WNDRVR

サポートネットワーク

Wind River Linux Binary Release Quick Start, LTS 21

Wind River Linux バイナリ・リリース クイックスタート LTS 21

2021年9月1日

著作権について

本社

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.
Toll free (U.S.A.): +1-800-545-WIND
Telephone: +1-510-748-4100

その他の連絡先については、下記のウェブサイトをご覧ください。
www.windriver.com
カスタマーサポートへのお問い合わせ
www.windriver.com/support

※　本ドキュメントは、参照目的のために英語版「Wind River Linux Platform Development Quick Start, LTS 21」を翻訳したものです。

Wind River Linux Binary Release Quick Start, LTS 21
2021年9月1日

1. クイックスタートの概要

このクイックスタートの説明を参考にして、Wind River® Linux Binary Release（バイナリ・リリース）をご利用ください。

このドキュメントでは、プラットフォームプロジェクトをゼロから作成することなく、デバイスやコンテナをWind River Linuxで起動する手順をご紹介します。バイナリ・リリースを搭載したターゲットシステムまたはコンテナイメージが起動したら、オプションでSDKをインストールしてアプリケーション開発に使用することができます。

Wind River Linuxの詳細および製品がどのような組込みシステムプロジェクト向けに設計されているかについては、ウインドリバーのウェブサイトをご覧ください。

英語： https://www.windriver.com/products/linux　　日本語： https://www.windriver.com/japan/products/linux

バイナリ・リリースの管理については『*Wind River Linux Binary Release Developer's Guide*』をご参照ください。

必要条件

- Wind River Linuxを使用するための要件を満たすLinuxホスト

  Gitバージョン1.9以上とPython 3を使用します。さらに、Python 3にまだ移行していない特定の依存関係を満たすために、Python 2も必要です。詳細については、*Wind River Linux Release Notes: Host System Recommendations and Requirements*をご参照ください。
- イメージのダウンロードや、システムのアップデートを行うためのインターネットアクセス
- コンテナイメージの場合

  インフラとLinuxホストにコンテナをデプロイするための依存関係が必要です。これには、使用するコンテナ技術に応じてDocker EngineまたはKubernetesフレームワークが含まれます。
- Linuxおよびコマンドラインに関する中級レベル知識

  これには、コマンドの実行、パッケージのインストール、プロジェクトの設定ファイルの編集などが含まれます。

ワークフロー

Wind River Linuxのバイナリイメージを稼働させるためのワークフローは以下の通りです。

- 圧縮されたシステムイメージまたはコンテナイメージファイルをダウンロードします。

  詳細については、「バイナリイメージとSDKのダウンロード（P.3）」をご覧ください。
- ハードウェアメディアにインストールするためのイメージを準備します。

  詳細については、「ターゲットシステムイメージのUSBフラッシュドライブへのロード（P.5）」をご参照ください。
- ハードウェア、QEMU、またはDockerコンテナとしてWind River Linuxイメージを起動します。
  - ハードウェア上でのターゲットシステムイメージの起動（P.6）
  - QEMUによるターゲットシステムイメージの起動（P.8）
  - Dockerを使ってコンテナイメージをデプロイする（P.12）
- Wind River Linux SDKをインストールして使用します。
  - SDKのインストール（P.13）
  - Hello Worldサンプルアプリケーションの作成（P.14）

ターゲットシステムイメージを使用したい場合は、「バイナリイメージとSDKのダウンロード（P.3）」から始めてください。コンテナイメージを使用する場合は、「Dockerを使ってコンテナイメージをデプロイする（P.12）」から始めてください。

ダウンロードしたイメージの種類によって、インストールと使用の方法が決まります。例えば、ターゲットシステムイメージをハードウェアやQEMUで起動したり、Dockerを使ってコンテナイメージを起動したりすることができます。

注： このドキュメントでは、コマンドラインの例として、ARMベースのシステムにはbcm-2xxx-rpi4 BSPを、IAベースのシステムにはintel-x86-64 BSPを使用しています。お使いのBSPに合わせてコマンドを変更する必要がある場合があります。

2. バイナリイメージとSDKのダウンロード

最小構成と完全構成のフルシステムのバイナリイメージと、バイナリ配布を管理するためのツールを備えたSDKを使用することができます。

始める前に

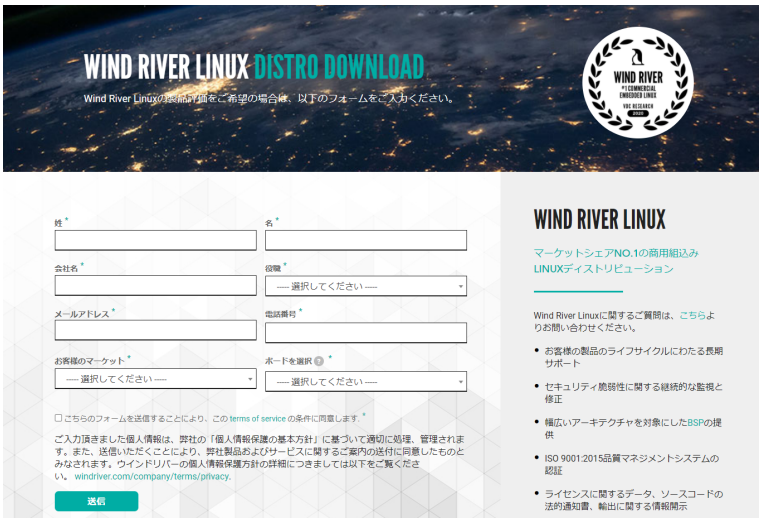
システムコンテナイメージをダウンロードするには、インターネットに接続されたLinuxホストシステムが必要です。コンテナのバイナリイメージはDockerHubで公開されており、手動でダウンロードする必要はなく、準備も必要ありません。詳しくは、「Dockerを使ってコンテナイメージをデプロイする（P.12）」をご覧ください。

