文書番号：014-0001-0025

ubinux
スタートアップガイド

2012 年 2 月 29 日

1.4 版

前書き

本マニュアルは、ubinux のインストール方法について説明しています。ubinux とは、ubinux カーネル、ubinux カーネル上で動作する ubinux ミドルウェア、ubinux をインストールするためのインストーラの総称です。

本マニュアルの内容は、ubinux 特有の内容に限定しています。Linux に関する基礎知識などについては Linux に関する書籍や Web サイトに記載されている情報などを参考にしてください。

本マニュアルの読者

本マニュアルは、ubinux の導入を検討されている方や、ubinux の導入作業を行う方を対象としています。

本マニュアルを参照することによって、ubinux の概要やインストール手順に関する情報を得ることができます。

輸出管理規制

本資料には、外国為替及び外国貿易管理法に基づく特定技術が含まれています。従って、本マニュアルまたはその一部を輸出する場合には、同法に基づく許可が必要となります。

お願い

- 本マニュアルの全体、または一部を無断で転載、複写することを禁じます。
- 本マニュアルの内容は予告なく変更されることがあります。

商標について

Linux は、Linus Torvals 氏の米国及びその他の国における登録商標あるいは商標です。

Red Hat、RPM、Red Hat をベースとした全ての商標とロゴ及び Fedora は、Red Hat, Inc. の米国及びその他の国における登録商標あるいは商標です。

ubinux は、富士通コンピュータテクノロジーズの日本における登録商標です。

その他の会社名及び製品名は、それぞれの会社の登録商標あるいは商標です。

改版履歴

版数	更新日	作成者	内容	更新箇所
1.0	2008/12/26	上羽	・下記マニュアルを統合してスタートアップガイドとして新規に作成。 －「組込み Linux 導入ガイド」 －「組込み Linux 設定ガイド」 －「組込み Linux カーネル ユーザーズマニュアル」	
1.1	2010/2/5	浅羽	・V10 向けに更新。	
1.2	2010/7/23	近沢	・P2020 追加の為更新。	
1.3	2011/2/28	森	・V11 向けに更新。	
1.4	2012/2/29	森	・V12 向けに更新。	

目次

1. 概要

本章では、ubinux の特長、クロス開発環境の構成、ホスト PC の要件、そして ubinux の構築手順の概要について説明します。

1.1. 特長

ubinux には、次のような特長があります。

- 豊富な最新パッケージ群
  組込み装置向けに選別した豊富な最新版のパッケージ群を提供していますので、独自にパッケージを調達する作業が不要になります。
- 小サイズ化済パッケージ群
  組込み装置向けに不要なモジュールやファイルを取り除いた状態のパッケージ群を提供しています。
- 著作権調査済パッケージ群
  提供する全てのパッケージ群は著作権の調査がなされたものですので、独自に著作権を調査する作業が不要になります。
- タイムリーなセキュリティ脆弱性対応
  パッケージ群に含まれるセキュリティ脆弱性に対して、タイムリーに対応しています。
- カスタマイズが容易
  バイナリパッケージと同時にソースパッケージも提供していますので、パッケージ群を独自にカスタマイズすることが可能です。

1.2. クロス開発環境の構成

ubinux では、開発環境の種別として、クロス開発環境をサポートしています。
クロス開発環境では、以下の 2 つの装置を使用します。

- ターゲット装置
  ubinux を動作させる装置です。
- ホスト PC
  ubinux 用のルートファイルシステムの構築、アプリケーションの開発、リモートデバッグなどを行う装置です。

クロス開発環境の構成例を図 1-1 に示します。
図 1-1　クロス開発環境の構成例では、ターゲット装置が PowerPC アーキテクチャの評価ボード、ホスト PC が x86 アーキテクチャの PC という構成になっています。

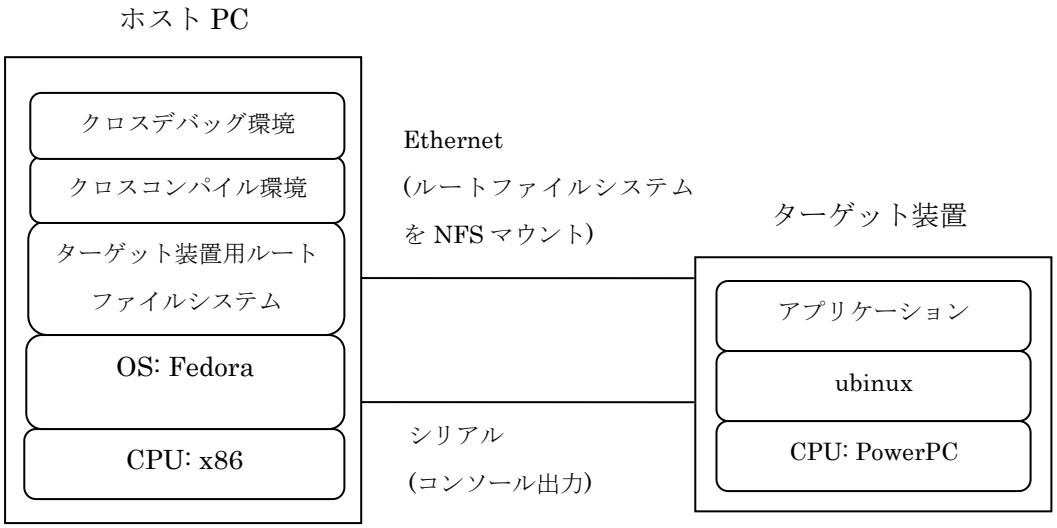


図 1-1　クロス開発環境の構成例

1.3. ホスト PC の要件

ホスト PC の要件を表 1-1 に示します。

表 1-1　ホスト PC の要件

種別	要件	備考
装置	IBM PC/AT 互換機	
OS	・Fedora 16 (i386, x86_64) ・Fedora 13 (i386) ・CentOS 6.2 (i386, x86_64) ・CentOS 5.7 (i386) ・Red Hat Enterprise Linux 6.2 (i386, x86_64) ・Red Hat Enterprise Linux 5.7 (i386)	
メモリ	1GB 以上	
HDD 空き容量	80GB 以上	OS のインストールに必要な容量含む。
通信手段	(ⅰ) Ethernet インターフェース (ⅱ) シリアルインターフェース	(ⅰ) ターゲット装置のルートファイルシステムを NFS マウントする際に必要。 (ⅱ) 開発時のコンソール出力、リモートデバッグ時に必要。

