文書番号：014-0001-0027

ubinux
インストーラ
マニュアル

第 2.9 版　2012 年 2 月 29 日

FUJITSU

［改版履歴］

資料名		ubinux インストーラマニュアル	
対応プログラム名		ubinux インストーラ	
版数	作成日付	作成者	変更内容
1.0	2005/8/30	PST) 宮家	初版作成
1.2	2005/10/18	FST) 宮家	ルートファイルシステム作成支援ツールのログ出力機能追加に伴う修正
1.3	2005/10/26	富士通 加藤	表紙の承認欄・部署名・関係者外秘表示を削除。
1.4	2005/11/1	富士通 加藤	P.32 「"Y"を押下して、インストールを開始してください。」を「"S"を押下して、インストールを開始してください。」に訂正。
1.5	2006/1/19	富士通 加藤	V4.1-HGW 版（FJAC には、版数 1.0 として提示）
1.6	2006/1/19	富士通 加藤	V4.1-HGW 版の内容を反映（ホスト環境を FC2 から FC3 に変更)。
1.7	2006/7/18	富士通 加藤	・「組み込み」を「組込み」に修正（「はじめに」、「第 1 章」)。 ・本文、図中の fje を ubq に変更（2.1.節、3.2.節、4.1.節、4.2.節、5.1.節、5.2.節、6.3.節、6.8.節)。 ・overwrite のスペルミスを修正（3.9.節） ・rpm のタイプミス（prm）を修正（5.2.節)。
1.8	2006/8/1	富士通 加藤	・ 付録１．「shread ライブラリ不足時の異常動作」を「共有ライブラリ不足時の異常動作」に変更 ・ 付録１．PAM のプラグイン共有オブジェクトに関する記述を追記。

1.9	2006/9/29	FST) 福田	・ 「動作要件」に「動作確認済みディストリビューション」を追加(1.2 節)。 ・ FC4 以降における rpm ライブラリの不具合の回避方法を追記（1.2 節、付録 2)。 ・ パッケージインストーラに -l オプションを追記（2.1 節)。 ・ 罫線の枠の太さを修正（3.6 節)。 ・ インストール完了時にインストール済みパッケージの一覧がファイルに出力される旨の説明を追記（3.9 節)。 ・ インストール済みパッケージ一覧ファイルの形式について説明を追記（3.11 節)。 ・ 「Shared ライブラリ」を「共有ライブラリ」に修正（6.1 節、6.5 節、6.6 節、6.8 節、6.9 節、6.10 節、6.11 節、6.13 節)。 ・ バッチインストールのためのサンプルスクリプトを追記（付録３）
2.0	2007/8/8	FCT) 加藤	・ 付録４に ubinux_make_rootfs.dev.conf のサンプルファイルの例と所在を明記
2.1	2007/8/9	FCT) 加藤	・ 付録１． X Window 共有オブジェクトに関する記述を追記
2.2	2007/8/14	FCT) 加藤	・ FC4 以降における rpm ライブラリの不具合の回避方法を FC4/FC5 に限定（1.2 節、付録 2)。
2.3	2007/8/29	FCT) 加藤	・ 付録１． libuser に関する共有オブジェクトに関する記述を追記
2.4	2008/3/27	FCT) 浅羽	・ 動作確認済みディストリビューションを修正(1.2 節)
2.5	2008/10/24	FCT) 浅羽	・ 動作確認済みディストリビューションの誤記を修正(1.2 節) ・ ubinux_make_rootfs.dev.conf に関する説明を追記(6.3 節) ・ 付録 4 の誤記を修正
2.6	2008/10/27	FCT) 船越	・ パッケージインストーラ、ルートファイルシステム作成ツールの機能追加に伴う修正。 ・ 付録１． X Window 共有オブジェクトに関する記述を仕様変更にともなって削除。

2.7	2010/1/14	FCT）船越	・パッケージインストーラ、ファイルシステムイメージ作成ツールの機能追加に伴う修正。
2.8	2011/2/28	FCT）船越	・ファイルシステムイメージ作成ツールの機能追加（Logfs）に伴う修正 ・組込み Linux から ubinux へ表記を統一
2.9	2012/2/29	FCT）船越	・パッケージインストーラ、ファイルシステムイメージ作成ツールの機能追加に伴う修正。 ・付録１、共有ライブラリ不足時の異常動作とその対策についての記述内容の見直し。

目　　次

はじめに

本書では、ubinux インストーラの使用方法について説明します。

第 2.9 版　　2012 年 2 月

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。したがって、本資料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されます。

お願い

- 本書の全体または一部を、無断で転載または複写することを禁じます。
- 本書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。
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。

第1章　概要

1.1　提供ツール

ubinux インストーラでは、以下のツールを提供します。

- パッケージインストーラ
- パッケージアップグレードツール
- パッケージアンインストールツール
- ルートファイルシステム作成ツール
- ファイルシステムイメージ作成ツール

1.2　動作要件

提供ツールの動作要件については「スタートアップガイド」を参照ください。

制限事項については「リリースノート」を参照ください。

1.3 機能概要

提供ツールの機能概要は、以下の通りです。

- 設定ファイル（サンプル等）

  パッケージインストール等に使用する為の各種定義ファイル。必要に応じてコピーして使用します。

  設定ファイルは以下のディレクトリにインストールされています。

  - /opt/ubinux_pkg_tools/share/tools/sample-config
  - /opt/ubinux_pkg_tools/share/tools/sample-script

- パッケージインストーラ（ubinux_pkg_install）

  パッケージインストーラは、rpm 形式のアーカイブによって提供されているターゲットボード用の ubinux ミドルウェアと、ホスト用の ubinux ミドルウェア（クロス環境ツール）をホスト上の開発用ディレクトリにインストールするツールです。

  キャラクタ・ユーザ・インタフェースにより、インストールするパッケージを選択することができます。

- パッケージアップグレードツール（ubinux_pkg_upgrade）

  パッケージインストーラでインストールしたパッケージをアップグレードするツールです。

- パッケージアンインストールツール（ubinux_pkg_uninstall）

  パッケージインストーラでインストールしたパッケージをアンインストールするツールです。

- ルートファイルシステム作成ツール（ubinux_make_rootfs）

  パッケージインストーラが ubinux ミドルウェアを展開した開発用ディレクトリを元にして、ターゲット向けのルートファイルシステムを作成するツールです。

  一度、作成したルートファイルシステムを元にして、新たにルートファイルシステムを作成することも可能です。

  キャラクタ・ユーザ・インタフェースにより、ルートファイルシステムに組み込むファイルを選択することができます。

- ファイルシステムイメージ作成ツール（ubinux_make_ fsimage）

  ルートファイルシステム作成ツールで作成したターゲット向けのルートファイルシステムを元にして、フラッシュメモリやディスクに配置するファイルシステムイメージを作成するツールです。

  本ツールがサポートするイメージ種別は、以下の通りです。

  - JFFS2
  - INITRAMFS
  - INITRD
  - RAW イメージ（ext2、ext3、ext4）
  - SquashFS
  - UBIFS
  - Cramfs
  - Logfs

パッケージのインストールからファイルシステムイメージ作成までの手順は、以下のようになります。

① パッケージのインストール（使用ツール：ubinux_pkg_install）
② 開発用ディレクトリのカスタマイズ
- ネットワーク定義など各種システムコンフィグレーションファイルの変更
- 利用者が作成したアプリケーションのコピー
- カーネルのコピー
- その他

③ ターゲット向けルートファイルシステムの作成（使用ツール：ubinux_make_rootfs）
④ ファイルシステムイメージの作成（使用ツール：ubinux_make_fsimage）
⑤ ターゲット上で動作を確認
⑥ ファイルシステムイメージが完成するまで②～⑤を繰り返す。

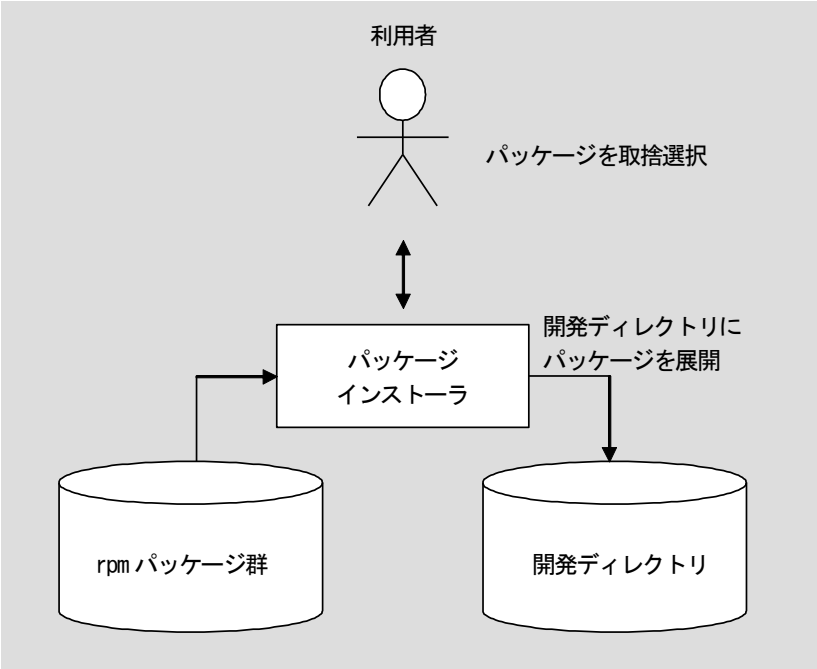


図 1 パッケージのインストール

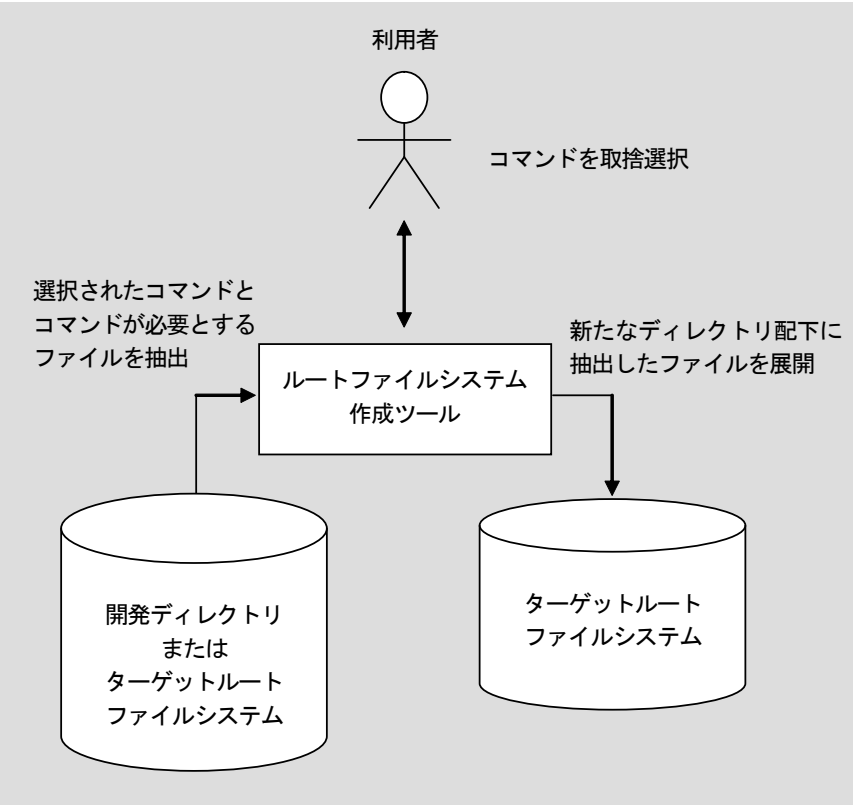


図 2 ターゲットルートファイルシステムの作成

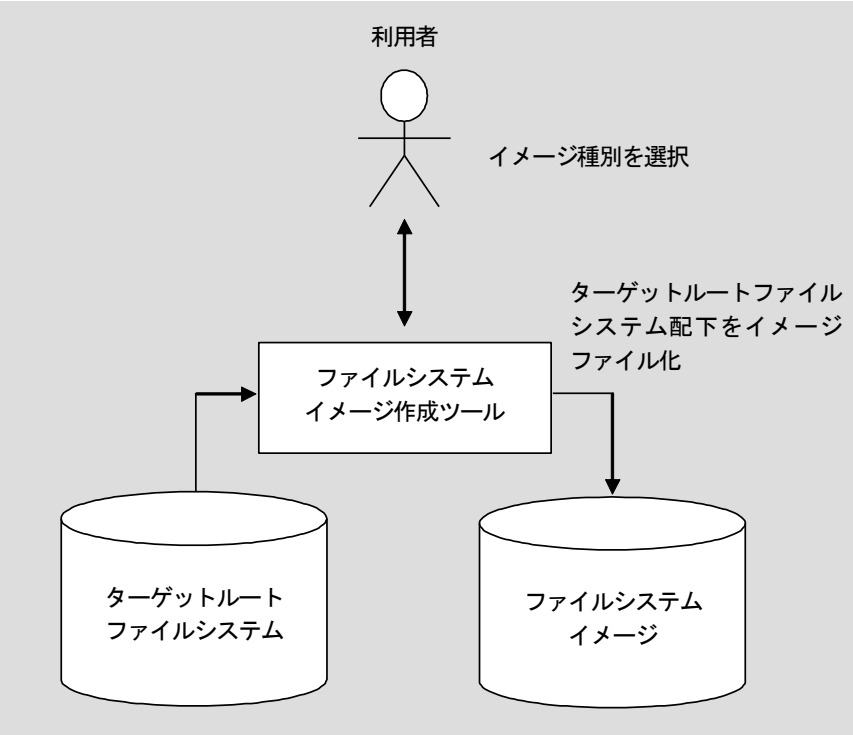


図 3 ファイルシステムイメージの作成

第2章　ツール仕様

2.1　パッケージインストーラ

名前

ubinux_pkg_install - CUI パッケージインストーラ

書式

ubinux_pkg_install [-o log-file] [-l list-file] [-i target-install-dir]
[-h host-rpmdb-dir] [-t target-rpmdb-dir]
[-p install-pkg-dir] [--info install-information-file] [--nohost]

説明

ubinux_pkg_install は、ディレクトリ install-pkg-dir 配下の rpm パッケージを、ターゲット開発ディレクトリ target-install-dir にインストールする。インストールするパッケージは、キャラクタ・ユーザ・インターフェースで選択する。
ubinux_pkg_install は、rpmlib の機能を使用して、rpm パッケージのインストールを行う。
なお、ubinux_pkg_install は、root ユーザでのみ使用可能である。
また —info オプションを指定されたときは、指定された install-information-file の内容に従い自動的にインストールを行う。

オプション

[-o log-file]
log-file にインストール時のログを出力する。省略時は、~/ubinux_pkg_install.YYYYMMDD-hhmmss.log に出力する（YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）。

[-l list-file]
list-file にインストール済みの rpm パッケージの一覧を出力する。省略時は、~/ubinux_pkg_install.YYYYMMDD-hhmmss.pkglist に出力する（YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）。

[-i target-install-dir]
target-install-dir にターゲット向けパッケージをインストールする。ただし、クロスツールチェインは/opt/ubq/devkit/*ARCH*/*PROC*/target 配下のヘッダ、ライブラリを参照するため、本オプションで指定した target-install-dir へのリンクを作成する必要がある。
（*ARCH*、*PROC*については、スタートアップガイドを参照）
例：target-install-dir に/home/test を指定する場合
インストール後に以下のコマンドでリンクを作成する。
ln -s /home/test/target /opt/ubq/devkit/*ARCH*/*PROC*/target

[-h host-rpmdb-dir]
host-rpmdb-dir にホスト向けパッケージの rpm データベースを構築する。--nohost オプションが設定されている時は省略可。

[-t target-rpmdb-dir]
target-install-dir/target-rpmdb-dir にターゲット向けパッケージの rpm データベースを構築する。また ubinux_inst_info.xml も当ディレクトリに作成される。
[-p install-pkg-dir]
install-pkg-dir 配下（サブディレクトリを含む）に格納されている rpm パッケージをインストールする。
[—info install-information-file]
install-information-file に格納されている情報に基づいた自動インストールを行う。
[—nohost]
当オプションが設定された場合には、ホストパッケージのインストールを行わない。
[—allall]
当オプションが設定された場合には、インストール時にファイルの競合があった時に上書き(All)で処理を自動的に進める。上書き動作の詳細については 3.13 を参照。
[—allnone]
当オプションが設定された場合には、インストール時にファイルの競合があった時に上書きせず(None)に処理を自動的に進める。上書き動作の詳細については 3.13 を参照。

環境変数と設定ファイル

—info オプションを除く上記オプションで指定する内容は、環境変数および定義ファイルでも設定が可能である。
参照する順番は、優先順位の高い方から、コマンドラインオプション、環境変数、定義ファイルの順である。
target-install-dir、host-rpmdb-dir、target-rpmdb-dir および install-pkg-dir は、デフォルト値を持たない。コマンドラインオプション、環境変数または定義ファイルのいずれかでの指定が必須である。
—info オプションを指定した場合には install-information-file ファイルに記述されているオプションが採用される。その内容を変更したい場合は、コマンドラインオプションに記述する。install-information-file の記述内容については 3.14 を参照。

環境変数

コマンドラインオプションに対応する環境変数は、以下の通り。

UBINUX_PKG_INSTALL_LOG	⇔ [-o log-file]
UBINUX_PKG_INSTALL_LOG_LIST	⇔ [-l list-file]
UBINUX_PKG_TARGET_INSTALL_DIR	⇔ [-i target-install-dir]
UBINUX_PKG_HOST_RPMDB_DIR	⇔ [-h host-rpmdb-dir]
UBINUX_PKG_TARGET_RPMDB_DIR	⇔ [-t target-rpmdb-dir]
UBINUX_PKG_INSTALL_PKG_DIR	⇔ [-p install-pkg-dir]

定義ファイル

定義ファイルは、/root/.ubinux_pkg_tools.rc である。書式は、以下の通り。
ディレクトリおよびファイル名は絶対パスを指定する。シェルの環境変数の指定はできない。

```
### /root/.ubinux_pkg_tools.rc ###
UBINUX_PKG_INSTALL_LOG=/tmp/ubinux_pkg_install.log
UBINUX_PKG_INSTALL_LOG_LIST=/tmp/ubinux_pkg_install.pkglist
UBINUX_PKG_TARGET_INSTALL_DIR=/opt/ubq/devkit/ARCH/PROC/target
UBINUX_PKG_HOST_RPMDB_DIR=/opt/ubq/devkit/ARCH/PROC/hrpmdb
UBINUX_PKG_TARGET_RPMDB_DIR=/opt/ubq/trpmdb
UBINUX_PKG_INSTALL_PKG_DIR=/opt/ubq/devkit/ARCH/PROC/rpms
### /root/.ubinux_pkg_tools.rc ###
```