使用可能なイメージとその機能については、「リリース情報（P.21）」をご覧ください。

手順

1. Webブラウザを開いて、https://www.windriver.com/japan/products/linux/download にアクセスします。

   オンラインリクエストフォームに必要事項を入力し、ハードウェアボード（BSP）を選択して、利用規約に同意し、送信をクリックします。



2. 前のステップで選択したハードウェアボード向けの、ダウンロードリンクがメールで送られてきます。

Wind River Linux ダウンロードリンクのご連絡

SI salesinquiry@windriver.com
To

Wind River Linux for の製品評価をリクエスト頂き、誠にありがとうございます。

以下リンクよりダウンロードください。

- Minimal Image
- Full Image
- Development SDK
- Source Code

プラットフォームの開発手順は、Quick Start guideをご参照ください。
Wind River Linuxに関するドキュメントは、Wind River Support Networkをご覧ください。

充実した有償サポートをご希望の場合は、お問い合わせフォームよりご連絡ください。

どうぞよろしくお願いいたします。
Wind River Linux チーム

使用したいイメージまたはSDKの***.tar.bz2**ファイルを保存します。

各圧縮ファイルには以下の内容が含まれています。

bcm-2xxx-rpi4（またはその他のARM）ターゲットシステムイメージ

- **qemu-u-boot-***bspName* **.bin** - QEMUでイメージを展開する際に使用する、バイナリコンパイルされたBIOSファイルです。
- *imageType* **. ustart.img.gz** - イメージをデバイスにインストールするための圧縮されたイメージソースファイルを含むARMに対応 するSDKです。

intel-x86-64 ターゲットシステムイメージ

- ovfm.qcow2 - QEMUでイメージを展開する際に使用する、バイナリコンパイルされたBIOSファイルです。
- *imageType* **. ustart.img.gz** - イメージをデバイスにインストールするための圧縮されたイメージソースファイルを含むx86に対応するSDKです。

3. すべてのダウンロードディレクトリにある追加ファイル
   - *sdkType* **. sh** - LinuxホストシステムにSDKとツールをインストールするためのセルフインストールのシェルスクリプトを提供します。
   - *sdkType* **host.manifest** - ホストパッケージマニフェストが含まれ、Linuxホストシステムでの開発をサポートするために含まれるSDKのすべてのパッケージがリストアップされます。
   - *sdkType* **target.manifest** - ターゲットパッケージのマニフェストを含み、SDKでサポートされているすべてのターゲットパッケージをリストアップします。
   - *imageType***.README.md** - イメージのREADMEファイルを含みます。
   - *imageType* **.manifest** - パッケージマニフェストを含み、イメージにインストールされたすべてのパッケージをリストアップします。
   - **sha256sum.txt** – イメージの完全性を確認するためのチェックサムファイルが含まれています。

4. ターゲットシステムイメージとSDKについては、前の手順で***.tar.bz2**の内容を抽出します。イメージをハードウェア・デバイスやQEMUにデプロイするには、*imageType* **.ustart.img**ファイルが必要になります。

次のステップ

イメージやSDKをダウンロードし、ファイルを解凍後に、用途に応じて以下のステップに進んでください。

- USBメモリーにコピーして、ハードウェアデバイスへ展開します。詳細は、「ターゲットシステムイメージのUSBフラッシュドライブへのロード（P.5）」をご覧ください。
- QEMUを使用して直接デプロイします。詳細については、「QEMUによるターゲットシステムイメージの起動（P.8）」をご参照ください。
- SDKとツールについては、Linuxホストシステムにインストールして、ユーザースペースアプリケーションの開発を開始してください。詳しくは、「SDKのインストール（P.13）」をご覧ください。

3. ターゲットシステムイメージ クイックスタート

3.1. ターゲットシステムイメージのUSBフラッシュドライブへのロード

Wind River Linuxのバイナリターゲットシステムイメージを USBフラッシュドライブにコピーします。

本作業について

Wind River Linux ターゲットシステムイメージをハードウェアデバイスにインストールする場合、この手順を実行して起動可能な USB フラッシュドライブを作成する必要があります。

始める前に

- 使用するデバイスに適したターゲットシステムイメージをダウンロードして解凍してください。詳しくは、「バイナリーイメージとSDKのダウンロード（P.3）」をご覧ください。
- ディスク容量が8GB以上のUSBフラッシュドライブを用意してください。サポートされているメモリーデバイスの詳細については、本クイックスタートドキュメントの「リリース情報（P.21）」をご参照ください。

手順

1. USBフラッシュドライブをホストシステムに挿入します。

   注：bcm-2xxx-rpi4 の場合はSDカードをご使用下さい。

2. ホストシステム上でUSBフラッシュドライブがマッピングされているデバイスノードを確認します。Linuxでは、lsblkコマンドを実行して確認できます。
   例えば、以下のようになります。

   ```
   $ lsblk
   ```

   出力には、システム上のすべてのドライブとそのパーティションが一覧表示されます。メインのドライブには、**sda**やmixiなどの3文字の名前がついています。

   sdb、各ドライブ内のパーティションには、**sda1**や**sdb2**のように追加の数字が指定されています。

   以下の例では、**/dev/sdb1**がUSBフラッシュドライブのデバイスノードとして使用されています。

3. Linuxホストシステムが USBフラッシュドライブをオートマウントしている場合は、umountコマンドを実行して、そのファイルシステムをファイル階層から切り離します。

   a. **dev/sdb1**にマウントされている全てのパーティションを特定します。

   ```
   $ mount | grep sdb1
   /dev/sdb1 on /media/username/usbdrive type fuseblk ...
   ```