1.4. ホスト OS のインストール

ホスト OS のインストール時に選択するパッケージを示します。

ubinuxlive を使用する場合は「ubinuxlive インストールマニュアル」を参照して下さい。

1.4.1. Fedora 16

Fedora 16 (i386, x86_64) インストール時に選択するパッケージは、**図 1-2 Fedora 16 (x86_64) のインストール画面**のように『ソフトウェア開発』を選択してください。

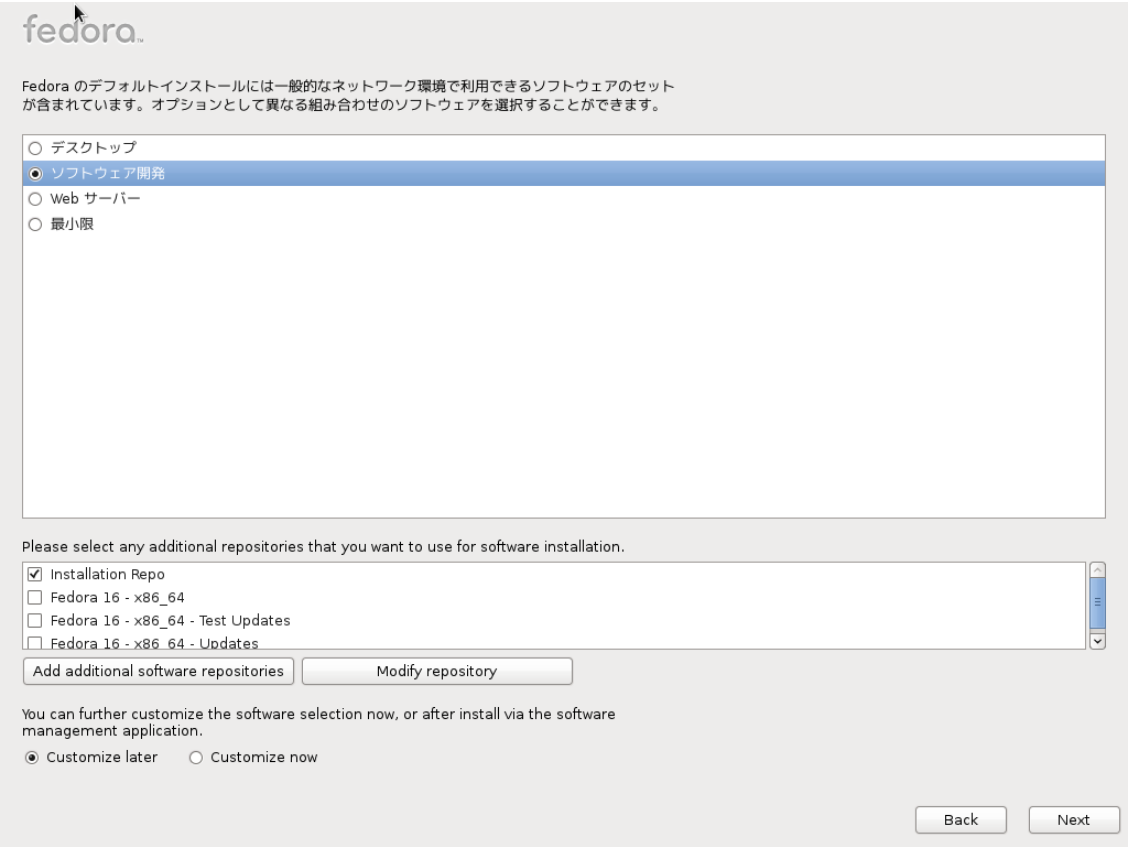


図 1-2 Fedora 16 (x86_64) のインストール画面

1.4.2. Fedora 13

Fedora 13 (i386) インストール時に選択するパッケージは、**図 1-3 Fedora 13 (i386) のインストール画面**のように『ソフトウェア開発』を選択してください。

図 1-3 Fedora 13 (i386) のインストール画面

1.4.3. CentOS 6.2

CentOS 6.2 (i386, x86_64) インストール時に選択するパッケージは、**図 1-4 CentOS 6.2 (x86_64) のインストール画面**のように『Software Development Workstation』を選択してください。

図 1-4 CentOS 6.2 (x86_64) のインストール画面

『Software Development Workstation』を選択すると、『仮想化』に関するパッケージもインストールします。弊社確認時は、ネットワーク設定の煩雑化を避けるため、**図 1-4 CentOS 6.2 (x86_64) のインストール画面**のように『今すぐカスタマイズ』を選択し、『⇒次(N)』を押し、**図 1-5 CentOS 6.2 (x86_64) のインストール画面**で、『仮想化』メニューのチェックを外しています。