上記以外の関連ファイル

/var/lock/ubinux_pkg_tools/pkg_install

ツールの多重起動を防止するための排他獲得ファイル

install-pkg-dir/ubinux_pkg_info.xml

以下の情報を記述した外部 xml ファイル

- インストールパッケージの選択例
- プレインストールパッケージ
- ポストインストールパッケージ

サンプル定義ファイルはインストール時に以下に配置されています。ubinux ミドルウェアアーカイブ展開ディレクトリの rpms にコピーして使用してください。(推奨)

/opt/ubinux_pkg_tools/share/tools/sample-config/ubinux_pkg_info.xml

また、install-ubq-*ARCH*.sh を実行する場合には指定した ubinux ミドルウェアアーカイブ展開ディレクトリの rpms に当ファイルが無ければコピーされます。

target-install-dir/target-rpmdb-dir/ubinux_inst_info.xml

以下の情報を記述した xml ファイル(インストール時に自動生成される)。―info オプション時にも使用する。

- インストール時の環境情報
- インストールされているパッケージ
- インストール時に競合したファイルの優先順位

2.2 パッケージアップグレードツール

ubinux_pkg_install は、アップグレード機能を持ちません。パッケージをアップグレードする場合は、本ツールを使用してください。

名前

ubinux_pkg_upgrade - パッケージアップグレードツール

書式

ubinux_pkg_upgrade -T [-i target-install-dir] [-t target-rpmdb-dir]
パッケージファイル名
ubinux_pkg_upgrade -H [-h host-rpmdb-dir] パッケージファイル名

説明

ubinux_pkg_upgrade は、ubinux_pkg_install でインストールした rpm パッケージのアップグレードを行う。ubinux_pkg_upgrade は、root ユーザでのみ使用可能である。

オプション

-T
ターゲット向けパッケージのアップグレードを行う。
-H
ホスト向けパッケージのアップグレードを行う。
[-i target-install-dir]
ターゲット向けパッケージがインストールされているディレクトリを指定する。
[-h host-rpmdb-dir]
ホスト向けパッケージの rpm データベース格納先ディレクトリを指定する。
[-t target-rpmdb-dir]
ターゲット向けパッケージの rpm データベース格納先ディレクトリを指定する。
実際のターゲット向けパッケージの rpm データベースのパスは、target-install-dir/target-rpmdb-dir となる。

環境変数と設定ファイル

上記オプションで指定する内容は、環境変数および定義ファイルでも設定が可能である。参照する順番は、優先順位の高い方から、コマンドラインオプション、環境変数、定義ファイルの順である。
target-install-dir、host-rpmdb-dir、target-rpmdb-dir は、コマンドラインオプション、環境変数または定義ファイルのいずれかでの指定が必須である。

コマンドラインオプションに対応する環境変数は、以下の通り。

UBINUX_PKG_TARGET_INSTALL_DIR ⇔ [-i target-install-dir]
UBINUX_PKG_HOST_RPMDB_DIR ⇔ [-h host-rpmdb-dir]
UBINUX_PKG_TARGET_RPMDB_DIR ⇔ [-t target-rpmdb-dir]

定義ファイルによる定義は、/root/.ubinux_pkg_tools.rc を ubinux_pkg_install と共有する。

上記以外の関連ファイル

/var/lock/ubinux_pkg_tools/pkg_install
ツールの多重起動を防止するための排他獲得ファイル

target-install-dir/target-rpmdb-dir/ubinux_inst_info.xml
ubinux_pkg_install にてインストール時に自動生成された xml ファイル。
- インストール時の環境情報
- インストールされているパッケージ
- インストール時に競合したファイルの優先順位

2.3 パッケージアンインストールツール

ubinux_pkg_install は、アンインストール機能を持ちません。パッケージをアンインストールする場合は、本ツールを使用してください。

名前

ubinux_pkg_uninstall - パッケージアンインストールツール

書式

ubinux_pkg_uninstall -T [-i target-install-dir] [-t target-rpmdb-dir] パッケージ名
ubinux_pkg_uninstall -H [-h host-rpmdb-dir] パッケージ名

説明

ubinux_pkg_uninstall は、ubinux_pkg_install でインストールした rpm パッケージをアンインストールする。ubinux_pkg_uninstall は、root ユーザでのみ使用可能である。

オプション

-T
ターゲット向けパッケージのアンインストールを行う。

-H
ホスト向けパッケージのアンインストールを行う。

[-i target-install-dir]
ターゲット向けパッケージがインストールされているディレクトリを指定する。-T オプション指定持にのみ有効。

[-h host-rpmdb-dir]
ホスト向けパッケージの rpm データベース格納先ディレクトリを指定する。

[-t target-rpmdb-dir]
ターゲット向けパッケージの rpm データベース格納先ディレクトリを指定する。
実際のターゲット向けパッケージの rpm データベースのパスは、target-install-dir/target-rpmdb-dir となる。

パッケージ名
ワイルドカード(アスタリスクによる)を指定することが出来る。(※ bash での展開を避ける為、ワイルドカード指定する場合には、ダブルクォーテーション等での展開を抑止する必要があります。)

環境変数と設定ファイル

上記オプションで指定する内容は、環境変数および定義ファイルでも設定が可能である。
参照する順番は、優先順位の高い方から、コマンドラインオプション、環境変数、定義ファイルの順である。
target-install-dir、host-rpmdb-dir、target-rpmdb-dir は、コマンドラインオプション、環境変数または定義ファイルのいずれかでの指定が必須である。

コマンドラインオプションに対応する環境変数は、以下の通り。

UBINUX_PKG_TARGET_INSTALL_DIR ⇔ [-i target-install-dir]
UBINUX_PKG_HOST_RPMDB_DIR ⇔ [-h host-rpmdb-dir]
UBINUX_PKG_TARGET_RPMDB_DIR ⇔ [-t target-rpmdb-dir]

定義ファイルによる定義は、/root/.ubinux_pkg_tools.rc を ubinux_pkg_install と共有する。

上記以外の関連ファイル

/var/lock/ubinux_pkg_tools/pkg_install
ツールの多重起動を防止するための排他獲得ファイル

target-install-dir/target-rpmdb-dir/ubinux_inst_info.xml
ubinux_pkg_install にてインストール時に自動生成された xml ファイル。
- インストール時の環境情報
- インストールされているパッケージ
- インストール時に競合したファイルの優先順位

2.4 ルートファイルシステム作成ツール

名前

ubinux_make_rootfs - ルートファイルシステム作成ツール

書式

ubinux_make_rootfs [-f from-dir] [-t to-dir] [-r target-rpmdb-dir]
[-k kernel-source-dir]

説明

ubinux_make_rootfs は、ディレクトリ from-dir 配下から利用者が指定したコマンド、コマンドが使用する shared ライブラリ、これらに関連するその他ファイルを抽出し、
to-dir /root 配下にターゲット向けのルートファイルシステムを構築する。
ルートファイルシステムに組み込むコマンドは、キャラクタ・ユーザ・インタフェースにより選択する。
また、デバイスファイルは、カーネルコンフィグレーションの情報を元に自動生成する。
なお、ubinux_make_rootfs は、root ユーザでのみ使用可能である。

オプション

[-f from-dir]
from-dir 配下からルートファイルシステムに組み込むファイルを抽出する。

[-t to-dir]
to-dir/root にターゲット向けルートファイルシステムを構築する。

[-r target-rpmdb-dir]
コマンドに関連する shared ライブラリ以外のファイルの情報を target-rpmdb-dir 配下の rpm データベースから求める。

[-k kernel-source-dir]
kernel-source-dir/.config と kernel-source-dir/ubinux_make_rootfs.dev.conf の情報を元に、必要なデバイスファイルを作成する。

環境変数と設定ファイル

上記オプションで指定する内容は、環境変数および定義ファイルでも設定が可能である。
参照する順番は、優先順位の高い方から、コマンドラインオプション、環境変数、定義ファイルの順である。
上記オプションが示す情報は、コマンドラインオプション、環境変数または定義ファイルのいずれかでの指定が必須である。

環境変数

コマンドラインオプションに対応する環境変数は、以下の通り。

UBINUX_PKG_MKROOTFS_FROM_DIR ⇔ [-f from-dir]
UBINUX_PKG_MKROOTFS_TO_DIR ⇔ [-t to-dir]
UBINUX_PKG_TARGET_INSTALL_DIR + UBINUX_PKG_TARGET_RPMDB_DIR
⇔ [-r target-rpmdb-dir]
UBINUX_PKG_KERNEL_SRC_DIR ⇔ [-p kernel-source-dir]

定義ファイル

定義ファイルは、/root/.ubinux_pkg_tools.rc である。書式は、以下の通り。
ディレクトリ名は、絶対パスを指定する。シェルの環境変数の指定はできない。

```
### /root/.ubinux_pkg_tools.rc ###
UBINUX_PKG_MKROOTFS_FROM_DIR=/opt/ubq/devkit/ARCH/PROC/target
UBINUX_PKG_MKROOTFS_TO_DIR=/tmp/build-dir
UBINUX_PKG_TARGET_INSTALL_DIR=/opt/ubq/devkit/ARCH/PROC/target
UBINUX_PKG_TARGET_RPMDB_DIR=/opt/ubq/trpmdb
UBINUX_PKG_KERNEL_SRC_DIR=/tmp/ubinux-kernel
### /root/.ubinux_pkg_tools.rc ###
```

上記以外の関連ファイル

to-dir/ubinux_make_rootfs.YYYYMMDD-hhmmss.log
ルートファイルシステム作成時のログ出力ファイル（YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）。

to-dir/ubinux_make_rootfs.select.YYYYMMDD-hhmmss.log
キャラクタ・ユーザ・インターフェースで選択した内容を出力するログファイル（YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）。

to-dir/ubinux_make_rootfs.file.YYYYMMDD-hhmmss.log
作成したルートファイルシステムのファイル一覧を出力するログファイル（YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）。

from-dir/../ubinux_make_rootfs.select.conf
組み込みファイル選択情報。再度、from-dir を使用してルートファイルシステムを作成する場合に、前回作成時のファイル選択状況を再現する。

kernel-source-dir/ubinux_make_rootfs.dev.conf
自動作成するデバイスファイルの情報。ubinux_make_rootfs.dev.conf と kernel-source-dir/.config の情報を基に、デバイスファイルを自動生成する。

from-dir/../ubinux_make_rootfs.mandatory.conf
　　強制的に組み込むファイルの情報。利用者の選択に依らず、記述されたファイルを強制的にルートファイルシステムに組み込む。
from-dir/../ubinux_make_rootfs.mandatory.xml
　　強制的に組み込むファイルの情報。インストールされているパッケージまたはファイルに応じて、記述されたファイルを利用者の選択に依らず、強制的にルートファイルシステムに組み込む。
/var/lock/ubinux_pkg_tools/make_rootfs
　　ツールの多重起動を防止するための排他獲得ファイル

2.5 ファイルシステムイメージ作成ツール

名前

ubinux_make_fsimage - ファイルシステムイメージ作成ツール

書式

ubinux_make_fsimage [-f from-dir] [-t to-dir] [-o image-file]

説明

ubinux_make_fsimage は、ディレクトリ from-dir をルートとしたファイルシステムイメージを作成する。サポートするイメージ種別は、JFFS2、INITRAMFS、INITRD、RAW、SQUASHFS、UBIFS、CRAMFS、および LOGFS イメージである。作成するイメージ種別は、キャラクタ・ユーザ・インタフェースで指定する。

なお、ubinux_make_fsimage は、root ユーザでのみ使用可能である。

オプション

[-f from-dir]

from-dir 配下のファイルをイメージファイルに組み込む。

[-t to-dir]

to-dir 配下にイメージファイルを作成する。

[-o image-file]

指定された名前でイメージファイルを作成する。

省略した場合のイメージファイル名は以下の通り。

JFFS2	: rootfs.jffs2.YYYYMMDD-hhmmss.bin
INITRAMFS	: rootfs.initramfs.YYYYMMDD-hhmmss.bin.gz
INITRD	: rootfs.initrd.YYYYMMDD-hhmmss.bin.gz
RAW	: rootfs.raw.YYYYMMDD-hhmmss.bin
SQUASHFS	: rootfs.squashfs.YYYYMMDD-hhmmss.bin
UBIFS	: rootfs.ubifs.YYYYMMDD-hhmmss.bin
CRAMFS	: rootfs.cramfs.YYYYMMDD-hhmmss.bin
LOGFS	: rootfs.logfs.YYYYMMDD-hhmmss.bin

（YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）

環境変数と設定ファイル

上記オプションで指定する内容は、環境変数および定義ファイルでも設定が可能である。

参照する順番は、優先順位の高い方から、コマンドラインオプション、環境変数、定義ファイルの順である。

image-file 以外のオプションが示す情報は、コマンドラインオプション、環境変数または定義ファイルのいずれかでの指定が必須である。

環境変数

コマンドラインオプションに対応する環境変数は、以下の通り。

```
UBINUX_PKG_MKIMAGE_FROM_DIR  ⇔ [-f from-dir]
UBINUX_PKG_MKIMAGE_TO_DIR ⇔ [-t to-dir]
UBINUX_PKG_MKIMAGE_FILE   ⇔ [-o image-file]
```

定義ファイル

定義ファイルは、/root/.ubinux_pkg_tools.rc である。書式は、以下の通り。
ディレクトリ名は、絶対パスを指定する。シェルの環境変数の指定はできない。

```
### /root/.ubinux_pkg_tools.rc ###
UBINUX_PKG_MKIMAGE_FROM_DIR=/tmp/build-dir
UBINUX_PKG_MKIMAGE_TO_DIR=/tmp/image-dir
UBINUX_PKG_MKIMAGE_FILE=rootfs.image
### /root/.ubinux_pkg_tools.rc ###
```

上記以外の関連ファイル

to-dir/image-file.log
　イメージファイル作成時のログ出力ファイル。
/var/lock/ubinux_pkg_tools/make_image
　ツールの多重起動を防止するための排他獲得ファイル

第3章　パッケージインストーラの使用方法

3.1　パッケージインストールの手順

パッケージをインストール手順は、以下の通りです。

① パッケージインストーラを起動
② インストールパッケージの選択例を選択
③ 選択のカスタマイズ
④ デバッグインフォパッケージの選択
⑤ マニュアルパッケージをインストールするか否かを選択
⑥ インストールパッケージの確認
⑦ インストール

3.2　パッケージインストーラの起動

パッケージインストーラの起動例を以下に示します。

```
$ ubinux_pkg_install -i /opt/ubq/devkit/x86/x86/target -t /opt/ubq/trpmdb
                                                  ①                  ②
 -h /opt/ubq/devkit/x86_x86/hrpmdb -p /opt/ubq/devkit/x86/x86/rpms
                    ③                              ④
```

① ターゲット向けパッケージをインストールするディレクトリを指定します。
② ターゲット向けパッケージの rpm データベースを作成するディレクトリを指定します。
③ ホスト向けパッケージ（クロス環境ツール）の rpm データベースを作成するディレクトリを指定します。ホスト向けパッケージは、パッケージ内に定義されたディレクトリにインストールされます。(--nohost オプション指定時には不要。詳細については、2.1 を参照してください。)
④ パッケージファイルが格納されているディレクトリを指定します。

- ターゲット向けパッケージの rpm データベースは、実際には、① + ②のディレクトリに作成されます。上記例の場合、ターゲット向けパッケージの rpm データベースは、**/opt/ubq/devkit/x86/x86/target/opt/ubq/trpmdb 配下に作成されます。**
- 上記オプションは、定義ファイルまたは環境変数に定義することが可能です。詳細については、2.1 を参照してください。
- --info を用いた自動インストールの詳細については、3.15 を参照してください。

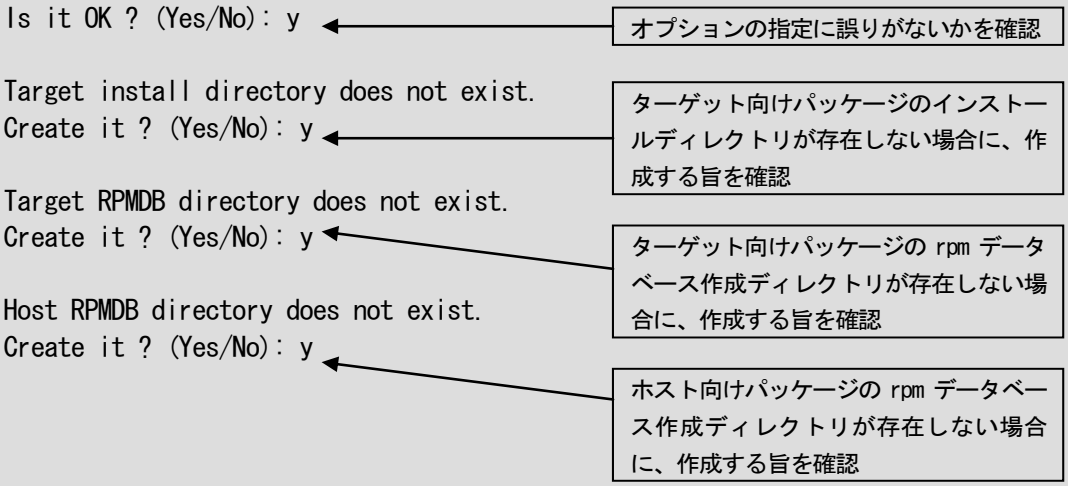
パッケージインストーラを起動すると、以下の確認メッセージが出力されます。メッセージの内容について問題がなければ、"y" を入力してください。

```
$ ubinux_pkg_install -i /opt/ubq/devkit/x86/x86/target -t /opt/ubq/trpmdb -h /opt
/ubq/devkit/x86/x86/hrpmdb -p /opt/ubq/devkit/x86/x86/rpms
********************************************************************************
 Target install directory      : /opt/ubq/devkit/x86/x86/target
 Target RPMDB directory        : /opt/ubq/devkit/x86/x86/target/opt/ubq/trpmdb
 Host RPMDB directory          : /opt/ubq/devkit/x86/x86/hrpmdb
 RPM packages directory        : /opt/ubq/devkit/x86/x86/rpms
 Log file                      : /root/ubinux_pkg_install.20110829-115317.log
 Log file (Installed packages) : /root/ubinux_pkg_install.20110829-115317.pkglist
********************************************************************************
Is it OK ? (Yes/No): y

Target install directory does not exist.
Create it ? (Yes/No): y

Target RPMDB directory does not exist.
Create it ? (Yes/No): y

Host RPMDB directory does not exist.
Create it ? (Yes/No): y
```



3.3 キャラクタ・ユーザ・インタフェース

パッケージインストーラを起動すると、キャラクタ・ユーザ・インタフェース（CUI）画面が表示されます。

図 4 CUI 画面

CUI 画面の基本操作は、以下の通りです。

① 選択項目に対してカーソルを合わせます。

[↑][↓]キーによりカーソルを上下させます。

② ホットキーに定義される動作を実行します。

ホットキーにより、チェックボックスのトグルや次の画面への移行などの動作を行います。

なお、各画面共通のホットキーの動作は、以下の通りです。

ホットキー	動作
SPACE or ENTER	項目を選択します。
N	次の画面に進みます。
B	前の画面に戻ります。
I	項目に関する詳細情報を表示します。
X	ツールを終了します。