   この例では、1つのパーティションのみがマウントされています。

   b. 前のステップでリストアップされたマウントされた各パーティションをアンマウントします。

   ```
   $ sudo umount /dev/sdb1
   ```

   この例では、Ubuntu Linuxを実行しているホストシステム上のファイル階層から、**/dev/sdb1**ファイルシステムを切り離します。このコマンドは、前のステップで挙げた各パーティションに対して実行します。

4. イメージをメモリーデバイスにコピーします。

以下の例では、Linux ホストシステムのインストーライメージを **/dev/sdb** デバイスノードにコピーしています。

BSP	実行するコマンド
intel-x86-64	$ **zcat** *path_to* **/wrlinux-***imageType* **-intel-x86-64.ustart.img.g z | sudo dd of=/dev/sdb bs=1M status=progress**
bcm-2xxx-rpi4またはその他のARMベースのBSP	$ **zcat** *path_to* **/wrlinux-***imageType* **-bcm-2xxx-rpi4.ustart.img. gz | sudo dd of=dev/sdb bs=1M status=progress**

これらの例では、コピーしたいイメージに応じて、*imageType*を**full**または**minimal**に置き換えてください。

5. syncコマンドを実行して、メモリ上のデータが確実にディスクに書き込まれるようにします。

```
$ sync
```

6. ホストのファイルシステムからUSBフラッシュドライブを取り出します。

```
$ eject /dev/sdb
```

7. USBフラッシュメモリーを安全に取り出します。

次のステップ

USBフラッシュドライブにイメージが格納されたら、それを使ってハードウェアまたはエミュレートされたターゲットデバイスを起動し、GRUBブートメニューを表示することができます。詳細については、「ハードウェア上でターゲットシステムイメージの起動（P.6）」をご参照ください。

3.2. ハードウェア上でのターゲットシステムイメージの起動

ターゲットシステムイメージをハードウェアデバイスのストレージにコピーしたら、そのイメージを起動して使い始めることができます。

本作業について

Wind River Linuxイメージでは、入力用のキーボードとマウス、そしてコンソール出力を見るためのディスプレイが必要です。

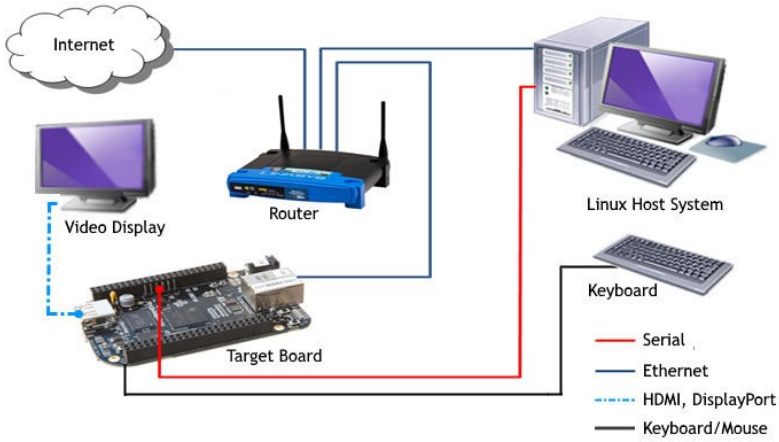
始める前に

- 該当するデバイス用のWind River Linuxイメージを格納したUSBフラッシュドライブを用意しました。詳細については、「ターゲットシステムイメージのUSBフラッシュドライブへのロード（P.6）」を参照してください。
- 対応するデバイスとそのための電源を準備して下さい。詳細は「リリース情報（P.21）」をご参照ください。
- お使いのハードウェアデバイスに適したキーボードとマウスがあるかをご確認ください。
- ハードウェアからディスプレイデバイスまでシリアル接続されているかをご確認ください。
- 本機がイーサネットネットワークに接続されているかをご確認ください。
- (オプション）ボードからモニターへのHDMIやDisplayPortなどのビデオ接続をご確認ください。

手順

1. ハードウェアデバイスとLinuxホストシステムの接続を設定します。

   以下のイメージは、フルターゲットイメージに必要な接続を示しています。minimal imagesでは、Video Displayは不要です。



   a. キーボードとマウス（オプション）を取り付けます。

   b. デバイスからのコマンドライン出力をモニターするために、デバイスからLinuxホストシステムにシリアルポートを接続します。

   c. (オプション）ビデオディスプレイを本機に接続します。

      お使いのハードウェアボードやディスプレイで利用可能なポートに応じて、適切なHDMIまたはDisplayPortアダプターが必要になる場合があります。これはオプションですが、XFCEデスクトップでfull imageを使用する場合は推奨します。

2. ディスプレイの電源を入れます。

3. 本体の電源を入れて、Wind River Linuxを起動します。

   full imageを使用している場合は、XFCE デスクトップで起動します。minimal imageでは、コマンドプロンプトで起動します。

4. ユーザー名に **root** を使用して Wind River Linux システムにログインします。初回のログイン時には、新しいパスワードを設定するよう促されます。