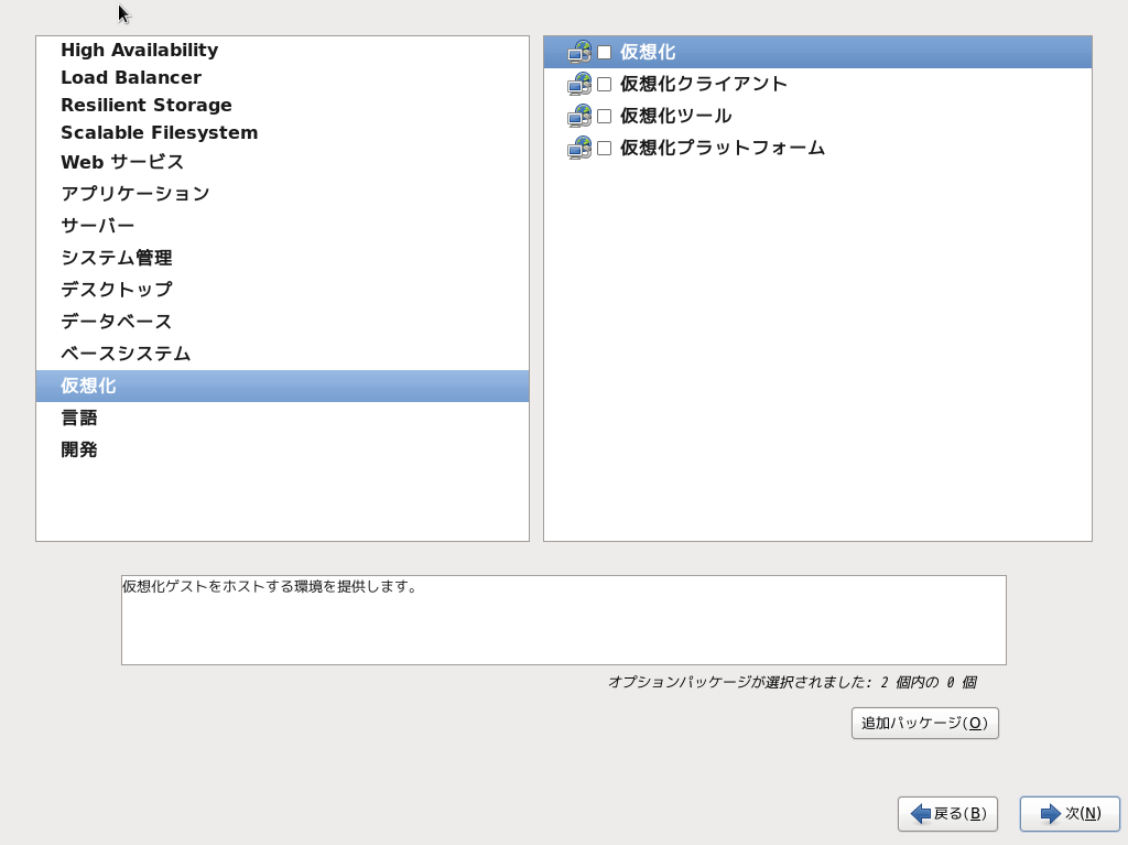


図 1-5 CentOS 6.2 (x86_64) のインストール画面

1.4.4. CentOS 5.7

CentOS 5.7 (i386) インストール時に選択するパッケージは、**図 1-6 CentOS 5.7 (i386) のインストール画面**のように『今すぐカスタマイズ』を選択し『⇒次(N)』を押してください。

図 1-6 CentOS 5.7 (i386) のインストール画面

図 1-7 CentOS 5.7 (i386) のインストール画面に移行しますので、サブメニューの中から『開発ツール』と『開発ライブラリ』を選択してください。

図 1-7 CentOS 5.7 (i386) のインストール画面

1.5. 構築手順の概要

ubinux の構築手順を以下に示します。

(1) ホスト PC 上での構築手順

(1-a) ubinux インストーラと ubinux ミドルウェアのインストール

(1-b) ubinux カーネルのビルド、インストール

(2) ターゲットの起動

(2-a) ローカルデバイス方式による起動、NFS マウント方式による起動

(2-b) ubinux の起動

(3) ターゲット装置上での構築手順

(3-a) 初期設定済アカウントの変更

(3-b) 各種サービスプログラムの設定

(3-c) 各種パッケージの設定

2. ubinux のインストール

本章では、ubinux のインストール方法について説明します。

2.1. ubinux インストーラと ubinux ミドルウェアのインストール

本節では、ubinux インストーラと ubinux ミドルウェアのインストール方法について説明します。

ubinuxlive では、ubinux インストーラと全ての ubinux ミドルウェアのパッケージをインストール済みの為、下記の実施は不要です。再インストールなどを実施する場合に参照して下さい。

2.1.1. ubinux ミドルウェアアーカイブの展開

ubinux ミドルウェアアーカイブ *Version-PROC*-packages_*Number*.tar.gz をホスト PC の任意のディレクトリに配置し展開します。*Version* は、ubinux 版数です。*PROC* は、プロセッサ識別名です。プロセッサ識別名の一覧は表 2-1 を参照ください。*Number* は、ビルド番号です。

```
$ tar xvzf Version-PROC-packages_Number.tar.gz
```

上記コマンドを実行すると展開したディレクトリ配下に rpms ディレクトリが配置されます。rpms の配下には、ターゲット装置用の ubinux ミドルウェアを含む全てのバイナリパッケージファイルが格納されています。

2.1.2. ubinux インストーラアーカイブの展開

ubinux インストーラアーカイブ *Version-PROC*-installer_*Number*.tar をホスト PC の任意のディレクトリに配置し展開します。*Version* は、ubinux 版数です。*PROC* は、プロセッサ識別名です。プロセッサ識別名の一覧は表 2-1 を参照ください。*Number* は、アーカイブ番号です。

```
$ tar xvf Version-PROC-installer_Number.tar
```

上記コマンドを実行すると展開したディレクトリ配下に ubinux インストーラとインストールスクリプトが配置されます。

2.1.3. ubinux インストーラのインストール

ubinux インストーラ ubinux_pkg_tools-*Version*.i686.rpm をホスト PC の任意のディレクトリに配置し展開します。*Version* は、ubinux インストーラ版数です。

```
# rpm -Uvh ubinux_pkg_tools-Version.i686.rpm
```

上記コマンドを実行すると/opt ディレクトリ配下に ubinux_pkg_tools ディレクトリが配置されます。ubinux_pkg_tools の配下には、ubinux インストーラが格納されています。

上記コマンドは root 権限が必要となります。

2.1.4. ubinux ミドルウェアのインストール

インストールスクリプト install-ubq-*PROC*.sh をホスト PC の任意のディレクトリに配置します。*PROC* は、プロセッサ識別名です。プロセッサ識別名の一覧は表 2-1 を参照ください。

インストールスクリプトを実行し、ubinux パッケージインストーラ(ubinux_pkg_install)の起動を行います。

```
# ./install-ubq-PROC.sh -p ubinux ミドルウェアアーカイブ展開ディレクトリ/rpms
```

上記コマンド例は ubinux ミドルウェアアーカイブ展開後のディレクトリで、-p オプションをつけて実行しています。ubinux ミドルウェアアーカイブの展開ディレクトリ／インストールログ出力先を変更する場合は、引数をつけてインストールスクリプトを実行してください。

```
"./install-ubq-PROC.sh [オプション] ディレクトリのパス..."
    : オプション
    :       -p  ubinux ミドルウェアアーカイブの展開ディレクトリの rpms を
                絶対パスで指定
    :       -o  インストールログ出力先を絶対パスで指定
```