3.4 インストールパッケージ選択例の選択

インストールパッケージ選択の推奨例、またはカスタマイズして保存したパッケージ選択が表示されます。ターゲットの目的に応じて、どの選択例を使用するか選択してください。

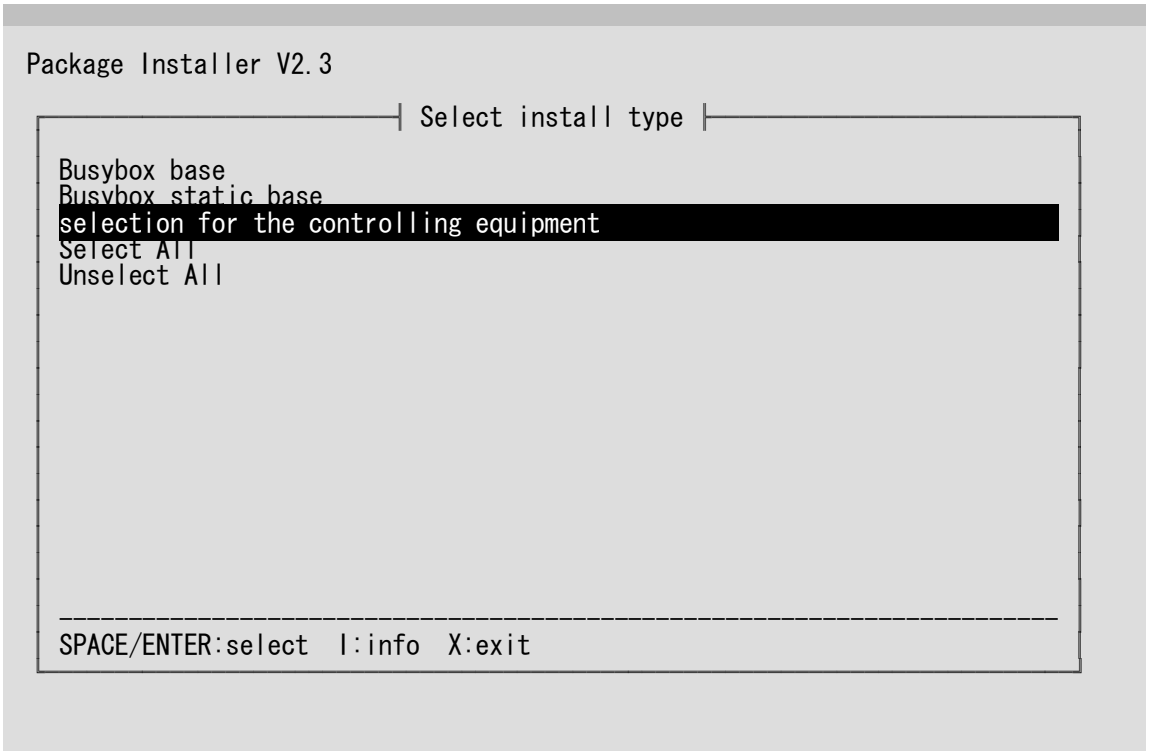


図 5 推奨例選択画面

選択項目には、パッケージ選択例の名前が表示されます。いずれかを選択すると、選択例に応じたパッケージがインストール対象として選択されます。

選択例に関する説明を参照する場合は、"I" を押下してください。

パッケージ選択例は、パッケージファイル格納ディレクトリ配下の XML ファイル ubinux_pkg_info.xml に定義されています。この XML ファイルは後述する選択状況保存にて選択例を追加したり、直接この XML ファイルを変更したりすることにより、選択例をカスタマイズすることが可能です。詳細については、3.14 を参照してください。

"Select All" および "Unselect All" は、XML の定義によらず、必ず表示される項目です。"Select All" を選択すると、パッケージファイル格納ディレクトリ配下のすべてのパッケージがインストール対象に選択されます。"Unselect All" を選択すると、すべてのパッケージがインストール対象に選択されません。

なお、未インストールのパッケージが存在しない場合、本画面は表示されません。

3.5 カテゴリの選択

インストールするパッケージの選択画面で、全てのカテゴリを表示するか、カテゴリ単位で表示するかを選択する事ができます。

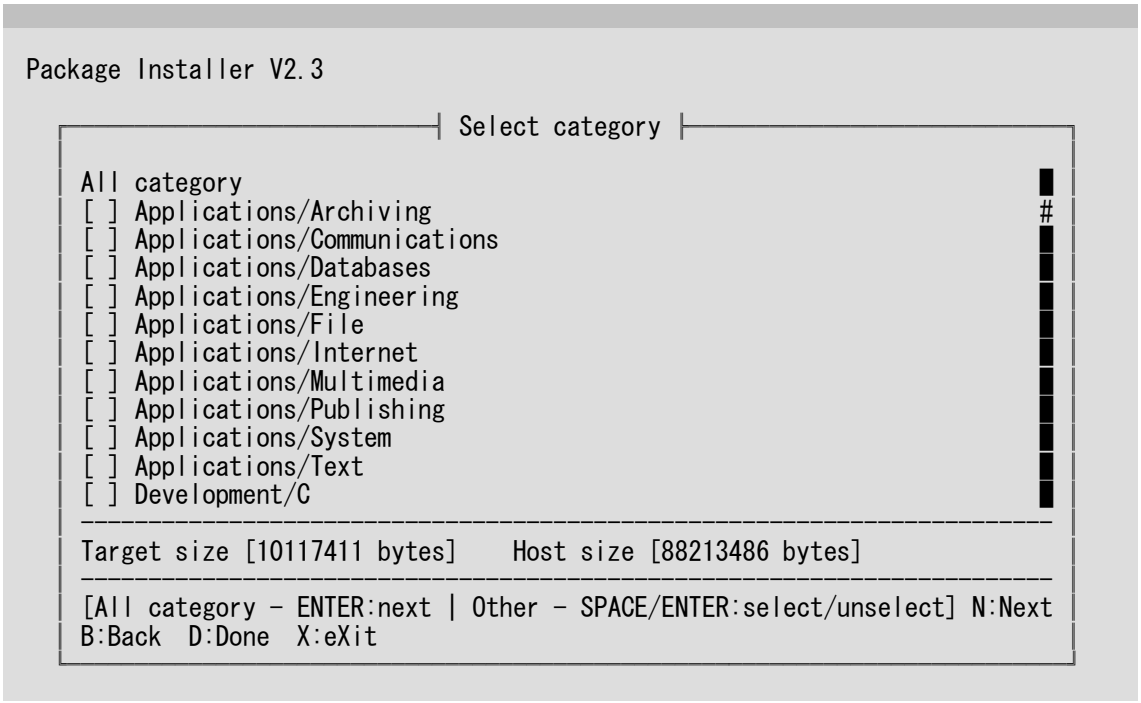


図 6 カテゴリの選択画面

選択項目には全てのカテゴリ（All category）とパッケージ群のカテゴリ名が表示されます。

パッケージ一覧の下部 “Target size” および “Host size” には、インストール済み および インストール対象に選択されたパッケージのサイズの合計値が表示されます。サイズは、rpm パッケージのヘッダ情報から取得できるサイズです。“Target size”には、ターゲット向けパッケージのサイズが表示され、“Host size”には、ホスト向けパッケージのサイズが表示されます。

チェックボックスは、以下を意味します。

表示	意味
[*]	該当カテゴリは表示対象です。
[ ]	該当カテゴリは表示対象ではありません。

ホットキーの動作は、以下の通りです。

ホットキー	動作
ENTER	（カーソルが All category の時）全てのカテゴリが一覧されるパッケージ選択画面へ。
SPACE/ENTER	（カーソルが All category 以外の時）カテゴリの選択／非選択を切り替えます。
N	パッケージ選択画面に進みます。
B	前の画面に戻ります。現在の選択状態は破棄されます。
D	選択されたパッケージに対応するデバッグインフォパッケージが存在する場合デバッグインフォパッケージの選択画面に進みます。対応するデバッグインフォパッケージが存在しない場合にはマニュアルインストールの選択に進みます。
X	ツールを終了します。

3.6 選択のカスタマイズ

ターゲット向けパッケージの一覧と現在の選択状態が表示されます。必要に応じてインストールするパッケージの選択状態を変更してください。

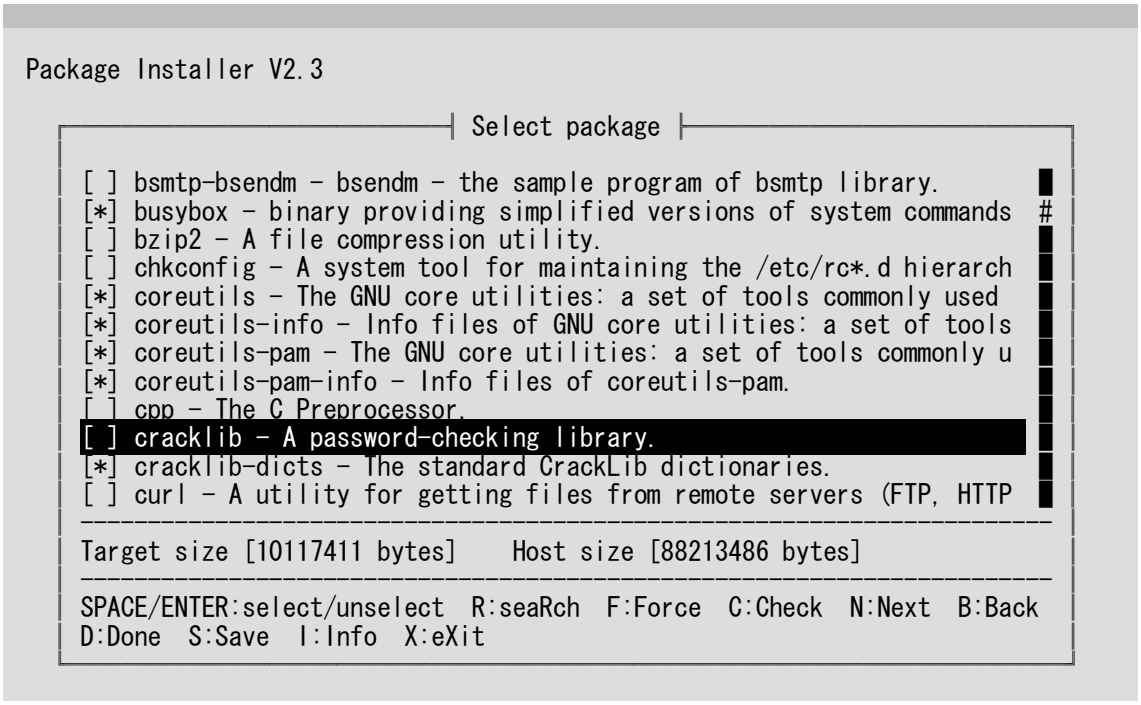


図 7 選択のカスタマイズ画面

選択項目には、パッケージ名とサマリーが表示されます。

パッケージ一覧の下部 “Target size” および “Host size” には、インストール済み および インストール対象に選択されたパッケージのサイズの合計値が表示されます。サイズは、rpm パッケージのヘッダ情報から取得できるサイズです。“Target size”には、ターゲット向けパッケージのサイズが表示され、“Host size”には、ホスト向けパッケージのサイズが表示されます。

このサイズは、"C" または "N" を押下すると更新されます。

チェックボックスは、以下を意味します。

表示	意味
[I]	既にインストールされています。状態は変更できません。
[*]	該当パッケージはインストール対象です。
[ ]	該当パッケージはインストール対象ではありません。
[F]	該当パッケージは必ずインストールするよう選択されています（3.7 参照）。

ホットキーの動作は、以下の通りです。

ホットキー	動作
SPACE/ENTER	選択／非選択を切り替えます。
F	必ずインストールするパッケージとして選択します（3.7 参照）。
R	パッケージ検索画面に進みます。
C	選択されたパッケージについて依存関係をチェックし、依存関係に応じて選択状態を変更します（3.7 参照）。
N,B	カテゴリ選択画面に戻ります。
D	選択されたパッケージに対応するデバッグインフォパッケージが存在する場合デバッグインフォパッケージの選択画面に進みます。 対応するデバッグインフォパッケージが存在しない場合にはマニュアルインストールの選択に進みます。 依存関係のチェックが行われていない場合は、依存関係のチェック後、デバッグインフォパッケージ選択画面に進みます。
S	選択されたパッケージ状態を保存します（3.7 参照）。
I	パッケージの詳細情報を表示します。
X	ツールを終了します。

未インストールのパッケージが存在しない場合、本画面は表示されません。

パッケージの詳細情報を参照する場合は、"I" を押下してください。パッケージの名前、バージョン番号、リリース番号、サイズ、ライセンス、サマリー、詳細情報およびファイル一覧、パッケージ依存状況が表示されます。

表示されるパッケージ一覧は、前画面のカテゴリ選択の状態によって変わります。必要に応じてカテゴリ別に使い分けてください。

選択項目には、ターゲット向けのデバッグインフォパッケージ、マニュアルパッケージ、ライセンステキストのパッケージおよびホスト向けパッケージは表示されません。これらについては、以下のようにインストールされます。

- ターゲット向けのデバッグインフォパッケージは、対応する主パッケージがインストール対象に選択された場合に、インストール対象候補になります。実際にデバッグインフォパッケージをインストールするか否かは、次の画面で選択します。
- ターゲット向けのマニュアルパッケージは、対応する主パッケージがインストール対象に選択された場合に、インストール対象候補になります。実際にマニュアルパッケージをインストールするか否かは、デバッグインフォパッケージ選択後の画面で選択します。

- ターゲット向けのライセンステキストのパッケージは、対応する主パッケージがインストール対象に選択された場合に、自動的にインストール対象になります。
- ホスト向けパッケージは、無条件に全てがインストール対象になります。ホスト向けパッケージをインストールする必要のない環境で使用する場合には--nohost オプションを使用してください。

ホスト向けパッケージのファイル名　　：　*-cross-*.rpm
ターゲット向けマニュアルパッケージのファイル名　：　上記以外の *-licence.*.rpm
ターゲット向けライセンスパッケージのファイル名　：　上記以外の *-manual.*.rpm
ターゲット向けデバッグインフォパッケージのファイル名
：　上記以外の *-debuginfo.*.rpm
ターゲット向け主パッケージのファイル名　　：　上記以外の *.rpm

3.7　依存関係のチェックについて

本ツールは、インストール対象に選択されたパッケージについて依存関係をチェックし、必要なパッケージが選択されていなければ、以下のポリシーで選択状態を自動的に変更します。

- パッケージを 非選択 → 選択 に変更した場合
  選択したパッケージの前提パッケージがインストール対象ではない場合は、前提パッケージを自動的に選択状態に変更します。

- パッケージを 選択 → 非選択 に変更した場合
  非選択にしたパッケージを前提とするパッケージが存在する場合、これらのパッケージも自動的に非選択状態に変更します。
  ただし、利用者が必ずインストールするパッケージとして選択（[F]指定）したものに対しては、非選択状態に変更しません。この場合は、利用者が非選択にした前提パッケージを選択状態に戻します。

自動変更の例を以下に示します。表中のパッケージ **A**,b,c は次の依存関係にあるとします。

- pkg **A** は pkg b に必要である。
- pkg **A** は pkg c に必要である。

	利用者操作			動作	チェック後
1	[ ]pkg **A** [ ]pkg b [ ]pkg c	⇒	[ ]pkg **A** **[*]pkg b** [ ]pkg c	利用者：パッケージ b を選択。 ツール：パッケージ b の前提パッケージ **A** を自動的に選択する。	[*]pkg **A** [*]pkg b [ ]pkg c
2	[*]pkg **A** [*]pkg b [ ]pkg c	⇒	**[ ]pkg A** [*]pkg b [ ]pkg c	利用者：パッケージ **A** を非選択。 ツール：パッケージ **A** を前提とするパッケージ b を自動的に非選択にする。	[ ]pkg **A** [ ]pkg b [ ]pkg c
3	[*]pkg **A** [*]pkg b [ ]pkg c		**[ ]pkg A** [*]pkg b **[*]pkg c**	利用者：パッケージ **A** を非選択にし、かつ、パッケージ c を選択。 （選択と非選択の競合ケース） ツール：パッケージ **A** を前提とするパッケージ b を自動的に非選択にする。 また、パッケージ c の前提パッケージ **A** を再び選択する。	[*]pkg **A** [ ]pkg b [*]pkg c

4	[*]pkg **A** [*]pkg b [F]pkg c	⇒	[ ]pkg **A** [*]pkg b [F]pkg c	利用者：パッケージ A を非選択。 （「F」指定ありのケース） ツール：パッケージ A を前提とするパッケージ b を自動的に非選択にする。 「F」のパッケージ c は非選択にせず、パッケージ c の前提パッケージ A を再び選択する。	[*]pkg **A** [ ]pkg b [F]pkg c

ツールによって選択状態が自動的に変更される場合は、以下の画面が表示されます。変更内容に問題がなければ、"Y" を押下してください。"N" を押下した場合は、変更は反映されず、利用者が指定した状態のまま、選択のカスタマイズ画面に戻ります。

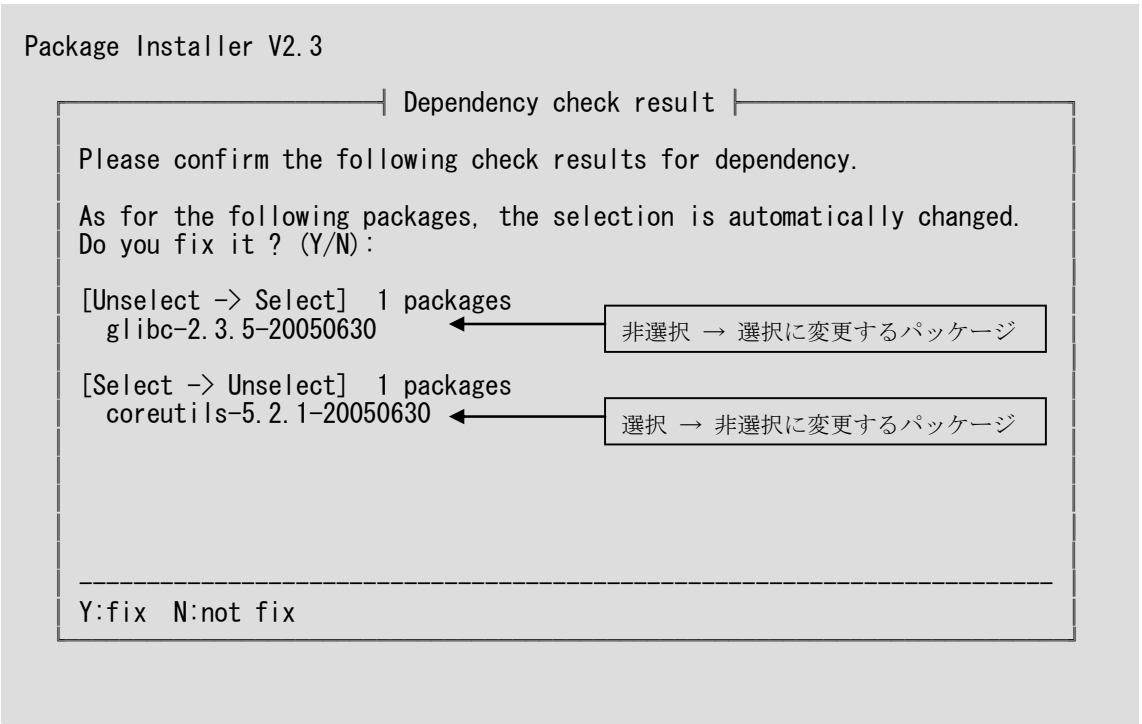


図 8 依存関係チェック結果の確認画面

パッケージファイル格納ディレクトリからファイルを削除したなどの理由により、前提パッケージが見つからず、依存関係が解決できなかった場合は、以下のエラー画面が表示されます。

