Toradex
Linux OS開発環境構築
マニュアル
BSP5用

本マニュアルは岡本無線電機株式会社が独自作成したものでありメーカーが保証した内容ではありません。万が一本マニュアルに間違いがあり、事故が生じたとしても岡本無線電機株式会社は一切の責任を問われないものとさせていただきます。

変更履歴

バージョン	更新日付	変更内容
1.00	2022/11/07	BSP5用に作成

本マニュアルについて

本マニュアルはトラデックスのCPU**モジュール上で動作する**Linux OS**の開発環境構築および**OS**イメージの作成をする手順を記述しています。**

参考：
https://developer.toradex.com/knowledge-base/board-support-package/openembedded-core
https://docs.yoctoproject.org/ref-manual/index.html

1. 実行環境

本マニュアルの実行環境は下記です。

仮想化ソフト：VMWARE Player v15.5.7
Host OS： Windows 10 21H2
Guest OS： Ubuntu Desktop 20.04LTS 64bit(**英語版**)
BSP：v5.7
CPU**モジュール**：Verdin-iMX8M Plus Quad 4GB Wi-Fi / Bluetooth IT 1.1A
キャリアボード：Verdin**開発ボード** Rev 1.1C + **アクセサリーキット**

本マニュアルとは異なるモジュールや評価ボード以外のキャリアボードを使われても大雑把には同じ操作となります。
BSP**や**Guest OS**のバージョンが異なると本マニュアルの操作と異なる可能性があります。**

インターネット接続環境が必要になります。

・VMwareの設定について

下記のような設定を推奨いたします。

CPUコア数　割り当てられる最大値

メモリー8GB以上

HDD300GB以上

・パソコンのスペックについて

SDカードスロットが必要です。本開発ではパソコンのスペックをかなり要求されます。コンパイルに数時間要するのでできるだけ開発効率を上げるために高性能なパソコンを使用することを推奨いたします。参考まで下記パソコンのスペックでコンパイルに4時間ほどかかります。

CPU：2.9GHz　8コア(Hyper Threading)

メモリー：64GB

SSD & HDD：6TB

コンパイルは並列で行われますのでCPUの速度がコンパイル時間に影響します。上記のスペックではHDDやメモリーは十分ですが全CPUコアが使用率100%になることも多々あります。

2．事前準備

事前にVMWARE Playerをインストールし Ubuntuが動作する状態にしてください。本マニュアルではこれらがインストール済みの状態から始まります。インターネットに接続できるように設定も行ってください。

3. 前提知識

本マニュアルは下記に理解がある方を前提としています。

・Ubuntuの基本的な操作

・CPUボードの大雑把な仕組み

・Linuxの大雑把な仕組み

・gitの使い方

4. 注意事項

マニュアルで進めている内容はあくまで一例ですが同様の方法をとっていただくとトラブル発生時などにご質問にお答えしやすいです。

開発環境と実行環境の違いをわかりやすくするためにコマンドの表記の前に下記をつけています。

開発環境(パソコン)上で入力するコマンド：[Ubuntu]$

実行環境(モジュール)上で入力するコマンド(Linux) ：[Module]#

実行環境(モジュール)上で入力するコマンド(U-Boot)：[U-Boot]#

・仮想環境

VMwareなど仮想環境を使用せずパソコンに直接Ubuntuをインストールされてもよいですが環境が壊れるなどのトラブルが発生する可能性もあります。仮想環境を使用すればイメージのバックアップを取ることでこのようなトラブルの回避が可能です。

・開発OSの選定について

特にUbuntu20.04でないと開発ができないわけではありませんがトラデックスのBSPと親和性が高く、開発情報が多い安定したものを選定しています。別のディストリビューションを使われると本マニュアルとは違う方法で進めないといけない可能性があります。またトラブルが発生する可能性があります。言語は英語版を推奨します。(エラーメッセージやログなどが英語で出力されるため問題発生時にインターネットで情報を捜しやすいため。また日本語版を使うとIDEなどのGUIのレイアウトが崩れる場合があります。)

・インターネット接続環境について

トラデックスのLinux開発環境ではさまざまな(gitやftp,svnなど)プロトコルを使用します。通常、社内LANをご使用いただく場合、ファイヤーウォールやプロキシサーバなどを使用しているケースが多くこれらのプロトコルで通信できない可能性があります。ファイヤーウォールなどの影響を受けないフリーな回線をご用意ください。