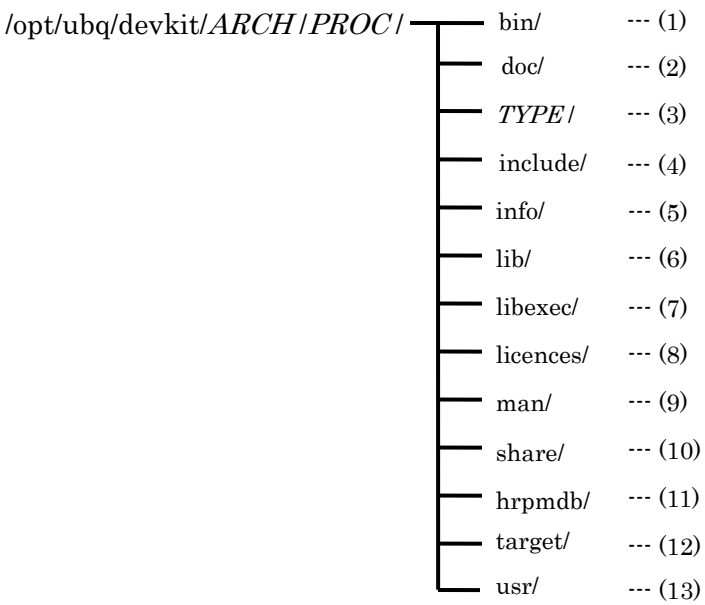
引数なしで実行することで、カレントディレクトリの rpms ディレクトリをパッケージ展開ディレクトリとして動作します。実行時のカレントディレクトリにインストールログを出力します。

上記コマンドは root 権限が必要となります。

インストールスクリプトは、ubinux ミドルウェアのインストールを簡略化する為に用意しています。インストールスクリプトを使用せず、ubinux パッケージインストーラ(ubinux_pkg_install)を起動することで豊富なオプションを指定することが可能です。

インストーラの操作方法については「ubinux インストーラマニュアル」の「パッケージインストーラの起動」、「キャラクタ・ユーザ・インターフェース」を参照して下さい。

インストールを完了すると以下のディレクトリに ubinux ミドルウェア、クロスツールチェーンがインストールされます。



上記の *ARCH*、*PROC*、*TYPE* は、それぞれ、アーキテクチャ識別名、プロセッサ識別名、ターゲット識別名です。これらの識別名の一覧を表 2-1 に示します。

(1)～(10)には、クロスツールチェーンがインストールされます。クロスコンパイラなどのバイナリファイルは(1)に配置されます。このディレクトリを環境変数 PATH に追加してください。

(11)には、インストール済みのバイナリパッケージファイルに関する情報が格納されます。

(12)には、ターゲット装置用パッケージがインストールされます。

(13)には、ホスト用パッケージがインストールされます。

ターゲット装置用ルートファイルシステムの作成については「ubinux インストーラマニュアル」を参照ください。

表 2-1　識別名一覧

アーキテクチャ	CPU ファミリ	プロセッサ	アーキテクチャ識別名	プロセッサ識別名	ターゲット識別名
x86	x86	Atom N280	x86	x86	i686-unknown-linux-gnu
	x86_64	Xeon 5260	x86	x86_64	x86_64-unknown-linux-gnu
PowerPC	PowerQUICC II Pro	MPC8313E	powerpc	e300	powerpc-e300-linux-gnu
		MPC8377E	powerpc	e300	powerpc-e300-linux-gnu
	QorIQ	P1020	powerpc	e500v2	powerpc-e500v2-linux-gnuspe
		P2020	powerpc	e500v2	powerpc-e500v2-linux-gnuspe
		P4080	powerpc	e500mc	powerpc-e500mc-linux-gnu
ARM	ARMv7	OMAP4	arm	armv7	arm-armv7-linux-gnueabi

2.2. ubinux カーネルのビルド、インストール

本節では、ubinux カーネルをビルドする手順について説明します。

2.2.1. ubinux カーネルアーカイブの展開

ubinux カーネルアーカイブ *Version*-kernel_*Number*.tar.bz2 をホスト PC の任意のディレクトリに配置し展開します。*Version* は、ubinux 版数です。*Number* は、アーカイブ番号です。

```
$ tar jxvf Version-kernel_Number.tar.bz2
```

上記コマンドを実行すると展開したディレクトリ配下にubinuxカーネルのソースが配置されます。

2.2.2. 環境変数の設定

ubinux カーネルをビルドする前に各ターゲット装置に合わせて以下の環境変数を設定する必要があります。

・CROSS_COMPILE：クロスコンパイラのプレフィックス名

・ARCH：ターゲットのアーキテクチャ名

各ターゲット装置と設定する環境変数の一覧を表 2-2 に示します。

表 2-2　各ターゲット装置の環境変数一覧

アーキテクチャ	CPU ファミリ	プロセッサ	ARCH	CROSS_COMPILE
x86	x86	Atom N280	x86	i686-unknown-linux-gnu-
	x86_64	Xeon 5260	x86	x86_64-unknown-linux-gnu-
PowerPC	PowerQUICC II Pro	MPC8313E	powerpc	powerpc-e300-linux-gnu-
		MPC8377E	powerpc	powerpc-e300-linux-gnu-
	QorIQ	P1020	powerpc	powerpc-e500v2-linux-gnuspe-
		P2020	powerpc	powerpc-e500v2-linux-gnuspe-
		P4080	powerpc	powerpc-e500mc-linux-gnu-
ARM	ARMv7	OMAP4	arm	arm-armv7-linux-gnueabi-

x86 アーキテクチャのプロセッサを搭載したターゲット装置用に環境変数を設定する場合の例を以下に示します。

```
$ ARCH=x86
$ CROSS_COMPILE=i686-unknown-linux-gnu-
$ export CROSS_COMPILE ARCH
```

2.2.3. コンフィグレーションファイルの読込みと設定内容の変更

ubinux カーネルアーカイブを展開したディレクトリに移動して、各ターゲット装置用のカーネルコンフィグレーションファイルを読込みます(make “コンフィグレーションファイル名”)。その後、カーネルコンフィグレータを起動し、それぞれのターゲット装置に適した設定を行います(make menuconfig)。