以下の画面で示されるパッケージのファイルをパッケージファイル格納ディレクトリに追加して、本ツールを再起動してください。

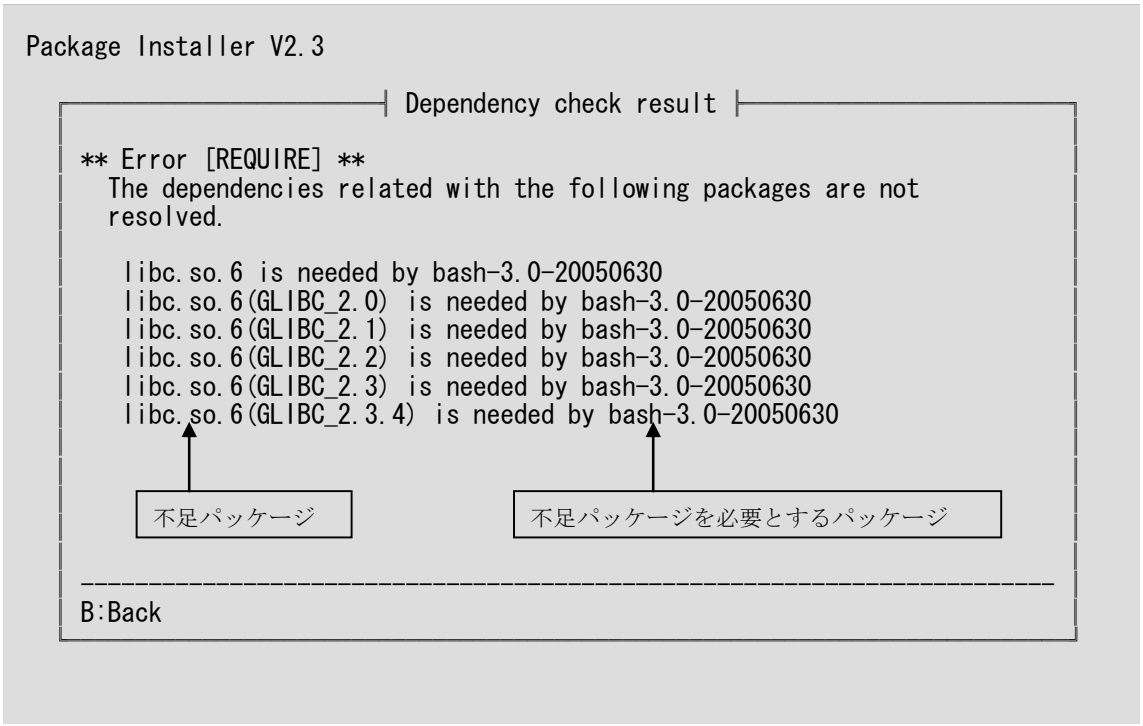


図 9 前提パッケージ不足の通知画面

パッケージ間に競合（conflict）がある場合は、以下のエラー画面が表示されます。以下の画面に示されるパッケージは、機能が競合しており、いずれか一方のみインストール可能です。いずれか一方のみを選択して、再度、依存関係のチェックを実施してください。

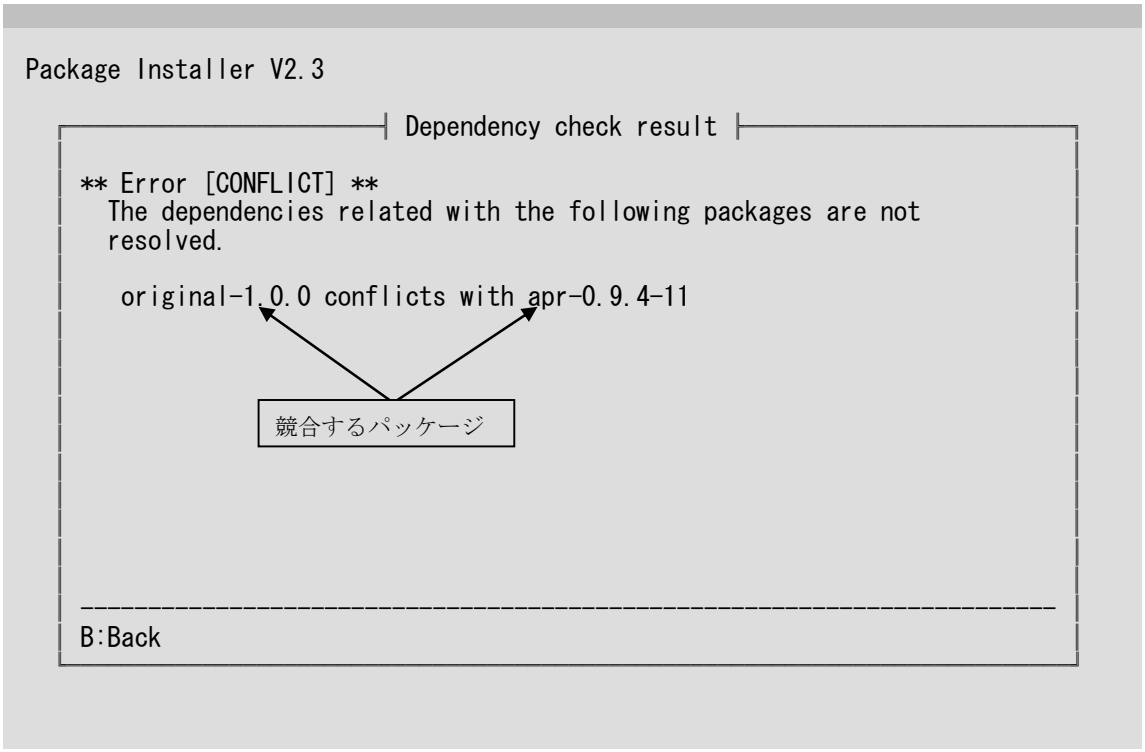


図 10 パッケージ競合の通知画面

3.8 パッケージ検索

パッケージ検索では 図 11 の、キャラクタ・ユーザ・インタフェース（CUI）画面が表示されます。

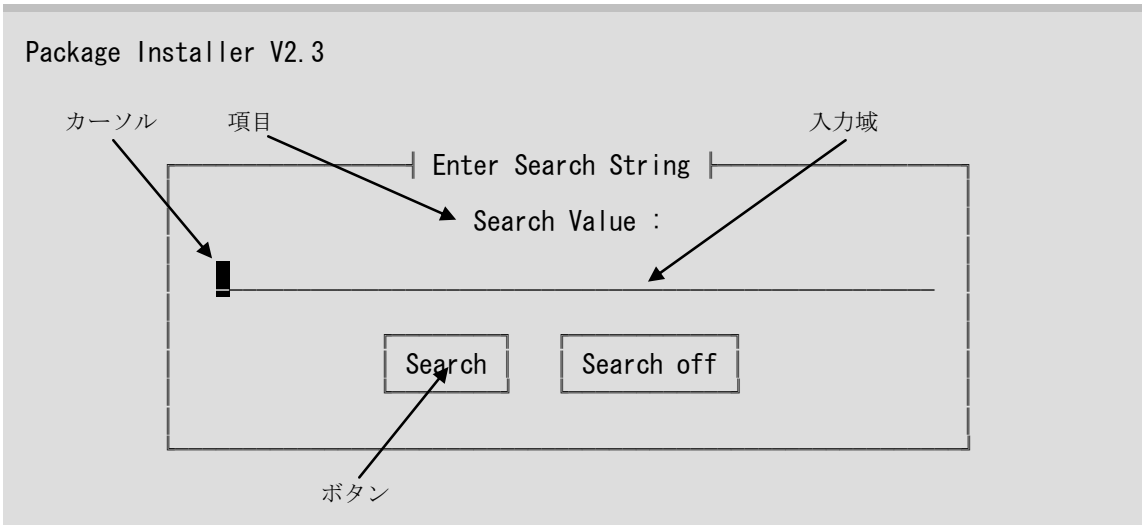


図 11 CUI 画面

パッケージ検索画面の基本操作は、以下の通りです。

① 検索文字列入力を行います。
アンダーバー付きの項目にカーソルを合わせ、検索文字を入力します。

② 動作を行います。
下部に表示されるボタンに[↑] [↓] [TAB]キーを使用してカーソルを合わせ、[ENTER]キーにより、ボタンが示す動作を実行します。

検索文字を含むパッケージ名をパッケージ選択画面から選択出来るようになります。
以下のパラメータを入力してください。
パラメータの入力後に“Search”ボタンを押すと検索モードでパッケージ選択が出来るようになります。図 12 で示す検索モード画面では画面上部に検索文字列が表示されるようになります。
“Search off”ボタンを押すと通常のモードに戻ります。

入力項目	入力値
Search Value	検索するパッケージの文字列を入力してください。入力した値と部分一致するパッケージが絞り込まれて表示されます。

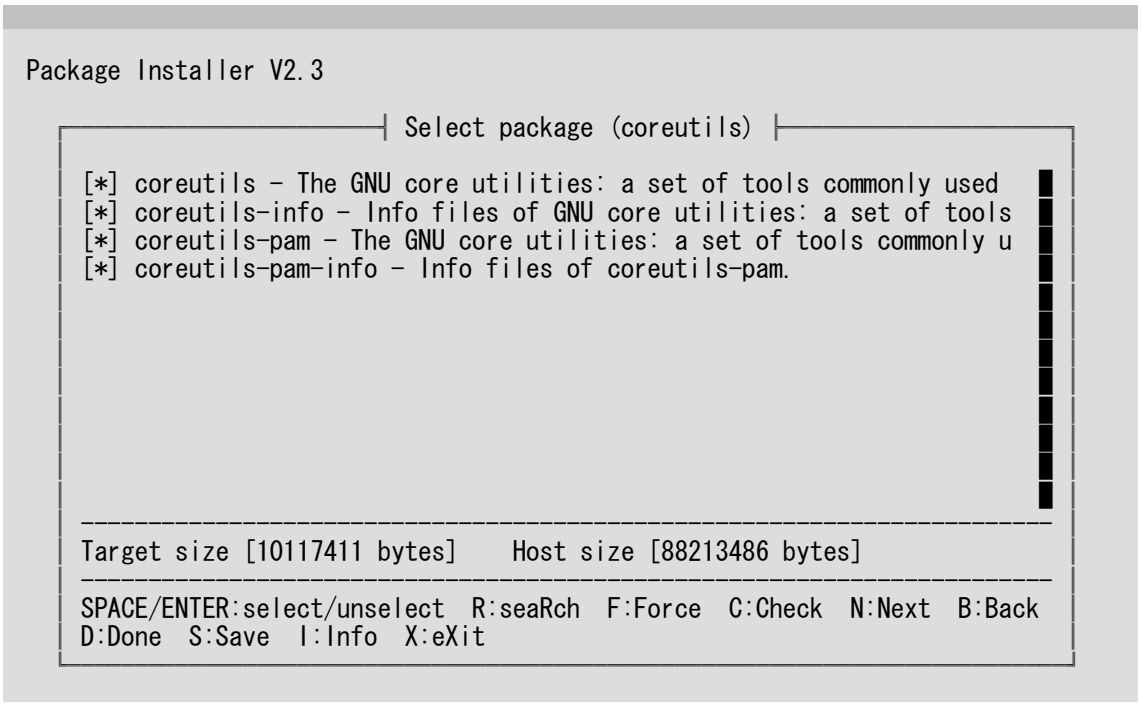

Package Installer V2.3
Select package (coreutils)
[*] coreutils - The GNU core utilities: a set of tools commonly used
[*] coreutils-info - Info files of GNU core utilities: a set of tools
[*] coreutils-pam - The GNU core utilities: a set of tools commonly u
[*] coreutils-pam-info - Info files of coreutils-pam.
Target size [10117411 bytes] Host size [88213486 bytes]
SPACE/ENTER:select/unselect R:seaRch F:Force C:Check N:Next B:Back
D:Done S:Save I:Info X:eXit

図 12 検索モード画面

3.9 選択されたパッケージ状態の保存

選択されたパッケージ状態の保存では 図 13 の、キャラクタ・ユーザ・インタフェース（CUI）画面が表示されます。

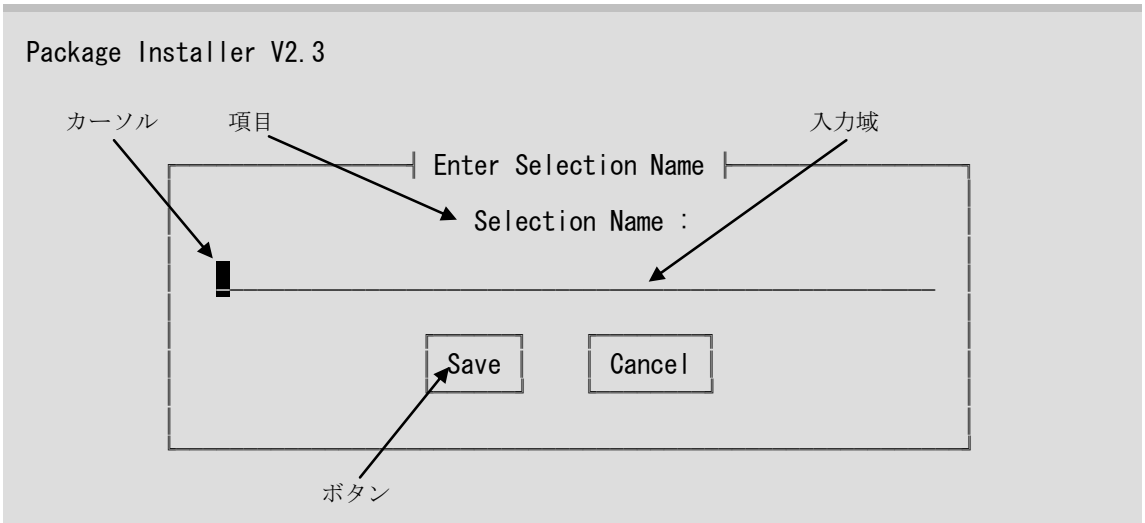


図 13 CUI 画面

選択されたパッケージ状態の保存画面の基本操作は、以下の通りです。

① 入力を行います。
アンダーバー付きの項目にカーソルを合わせ、文字を入力します。

② 動作を行います。
下部に表示されるボタンに[↑] [↓] [TAB]キーを使用してカーソルを合わせ、[ENTER]キーにより、ボタンが示す動作を実行します。

インストール対象に選択されたパッケージの情報を保存することで、インストーラ起動時に選択できるようにします。

以下のパラメータを入力してください。

パラメータの入力後に“Save”ボタンを押して保存します。

入力項目	入力値
Selection Name	希望するパッケージ選択の名称を入力してください。パッケージ推奨例の選択（3.4）で表示されるラベルの名称になります。（英数字記号 32 文字まで） 既に存在する名称を入力するとその名称の選択例を上書きします。

3.10 デバッグインフォパッケージの選択

選択したターゲット向けパッケージに対応するデバッグインフォパッケージの一覧と現在の選択状態が表示されます。必要に応じてインストールするデバッグインフォパッケージの選択状態を変更してください。

デバッグインフォパッケージは各 ubinux ミドルウェアをソースレベルデバッグするために必要なデバッグ情報を含んだパッケージです。パッケージごとに対応するデバッグインフォパッケージが存在します。

gdb を使用して ubinux ミドルウェアをソースレベルデバッグしたい場合、デバッグ対象となるパッケージのデバッグインフォパッケージを選択ください。

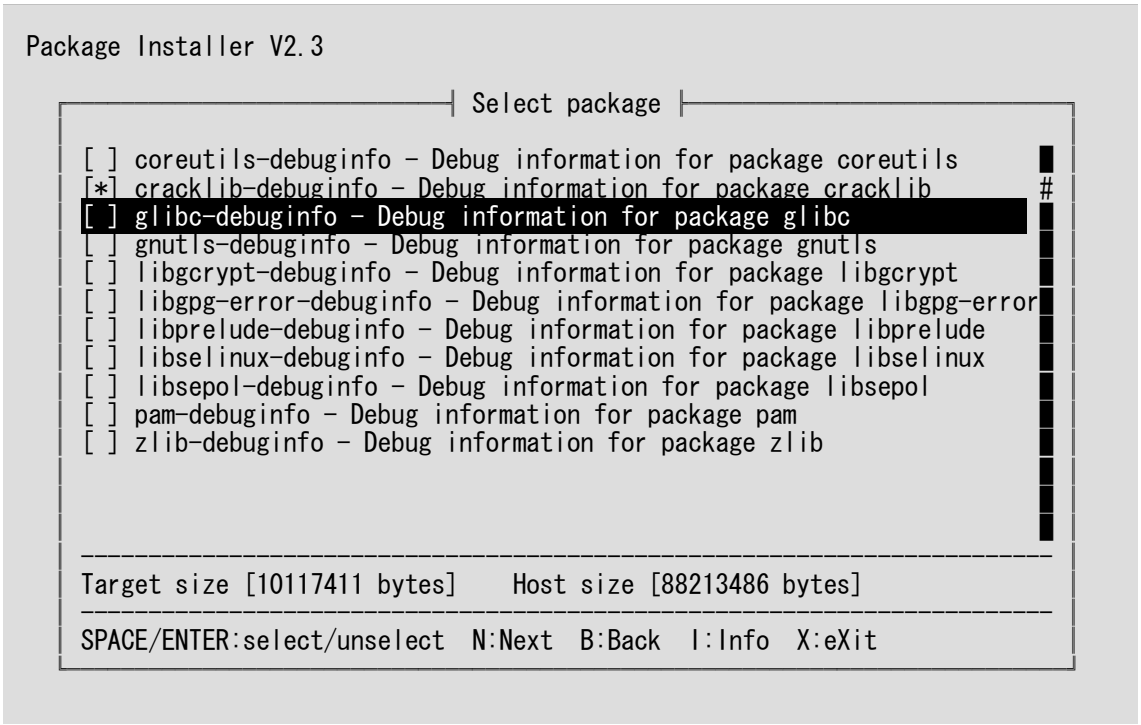


図 14 デバッグインフォパッケージ選択画面

選択項目には、パッケージ名とサマリーが表示されます。

チェックボックスは、以下を意味します。

表示	意味
[I]	既にインストールされています。状態は変更できません。
[*]	該当パッケージはインストール対象です。
[ ]	該当パッケージはインストール対象ではありません。