コピーについて

本マニュアル内のコマンドなどをコピーした場合、改行が入ったり「-」が抜けてしまうことがあるのでご注意ください。一度テキストエディタなどに張り付けてコピーした内容をご確認ください。

Open Embeddedについて

トラデックスのCPUモジュール上で動作するLinux OSは組み込み向けディストリビューション(Poky)であり、パソコンなどで動作するDebian系ディストリビューションなどと大きく異なります。

例えばaptなどパッケージ管理ソフトで容易にパッケージのインストールを行うような仕組みがありません。

また開発環境(x86系)と実行環境(ARM32 or ARM64)が異なるためOSやミドルウェア、アプリケーションなどはすべてARM向けにソースコードからクロスコンパイルを行って作成する必要があります。

トラデックスの開発環境ではそれらの開発を容易にするためにOpen Embeddedという開発プラットフォームを使用します。

Open Embeddedは主にリソースが限られた組み込み機器に必要な機能のみが搭載された軽量なOSを作るために用意された開発プラットフォームです。

CPUボードの開発をする場合、目的のアプリケーション向けに必要なCPUの速度やメモリ容量などハードウェアスペックを確認してデバイス選定を行いますがソフトウェア(主にOSとミドルウェア)も目的のアプリケーションを動かすための必要最小限のものにしないと余計なリソースを奪われ性能が損なわれる可能性があります。

例えばトラデックス標準OSはデスクトップ環境やQTなどが搭載されていますがGUIが必要ないシステムを作る場合これらは不要でかつリソースの無駄使いになります。

CPUボードを使ってGUIを不要とするルーターを開発する場合、GUI機能は不要ですがさまざまな通信プロトコルやwebサーバなどの機能が要求されます。トラデックス標準OSではこのようなルーターを作ることはできません。

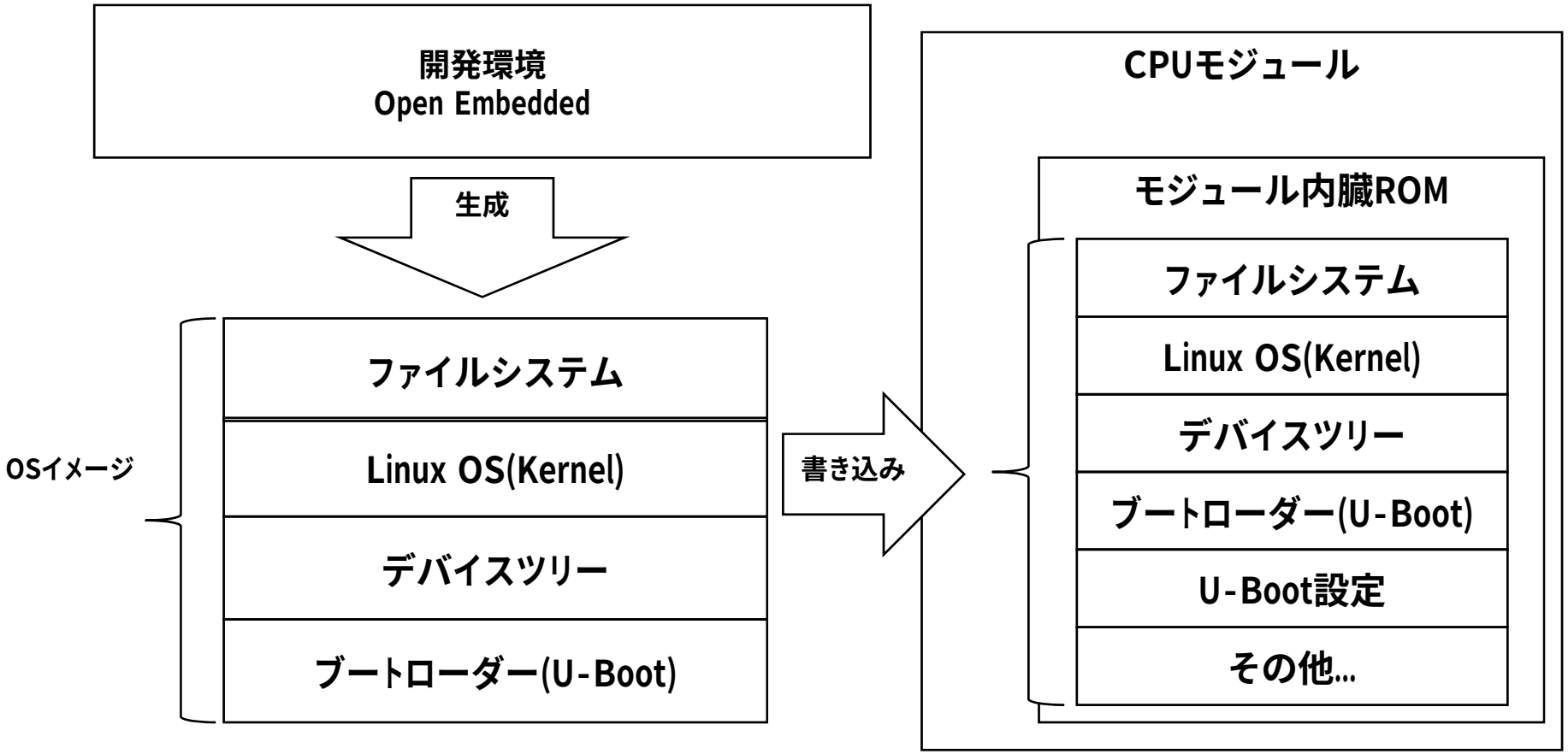
Open Embeddedではこのような要求される機能を丁度よく盛り込んだOSを容易に作成できるようになっています。