各ターゲット装置用のコンフィグレーションファイル名を表 2-3 に示します。

以下に x86 ターゲット装置用のカーネル設定ファイルの読込みとコンフィグレータの起動コマンドを示します。

```
$ make ubq_defconfig          // コンフィグレーションファイル
                              // ubq_defconfig の読込み
$ make menuconfig             // カーネルコンフィグレータの起動
```

表 2-3　各ターゲット装置用コンフィグレーションファイル

アーキテクチャ	ターゲット装置	コンフィグレーションファイル名
x86	x86	ubq_defconfig
	x86_64	ubqx86_64_defconfig
PowerPC	MPC8313E-RDB	83xx/ubqmpc8313erdb_defconfig
	MPC8377E-MDS	83xx/ubqmpc8377emds_defconfig
	P1020RDB	85xx/ubqp1020rdb_defconfig
	P2020RDB	85xx/ubqp2020rdb_defconfig
	P4080DS	85xx/ubqp4080ds_defconfig
ARM	Blaze	ubqomap4_defconfig

make menuconfig を用いてカーネルコンフィグレーションを変更した場合、その設定内容はカーネルトップディレクトリ配下の.config というファイルに保存されます。カーネルの設定を再利用したい場合などは本ファイルを保管してください。(make distclean コマンドでカーネルオブジェクトを削除する際に本ファイルも自動的に削除されます。)

2.2.4. カーネルのビルドとインストール

ターゲット装置で実行するカーネルイメージファイル、モジュールを作成します。

カーネルイメージ生成方法はターゲット装置毎に異なります。表 2-4 の「ターゲット名」を参考にターゲット装置に適したカーネルイメージを生成してください。

以下に x86 ターゲット装置用のカーネルビルドコマンドの例を示します。

この例では、カーネルモジュールはカーネルトップディレクトリの rootfs というディレクトリにインストールされます。

```
$ make bzImage            // カーネルイメージのビルド
$ make modules            // モジュールのビルド
$ make modules_install INSTALL_MOD_PATH=`pwd`/rootfs
                          // モジュールのインストール
```

また上記モジュールのインストールまでを一度の make コマンドで実行することも可能です。その場合は、以下のようにコマンドを実行してください。

```
$ make bzImage modules modules_install INSTALL_MOD_PATH=`pwd`/rootfs
```

カーネルのビルドが完了すると表 2-4 の「カーネルイメージが生成される場所」に示されているディレクトリに「ターゲット名」で示されるカーネルイメージが生成されます。ターゲット装置では、このカーネルイメージを用いて起動することになります。

作成したカーネルモジュールはターゲット装置用ルートファイルシステムの/lib 配下に配置します。以下に x86 ターゲット装置用のコマンド例を示します。

```
# cp -a `pwd`/rootfs/lib/* /opt/ubq/devkit/x86/x86/target/lib
```

上記コマンドは root 権限が必要となります。

表 2-4　各ターゲット装置用ターゲット名とカーネルイメージが生成される場所

アーキテクチャ	ターゲット装置	ターゲット名	カーネルイメージが生成される場所
x86	x86	bzImage	arch/x86/boot/
	x86_64	bzImage	arch/x86/boot/
PowerPC	MPC8313E-RDB	cuImage.mpc8313erdb (※)	arch/powerpc/boot/
	MPC8377E-MDS	uImage (※) (※※)	arch/powerpc/boot/
	P1020RDB	uImage (※) (※※)	arch/powerpc/boot/
	P2020RDB	uImage (※) (※※)	arch/powerpc/boot/
	P4080DS	uImage (※) (※※)	arch/powerpc/boot/
ARM	Blaze	uImage (※)	arch/arm/boot/

(※)のターゲット名は U-Boot の版数に依存します。
弊社で動作確認時に使用した U-Boot の版数は表 3.1 を参照ください。

(※※)が記載されているビルド方法の場合，カーネルを起動するためにカーネルイメージ以外に Device Tree Blob（以下、dtb とします。）というデバイス定義ファイルが必要となります。

dtb はカーネルのビルド後に dtc コマンドでビルドしてください。弊社の動作確認時に実施したコマンドを下記に示します。

・ターゲット装置：MPC8377E-MDS、P1020RDB、P2020RDB

```
$ ./scripts/dtc/dtc -O dtb -I dts -S 0x3000 -o dtb ファイル名 ¥
  ./arch/powerpc/boot/dts/dts ファイル
```

・ターゲット装置：P4080DS

```
$ ./scripts/dtc/dtc -O dtb -I dts -o dtb ファイル名 ¥
  ./arch/powerpc/boot/dts/dts ファイル
```

上記コマンドを実行するとカーネルトップディレクトリに”dtb ファイル名”で指定したファイルが生成されます。

dts ファイルはターゲット装置毎に異なります。下記対応一覧を参照ください。

- MPC8377E-MDS : mpc8377_mds.dts
- P1020RDB : p1020rdb.dts
- P2020RDB : p2020rdb.dts
- P4080DS : p4080ds.dts

dtc コマンドの詳細については-h オプションなどで確認してください。

-S オプションを使用すると下記のメッセージが出力されますが、動作に影響はありません。

```
DTC: Use of "-S" is deprecated; it will be removed soon, use "-p" instead
```

3. ubinux の起動

本章では、HDD やコンパクトフラッシュのようなターゲット装置に接続されたローカルデバイスを用いたブート方式（以下、ローカルデバイス方式とします）と、NFS マウントを用いたブート方式（以下、NFS マウント方式とします）で ubinux を起動する手順について説明します。