5. (minimal imagesではオプション）XFCEデスクトップをインストールします。

   a. イメージのロックを解除します。

```
# ostree admin unlock --hotfix
```

   b. 必要なXFCEのパッケージをインストールします。

```
# dnf install -y packagegroup-xfce-base packagegroup-core-x11-base gsettings-desktop-schemas w
r-themes
```

   c. XFCEデスクトップをデフォルトに設定します。

```
# systemctl set-default graphical.target
```

   d. システムを再起動します。

      システムが起動すると、XFCEのデスクトップが直接起動されます。

次のステップ

インターネットへの接続を確立した後、インストール可能なアップデートがあるかどうかを確認します。詳細は「デバイスのアップデート（P.18）」をご参照ください。

3.3. QEMUによるターゲットシステムイメージの起動

ターゲットシステムイメージをLinuxホストシステムに展開すると、一時的なストレージを作成してQEMUでイメージを起動することができます。

始める前に

- お使いのLinuxホストシステムには、**/usr/bin/qemu-system-*arch***のバイナリ（最低バージョン2.11）がインストールされているかをご確認ください。

  インストールされていない場合は以下のコマンドを実行して下さい。

  $ sudo apt install qemu-utils

  intel-x86-64の場合：

  $ sudo apt install qemu-system-x86

  bcm-2xxx-rpi4の場合：

  $ sudo apt install qemu-system-arm

- お使いの Linux ホストシステムに Wind River Linux ターゲットシステムイメージをダウンロードして展開してください。詳細については、「バイナリイメージとSDKのダウンロード（P.3）」を参照してください。
- インターネットに接続しているかをご確認ください。

> 注： 現在、QEMUの導入をサポートしているのはintel-x86-64およびbcm-2xxx-rpi4イメージのみです。

手順

1. 8GBのディスクイメージを作成します。

BSP	実行するコマンド
intel-x86-64	$ **qemu-img create -f raw** *path_to* **/boot-image-qemu.hddimg 8G** このコマンドは、指定されたパスに**boot-image-qemu.hddimg**という名前のQEMUディスクイメージを作成します。
bcm-2xxx-rpi4またはその他のARMベースのBSP	$ **qemu-img create -f raw** *path_to* **/img 8G** このコマンドは、指定されたパスに**img**という名前のQEMUディスクイメージを作成します。

2. ダウンロードしたイメージを、前のステップで作成した8GBのディスクイメージにコピーします。

BSP	実行するコマンド
intel-x86-64	$ **zcat wrlinux-***imageType* **-intel-x86-64.ustart.img.gz | sudo dd of=***path_to* **/boot-image-qemu.hddimg conv=notrunc**

bcm-2xxx-rpi4	$ **zcat wrlinux-***imageType* **-bcm-2xxx-rpi4.ustart.img.gz | sudo dd of=***path_to* **/img conv=notrunc**

これらの例では、コピーしたいイメージに応じて、*imageType*を**full**または**minimal**に置き換えてください。

3. 前のステップでコピーしたイメージをデプロイします。

BSP	実行するQEMUコマンド
intel-x86-64	**オプション1 – KVMあり**（推奨） **$ /usr/bin/qemu-system-x86_64 -net nic -net user -m 512 ** **-drive if=none,id=hd,file=*path_to* /boot-image-qemu.hddimg,forma t=raw ** **-device virtio-scsi-pci,id=scsi -device scsi-hd,drive=hd ** **-cpu kvm64 -enable-kvm ** **-drive if=pflash,format=qcow2,file=ovmf.qcow2** この例では、kvm オプションを使用してブート時間を短縮し、ターゲットの応答性を高めています。このオプションを使用するには、Linux ホストシステムの **/dev/kvm** へのアクセス権が必要です。詳細については、https://wiki.yoctoproject.org/wiki/ How_to_enable_KVM_for_Poky_qemuを参照してください。 **オプション2 - KVMなし** **$ /usr/bin/qemu-system-x86_64 -net nic -net user -m 512 ** **-drive if=none,id=hd,file=*path_to* /boot-image-qemu.hddimg,forma t=raw ** **-device virtio-scsi-pci,id=scsi -device scsi-hd,drive=hd ** **-cpu Nehalem ** **-drive if=pflash,format=qcow2,file=ovmf.qcow2**
bcm-2xxx-rpi4	注　このBSPは、QEMUでARMを正常に起動するために、**qemu-u-boot-bcm-2xxx-rpi4.bin** BIOSファイルを必要とします。 **$ /usr/bin/qemu-system-aarch64 -machine virt -cpu cortex-a57 ** **-device virtio-net-device,netdev=net0 -netdev user,id=net0 ** **-m 512 ** **-bios *path_to* /qemu-u-boot-bcm-2xxx-rpi4.bin ** **-nographic ** **-drive id=disk0,file=*path_to* /img,if=none,format=raw -device virtio-blk-device,drive=disk0**

QEMUのコマンドとその意味についての詳細は、オンラインのREADMEファイルをご参照ください。

- target_intel-x86-64.README.md
- target_bcm_2xxx-rpi4.README.md

full imageを使用している場合は、XFCE デスクトップで起動します。minimal imageでは、コマンドプロンプトで起動します。

4. ユーザー名に **root** を使用して Wind River Linux システムにログインします。初回のログイン時には、新しいパスワードを設定するよう促されます。
5. (最小限のイメージではオプション）XFCEデスクトップをインストールします。
   a. イメージのロックを解除します。

```
# ostree admin unlock --hotfix
```

b. 必要なXFCEのパッケージをインストールします。

```
# dnf install -y packagegroup-xfce-base packagegroup-core-x11-base gsettings-desktop-schemas w
r-themes
```

c. XFCEデスクトップをデフォルトに設定します。

```
# systemctl set-default graphical.target
```

d. システムを再起動します。

システムが起動すると、XFCEのデスクトップが直接起動します。

次のステップ

QEMUが起動したら、インストール可能なアップデートがあるかどうかを確認します。詳細については、「デバイスのアップデート（P.18）」をご参照ください。

4. コンテナイメージ クイックスタート

4.1. Dockerを使ってコンテナイメージをデプロイする

コンテナイメージをLinuxホストシステムにダウンロードし、Dockerfileを作成すると、Dockerでイメージを起動する準備が整います。

始める前に

- LinuxホストシステムにDocker Engineがインストールされている必要があります。

  詳しい情報は、https://docs.docker.com/ engine/install/をご覧ください。
- Linux ホストシステムは、コンテナイメージと同じアーキテクチャである必要があります。例えば、IAベースのホストでは、x86-64ベースのコンテナイメージのみを実行する必要があります。
- インターネットに接続しているかを確認します。

手順

1. 起動したいコンテナイメージタイプのdocker pullコマンドを実行します。

   以下の例では、*imageType*を**full**または**minimal**に、*version*をリリースバージョンなどに置き換えてください。
   10.21.20.0.