ホットキーの動作は、以下の通りです。

ホットキー	動作
SPACE/ENTER	選択／非選択を切り替えます。
N	マニュアルインストールの選択に進みます。
B	前の画面に戻ります。現在の選択状態は破棄されます。
I	パッケージの詳細情報を表示します。
X	ツールを終了します。

全てのデバッグインフォパッケージがインストール済である場合、本画面は表示されません。

パッケージの詳細情報を参照する場合は、"I" を押下してください。パッケージの名前、バージョン番号、リリース番号、サイズ、ライセンス、サマリー、詳細情報およびファイル一覧、パッケージ依存状況が表示されます。

3.11 マニュアルインストールの選択

マニュアルパッケージをインストールするか否かを問い合わせるメッセージが表示されます。ターゲットの目的に応じて、マニュアルパッケージをインストールするか否かを選択してください。

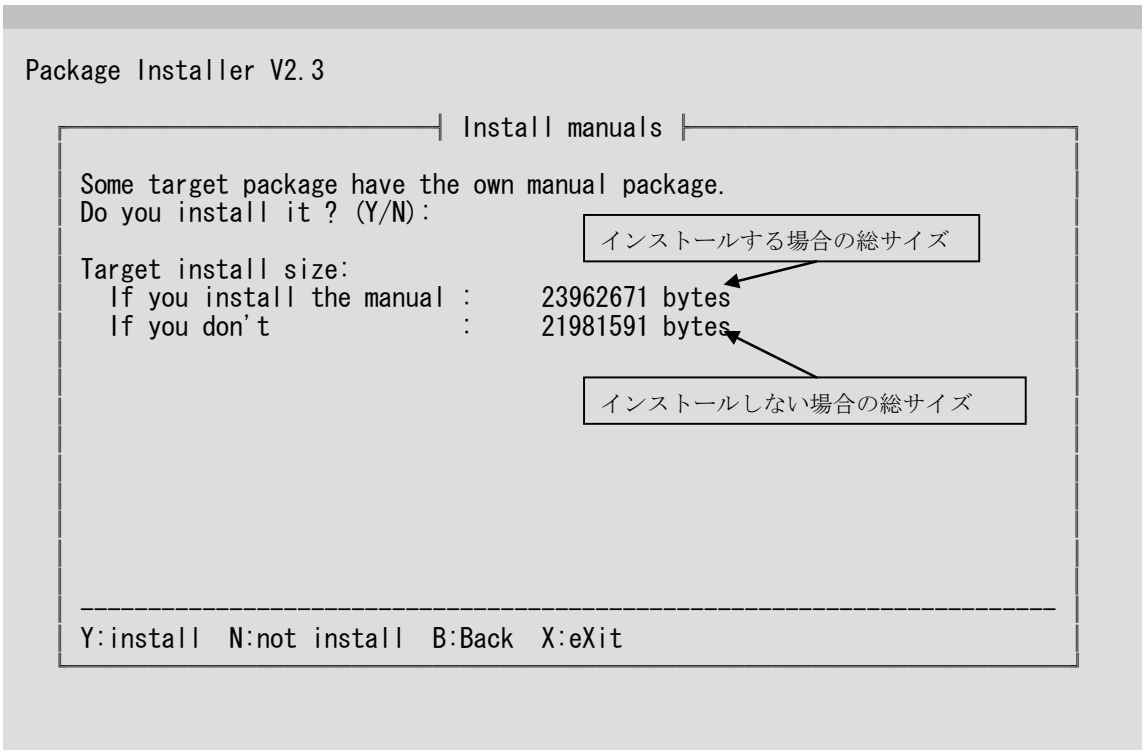


図 15 マニュアルインストールの選択画面

画面には、マニュアルパッケージをインストールする場合とインストールしない場合のターゲット向けパッケージの総サイズが表示されます。

マニュアルパッケージをインストールする場合は "Y" を、インストールしない場合は "N" を押下してください。

"Y" を押下した場合、前節までの作業でインストール対象に選択されたパッケージに関するすべてのマニュアルパッケージがインストール対象に選択されます。

前節までの作業でインストール対象に選択ターゲットされたパッケージについて、関連するマニュアルパッケージが存在しない場合、本画面は表示されません。

3.12 インストールパッケージの確認

インストール対象パッケージの一覧が表示されます。選択されたパッケージに問題がないか確認してください。

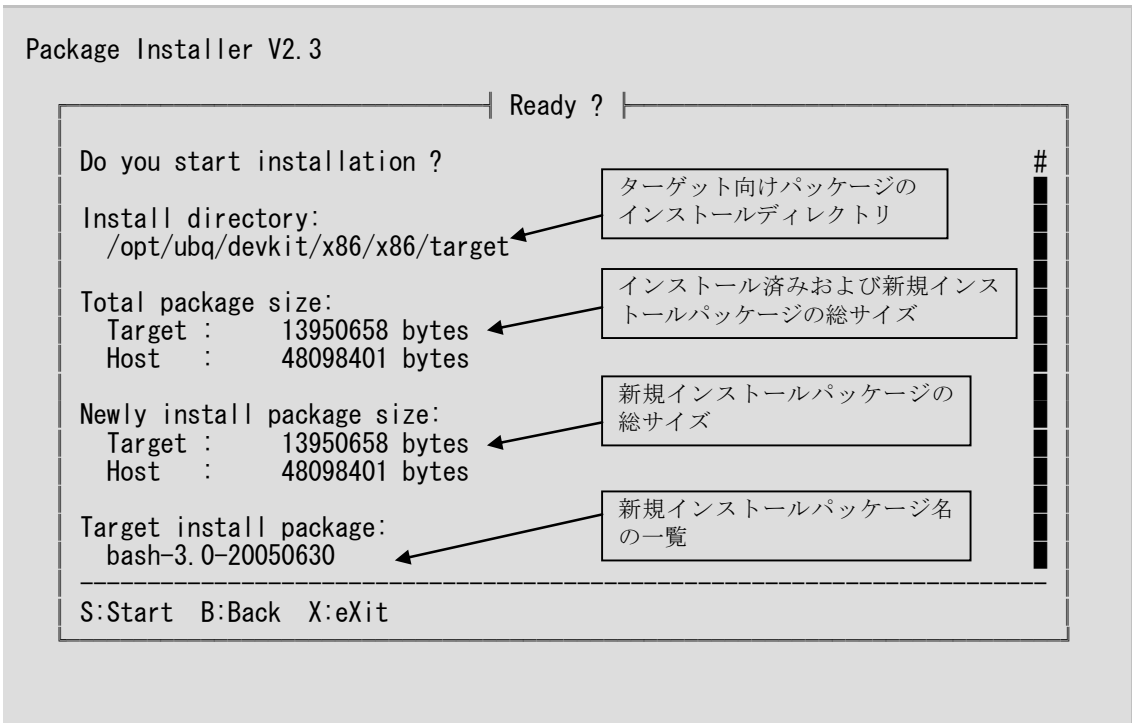


図 16 インストールパッケージの確認画面

選択されたパッケージに問題がない場合は、"S" を押下して、インストールを開始してください。

3.13 インストール

インストール中はインストール状況が表示されます。

```
Install start.
(   1/  10)setup                ######################################## [100%]
(   2/  10)filesystem           ######################################## [100%]
(   3/  10)glibc                ######################################## [100%]
(   4/  10)coreutiles           ######################################## [100%]
(   5/  10)busybox              ######################################## [100%]
(   6/  10)bash                 ######################################## [100%]
(   7/  10)coreutils
  [File conflict] Is it OK overwrite the following files?
    /bin/cat - busybox (Yes/No/All/nOne) y
    /bin/chgrp - busybox (Yes/No/All/nOne) y      ←①
    /bin/chmod - busybox (Yes/No/All/nOne) y
    /bin/chown - busybox (Yes/No/All/nOne) a
(   7/  10)coreutils            ######################################## [100%]
(   8/  10)armv5b-cross-binutils######################################## [100%]
(   9/  10)armv5b-cross-gcc     ######################################## [100%]
(  10/  10)armv5b-cross-gdb     ######################################## [100%]

Install succeeded.
Package installer finish.
```

図 17 インストール中の画面

本ツールは、インストールパッケージによって既存ファイルが上書きされる状況が発生した場合、上記①のように各ファイルに対して、上書きするか否かの問い合わせを行います。起動オプションに --allall 又は --allnone を指定した場合には問い合わせは発生せずに指定した動作を行います。

以下のいずれかを入力してください。

入力	動作
y or yes	問い合わせ中のファイルを上書きする。
n or no	問い合わせ中のファイルを上書きしない。
a or all	処理中パッケージの以降のファイルに関して、すべて上書きを行う。
o or none	処理中パッケージの以降のファイルに関して、すべて上書きを行わない。

インストール後に以下の情報がファイルへ保存されます。

・ インストールログ
  - -o オプションで指定したログファイル、または以下
  - ~/ubinux_pkg_install.YYYY MMDD-hhmmss.log
    （YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）

・ インストール済みの rpm パッケージの一覧
  - -l オプションで指定したログファイル、または以下
  - ~/ubinux_pkg_install.YYYY MMDD-hhmmss.pkglist
    （YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）

・ インストール情報
  - ubinux_inst_info.xml にインストール途中で問い合わせのあった上書きする／しないの情報を含むインストールに必要な情報が出力されます。

--info オプションにて上記、ubinux_inst_info.xml を起動時に読み込み、問い合わせ応答も含めてインストールを再現することが可能です。

3.14 外部 xml ファイルについて

パッケージファイル格納ディレクトリ直下の XML ファイル ubinux_pkg_info.xml には、以下を定義します。

- プレインストールパッケージ
- ポストインストールパッケージ
- 選択可プレインストールパッケージ
- 選択可ポストインストールパッケージ
- 競合するパッケージ
- 選択しないパッケージ
- インストールパッケージの選択例

プレインストールパッケージには、他のパッケージをインストールする前にインストールする必要がある必須パッケージを定義します。通常は、パッケージ "setup" と "filesystem" を指定します。

ポストインストールパッケージには、他のパッケージをインストールした後にインストールする必要がある必須パッケージを定義します。現在のところ、該当パッケージはありません。

プレインストールパッケージ、ポストインストールパッケージに記述したものは必須パッケージとなりインストールしないことは出来ませんが、選択可プレインストールパッケージ、選択可ポストインストールパッケージでは選択のカスタマイズにてインストールするかを選択可能です。

選択可プレインストールパッケージには、他のパッケージをインストールする前にインストールする必要があるパッケージを定義します。現在のところ、該当パッケージはありません。

選択可ポストインストールパッケージには、他のパッケージをインストールした後にインストールする必要がある必須パッケージを定義します。既定値では、パッケージ “busybox” が指定されています。

競合するパッケージには、--allnone、--allall オプションが有効な時にも強制的に問い合わせを行うパッケージを定義します。現在のところ、該当パッケージはありません。

選択しないパッケージには、”インストールパッケージ選択例”の選択で All を選んだ時でも既定で選択状態にならないパッケージを定義します。既定値ではパッケージ “busybox-static”、”busybox-su”、”opensaf-controller”、”opensaf-payload” が指定されています。

外部 XML ファイルの記述形式は、以下の通りです。

```
<?xml version="1.0"?>
<ubinux-packagelist>

    <preinstall-packagelist>
        <name>                              ## ← 通常は"setup"と"filesystem"を指定する
            プレインストールするパッケージ名
        </name>
    </preinstall-packagelist>

    <postinstall-packagelist>
        <name>
            ポストインストールするパッケージ名
        </name>
    </postinstall-packagelist>

    <selectable-preinstall-packagelist>
        <name>
            選択された時にプレインストールするパッケージ名
        </name>
    </selectable-preinstall-packagelist>

    <selectable-postinstall-packagelist>
        <name>
            選択された時にポストインストールするパッケージ名
        </name>
    </selectable-postinstall-packagelist>

    <conflict-packagelist>
        <name>
            ユーザーが選択する競合パッケージ名
        </name>
    </conflict-packagelist>

    <unselect-packagelist>
        <name>
            全て(All)を選択した場合でもデフォルトで選択状態にしないパッケージ名
        </name>
    </unselect-packagelist>

    <recommend-group>
        <id> パッケージ選択例名 </id>
        <description>
            選択例の説明
        </description>

        <list>
            <name>                          ## ← デフォルトの選択状態が [*] になる
                選択状態にするパッケージ名
            </name>
            <name status="Force">           ## ← デフォルトの選択状態が [F] になる
                選択状態にするパッケージ名
            </name>
        </list>
    </recommend-group>
```

- 本 XML ファイルに記述したパッケージは、すべてターゲット向けパッケージとして扱われます。ホスト向けパッケージを記述した場合は、無視されます。
- パッケージ名には、バージョン番号およびリリース番号を含めないでください。

- <preinstall-packagelist> と <postinstall-packagelist> に記述したパッケージは、記述した順番にインストールされます。
  <preinstall-packagelist> にパッケージを追加する場合は、"setup" と "filesystem" の後に追加してください。"setup" と "filesystem" は、削除しないでください。
- <preinstall-packagelist> と <postinstall-packagelist> に記述したパッケージについては、依存関係のチェックが行われません。前提パッケージを必要とするパッケージは記述しないでください。
- <preinstall-packagelist> と <postinstall-packagelist> に記述したパッケージは、本ツールの「選択のカスタマイズ画面」に表示されません。
- <postinstall-packagelist>は、省略可能です。
- <selectable-preinstall-packagelist> と <selectable-postinstall-packagelist> に記述したパッケージは「選択のカスタマイズ画面」に表示されるので通常のパッケージと同様の操作が行えます。
- <selectable-preinstall-packagelist> と <selectable-postinstall-packagelist> に記述したパッケージについては、通常の選択パッケージ同様に依存関係のチェックが行われますが、前提パッケージもプレインストール、またはポストインストールにはなりません。
- <selectable-preinstall-packagelist> と <selectable-postinstall-packagelist> は省略可能です。
- <conflict-packagelist> は省略可能です。
- <unselect-packagelist> に記述している busybox-static と busybox-su は削除しないで下さい。
- インストールパッケージの選択例は、<recommend-group> に記述します。
  選択状態にするパッケージを <list> の <name> に記述してください。
  <name>は、複数指定可能です。
- 「F」選択状態にする場合は、<name> に status="Force" を記述してください。
- <recommend-group> は、省略可能、かつ、複数指定可能です。
- <recommend-group> 内の <id> は、省略不可です。

記述例を以下に示します。

```
<?xml version="1.0"?>
<ubinux-packagelist>

    <preinstall-packagelist>
        <name> setup </name>
        <name> filesystem </name>
    </preinstall-packagelist>

    <selectable-postinstall-packagelist>
        <name> busybox </name>
    </selectable-postinstall-packagelist>

    <unselect-packagelist>
        <name> busybox-static </name>
        <name> busybox-su </name>
        <name> opensaf-controller </name>
        <name> opensaf-payload </name>
    </unselect-packagelist>

    <recommend-group>
        <id> Example1 </id>
        <description>
            This is a set for test.
        </description>
        <list>
            <name status="Force"> glibc </name>
            <name> coreutiles </name>
            <name> bash </name>
        </list>
    </recommend-group>

    <recommend-group>
        <id> Example2 </id>
        <description>
            This is a set for test.
        </description>
        <list>
```

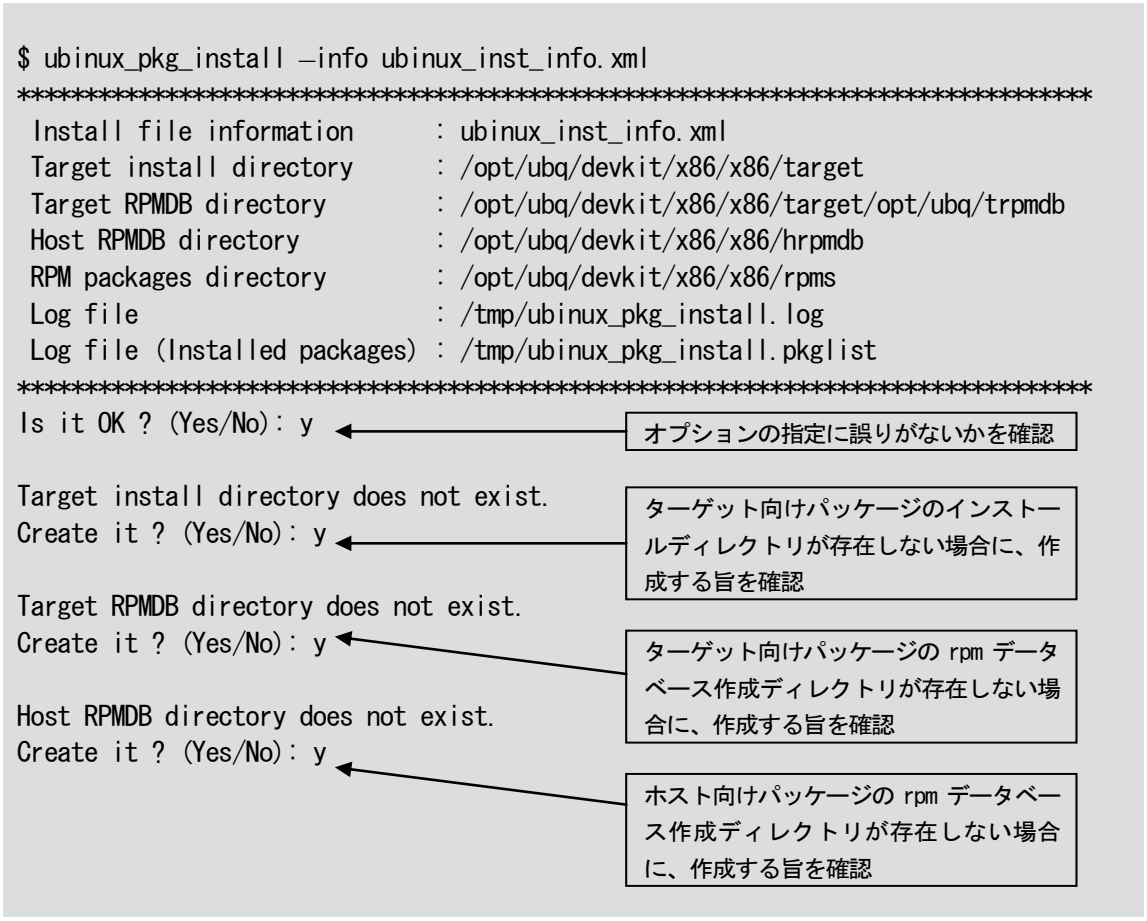
3.15 自動インストールオプションについて

ここでは、--info オプションを使用した自動インストールについて説明を致します。自動インストールを行うには事前に作成された ubinux_inst_info.xml ファイルが必要となります。