3.1. ローカルデバイス方式

3.1.1. ハードウェア構成　ローカルデバイス方式のハードウェア構成の例として、図 3.1 に示す構成をとるものとします。

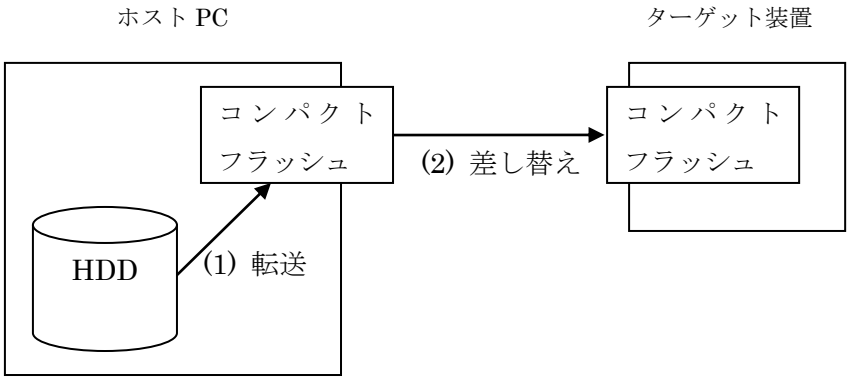


図 3-1 ローカルデバイス方式のハードウェア構成

ここでは、ブートデバイスとして、コンパクトフラッシュを使用しています。

ホスト PC の HDD 上には、ターゲット装置用ルートファイルシステムのデータが格納されているものとします。

3.1.2. コンパクトフラッシュへのデータ転送

ホスト PC の HDD 上に格納されているターゲット装置用のルートファイルシステムのデータをコンパクトフラッシュに転送します。

コンパクトフラッシュにファイルシステムを構築できていて、ホスト PC 上でマウントできる場合には、`cp -a` コマンドなどを使ってターゲット装置用のルートファイルシステムのデータをコピーします。

それ以外の場合には、HDD 上の独立したパーティションに `cp -a` コマンドなどを使ってコピーしておき、`dd` コマンドを使って、そのパーティションの情報をコンパクトフラッシュに転送することが可能です。

3.1.3. コンパクトフラッシュの差し替え

ホスト PC に接続されているコンパクトフラッシュをターゲット装置に差し替えます。

3.2. NFS マウント方式

3.2.1. ハードウェア構成

NFS マウント方式のハードウェア構成の例を図 3-2 に示します。

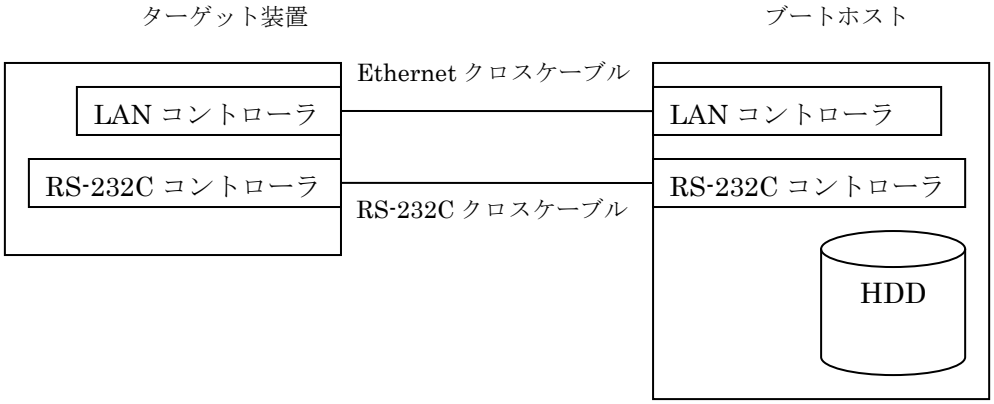


図 3-2 NFS マウント方式のハードウェア構成

ブートホストは、ブート時にターゲットに対してカーネルイメージやルートファイルシステムを供給するための端末です。

3.2.2. ブートホストの設定

ブートホストでは、以下のサービスを起動する必要があります。ご利用の環境に合わせてそれぞれのサービスを設定してください。

(a) TFTP サービス：ターゲット装置にカーネルイメージを転送するために使用します。

(b) NFS サービス ：ターゲット装置のルートファイルシステムを NFS マウントさせるために使用します。
ubinux をデフォルトのディレクトリにインストールした場合、下記ディレクトリを export します。

・/opt/ubq/devkit/*ARCH*/*PROC*/target

(c) DHCP サービス：ターゲット装置のネットワーク情報を DHCP で設定する場合に使用します。

3.3. ubinux の起動

本節では、ターゲット装置で ubinux を起動する方法を説明します。なお、ターゲット装置にはあらかじめブートローダが用意されているものとします。ここでは、組込み装置で主に利用される GRUB、U-Boot 用いた起動方法について説明します。弊社で ubinux の起動時に使用したブートローダを表 3-1 に示します。

表 3-1 各ターゲット装置用ブートローダ一覧

アーキテクチャ	ターゲット装置	ブートローダ
x86	x86	GRUB
	x86_64	GRUB
PowerPC	MPC8313E-RDB	U-Boot (1.3.0)
	MPC8377E-MDS	U-Boot (1.3.0)
	P1020RDB	U-Boot (2009.11)
	P2020RDB	U-Boot (2009.11-00017-ga100cd2)
	P4080DS	U-Boot (2010.12-00007-g4114d25)
ARM	Blaze	U-Boot (1.1.4-g255e28d6-dirty)

3.3.1. GRUB を用いた起動

本節では、bzImage を例に ubinux カーネルの起動方法について述べます。

bzImage を所定のブートディレクトリ(一般的には/boot 配下となります)に配置します。その後、以下の手順で ubinux カーネルを起動します。

(1) grub.conf の編集

“ターゲット装置用ルートファイルシステム”/boot/grub にある grub.conf ファイルを編集します。

下記例のように ubinux カーネルを実行する項目を追加します。ここでは、カーネルイメージを格納した「/boot パーティション」は/dev/sdc1、「/ パーティション」は/dev/sdc2 に配置されているものとしています。ご利用の環境に合わせて、それぞれの設定値を変更してください。

```
title ubinux
root (hd2,0)
kernel /bzImage root=/dev/sdc2
```

(2) ターゲット装置の起動

ターゲット装置を起動し、起動メニューから ubinux カーネルを選択します。

3.3.2. U-Boot を用いた起動

本節では、U-Boot を用いてブートホストから tftp でカーネルイメージをロードし、そのカーネルを起動する手順について説明します。

なお tftp でカーネルイメージをロードする前に U-Boot の環境変数（ipaddr、serverip など）をご利用の環境に合わせて設定し、ブートホストとの通信ができることを ping コマンドで確認しておいてください。