BSP	実行するコマンド
bcm-2xxx-rpi4	$ **docker pull windriver/wrlx-image:***imageType* **-rpi4-***version*
intel-x86-64	$ **docker pull windriver/wrlx-image:***imageType* **-x86-64-***version*

2. イメージがDockerに追加されていることを確認します。

```
$ docker images
REPOSITORY              タグ                IMAGE ID       CREATED          SIZE
windriver/wrlx-image    Full-Rpi4-10.20.30.0  ff67e356d9aa   1秒未満前        189MB
```

   コマンドの出力に新しいイメージが表示されます。この例では、完全な rpi4 イメージが Linux ホストシステムに追加されています。これで、デプロイの 準備が整いました。

3. このイメージをDockerコンテナで実行します。

   次の例では、*imageType*は、前のステップのイメージタグの名前を指しています。これには、イメージタイプ（fullまたはminimal）、アーキテクチャ（rpi4またはx86-64）、ビルド日が含まれます。

```
$ docker run -i -t windriver/wrlx-image:imageType /bin/bash
```

例えば、full rpi4イメージを起動するには、以下のコマンドを実行します。

```
$ docker run -i -t windriver/wrlx-image:full-rpi4-10.21.20.0 /bin/bash
```

新しいイメージが起動し、コマンドプロンプトが表示されます。

この例では、**-I** および **-t** オプションを使用すると、擬似ターミナルを使用して、従来の仮想マシンのようにコンテナを対話形式で実行することができます。すべてのオプションの詳細については、Dockerウェブサイトのドキュメントをご参照ください。

次のステップ

コンテナが起動したら、インストール可能なアップデートがあるかどうか確認してください。詳細は、「デバイスのアップデート（P.18）」をご参照ください。

5. SDK イメージクイックスタート

5.1. SDKのインストール

SDKを使用する前に、SDKをインストールし、開発環境を整える必要があります。

本作業について

ウインドリバーは、標準の Yocto Project SDK をベースにした Wind River Linux 用の SDK を提供しています。Yocto Project SDK の使用に関する追加情報については、 *Yocto Project Mega Manual: Yocto Project Application Development and the Extensible Software Development Kit (eSDK)*をご参照ください。

本SDKは、以下のバイナリイメージタイプのアプリケーションを開発するために使用します。

- container/wrlinux-image-minimal
- container/wrlinux-image-full
- target/wrlinux-image-minimal
- target/wrlinux-image-full

始める前に

- 推奨Linuxホストまたは同等の構成の他のLinuxホストを使用してください。詳細については、*Wind River Linux Release Notes*：Host System Recommendations and Requirements をご参照ください。
- 開発したいBSPアーキテクチャのSDKをダウンロードしました。詳しくは、「バイナリイメージとSDKのダウンロード（P.3）」をご覧ください。

手順

1. SDKのインストーラスクリプトに実行権を与えます。

```
$ chmod a+x wrlinux-*-wrlinux-image-full-sdk.sh
```

2. SDKをインストールします。

```
$sudo ./wrlinux-*-wrlinux-image-full-sdk.sh

Wind River Linux SDK installer version 21.20
==========================================================
Enter target dorectory for SDK (default /opt/windriver/wrlinux-graphics/21.20):
You are about to install the SDK to "/opt/wrlinux-graphics/21.20" - Proceed[Y/n]? Y
Extracting SDK.....................................................................................
.....................................................................................................
.............done
Setting it up.....done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment set
up script e.g.
 $ . /opt/wrlinux-graphics/21.20/environment-setup-corei7-64-wrs-linux
```

> 注： この出力は、インテル・アーキテクチャ・ベースのSDKをインストールした場合の結果です。他のアーキテクチャの場合は、出力が異なります。

3. SDKの環境を整えます。

   この手順は、SDK開発のために新しいターミナルを開くたびに行う必要があります。

   ```
   $ . environment-setup-*-wrs-linux
   ```

   エクスポートされた環境変数の一覧を表示するには、**Environment-setup-*-wrs-linux** ファイルをエディターで開きます。

次のステップ

これで、SDKがインストールされ、アプリケーションを開発する準備が整いました。サンプルの手順については、「Hello Worldサンプルアプリケーションの作成（P.15）」をご覧ください。

5.2. Hello Worldサンプルアプリケーションの作成

SDKを使用して、Wind River Linux SDKで使用するサンプルアプリケーションを作成します。

本作業について

この手順では、ソースコードとMakefileを使ってHello Worldアプリケーションのサンプルを作成します。

始める前に

- 推奨Linuxホストまたは同等の構成の他のLinuxホストを使用してください。詳細については、*Wind River Linux Release Notes*：Host System Recommendations and Requirements を参照してください。
- Linux ホストシステムに Wind River Linux SDK をインストールください。詳細については、「SDKのインストール（P.14）」をご参照ください。

手順

1. Linuxホストシステム上にアプリケーションの作業ディレクトリを作成し、そこに移動します。この場所を*appDir*と呼ぶことにします。
2. **hello.c**のソースファイルをテキストエディタで設定します。
   a. **hello.c**ファイルを作成します。