Open Embeddedの詳細に関しては下記をご参照ください。

http://docs.openembedded.org/usermanual/html/

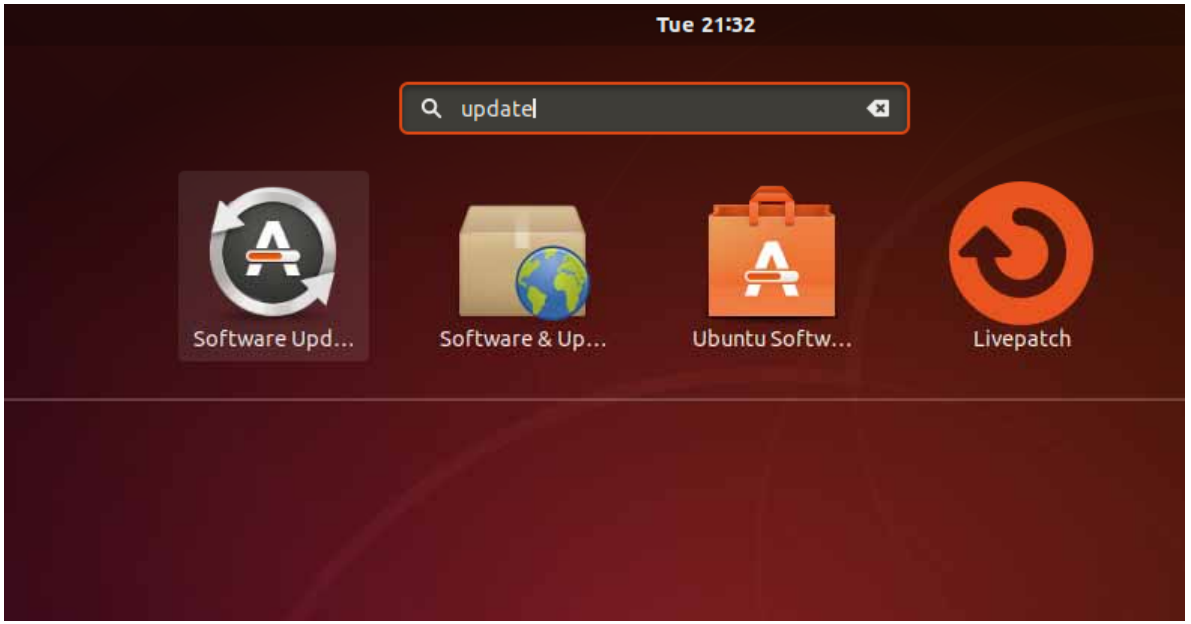
開発の流れ

トラデックスのCPUモジュールのソフトウェア郡は主にブートローダー(U-Boot)、Linux OS、ファイルシステムで構成されています。
本マニュアルではそれら3つを含めたOSイメージの作成について記載しています。
下記はOSイメージの開発の流れです。アプリケーションはファイルシステムに含まれます。