(1) 環境変数 bootargs の設定

カーネルの起動オプションを U-Boot の環境変数 bootargs に設定します。

MPC8313E-RDB ボードで起動する場合の設定例を下記に示します。ここでは、以下に設定しています。

・コンソールデバイス：ttyS0（ボーレートが 115200bps）
・ターゲット装置 IP アドレス：192.168.246.87
・ブートホストの IP アドレス：192.168.246.1
・NFS マウントするディレクトリ：/opt/ubq/devkit/powerpc/e300/target
・NFS マウントに使用するネットワークデバイス：eth1

```
=> setenv bootargs console=ttyS0,115200 root=/dev/nfs
   nfsroot=192.168.246.1:/opt/ubq/devkit/powerpc/e300/target
   ip=192.168.246.87:192.168.246.1:192.168.246.1:255.255.255.0::eth1:
```

bootargs の設定値を保存したい場合は、上記コマンド実行後に saveenv コマンドを実行してください。

(2) カーネルイメージのロード

カーネルイメージをブートホストから tftp で転送します。

下記例では、カーネルイメージ cuImage.mpc8313erdb を *addr* で指定したアドレスにロードします。

```
=> tftp addr cuImage.mpc8313erdb
```

弊社の動作確認時に *addr* で指定したアドレスを下記に示します。

・PowerQUICC II Pro CPU ファミリのターゲット装置：0x200000
・QorIQ CPU ファミリのターゲット装置：0x1000000
・ARMv7 CPU ファミリのターゲット装置：0x80000000

(3) ubinux カーネルの起動

bootm コマンドにより、ubinux カーネルを起動します。

```
=> bootm addr
```

「2.2.3 カーネルのビルドとインストール」で dtb の生成を必要としたターゲット装置では、この dtb ファイルもブートホストから転送する必要あります。
ここでは、P2020RDB で dtb を用いて起動する例を示します。
「(1) 環境変数 bootargs の設定」までは設定されているものとします。

```
=> tftp 0x1000000 uImage    //カーネルイメージのロード
=> tftp 0xc00000 p2020rdb.dtb  //dtb ファイルのロード
      //（P1020RDB、P4080DS も dtb ファイルのロードするアドレスは同じです。）
      //（MPC8377E-MDS の dtb ファイルのロードするアドレスは 0x500000 です。）
=> bootm 0x1000000 - 0xc00000
           ①      ②    ③
    ① カーネルをロードしたアドレス
    ② initrd で起動する場合にアドレスを指定. (不要な場合は"-")
    ③ dtb をロードしたアドレス
```

上記 bootm コマンド実行後、ubinux カーネルが起動します。

4. ターゲット装置上での設定

本章では、ターゲット装置上で動作する ubinux で設定する手順について説明します。

4.1. 初期設定済アカウントの変更

ubinux では、初期設定済アカウントとして、**表 4-1　初期設定済のアカウント**のアカウントが設定されています。

表 4-1　初期設定済のアカウント

権限	ユーザ名	パスワード
スーパユーザ	root	なし

初期設定のままの状態にしておくと、セキュリティの問題が生じる可能性がありますので、passwd コマンド等を使い、適宜変更を行ってください。

4.2. 共有ライブラリ位置情報の更新

ubinux では、共有ライブラリの位置情報を/etc/ld.so.cache に格納しており、プログラムロード時に当該のファイルに基づいて、ライブラリのロードを実施します。

次のいずれかに該当する場合、共有ライブラリの位置情報を更新する必要があります。

- はじめてターゲット装置を起動したとき
- ライブラリの配置情報や版数を更新したとき
- /etc/ld.so.cache が無い、または当該ファイルに欠損がある場合

/etc/ld.so.cache を更新するためには、次のコマンドを実行します。

```
# /sbin/ldconfig
```

4.3. カーネルモジュール依存関係の更新

ubinux でカーネルモジュール化されたデバイスドライバなどを利用する場合、カーネルモジュール間の依存関係情報を作成する必要があります。カーネルモジュールの依存関係情報は以下のファイルに格納されます。

```
/lib/modules/ベースカーネル版数/modules.dep
```

ベースカーネル版数は、ubinux のリリースノートを参照するか、ターゲット上で次のコマンドを実行することで確認できます。

```
# uname -r
```

/lib/modules 配下に、ベースカーネル版数のディレクトリがない場合は、次のコマンドを実行し作成するか、カーネルモジュールを配置してください。

```
# mkdir /lib/modules/ベースカーネル版数
```

次のいずれかに該当する場合、カーネルモジュールの依存関係情報を更新する必要があります。

- はじめてターゲット装置を起動したとき
- カーネルモジュールの追加や削除をおこなったとき
- /lib/modules 配下に modules.dep がない、または当該ファイルに欠損がある場合

カーネルモジュールの依存関係情報を更新するためには、次のコマンドを実行します。

```
# depmod -a
```

4.4. 各種パッケージの設定

本節では、ubinux に固有なミドルウェアの設定について説明します。

4.4.1. cracklib の設定

passwd パッケージに含まれる passwd コマンドは、cracklib を使用して、ユーザパスワードの安全性検証を実施します。

そこで、passwd パッケージに含まれる passwd コマンドを使用する場合や、装置固有のアプリケーションが cracklib を使用するようになっている場合は、ターゲット装置上で cracklib が使用する DB を作成する必要があります。

root アカウントでターゲットにログイン後、以下のコマンドを実行することで cracklib が使用する DB を作成することが可能です。

```
# /usr/sbin/create-cracklib-dict /usr/share/cracklib/words/*
```

上記コマンドを実行すると/usr/share/cracklib 配下に以下のファイルが作成されます。

- pw_dict.hwm
- pw_dict.pwd
- pw_dict.pwi

4.4.2. 地域情報の設定

ubinux では、組込み用にフットプリントサイズを小さくするために、NLS(Native Language Support)機能を**表 4-2 NLS サポート方針**に示す方針でサポートするようにしています。

表 4-2 NLS サポート方針

種別	NLS 機能
バイナリプログラム	無効
ライブラリ	有効

また、各種地域情報関連設定ファイル（ロカール情報ファイル）は、パッケージファイルに含まれておりません。ターゲット装置に root でログインした後、localedef コマンドを実行することで、各種地域情報関連設定ファイルを作成することが可能です。

localedef コマンドは、次のようにして使用します。

```
# /usr/bin/localedef -i ロカール名 -f 文字コード /usr/lib/locale/文字コード
# /usr/bin/localedef -i ロカール名 -f 文字コード /usr/lib/locale/ロカール名.文字コード
```