当ファイルは通常のインストール処理を行った際に target_dir/target_rpmdb_dir/の中に自動的に作成されます。

当オプションを使用してインストールする時には target_dir/target_rpmdb_dir がインストール済みのディレクトリは選択できません。

パッケージインストーラを起動すると、以下の確認メッセージが出力されます。メッセージの内容について問題がなければ、"y" を入力してください。



この後のインストールの流れは 3.12 からと同様となります。インストール時の問い合わせは ubinux_inst_info.xml に則って行われます。パッケージの追加／改版などによって新たな問い合わせが必要な場合には入力が必要になります。

ubinux_inst_info.xml ファイルの記述形式は、以下の通りです。

```
<?xml version="1.0" encoding="UTF-8"?>
<ubinux_install_infomation>
 <target-dir dir="/opt/ubq/devkit/x86/x86/target" />
 <target-rpm-dir dir="/opt/ubq/trpmdb" />
 <host-rpm-dir dir="/opt/devkit/x86/x86/hrpmdb" />
 <rpm-pkg-dir dir="/opt/devkit/x86/x86/rpms" />
 <log-file path="/tmp/ubinux_pkg_install.log"/>
 <log-file-pkg path="/tmp/ubinux_pkg_install.pkglist"/>
 <manual-install flag="0"/>
 <installation-sequence>
  <list cat="pre" name="setup" seq="1"/>
  <list cat="pre" name="filesystem" seq="2"/>
   ......
  <list cat="post" name="busybox" seq="628"/>
 </installation-sequence>
 <conflict-files>
  <file name="/bin/cat">
   <overwrited package="coreutils">
    <package name="busybox"/>
   </overwrited>
  </file>
  <file name="/bin/chgrp">
   <overwrited package="coreutils">
    <package name="busybox"/>
   </overwrited>
  </file>
   ......
  <file name="/var/named/chroot/etc/named.conf">
   <overwrited package="bind-chroot">
    <package name="caching-nameserver"/>
   </overwrited>
  </file>
 </conflict-files>
</ubinux_install_infomation>
```

3.16 インストール済みの rpm パッケージの一覧について

インストール終了時には、インストール済みの rpm パッケージの一覧が以下のファイルに出力されます。

- -l オプションで指定したログファイル、または、
- ~/ubinux_pkg_install.YYYY MMDD-hhmmss.pkglist
  （YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）

一覧ファイルの出力内容は、以下の通りです。

一覧ファイルには、rpm パッケージのファイル名が出力されます。ディレクトリ名は含まれません。

```
PRE: パッケージファイル名       ← ターゲット向けプレインストールパッケージ
PRE: パッケージファイル名
・・・・
NORMAL: パッケージファイル名   ← ターゲット向け通常インストールパッケージ
NORMAL: パッケージファイル名
NORMAL: パッケージファイル名
・・・・
POST: パッケージファイル名     ← ターゲット向けポストインストールパッケージ
POST: パッケージファイル名
・・・・
HOST: パッケージファイル名     ← ホスト向けパッケージ
HOST: パッケージファイル名
・・・・
```

一覧ファイルを利用して、パッケージインストーラを使用せずに、rpm パッケージをバッチインストールすることが可能です。バッチインストールするためのサンプルスクリプトを「付録２ バッチインストールのサンプルスクリプト」に示します。

第4章　パッケージアップグレードツールの使用方法

4.1　ターゲット向けパッケージをアップグレードする場合

ターゲット向けパッケージをアップグレードする場合のツールの起動例を以下に示します。

```
$ ubinux_pkg_upgrade -T -i /opt/ubq/devkit/x86/x86/target -t /opt/ubq/trpm_db
                                                ①                    ②
/tmp/new/rpms/target/bash-3.0-20120229.x86.rpm
                          ③
```

① ターゲット向けパッケージをインストールしたディレクトリを指定します。
② ターゲット向けパッケージの rpm データベースを作成したディレクトリを指定します。
③ アップグレードするパッケージのファイル名を指定します。

- ①と②には、ubinux_pkg_install 実行時に指定したディレクトリを指定してください。
- ①と②は、定義ファイルまたは環境変数に定義することが可能です。詳細については、2.2 を参照してください。
- 依存関係に問題がある場合、本ツールは、エラーメッセージを出力して、終了します。エラーメッセージが示すパッケージを③の指定に追加して、本ツールを再実行してください。
- 本ツールは、ログファイルへの出力を行いません。
- 本ツールは、インストール時のファイル競合時に（3.13 参照）インストール時に作成された ubinux_inst_info.xml を参照し判断を行います。インストール時に上書きされる側のファイルがパッケージ内にある場合にはそのファイルは既存のファイルが優先され置き換わりません。
- 本ツールで、複数のパッケージをアップグレードする際にパッケージ間で競合があった場合には警告が出ます。継続することも出来ますが本ツールでは ubinux_inst_info.xml を更新出来ない為個別にアップグレードをすることを推奨します。

4.2 ホスト向けパッケージをアップグレードする場合

ホスト向けパッケージをアップグレードする場合のツールの起動例を以下に示します。

```
$ ubinux_pkg_upgrade -H -h /opt/ubq/devkit/x86/x86/hrpmdb /tmp/new/rpms/target
                           ①                              ②
/x86-cross-gcc-4.4.6-20120229.i686.rpm
```

① ホスト向けパッケージの rpm データベースを作成したディレクトリを指定します。
② アップグレードするパッケージのファイル名を指定します。

- ①には、ubinux_pkg_install 実行時に指定したディレクトリを指定してください。
- ①は、定義ファイルまたは環境変数に定義することが可能です。詳細については、2.2 を参照してください。
- 本ツールは、ログファイルへの出力を行いません。

第5章　パッケージアンインストールツールの使用方法

5.1　ターゲット向けパッケージをアンインストールする場合

ターゲット向けパッケージをアンインストールする場合のツールの起動例を以下に示します。

```
$ ubinux_pkg_uninstall -T -i /opt/ubq/devkit/x86/x86/target -t /opt/ubq/trpmdb bash
                            ①                               ②                 ③
```

① ターゲット向けパッケージをインストールしたディレクトリを指定します。

② ターゲット向けパッケージの rpm データベースを作成したディレクトリを指定します。

③ アンインストールするパッケージ名を指定します。

- ①と②には、ubinux_pkg_install 実行時に指定したディレクトリを指定してください。
- ①と②は、定義ファイルまたは環境変数に定義することが可能です。詳細については、2.3 を参照してください。
- ③のパッケージ名にアスタリスク（”＊”）を含めることでワイルドカード指定をすることが可能です。その際には shell でワイルドカードが展開されることを抑止するためにパッケージ名をダブルクォーテーションで囲ってください。
- アンインストール対象のパッケージを前提とするパッケージが存在する場合、本ツールは、エラーメッセージを出力して終了します。
- 本ツールは、ログファイルへの出力を行いません。
- 本ツールは、ubinux_inst_info.xml を参照し、削除時に該当パッケージのデータを ubinux_inst_info.xml から削除します。ただし、削除対象が他のパッケージと共有しているファイルの場合にはそのファイルを削除しません。

5.2 ホスト向けパッケージをアンインストールする場合

ホスト向けパッケージをアンインストールする場合のツールの起動例を以下に示します。

```
$ ubinux_pkg_uninstall -H -h /opt/ubq/devkit/x86/x86/hrpmdb x86-cross-gcc
                                                          ①                ②
```

① ホスト向けパッケージの rpm データベースを作成したディレクトリを指定します。

② アンインストールするパッケージ名を指定します。

- ①には、ubinux_pkg_install 実行時に指定したディレクトリを指定してください。
- ①は、定義ファイルまたは環境変数に定義することが可能です。詳細については、2.3 を参照してください。
- 本ツールは、ログファイルへの出力を行いません。

5.3 全パッケージをアンインストールする場合

開発環境からすべてのパッケージをアンインストールする場合は、本ツールは使用せず、以下のディレクトリを rm コマンドで削除してください。

- ターゲット向けパッケージのインストールディレクトリ
- ホスト向けパッケージの rpm データベース作成ディレクトリ
- ホスト向けパッケージが展開されたディレクトリ

第6章　ルートファイルシステム作成ツールの使用方法

6.1　ルートファイルシステム作成の手順

ターゲット向けルートファイルシステム作成の手順は、以下の通りです。

① ファイルシステム作成元ディレクトリのカスタマイズ

② ルートファイルシステム作成ツールを起動

③ ルートファイルシステムに組み込むコマンドの選択

④ コマンドと直接関連しないファイルの選択

⑤ マニュアルとライセンステキストの組み込みの選択

⑥ 組み込みファイルの確認

⑦ ルートファイルシステムの作成

⑧ ターゲット上で共有ライブラリキャッシュの更新

6.2　ファイルシステム作成元ディレクトリのカスタマイズ

ターゲット向けパッケージのインストールディレクトリや、本ツールで作成されたルートファイルシステムなど、ファイルシステム作成元にするディレクトリに対して、以下のカスタマイズを実施してください。

- ネットワーク定義など各種システムコンフィグレーションファイルの変更
- 利用者が作成したアプリケーションのコピー
- カーネルのコピー
- その他

6.3 ルートファイルシステム作成ツールの起動

ルートファイルシステム作成ツールの起動例を以下に示します。

```
$ ubinux_make_rootfs -f /opt/ubq/devkit/x86/x86/target -t /tmp/build-dir
                                                ①                  ②
 -r /opt/ubq/devkit/x86/x86/target/opt/ubq/trpmdb -k /tmp/x86-kernel
                                ③                   ④
```

① ルートファイルシステム作成元のディレクトリを指定します。
② ルートファイルシステム作成先のディレクトリを指定します。
③ ターゲット向けパッケージの rpm データベースを作成したディレクトリを指定します。
④ ターゲットのカーネルをコンパイルしたカーネルソースのディレクトリを指定します。

- ②に指定されたディレクトリは本ツールによって自動作成されます。②には、既存のディレクトリを指定しないでください。
- ③には、パッケージインストーラ（ubinux_pkg_install）の「-i オプションに指定したディレクトリ ＋-t オプションに指定したディレクトリ」を指定してください。
- 上記オプションは、定義ファイルまたは環境変数に定義することが可能です。詳細については、2.4 を参照してください。
- デバイスファイルは、④に指定されたディレクトリ配下のファイル.config と ubinux_make_rootfs.dev.conf の情報から自動作成されます。
  ubinux_make_rootfs.dev.conf は、事前に作成しておく必要があります。記述方法については、付録３ ubinux_make_rootfs.dev.conf の例を参考にしてください。

ルートファイルシステム作成ツールを起動すると、以下の確認メッセージが出力されます。メッセージの内容について問題がなければ、"y" を入力してください。

```
$ ubinux_make_rootfs -f /opt/ubq/devkit/x86/x86/target -t /tmp/build-dir -r /opt
/ubq/devkit/x86/x86/target/opt/ubq/trpmdb -k /tmp/x86-kernel
*******************************************************************************
 From directory          : /opt/ubq/devkit/x86/x86/target
 To directory            : /tmp/build-dir
 Target RPMDB Directory  : /opt/ubq/devkit/x86/x86/target/opt/ubq/trpmdb
 Kernel Source Directory : /tmp/x86-kernel
*******************************************************************************
Is it OK ? (Yes/No): y
```

オプションの指定に誤りがないかを確認

```
To-directory does not exist.
Create it ? (Yes/No): y
```

ルートファイルシステム作成先ディレクトリの作成を確認

```
The last File Select Information exists in From-directory.
Do you use this Info ? (Yes/No): y

Information on From-directory is being collected.
Wait a few minutes.....
```

前回作成時の選択情報を使用するか否かの問い合わせ（※）

- 本ツールは、ルートファイルシステム作成時に、組み込み対象ファイルの選択状態を from-dir/../ubinux_make_rootfs.select.conf に保存します。次回、同じ作成元ディレクトを使用して、ルートファイルシステムを作成する場合は、前回の選択状態をキャラクタ・ユーザ・インタフェースに再現することができます。上記(※)のメッセージは、前回の選択状態を再現するか否かを問い合わせるものです。前回の選択状態を再現する場合は、"y"を入力してください。
- 最後のメッセージの表示後、本ツールは、作成元ディレクトリ配下の全ファイルに対して情報の取得を行います。この処理には、数分間要することがあります。

6.4 キャラクタ・ユーザ・インタフェース

パッケージインストーラを起動すると、キャラクタ・ユーザ・インタフェース（CUI）画面が表示されます。CUI 画面の基本操作は、パッケージインストーラと同じです。基本操作については、3.3 を参照してください。

6.5 ルートファイルシステムに組み込むコマンドの選択

作成元ディレクトリ配下に存在するコマンドの一覧が表示されます。ターゲットの目的に応じて、ルートファイルシステムに組み込むコマンドを選択してください。

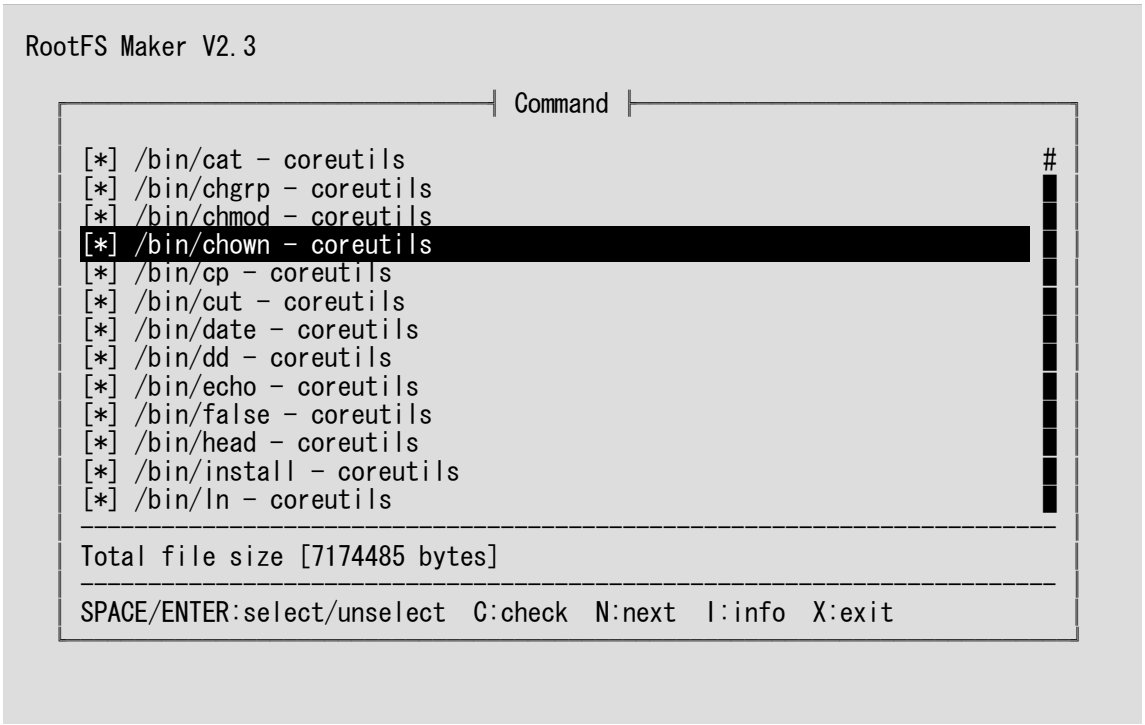


図 18 組み込みコマンドの選択画面

選択項目には、コマンドのパス名と所属するパッケージ名が表示されます。

コマンドとは、ELF 形式の実行ファイルを指します。ただし、a.out は表示されません。

パッケージ一覧の下部 “Total file size” には、組み込み対象ファイルのサイズの合計値が表示されます。

このサイズは、"C" または "N" を押下すると更新されます。

なお、作成元ディレクトリ配下にコマンドが存在しない場合、この画面は表示されません。

チェックボックスは、以下を意味します。

デフォルト（前回作成時の選択状態を反映させなかった場合）では、すべてのコマンドが組み込み対象に選択されます。

表示	意味
[*]	該当するコマンドは組み込み対象です。
[ ]	該当するコマンドは組み込み対象ではありません。

ホットキーの動作は、以下の通りです。

ホットキー	動作
SPACE or ENTER	選択／非選択を切り替えます。
C	以下のファイルを組み込み対象とし、"Total file size"を更新します。 組み込み対象ファイルの選択論理については、6.10 を参照してください。 ・選択されたコマンド ・選択されたコマンドが使用する共有ライブラリ ・選択されたコマンドと共有ライブラリが所属するパッケージ内の 非 ELF ファイル
N	コマンドに直接関連しないファイルの選択に進みます。
I	コマンドの詳細情報を表示します。
X	ツールを終了します。終了時は、現在の選択状態を破棄します。

6.6 コマンドと直接関連しないファイルの選択

コマンドとの関連性を機械的に検出できないファイルの一覧が表示されます。これらの中から組み込み対象にするファイルを選択してください。

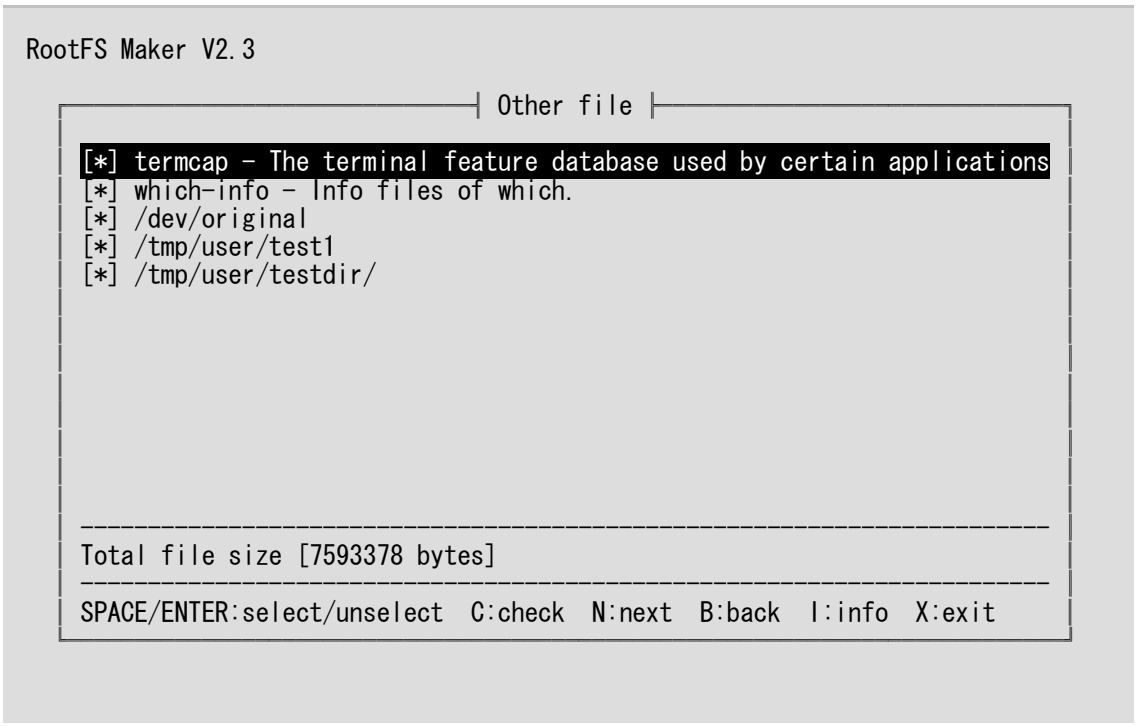


図 19 コマンドと直接関連しないファイルの選択画面

選択項目には、以下が表示されます。※

- コマンドおよび共有ライブラリを持たないパッケージの名前
- 利用者が追加したファイル（パッケージに所属しないファイル）のうち、コマンドおよび共有ライブラリ以外のファイルのパス名