開発環境構築

Ubuntuのアップデートを行います。画面左下のダッシュボードをクリックして「update」と入力しSoftware Updaterを起動しアップデートを行います。アップグレード(OSバージョンのアップデート)の案内がある場合もありますがUbuntu20.04のバージョンのまま使用しますのでアップグレードはしないようにご注意ください。

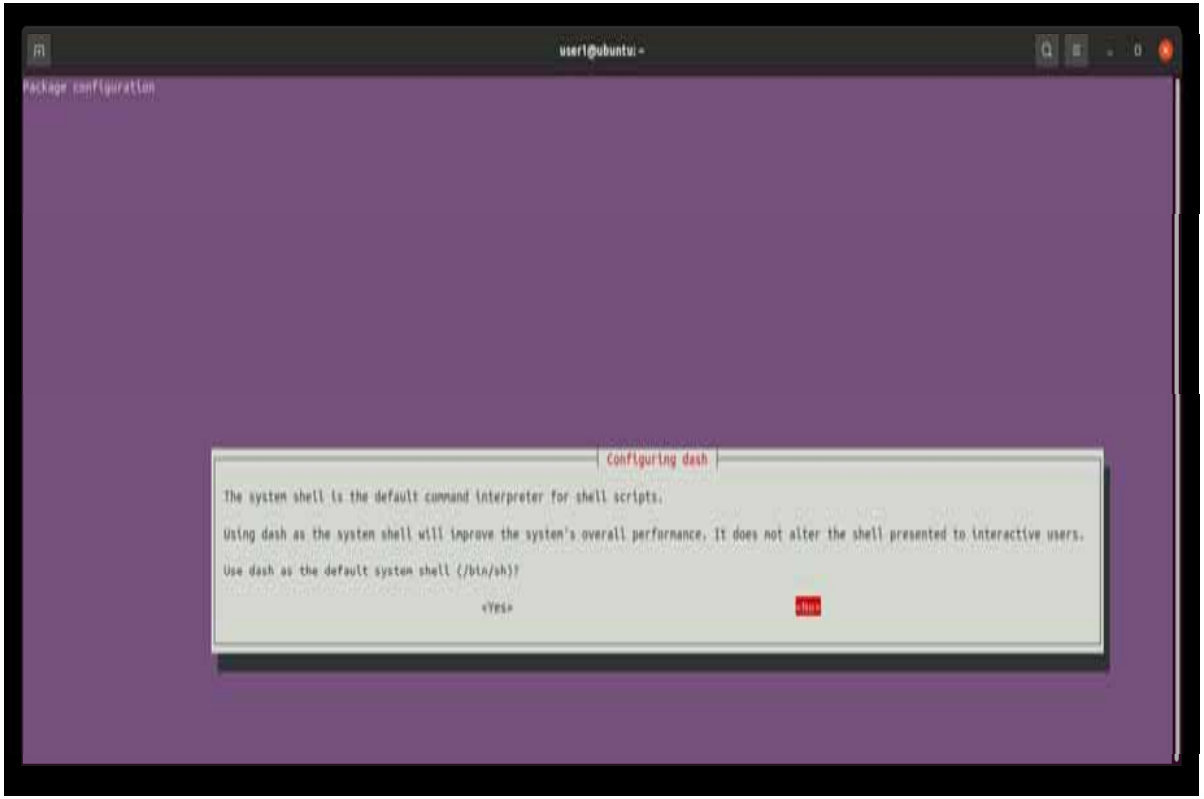


Software Updaterと同じようにダッシュボードからTerminalを立ち上げます。以後Terminalでさまざまなコマンド入力します。コマンドを入力するのは場合はTerminalを立ち上げてください。

Open Embeddedの開発においてシェルにUbuntuデフォルトのdashを使うとうまくいきません。またdashを使用しないといけないケースはありません。基本的にbashを使用すればよいため下記コマンドでdashからbashへ変更します。

[Ubuntu]$sudo dpkg-reconfigure dash

下記のような画面が表示されますのでNoを選択します。



必要機能のインストールのため下記コマンドを実行します。

[Ubuntu]$ sudo apt -y install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev xterm