日米で使用される代表的な地域情報ファイルを生成するために必要なロカール名と文字コードの組合せを**表 4-3 代表的なロカール名と文字コード**に示します。

表 4-3 代表的なロカール名と文字コード

地域	ロカール名	文字コード
米国	en_US	ANSI_X3.4-1968
米国	en_US	ISO-8859-1
日本	ja_JP	EUC-JP
日本	ja_JP	SHIFT_JIS
日本	ja_JP	UTF-8

ロカール名”en_US”、文字コード”ANSI_X3.4-1968”の地域情報ファイルを生成するコマンドを以下に示します。

```
# /usr/bin/localedef -i en_US -f ANSI_X3.4-1968 /usr/lib/locale/ANSI_X3.4-1968
# /usr/bin/localedef -i en_US -f ANSI_X3.4-1968 /usr/lib/locale/en_US.ANSI_X3.4-1968
```

4.4.3. gtk2 の設定

gtk2 及び pango パッケージを使用する場合、モジュール情報等の設定ファイルをターゲット装置上で作成する必要があります。

root アカウントでターゲット装置にログイン後、以下のコマンドを実行することで gtk2 及び pango の動作に必要な設定ファイルが生成されます。ターゲット識別名は**表 2-1　識別名一覧**を参照ください。

```
# /usr/bin/pango-querymodules-32 > /etc/pango/ターゲット識別名/pango.modules
# /usr/bin/gdk-pixbuf-query-loaders-32 > /etc/gtk-2.0/ターゲット識別名/gdk-pixbuf.loaders
# /usr/bin/gtk-query-immodules-2.0-32 > /etc/gtk-2.0/ターゲット識別名/gtk-immodules
```

4.4.4. xterm の設定

xterm を使用する場合、/usr/share/X11/fonts/misc/fonts.dir を作成する必要があります。

root アカウントでターゲット装置にログイン後、以下のコマンドを実行することで fonts.dir が生成されます。

```
# cd /usr/share/X11/fonts/misc/
# mkfontdir
```

また、「4.4.2. 地域情報の設定」も実施してください。

4.4.5. file の設定

PowerPC アーキテクチャの file パッケージは、クロスコンパイルで生成できないため、下記のファイルがありません。

- /usr/share/misc/magic.mgc

このファイルがない場合、/usr/share/misc/magic からマジックナンバーが読み込まれるため、file コマンドの動作には問題ありません。

magic.mgc ファイルが必要な場合は、root アカウントでターゲット装置にログイン後、以下のコマンドを実行することで生成されます。

```
# cd /usr/share/misc
# file --compile
```

本パッケージは、デバッグインフォを用いたデバッグは未サポートです。

4.4.6. udev の設定

udev パッケージを使用する場合、デバイス管理のためにグループの追加が必要です。root アカウントでターゲット装置にログイン後、以下のコマンドを実行することでグループが追加・設定されます。

```
# getent group video >/dev/null || /usr/sbin/groupadd -g 39 video || :
# getent group audio >/dev/null || /usr/sbin/groupadd -g 63 audio || :
# getent group cdrom >/dev/null || /usr/sbin/groupadd -g 11 cdrom || :
# getent group tape >/dev/null || /usr/sbin/groupadd -g 33 tape || :
# getent group dialout >/dev/null || /usr/sbin/groupadd -g 18 dialout || :
```

4.4.7. openldap の設定

openldap パッケージを使用する場合、openldap の起動に必要なディレクトリとファイルの作成が必要です。

openldap パッケージには slapd.conf ファイルが含まれていません。初期設定には、/etc/openldap/slapd.conf が必要となり、openldap を使用する環境に合わせて作成して頂く必要があります(※)。

以下の手順で openldap の起動ができます。root アカウントでターゲット装置にログイン後、以下のコマンドを実施します。

(1) ディレクトリとファイルを生成します。

```
# mkdir -p /var/lib/ldap
# mkdir -p /var/run/openldap
# cp -p /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
```

(2) openldap のデータベースを生成します。

```
# slaptest -f /etc/openldap/slapd.conf -F /etc/openldap/slapd.d
```

初回実施時は/var/lib/ldap にデータベースが存在しない為、以下のメッセージが出力されます。slaptest コマンドを実施することでデータベースが生成される為、問題ありません。

```
bdb_db_open: database "o=TestBase,c=JP": db_open(/var/lib/ldap/id2entry.bdb)
failed: No such file or directory (2).
backend_startup_one (type=bdb, suffix="o=TestBase,c=JP"): bi_db_open failed! (2)
slap_startup failed (test would succeed using the -u switch)
```

(3) ldap ユーザ権限に変更します。

```
# chown -R ldap.ldap /var/lib/ldap
# chown -R ldap.ldap /var/run/openldap
# chown -R ldap.ldap /etc/openldap/slapd.d
```

(4) openldap を起動します。

```
# /etc/init.d/slapd start
```

(※) 弊社の openldap 起動確認時に作成した slapd.conf ファイルを下記に示します。
起動に必要な最小限の設定を記述しています。

```
include         //etc/openldap/schema/core.schema
include         //etc/openldap/schema/cosine.schema
include         //etc/openldap/schema/inetorgperson.schema
include         //etc/openldap/schema/nis.schema

# Allow LDAPv2 client connections.  This is NOT the default.
allow bind_v2

# ldbm and/or bdb database definitions
pidfile         /var/run/openldap/slapd.pid
argsfile        /var/lib/ldap/slapd.args
database        bdb

suffix          "o=TestBase,c=JP"
rootdn          "cn=Manager,o=TestBase,c=JP"

# Cleartext passwords, especially for the rootdn, should
# be avoided.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw          secret

# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory       /var/lib/ldap

# Indices to maintain for this database
index objectClass                       eq,pres
index ou,cn,mail,surname,givenname      eq,pres,sub
index uidNumber,gidNumber,loginShell    eq,pres
index uid,memberUid                     eq,pres,sub
index nisMapName,nisMapEntry            eq,pres,sub
```