      以下は、テキストエディターviを使った例です。

      ```
      $ vi hello.c
      ```

   b. 以下の文章を入力またはコピーして、ファイルを保存します。

      ```
      #include <stdio.h>

      int main(void)
      {
              printf("Hello World with Wind River Linux!\n");
              return 0;
      }
      ```

3. テキストエディタで**Makefile**を設定します。

   以下は、テキストエディターviを使った例です。

   a. **Makefile**ファイルを作成します。

   ```
   $ vi Makefile
   ```

   b. 以下の文章を入力またはコピーして、ファイルを保存します。

   ```
    # Generated Makefile for "hello-Linux"
   # No selected build spec
   PROJ_DIR=$(shell pwd)
   CFLAGS?=-c

   AINCLUDES= SOURCES=hello.c

   OBJECTS=$(SOURCES:.c=.o)

   hello_Linux=hello_Linux

   all: $(hello_Linux) install

   $(hello_Linux):
           $(CC) $(CFLAGS) $(CINCLUDES) -c -o hello.o hello.c
           $(CC) $(LFLAGS) $(LINCLUDES) -o $(hello_Linux) hello.o

   install :
           mkdir -p  $(PROJ_DIR)/install/usr/bin
           cp $(hello_Linux) $(PROJ_DIR)/install/usr/bin
           chmod 755 $(PROJ_DIR)/install/usr/bin/$(hello_Linux)

   .PHONY: clean

   clean :
           @rm -f *.o
           @rm -f $(hello_Linux)
           @rm -rf $(PROJ_DIR)/install
           @echo Directory Cleaned!

   clean-wb :
           @for i in `ls -d */*/Debug 2> /dev/null`; do
             \ d=`dirname $$i`; d=`dirname $$d`; rm -rf $$d; \
           done;

           @for i in `ls -d */*/NonDebug 2> /dev/null`; do
             \d=`dirname $$i`; d=`dirname $$d`; rm -rf $$d; \
           done;
   ```

4. SDKの環境を整えます。

   ```
   $ . environment-setup-*-wrs-linux
   ```

5. アプリケーションを構築します。

```
$ make
```

コマンドが完了すると、プロジェクトディレクトリにはhello_Linuxのバイナリと、例えば**インストール**ディレクトリが含まれます。

```
$ ls
hello.c hello_Linux hello.o install Makefile
```

6. hello_Linuxアプリケーションのバイナリを、ターゲットシステムイメージ上の **/usr/bin** にコピーします。
7. Wind River Linuxにログインして、アプリケーションをテストします。

```
# hello_Linux
Hello World with Wind River Linux!
```

6. バイナリイメージの更新

Wind River Linuxのバイナリイメージを使用する利点の一つは、アップデートを行う機能が各イメージにデフォルトで含まれていることです。

本作業について

Wind River Linuxのバイナリイメージは、2種類のアップデートをサポートしています。

DNFによるオンターゲット・パッケージ・アップデート

このアプローチでは、Web リポジトリを使用して、デプロイされたターゲットデバイスで利用可能なユーザースペースパッケージを維持します。詳細については、「*Wind River Linux Platform Developer's Guide*：*Run Time Package Management Considerations*」をご参照ください。

OSTreeによるシステムレベルのバイナリ更新

この方法では、デュアルパーティション方式のGitリポジトリを利用して、再イメージの必要なしにシステムレベルのバイナリを更新します。intel-x86-64 イメージでは、**/boot/efi** パーティションが読み取り専用になっているため、システムレベルのアップグレードには後述の **ostree_upgrade.sh** スクリプトを使用する方法しかありません。詳細については、「*Wind River Linux Platform Developer's Guide*：*OSTree Implementation Considerations*」をご参照ください。

始める前に

Wind River Linuxのバイナリイメージでアップデートを実行するには、まずイメージをダウンロードし、ハードウェアボード、QEMU、またはDockerを使ってデプロイする必要があります。詳細については、「ハードウェア上でのターゲットシステムイメージの起動（P.6）」、「QEMUによるターゲットシステムイメージの起動（P.8）」、「Dockerを使ってコンテナイメージをデプロイする（P.12）」をご参照ください。

手順

1. Wind River Linuxのバイナリイメージが起動し、rootとしてログインしていることを確認します。
2. OSTreeアップグレードのラッパースクリプトを実行して、システムをアップグレードします。

> 注： この手順は、OSTreeリポジトリに利用可能なアップデートがある場合にのみ必要です。この手順は、システムイメージでのみサポートされており、コンテナイメージではサポートされていません。

このスクリプトは、カーネルと関連するシステムファイルが最新の状態であることを確認します。さらに、OStree の管理コマンドをラップし、**/sysroot/ostree/repo/config** で設定された指定のブランチを取得するために、シングルまたはマルチ パーティション方式でアップグレードを実行します。

これは、OSTree リポジトリで利用可能なシステムレベルの変更を更新するための同じコマンドです。

```
$ ostree_upgrade.sh
Validating refs...
Validating refs in collections...
Enumerating objects...
Verifying content integrity of 1 commit objects...
fsck objects (5629/5629) [=============] 100%
Validating refs...
Validating refs in collections...
Enumerating objects...
Verifying content integrity of 1 commit objects...
fsck objects (5629/5629) [=============] 100%

GPG: Verification enabled, found 1 signature:
  Signature made Fri July 16:06:14 2020 using RSA key ID CFA856DFC7CB87BE
```

```
  Good signature from "Wind-River-Linux-Sample <svc-linux@gmail.com>"
30 metadata, 13 content objects fetched; 89 KiB transferred in 0 seconds
Copying /etc changes: 7 modified, 1 removed, 4 added Bootloader updated;
bootconfig swap: yes; deployment count change: 1
```