[Ubuntu]$ sudo apt -y install curl python libncurses5-dev

Ubuntu20**では初期状態で**2GB**の**swap**がありますが足りなくなるため増量します。**

RAM**が不足していない状態でも**swap**が使用されるため**RAM**の大きさに関係なく必要です。**

下記例では8GB**を追加しています。**

領域確保

sudo fallocate -l 8G /swapfile2

権限付与

sudo chmod 600 /swapfile2

swap**として登録**

sudo mkswap /swapfile2

有効化

sudo swapon /swapfile2

確認

swapon -s

fstab**に追記**

sudo gedit /etc/fstab

/swapfile2 none swap sw 0 0

BSPの入手

トラデックスのBSP(Board Support Package)の内容はOpen Embeddedのレシピになっていますので
BSPの入手=Open Embeddedのレシピをダウンロードとご認識ください。

BSPをダウンロードするためのrepoを入手します。
実行ディレクトリ作成
[Ubuntu]$ mkdir ~/bin

repoのダウンロード
[Ubuntu]$ curl https://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo

実行権限追加
[Ubuntu]$ chmod a+x ~/bin/repo

一度Ubuntuを再起動します。再起動することでrepoコマンドが使用できるようになります。

ワークディレクトリを作成
(ワーキングディレクトリは都合のいい場所で構いません。本マニュアルではできるだけわかりやすいパスとしています。)
[Ubuntu]$ sudo mkdir /work

オーナーとグループを変更
[Ubuntu]$ sudo chown <user name> /work
[Ubuntu]$ sudo chgrp <user name> /work
[Ubuntu]$ cd /work

repoダウンロード用ディレクトリ作成
[Ubuntu]$ mkdir bsp && cd bsp

repo(git)の設定　メールアドレスと名前　(ダミーで問題ありません。変更せず下記をそのまま実行してください。)
[Ubuntu]$ git config --global user.email example@example.com
[Ubuntu]$ git config --global user.name example

repo初期化
[Ubuntu]$ repo init -u https://git.toradex.com/toradex-manifest.git -b dunfell-5.x.y -m tdxref/default.xml

バージョンを指定する場合は下記のような指定をします。
[Ubuntu]$ repo init -u https://git.toradex.com/toradex-manifest.git -b refs/tags/5.7.0 -m tdxref/default.xml

repoを実行すると下記のような質問があるのでyを入力
[Ubuntu]$ Enable color display in this user account (y/N)?

repo同期
[Ubuntu]$ repo sync

エラーが出る場合、いったんすべて削除して
[Ubuntu]$ rm -rf ./* && rm -rf ./.repo

[Ubuntu]$ repo sync -j1
を試してください。並列のダウンロード数を1つにします。

Open Embeddedの設定

Open Embedded用のディレクトリ作成&移動
cd /work
[Ubuntu]$ mkdir oe-core && cd ./oe-core

BSPコピー(repoのディレクトリはそのまま置いておき最新版や古いバージョンを入手するときに再利用します。不要なら削除しても問題ありません。)
[Ubuntu]$ cp -prf /work/bsp/* ./

BSP配下のファイルは下記のようになっています。
export：ビルド環境設定シェル
layers ：Open Embeddedのレシピ

/work/oe-core配下で下記を実行します。

Open Embeddedの環境設定
[Ubuntu]$. export

自動的にOpen Embeddedで使われるファイルの生成や環境変数の定義などがされ/work/oe-core/buildに移動します。

設定ファイルのlocal.confの内容を変更します。
[Ubuntu]$ gedit ./conf/local.conf

・ターゲットモジュールの選択(必須)
MACHINE変数はモジュールの設定を行います。ファイル内に候補がいくつかあるのでコメントアウトをはずして目的の設定を行ってください。コメントアウトは#で行います。下記はverdin-imx8mpの場合の設定です。ターゲットのモジュールにあわせて変更してください。
MACHINE ?= "verdin-imx8mp"

・GPUのライセンスへの同意(必須)
i.MX系のモジュールでGPUの機能を使用する場合、下記ファイルを参照の上ライセンスに同意しなければなりません。
/work/oe-core/layers/meta-freescale/EULA

同意できる場合、下記をlocal.conf内の任意の場所に追加します。これを設定しないとビルドが通りません。
この設定はライセンスへの同意の設定となります。
ACCEPT_FSL_EULA = "1"