注意）

- devlop パッケージ、マニュアルパッケージおよびライセンステキストパッケージは表示されません。
- 末尾が "/" の項目は、ディレクトリです。ディレクトリについては、配下にファイルが存在しないもののみ表示されます。

なお、作成元ディレクトリ配下に該当するファイルが存在しない場合、この画面は表示されません。

※ X-Window やカーネルモジュール等はファイル単位での選択が行えないのでルートファイルにコピーする必要の無いものはコピー元ディレクトリから削除する必要があります。

チェックボックスは、以下を意味します。

デフォルト（前回作成時の選択状態を反映させなかった場合）では、すべてのパッケージとファイルが組み込み対象に選択されます。

表示	意味
[*]	該当するパッケージ／ファイルは組み込み対象です。
[ ]	該当するパッケージ／ファイルは組み込み対象ではありません。

ホットキーの動作は、以下の通りです。

ホットキー	動作
SPACE or ENTER	選択／非選択を切り替えます。
C	選択されたパッケージに所属するファイルと選択されたファイルを組み込み対象に加え、"Total file size"を更新します。
N	マニュアルとライセンステキストの組み込みの選択に進みます。
B	前の画面に戻ります。
I	パッケージまたはファイルの詳細情報を表示します。
X	ツールを終了します。終了時は、現在の選択状態を破棄します。

パッケージに対して "I" を押下すると、パッケージを選択することにより組み込み対象になるファイルの一覧が表示されます。

6.7 マニュアルとライセンステキストの組み込みの選択

マニュアルとライセンステキストをルートファイルシステムに組み込むか否かを選択する項目が表示されます。ターゲットの目的に応じて、組み込むか否かを選択してください。

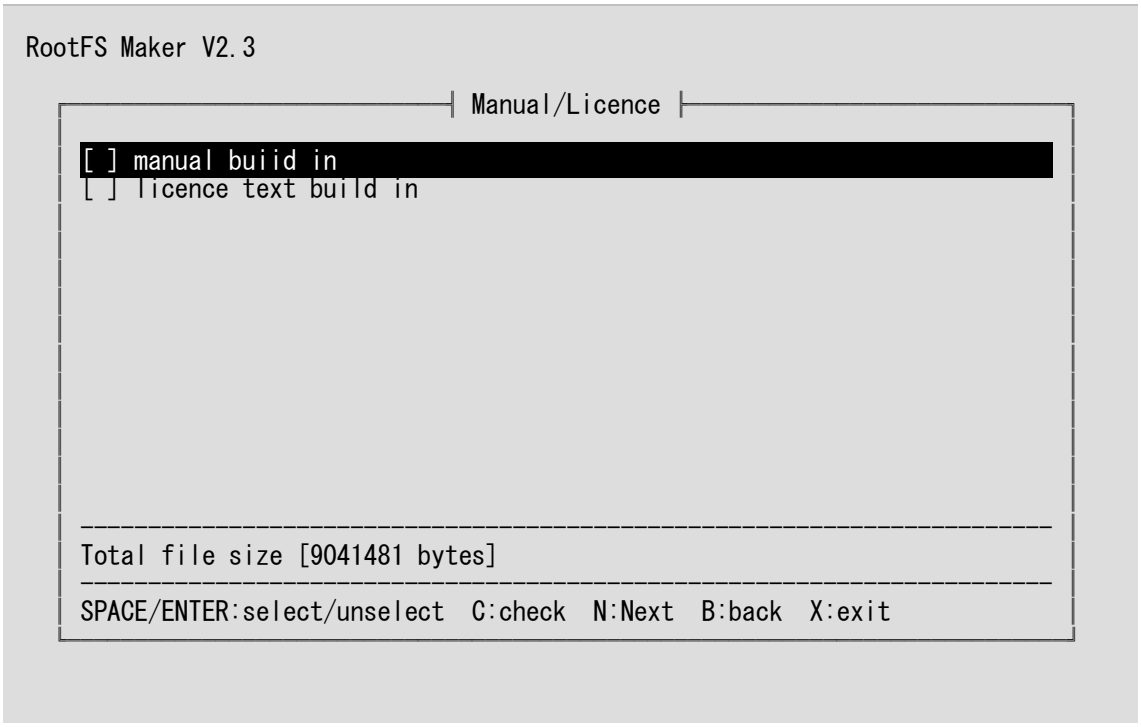


図 20 マニュアルとライセンステキストの組み込みの選択画面

“manual build in” を選択した場合、マニュアルが組み込み対象になります。
“licence text build in” を選択した場合、ライセンステキストが組み込み対象になります。

組み込み対象に選択されるマニュアル／ライセンステキストは、前節までの作業で組み込み対象に選択されたファイルが所属するパッケージのマニュアル／ライセンステキストです。

作成元ディレクトリ配下に組み込み対象候補になるマニュアル／ライセンステキストが存在しない場合、存在しない方の選択項目は表示されません。マニュアル／ライセンステキストが共に存在しない場合、この画面は表示されません。

チェックボックスは、以下を意味します。

デフォルト（前回作成時の選択状態を反映させなかった場合）では、マニュアル／ライセンステキスト共に組み込み対象に選択されません。

表示	意味
[*]	該当する項目のファイルは組み込み対象です。
[ ]	該当する項目のファイルは組み込み対象ではありません。

ホットキーの動作は、以下の通りです。

ホットキー	動作
SPACE or ENTER	選択／非選択を切り替えます。
C	マニュアル／ライセンステキストを組み込み対象に加え、"Total file size" を更新します。
N	組み込み対象ファイルの確認に進みます。
B	前の画面に戻ります。
X	ツールを終了します。終了時は、現在の選択状態を破棄します。

6.8 組み込みファイルの確認

組み込み対象ファイルの一覧が表示されます。組み込み対象に選択されたファイルに問題がないか確認してください。

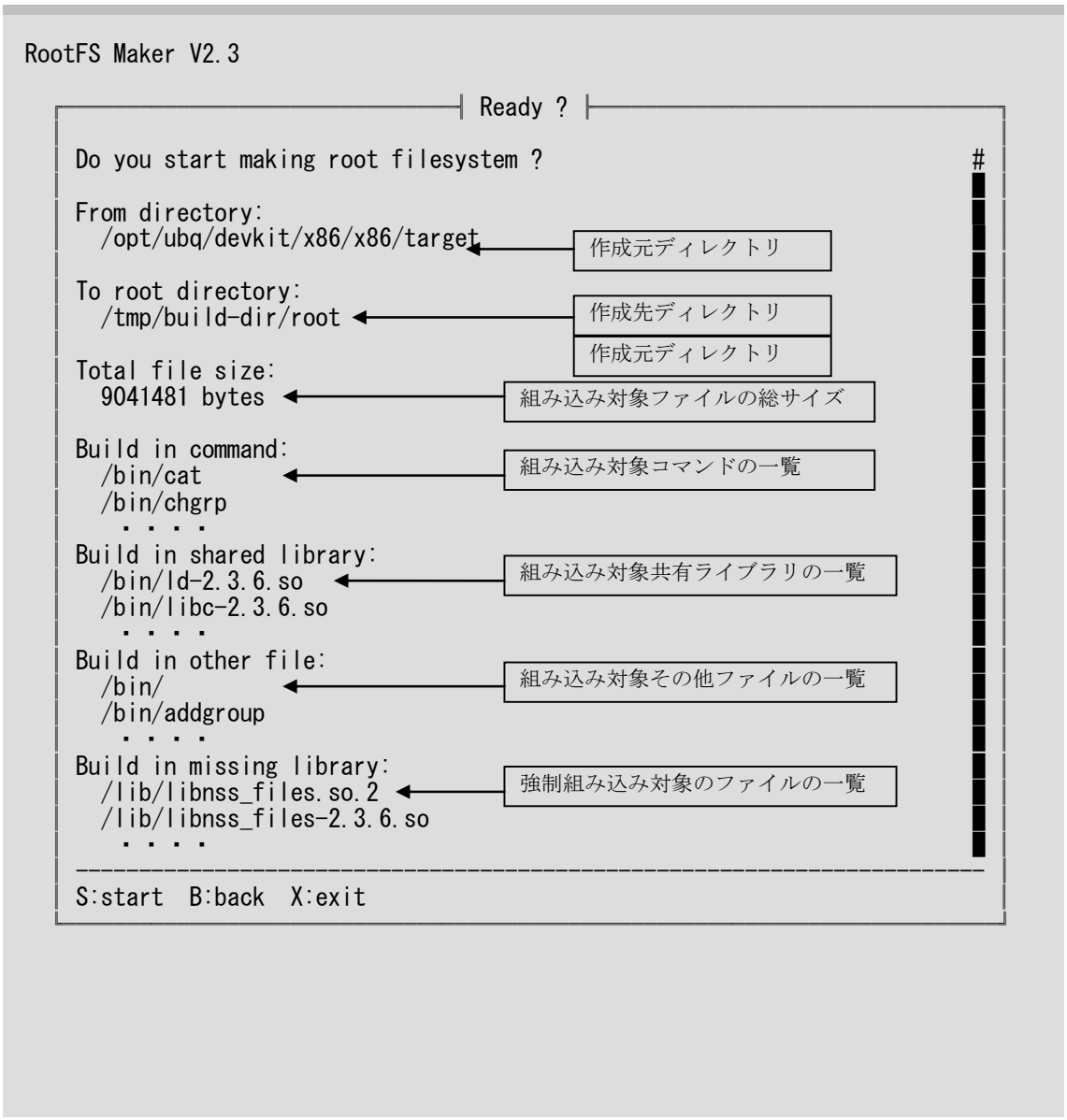


図 21 組み込み対象ファイルの確認画面

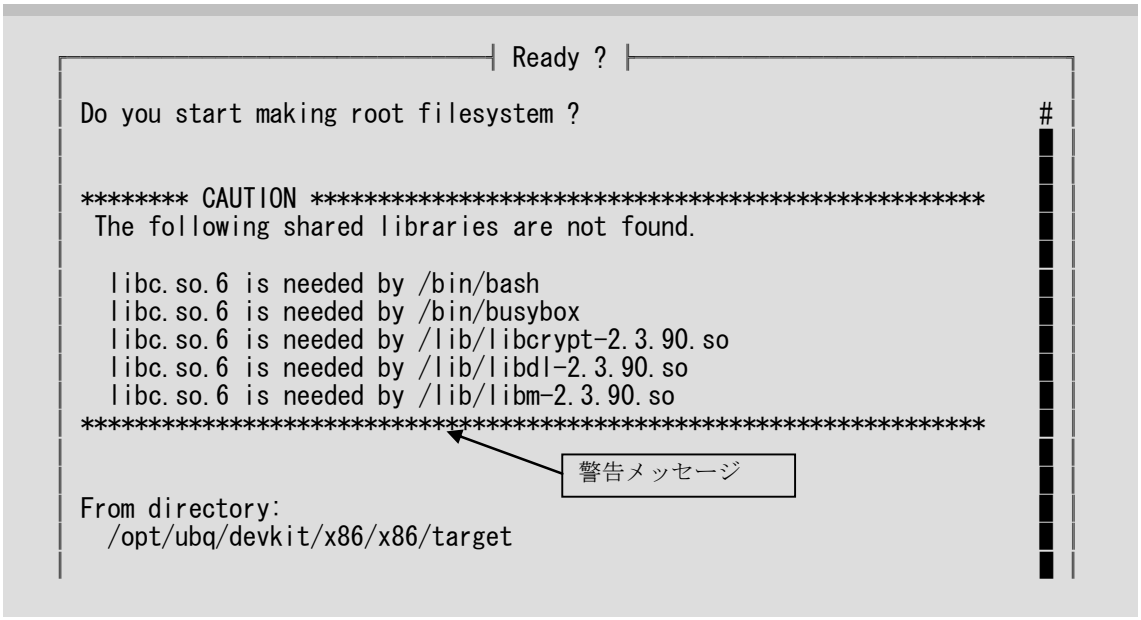
選択されたパッケージに問題がない場合は、"S" を押下して、インストールを開始してください。

この画面で "S" を選択すると、現在の選択状態が from-dir/../ubinux_make_rootfs.select.conf に保存されます。

上記組み込み対象ファイルの一覧には、本ツールによって自動作成されるデバイスファイルも含まれます。デバイスファイルは、本ツール起動時に指定したカーネルソースディレクトリ配下のファイル.config と ubinux_make_rootfs.dev.conf の情報を元に自動作成されます。ubinux_make_rootfs.dev.conf をカスタマイズすることにより、自動作成するデバイスファイルを追加／変更することができます。詳しくは、6.12 を参照してください。

なお、作成元ディレクトリから手作業で共有ライブラリを削除したなどの理由により、作成元ディレクトリ配下にコマンドが使用する共有ライブラリが存在しない場合は、以下のような警告メッセージが表示されます。

ターゲット向けパッケージのインストールディレクトリなどから、警告メッセージで示される共有ライブラリを作成元ディレクトリに追加して、本ツールを再起動してください。



6.9 ルートファイルシステムの作成

ルートファイルシステムは、to-dir/root 配下に作成されます。
作成先ディレクトリ to-dir 直下ではありません。ご注意ください。

作成中は以下のメッセージが出力されます。このメッセージは、ログファイル to-dir/ubinux_make_rootfs.YYYY MMDD-hhmmss.log（YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）にも出力されます。

- 作成を開始する旨のメッセージ
- 作成を終了する旨のメッセージ
- エラーメッセージ（エラー発生時のみ）

また、各選択画面で選択した内容が、ログファイル to-dir/ubinux_make_rootfs.select.YYYY MMDD-hhmmss.log に、作成したファイルシステム内のファイル一覧が、ログファイル to-dir/ubinux_make_rootfs.file.YYYY MMDD-hhmmss.log に出力されます。
この 2 つのログファイルの内容については、6.13 を参照してください。

以下は正常終了時のメッセージです。

```
Making the root filesystem start.
6891 blocks
Making the root filesystem succeeded.
RootFS Maker finish.
```

ルートファイルシステムの作成が終了しましたら、ファイルシステムイメージを作成する前に、以下の作業を実施してください。

① NFS ブートにより、シングルユーザモードにてターゲット上でシステムを起動する。
② ターゲット上で/sbin/ldconfig コマンドを実行する。

この作業により、以下が行われます。

- ライブラリキャッシュの更新
- 各種コマンド実行時に必要であるがパッケージには含まれていない共有ライブラリへのシンボリックリンクの作成

ldconfig の使用方法については、man マニュアルを参照してください。

6.10 組み込み対象ファイルの選択論理について

以下のファイルがルートファイルシステムに組み込まれます。

① 利用者によって組み込み対象に選択されたコマンド（※1）
② 上記コマンドが動的リンクする共有ライブラリ（※2）
③ 上記、共有ライブラリが動的リンクする共有ライブラリ（※2）
④ ②と③の共有ライブラリへのシンボリックリンク（※3）
⑤ ①～③のファイルが所属するパッケージ内の非 ELF 形式ファイル（※4）（※5）
⑥ 利用者によって組み込み対象に選択されたパッケージのファイル（※5）（※6）
⑦ ①～⑥のファイルが所属するパッケージに関連するマニュアル／ライセンステキストパッケージのファイル（※7）
⑧ 利用者によって組み込み対象に選択されたファイル（※8）
⑨ 基本必須パッケージである filesystem と setup パッケージに所属するファイル

（※1） a.out は選択項目に表示されず、組み込まれません。
（※2） コマンドから dlopen()によってリンクされる共有ライブラリは組み込まれません（6.11 参照）。
（※3） 名前にバージョン番号を持つシンボリックリンクのみ組み込まれます。
名前にバージョン番号を持たないシンボリックリンクは組み込まれません。
（※4） １つのコマンド／共有ライブラリが複数のパッケージに所属している場合、すべてのパッケージが対象になります。ただし、develop パッケージについては、他に所属パッケージが存在しない場合のみ対象になります。
（※5） シンボリックリンクについては、リンク先ファイルが組み込み対象になっているもの、または、作成元ディレクトリにリンク先ファイルが存在しないリンクファイルのみ組み込まれます。作成元ディレクトリにリンク先ファイルが存在し、かつ、リンク先ファイルが組み込み対象ではないリンクファイルは組み込まれません。
（※6） コマンド／共有ライブラリを持たないパッケージのみ利用者から直接組み込み対象の有無を選択されます。
（※7） 利用者がマニュアル／ライセンステキストを組み込むよう選択した場合のみ
（※8） パッケージに所属しない非 ELF 形式のファイルのみ利用者から直接組み込み対象の有無を選択されます。

6.11 共有ライブラリの組み込みに関する注意事項

コマンドから dlopen()によってリンクされる共有ライブラリは、コマンドから使用されることを機械的に検出することができません。このため、コマンドから dlopen()によってリンクされる共有ライブラリは、組み込まれません。(代表的な既知の問題に関しては付録１ 共有ライブラリ不足時の異常動作とその対策についてを参照してください。)

ターゲット上での動作確認の結果、共有ライブラリが不足していることが判明した場合は、以下の手順でルートファイルシステムを再作成してください。

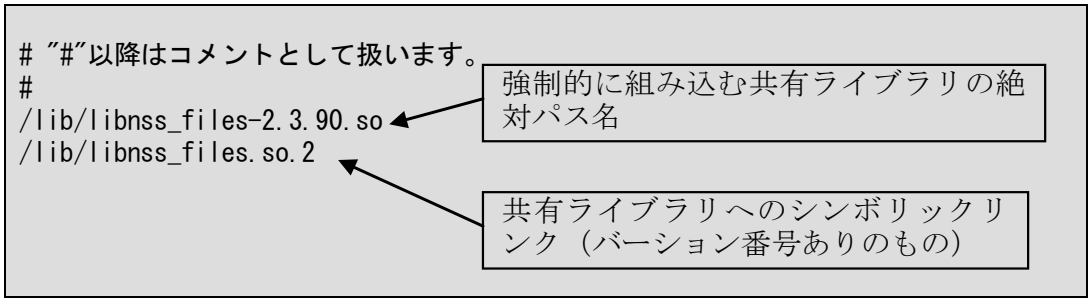
① 以下のパス名のテキストファイルを作成し、不足する共有ライブラリのパス名を記述する。