3. システムを再起動します。

```
$ reboot
```

再起動が完了したら、イメージにログインし直します。

4. イメージのロックを解除します。

デフォルトでは、イメージはロックされており、いかなるアップデートも許可されていません。パッケージのアップデートを成功させるためには、以下のコマンドを実行する必要があります。

```
$ ostree admin unlock --hotfix
```

5. DNFでパッケージのインストール、アップデート、削除を行います。

注意：DNFによるパッケージアップデートは、システムイメージとコンテナイメージの両方でサポートされています。

更新タイプ	実行するコマンド
インストール	$ **dnf install** *packageName*
削除	$ **dnf remove** *packageName*
パッケージインデックスの更新	DNFパッケージマネージャーは、パッケージのフィードインデックスをローカルにキャッシュします。このコマンドは、新しいフィードインデックスを読み込むために必要です。パッケージ管理ツールはこれらのインデックスを定期的に更新しますが、パッケージフィードの変更をすぐに確認するためには、明示的にキャッシュを更新する必要があります。 $ **dnf update**

パッケージのアップグレード	$ **dnf upgrade** *packageName*

7. リリース情報

各Wind River Linuxバイナリイメージの利用可能な機能とサポート情報を確認します。

イメージと特徴

Wind River Linuxでは、プロジェクトのニーズに合わせて開発を簡素化するために、Wind River Linuxのバイナリターゲットシステムとコンテナイメージを提供しています。これらのイメージには、minimal imageとfull imageの両方があります。それぞれのイメージタイプは、迅速に起動できるように設計されていますが、minimal imageは、起動可能なLinuxシステムを作成するために最低限必要なパッケージを提供するように設計されています。一方、full imageには、多くの便利な機能やパッケージがデフォルトで含まれており、カスタマイズしなくても使えるシステムになっています。

コンテナイメージは、リソースを共有することで、デプロイのコストだけでなく、開発やテストの多くの側面を簡素化します。Wind River Linuxのバイナリコンテナイメージは、Docker Hubを介して直接デプロイすることができ、追加の変更は必要ありません。

full imageタイプとminimalイメージタイプで提供されるパッケージの違いについては、BSPの*imageType* **.manifest**ファイルをご参照ください。このファイルの入手方法については、「バイナリーイメージとSDKのダウンロード（P.3）」をご参照ください。

指定のない限り、サポートされているすべてのBSPの各ターゲットシステムおよびコンテナイメージには、以下の機能が含まれています。

- **OStree** (ターゲットシステムのみ) - OSTreeは、オペレーティングシステム専用の完全なファイル管理システムです。最初のデプロイメントとアップグレードを行うことができます。システムバイナリに特化したGitに似たブランチを使用して、システムのアップデートや破損したファイルシステムの自動ロールバックを提供します。Wind River Linuxには、システムアップグレードを簡素化するためのOStreeが含まれています。

- **XFCEデスクトップ** - コマンドラインを使用するシステムは、お客様の用途によっては必ずしも実用的ではありません。そのため、各フルターゲットシステムイメージには、カスタマイズ性の高いデスクトップであるXFCEを搭載し、お客様の特定の開発要件に対応しています。この機能はminimal imageでも利用できますが、デフォルトではインストールされていません。追加するには、デプロイされたイメージ上のターミナルで次のように入力し、システムを再起動します。

  ```
  $ ostree admin unlock --hotfix
  $ dnf install packagegroup-core-x11 packagegroup-xfce-base
  $ systemctl set-default graphical.target
  ```

- **コンテナ**（ターゲットシステムのみ） - フルターゲットイメージには、Dockerのコンテナサポートがデフォルトでインストールされています。さらに、Kubernetesのフルサポートにより、プロジェクトイメージを計画するための追加オプションが提供されます。
- **パッケージ管理** - ランタイムパッケージ管理では、新しいターゲットイメージをリビルドして再デプロイすることなく、実行中のターゲットに個々のパッケージを追加または更新することができます。Wind River Linuxのバイナリイメージは、DNFによるパッケージ管理をサポートしています。
- **GLIBC** - GLIBCバージョン2.3.1をサポートしています。

SDKイメージには、上記のすべての機能のサポートが含まれています。本SDKは、Linuxホストシステムで使用することができますが、Microsoft Windowsではサポートされていません。

パッケージ情報

各イメージには、デフォルトでDNFによるパッケージ管理機能が搭載されており、パッケージの追加、更新、削除を行うことができるほか、追加のパッケージフィードも含まれています。パッケージのアップデートについては、「バイナリーイメージの更新（P.18）」をご覧ください。

full imageとminimal imageのパッケージのリストは、ダウンロードディレクトリーにある *imageType* **.manifest** ファイルと *imageType* **.tar.bz2** ファイルに含まれています。詳しくは、「バイナリーイメージとSDKのダウンロード（P.3）」をご覧ください。

ソースからのイメージ作成

ウインドリバーでは、イメージやパッケージをビルドできるソースを提供しています。詳細については、BSPに固有の**target**-*bspName* **-README.md**ファイルをご参照ください。このREADMEファイルを入手するための追加情報については、「バイナリイメージとSDKのダウンロード（P.3）」をご覧ください。