・一時ファイルの削除設定(任意)
下記設定を消すとビルド途中の一時ファイルなどが削除されなくなります。容量は大幅に増えますが一時ファイルが必要になる場合は本設定を削除します。通常はこのままの設定で問題ありません。
INHERIT += “rm_work”

・ビルドヒストリーの書き込みをなくす (任意)
下記設定を追記するとビルドの最後に書き込まれるビルドヒストリーの書き込みを行いません。その分若干ビルドが早くなります。
INHERIT_remove = "buildhistory"

ビルドヒストリーは/work/oe-core/build/buildhistory/配下にできます。主にビルドで出力されたファイルリストや依存関係の情報などが出力されます。

・並列処理オプション
下記の設定を変更することによりbitbakeの並列タスク数を変更できます。
初期設定はVMwareに割り当てられたCPUの数と同じ数になります。

bitbakeを並列で行うとエラーが発生する場合の原因の一つに並列タスクによる原因の場合があります。
並列スレッドが多すぎるとメモリー使用量の増加してメモリー不足が起きます。
その場合並列タスクを少なくするとエラーが回避できます。4つを指定する場合は下記のようになります。
BB_NUMBER_THREADS = "4"

・並列make設定
makeで指定するプロセッサ数を指定します。
初期設定はVMwareに割り当てられたCPUの数と同じ数になります。

4つを指定する場合は下記のようになります。
PARALLEL_MAKE ?= "-j 4"

Open Embeddedのビルド

Open Embeddedのビルドを行うと大雑把に下記のような手順が実行されます。

do_fetch: ソースコードをダウンロード
do_unpack: ソースコードを解凍(展開)
do_patch: パッチを適用
do_configure: configureを実行
do_compile: makeを実行
do_install: make install を実行
do_populate_sysroot: sysroot にインストール
do_package: パッケージ化用のディレクトリにインストール
do_package_write: パッケージの作成 (ipk, deb, rpm)
do_build: ビルド終了用のタスク
do_rm_work:一時ファイルの削除

ビルドに失敗する場合は最初のソースコードのダウンロードで失敗するケースがほとんどです。
ダウンロードにはgitやsvnなど使用しています。ファイヤーウォールやプロキシの利用している場合ダウンロードエラーとなる場合があります。最初に下記コマンドですべてのファイルのダウンロードだけをしておきエラーが発生しないかを検証します。

```
[Ubuntu]$ bitbake --runall fetch tdx-reference-multimedia-image
```

