TOPDIR}/sstate-cache"
# where to place the build output
#
# This option specifies where the bulk of the building work should be done and
# where BitBake should place its temporary files and output. Keep in mind that
# this includes the extraction and compilation of many applications and the toolchain
# which can use Gigabytes of hard disk space.
#
# The default is a tmp directory under TOPDIR.
#
TMPDIR = "${TOPDIR}/tmp"
```

Before continuing with the compilation process, add the following lines at the end of the local.conf file.

```
BB_NUMBER_THREADS = "X" PARALLEL_MAKE = "-j X"
```

Replace X with twice the number of processor / CPU in your computer. If you have 4 processors, then you would have statements like this:

```
BB_NUMBER_THREADS = '8' PARALLEL_MAKE = '-j 8'
```

To see the number of CPUs in the computer, execute the following command:

```
lscpu
```

## 6. Compile and build process

To start building the image, execute the command below in your / build directory.

## bitbake core-image-sato

```
tuts@fossilinux:~/Yocto/Project-One/poky/build/conf$ bitbake core-image-sato
Parsing recipes: 100% |#####| Time: 0:02:34
Parsing of 814 .bb files complete (0 cached, 814 parsed). 1282 targets, 44 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION      = "1.38.0"
BUILD_SYS       = "x86_64-linux"
NATIVELSBSTRING = "ubuntu-18.04"
TARGET_SYS      = "i586-poky-linux"
MACHINE         = "qemux86"
DISTRO          = "poky"
DISTRO_VERSION  = "2.5.3"
TUNE_FEATURES   = "m32 i586"
TARGET_FPU      = ""
meta
meta-poky
meta-yocto-bsp  = "sumo:b39f4146de84d7b36861859ec669d9c8e2ca77c6"

NOTE: Fetching uninative binary shim from http://downloads.yoctoproject.org/releases/uninative/2.4
/x86_64-nativesdk-libc.tar.bz2;sha256sum=06f91685b782f2ccfd3070b3ba0fe4a5ba2f0766dad5c9d1642ccf
95accd0
Initialising tasks: 100% |#####| Time: 0:00:13
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
Currently 8 running tasks (117 of 6129) 1% |
0: binutils-cross-i586-2.30-r0 do_fetch (pid 5052) 0% | 28.3K/s
1: gcc-cross-i586-7.3.0-r0 do_fetch (pid 5224) 23% | 124K/s
```

This command will start downloading and compiling packages for the target system. Please do not execute the above bitbake command with root privileges as it will cause an error.

For the first time build, this can take several hours (even more than 2 hours). Sometimes bitbake can fail. Do not panic! Please execute the above command again. The error can be caused by a particular site that is down or missing resources.

The resulting binary images are stored in the / build directory in poky / build / tmp / deploy / images / qemux86.

You finished reading the article "**How to Create Your Own Linux Distribution Using Yocto**" edited by the [TipsMake](#) team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.