ハードウェア機能

サポートされているハードウェア機能は、使用しているBSPとハードウェアによって異なります。Wind River Linuxでは、以下のBSPのバイナリイメージを提供しています。サポートされている各ボードの具体的なハードウェア情報については、Wind River Marketplace（https://bsp.windriver.com/bsps/product/wind-river-linux_lts-21）をご参照ください。

bcm-2xxx-rpi4

Raspberry Pi 4 Model B: BCM2711ボードに対応しています。

intel-x86-64

以下のボードに対応しています。

- Intel Harcuvar platform: Atom Pocessor, Denverton-NS SoC, (Harcuvar)
- Intel NUC Platform: KabyLake-U Processor, Sunrise Point-LP PCH, (NUC7i5DNK1E)
- Intel Coffee Lake-S: CoffeeLake Processor, Cannon Point PCH (Coffee Lake S)
- Intel Purley 4-Socket Server Refresh: Intel Cascade Lake Processor

intel-socfpga-64

Intel Stratix 10 SoCボードに対応しています

marvell-cn96xx

Marvell CN96xx CRBボードに対応しています。

nxp-imx8

i.MX 8QuadMaxボードのimx8qm-mekバージョンに対応しています。

nxp-s32g2xx

以下のボードに対応しています。

- NXP_S32G274A_EVB
- NXP_S32G274A_RDB2

ti-j72xx

Texas Instruments のJ721E-EVMボードをサポートしています。

xilinx-zynq

以下の種類のXilinx ZYNQ-7000ボードをサポートしています。

- ZYNQ-ZC702
- ZYNQ-ZC706

xilinx-zynqmp

ZCU102ボードに対応しています。

.

対応する外部メモリデバイス

メモリデバイスの種類	詳細
USBフラッシュドライブ	Wind River Linuxのインストールには、通常8GB以上のフラッシュドライブを使用します。 イメージを保存するのに十分な容量と、アプリケーションの追加やシステムログファイルの保存、システムのアップデートによる拡張に備えて、十分な容量があるものを選びましょう。

使用上の注意事項

コンテナでの**ecryptfs.service**の使用

コンテナ内の **ecryptfs.service** を正常に起動するためには、**ecryptfs.service** を使用するか、手動でまずホストシステムに **ecryptfs.ko** カーネルモジュールを挿入する必要があります。コンテナ内で実行されたら、ホストシステムの **ecryptfs.service** を停止する必要があります。これに失敗すると、**/dev/ecryptfs** のパーミッションが保持され、コンテナ内で実行されている **ecryptfs.service** が Linux ホストシステム上の **/dev/ecryptfs** にアクセスできなくなります。

バイナリの配布：intel-x86-64 DNFアップデートの失敗：/boot/efiファイルシステムへのパッケージのインストール (LIN1021-449)

intel-x86-64 バイナリイメージを最初に起動すると、現在のところ、DNF パッケージマネージャを使用してシステムパッケージを更新することができません。これは、**/boot/efi** パーティションが読み取り専用であることが原因です。システムパッケージをアップデートまたはインストールする前に、イメージをアンロックして、「*Wind River Linux Binary Release Quick Start*： *Updating Binary Image*」で説明されているように**ostree_upgrade.sh**スクリプトを使用する必要があります。アップグレードが完了したら、必要に応じて DNF を使用することができます。

nxp-s32g2xx ボードのustartイメージが起動できず、ネットブートに入る(LIN1021-315)

この問題は、最新のカーネルとの間でメモリアドレスの同期が取れていないために発生します。ブートプロセスを継続するには、u-bootコンソールで以下のように入力してください。

$ **env default -a; setenv ramdisk_addr 0x82000000; setenv initrd_addr 0x82000000; saveenv; boot**

ターゲットイメージでネットワークサービスが有効になっていない（LIN1021-342）

Linux Assembly Toolのgenimageコマンドオプションを使用してターゲットシステムのイメージを作成した場合、初期イメージではデフォルトでネットワークが有効になっていません。ネットワークを有効にするには、以下のコマンドで新しいイメージを生成します。

```
$ appsdk genimage --pkg dhcpcd --rootfs-post-script='systemctl
--root=$IMAGE_ROOTFS enable dhcpcd.service'
```

詳細については、『*Wind River Linux Binary Release Developer's Guide*』：*Linux Assembly Tool and Install* をご参照ください。

8. 追加情報の入手先

Wind River Linuxプラットフォームのプロジェクトイメージが完成したので、さらに開発を進めたいと思うかもしれません。詳細については、以下の場所をご参照ください。

Wind River Linux のドキュメント

すべてのドキュメントは、https://docs.windriver.comからオンラインで入手できます。

All Products > Wind River Linuxを選択し、「Wind River Linux LTS 21」を選択します。

サポートされているBSP、ワークフロー、製品のアップデートなど、一般的な製品情報については、Wind River Linux Getting Startedをご参照ください。

Wind River Linux の RSS フィードを購読するには、お使いの RSS リーダーに http://www.windriver.com/feeds/wrlinux_lts.xml を指定してください。
バイナリ・リリースの管理については、*Wind River Linux Binary Release Developer's Guide* をご参照ください。

外部ドキュメント

Wind River Linux は、OpenEmbedded Core Project をベースにした Yocto Project と同じビルドシステムを共有しています。そのため、Yocto Projectのドキュメントを参考にして、プラットフォーム・プロジェクト・イメージをさらに発展させることができます。

まずは、Yocto Project ビルドシステムを使ってプラットフォームプロジェクトのビルドを開発する際に使われる用語を理解することから始めましょう。詳しくは、*Yocto Project Mega Manual* : *Yocto Project Terms*をご覧ください。

外部の情報源	URL
Yocto Project	http://www.yoctoproject.org
Yocto Project Mega-Manual	https://docs.yoctoproject.org/singleindex.html
BitBake User Manual	https://docs.yoctoproject.org/3.3/bitbake.html
OpenEmbedded Core (OE-Core)	http://www.openembedded.org/wiki/OpenEmbedded-Core
QEMU	http://wiki.qemu.org