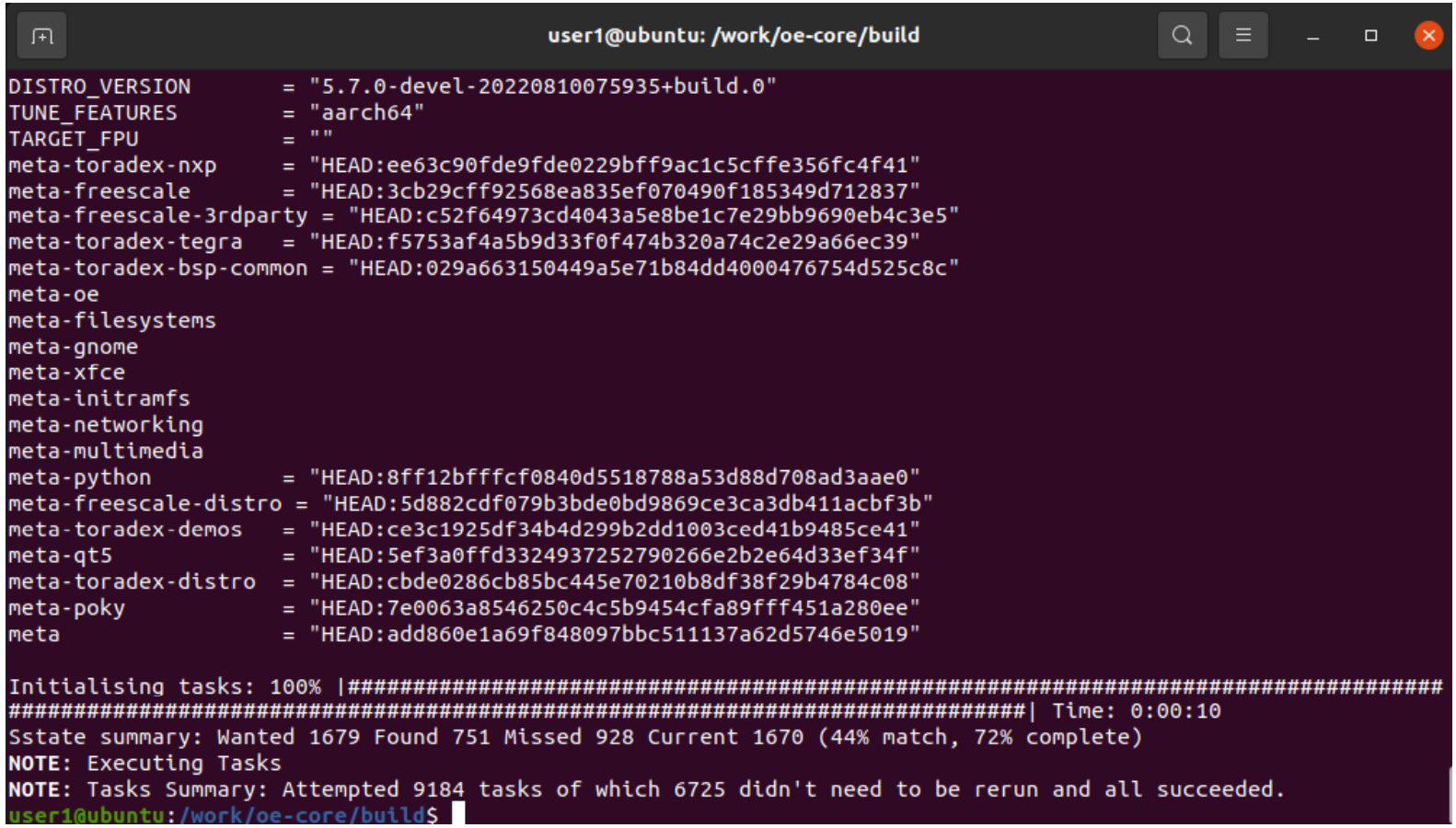
本コマンドでダウンロードエラーが出る場合はインターネット回線に何かしらの制限がかかっていると思われます。
ダウンロードが正常に終われば下記コマンドですべてのビルド作業を行います。

```
[Ubuntu]$ bitbake tdx-reference-multimedia-image
```

パソコンのスペックにもよりますがビルドには数時間かかります。正常にビルドが終わると下記にOSイメージを作成するためのtarファイルが出力されます。(Verdin-iMX8M Plusの場合) バージョンや日付はコンパイル環境によります。

/work/oe-core/build/deploy/images/verdin-imx8mp/Verdin-iMX8MP_Reference-Multimedia-Image-Tezi_<バージョン>-<日付>.tar

下記のようなメッセージが出るとビルド成功です。



OSイメージの書き込み

OSイメージ用ディレクトリ作成&移動

[Ubuntu]$ mkdir /work/oe-core/image && cd /work/oe-core/image

作成したイメージを展開します。必ずroot権限で展開してください。

[Ubuntu]$ sudo tar -xf ../build/deploy/images/verdin-imx8mp/Verdin-iMX8MP_Reference-Multimedia-Image-Tezi_<BSP Version>-<Build Version>.tar

解凍したファイルはTEZIでモジュールに書き込みます。
書き込み方法はTEZIマニュアルをご参照ください。

出力されたファイルの説明

u-boot-nand.imx
ブートローダーの実行ファイルです。

LA_OPT_NXP_SW.html
NXPのソフトウェアライセンスアグリメントについての書かれたhtmlファイル**です。**
ソフトウェアを使用するにあたり同意しなければならない内容です。

Reference-Multimedia-Image-verdin-imx8mp.bootfs.tar.xz
中にカーネルやデバイスツリーが入っています。

Reference-Multimedia-Image-verdin-imx8mp.tar.xz
ルートファイルシステムを圧縮したものです。

上記以外のファイルはTEZI**で使用するファイルです。**
詳しくはTEZI**マニュアルをご参照ください。**

以上がOS**イメージ開発の流れになります。本マニュアルではトラデックス標準の**OS**イメージを作成しましたが必要に応じて**OS**イメージのカスタムやアプリケーションの作成を行う必要があります。それらに関しては別途、対象のマニュアルをご参照ください。**