[作成元ディレクトリ]/../ubinux_make_rootfs.mandatory.conf

② 本ツールを使用して、再度、ルートファイルシステムを作成する。

- 上記ファイルに記述したファイルは、強制的にルートファイルシステムに組み込まれます。
- パス名は、作成元ディレクトリをルートディレクトリとみなした絶対パスで指定してください。
- 共有ライブラリに対してバージョン番号付きのシンボリックリンクが存在する場合は、シンボリックリンクのパス名も記述してください。
- 上記ファイルに記述したファイルは、本ツールの各選択画面において、選択項目に表示されなくなります。

上記ファイルの記述例を以下に示します。



なお、共有ライブラリが不足することより発生する異常動作（付録１ 共有ライブラリ不足時の異常動作とその対策について）に対応する共有ライブラリに関しては以下のファイルにて関連付けが行われ問題を回避出来るようになっているので作成元ディレクトリを基準とした以下の場所に配置してください。

・サンプル定義ファイルから次のファイルをコピーする

/opt/ubinux_pkg_tools/share/tools/sample-config/ubinux_make_rootfs.mandatory.xml

↓

[作成元ディレクトリ]/../ubinux_make_rootfs.mandatory.xml

6.12 デバイスファイルの自動作成について

デバイスファイルは、指定されたカーネルソースディレクトリ配下のファイル .config と ubinux_make_rootfs.dev.conf の情報を元に自動生成されます。ubinux_make_rootfs.dev.conf には、カーネルコンフィグレーションに対応するデバイスファイルが定義されています。

自動作成されるデバイスファイルを追加・変更する場合は、ubinux_make_rootfs.dev.conf を変更してください。

ubinux_make_rootfs.dev.conf の記述形式を以下に示します。

```
[キャラクタデバイスファイルの場合]
 ″カーネルコンフィグ″ ″名前″ c ″メジャー番号″ ″マイナー番号″ ″所有者″ ″グループ″ ″モード″

[ブロックデバイスファイルの場合]
 ″カーネルコンフィグ″ ″名前″ b ″メジャー番号″ ″マイナー番号″ ″所有者″ ″グループ″ ″モード″

[ディレクトリの場合]
 ″カーネルコンフィグ″ ″名前″ d ″所有者″ ″グループ″ ″モード″

[シンボリックリンクの場合]
 ″カーネルコンフィグ″ ″名前″ s ″リンク先への相対パス″
```

上記の "カーネルコンフィグ" に記述するコンフィグレーションが、.config 内に定義されている場合に、対応するファイルがディレクトリ to-dir/dev 配下に作成されます。

コンフィグレーションに依存しないで、常に作成する必要があるファイルについては、"カーネルコンフィグ" に対し、"-" (ハイフン) を指定してください。

以下に記述例を示します。

```
#-----------------------------------------------------------------
# Charactor device
#-----------------------------------------------------------------
# Memory devices (Major=1)
- mem c 1 1 root kmem 0640
- kmem c 1 2 root kmem 0640
- null c 1 3 root root 0666
- zero c 1 5 root root 0666

# Serial 8250/16550 (Major=4)
CONFIG_SERIAL_8250 ttyS0 c 4 64 root uucp 0660
CONFIG_SERIAL_8250 ttyS1 c 4 65 root uucp 0660

 （つづく）
```

（つづき）

```
#-----------------------------------------------------------------------
# Block device
#-----------------------------------------------------------------------
# RAM disk (Major=1)
CONFIG_BLK_DEV_RAM ram0 b 1 0 root disk 0640
CONFIG_BLK_DEV_RAM ram1 b 1 1 root disk 0640
CONFIG_BLK_DEV_RAM ram2 b 1 2 root disk 0640
CONFIG_BLK_DEV_RAM ram3 b 1 3 root disk 0640

#-----------------------------------------------------------------------
# Directory
#-----------------------------------------------------------------------
- shm d root root 01777
CONFIG_UNIX98_PTYS pts d root root 0755

#-----------------------------------------------------------------------
# Symbolic link
#-----------------------------------------------------------------------
CONFIG_PROC_FS fd s ../proc/self/fd
CONFIG_PROC_FS stdin s ../proc/self/fd/0
CONFIG_PROC_FS stdout s ../proc/self/fd/1
CONFIG_PROC_FS stderr s ../proc/self/fd/2
CONFIG_PROC_FS core s ../proc/kcore

※サブディレクトリ配下に作成するケース
CONFIG_IEEE1394_DV1394 ieee1394 d root root 755
CONFIG_IEEE1394_DV1394 ieee1394/dv d root root 755
CONFIG_IEEE1394_DV1394 ieee1394/host0 d root root 755
CONFIG_IEEE1394_DV1394 ieee1394/host0/NTSC d root root 755
CONFIG_IEEE1394_DV1394 ieee1394/host0/NTSC/in c 171 32 root root 600
```

- ファイルのモードは、８進数で指定してください。
- １つのコンフィグレーションが定義された場合に、複数のファイルを作成する場合には、その個数分記述してください。
- "#" 以降はコメントとして扱われます。

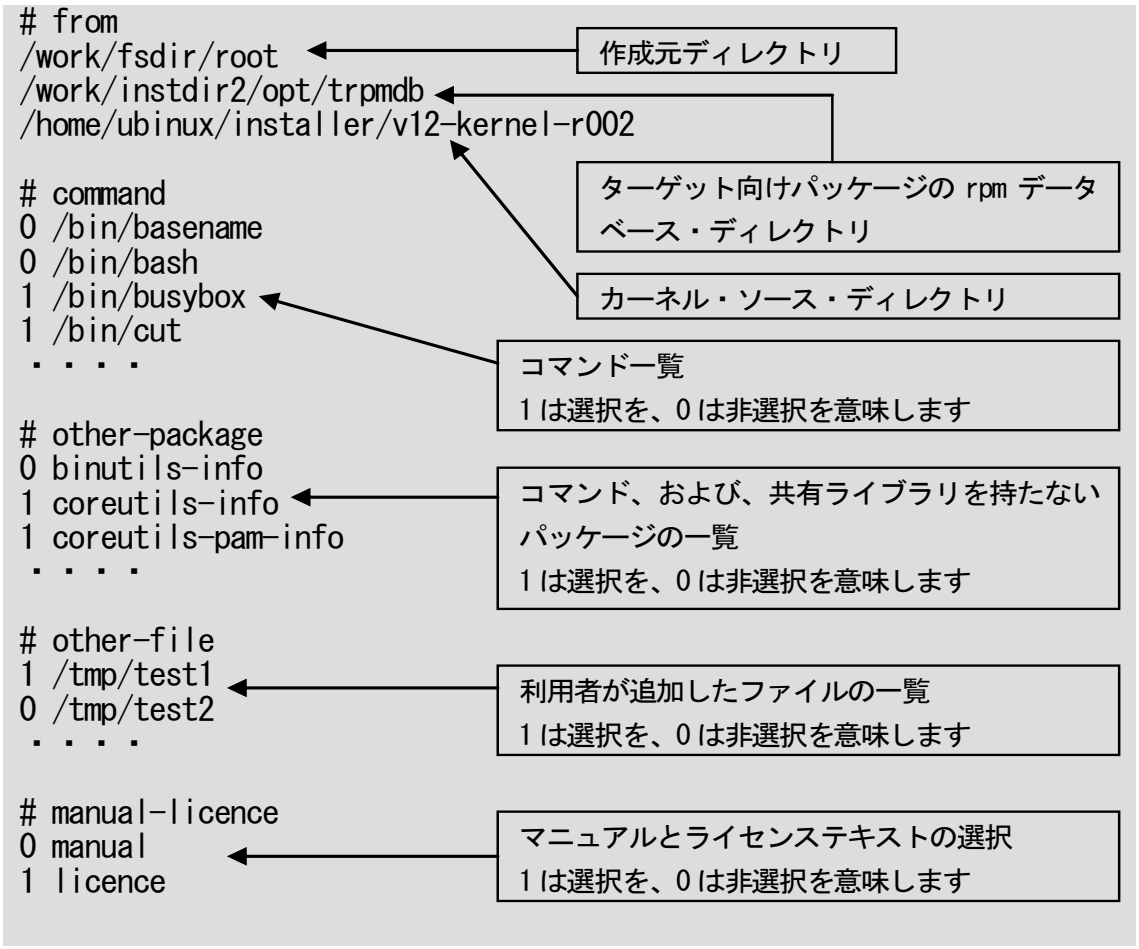
6.13 選択内容ログとファイル一覧ログについて

ルートファイルシステムの作成時には、各選択画面で選択した内容と、作成したルートファイルシステム内のファイル一覧がログファイルに出力されます。それぞれの出力内容は以下の通りです。

- 選択内容ログ

  ファイル名：to-dir/ubinux_make_rootfs.select.YYYY MMDD-hhmmss.log

  （YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）

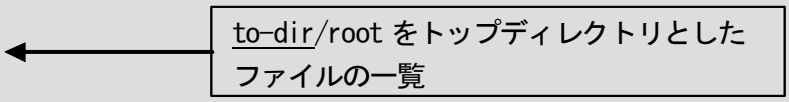


- ファイル一覧ログ

  ファイル名：to-dir/ubinux_make_rootfs.file.YYYY MMDD-hhmmss.log

  （YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒）

```
/
/bin
/bin/[
/bin/addgroup
/bin/adduser
/bin/arch
/bin/ash
/bin/awk
/bin/basename
/bin/bash
/bin/bashbug
/bin/busybox
/bin/cat
/bin/chgrp
/bin/chmod
/bin/chown
/bin/cp
 ・・・・
```



第7章　ファイルシステムイメージ作成ツールの使用方法

7.1　ファイルシステムイメージ作成の手順

ターゲット向けファイルシステムイメージ作成の手順は、以下の通りです。

① 各ファイルシステム向けのコマンドの導入
② ファイルシステムイメージ作成ツールを起動
③ 作成するイメージの種別を選択
④ イメージ作成時のパラメータを設定
⑤ イメージの作成
⑥ ターゲット上で起動を確認

7.2　各ファイルシステム向けのコマンドの導入

本ツールでは内部で各ファイルシステム用のコマンドを使用します。以下のコマンドが導入されていない場合には対応するパッケージを導入してください。

ファイルシステム	コマンド	導入方法
JFFS2	/sbin/mkfs.jffs2	各ディストリビューション向けの mtd-utils パッケージを導入してください。
SquashFS	/sbin/mksquashfs	各ディストリビューション向けの squashfs-tools パッケージを導入してください。
UBIFS	/sbin/mkfs.ubifs	ubinux の *ARCH*-cross-mtd-utils パッケージを導入しコマンドに /sbin/ へのリンクを作成してください。(下記参照①)
LOGFS	/sbin/mklogfs	ubinux の *ARCH*-cross-logfsprogs パッケージを導入しコマンドに /sbin/ へのリンクを作成してください。(下記参照②)

```
# ln –s /opt/ubq/devkit/ARCH/PROC/usr/bin/mkfs.ubifs /sbin/  ①
# ln –s /opt/ubq/devkit/ARCH/PROC/usr/bin/mklogfs /sbin/  ②
```

7.3 ファイルシステムイメージ作成ツールの起動

ファイルシステムイメージ作成ツールの起動例を以下に示します。

```
$ ubinux_make_fsimage -f /tmp/build-dir/root -t /tmp/image-dir
                                  ①              ②
```

① ルートファイルシステム作成ツールで作成したルートファイルシステムのディレクトリを指定します。

注意：ルートファイルシステム作成ツールの -t オプションに指定したディレクトリではなく、その配下の root サブディレクトリを指定してください。

② ファイルシステムイメージ作成先のディレクトリを指定します。

ファイルシステムイメージ作成ツールを起動すると、以下の確認メッセージが出力されます。メッセージの内容について問題がなければ、"y" を入力してください。

```
$ ubinux_make_rootfs -f /tmp/build-dir/root -t /tmp/image-dir
*******************************************************************************
 From directory : /tmp/build-dir/root
 To directory   : /tmp/image-dir
 File name      : not specified
*******************************************************************************
Is it OK ? (Yes/No): y ◄──────── オプションの指定に誤りがないかを確認
```

7.4 キャラクタ・ユーザ・インタフェース

ファイルシステムイメージ作成ツールを起動すると、キャラクタ・ユーザ・インタフェース（CUI）画面が表示されます。

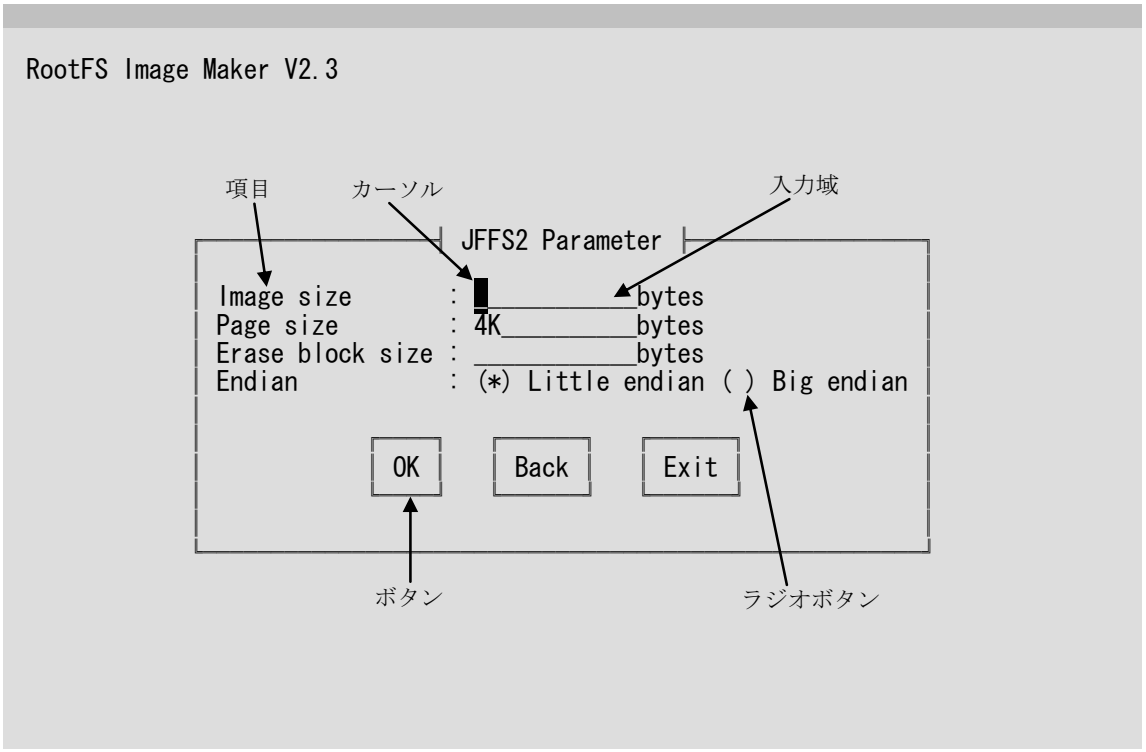


図 22 CUI 画面

CUI 画面の基本操作は、以下の通りです。

① 選択項目に対してカーソルを合わせます。
[↑] [↓] [TAB]キーによりカーソルを上下させます。

② 入力を行います。
アンダーバー付きの項目にカーソルを合わせ、文字を入力します。

③ 択一選択を行います。
ラジオボタン（括弧付きの項目）の項目にカーソルを合わせ、[SPACE]キーにより選択を切り替えます。

④ 動作を行います。
下部に表示されるボタンにカーソルを合わせ、[ENTER]キーにより、ボタンが示す動作を実行します。

7.5 イメージ種別の選択

イメージ種別の選択画面が表示されます。

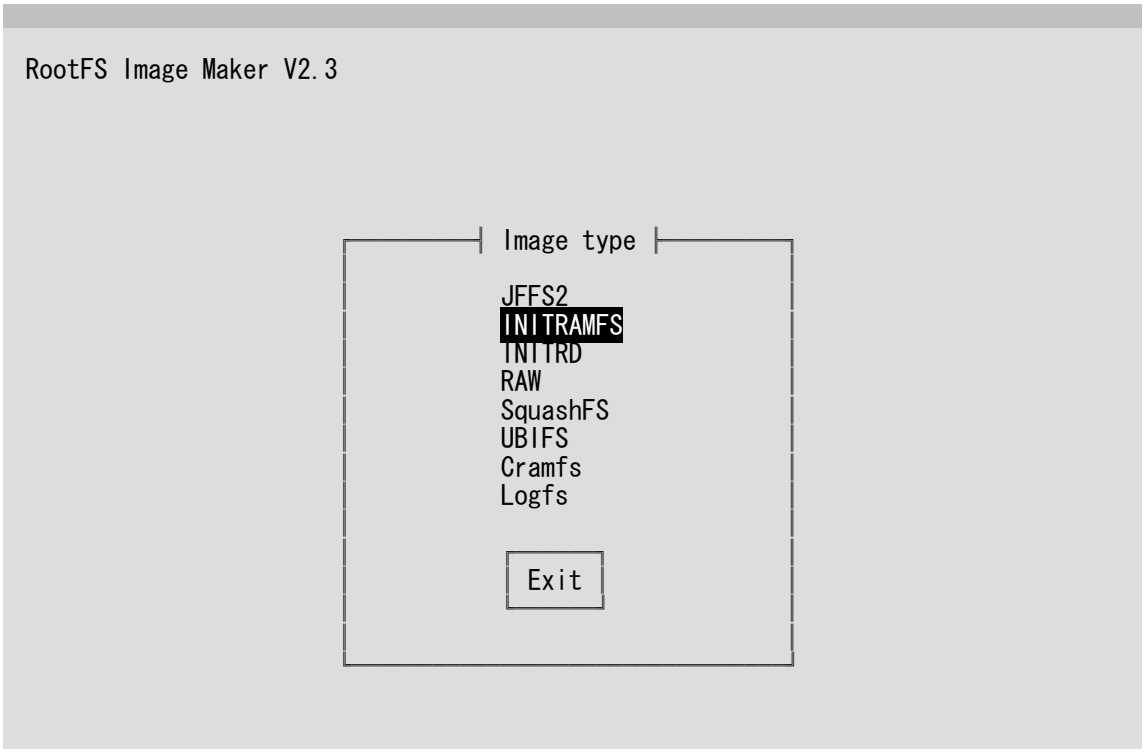


図 23 イメージ種別選択

作成するイメージの種別を選択してください。

7.6 JFFS2 イメージ

7.6.1 JFFS2 パラメータの設定

イメージの作成に必要な各種パラメータを問い合わせる画面が表示されます。

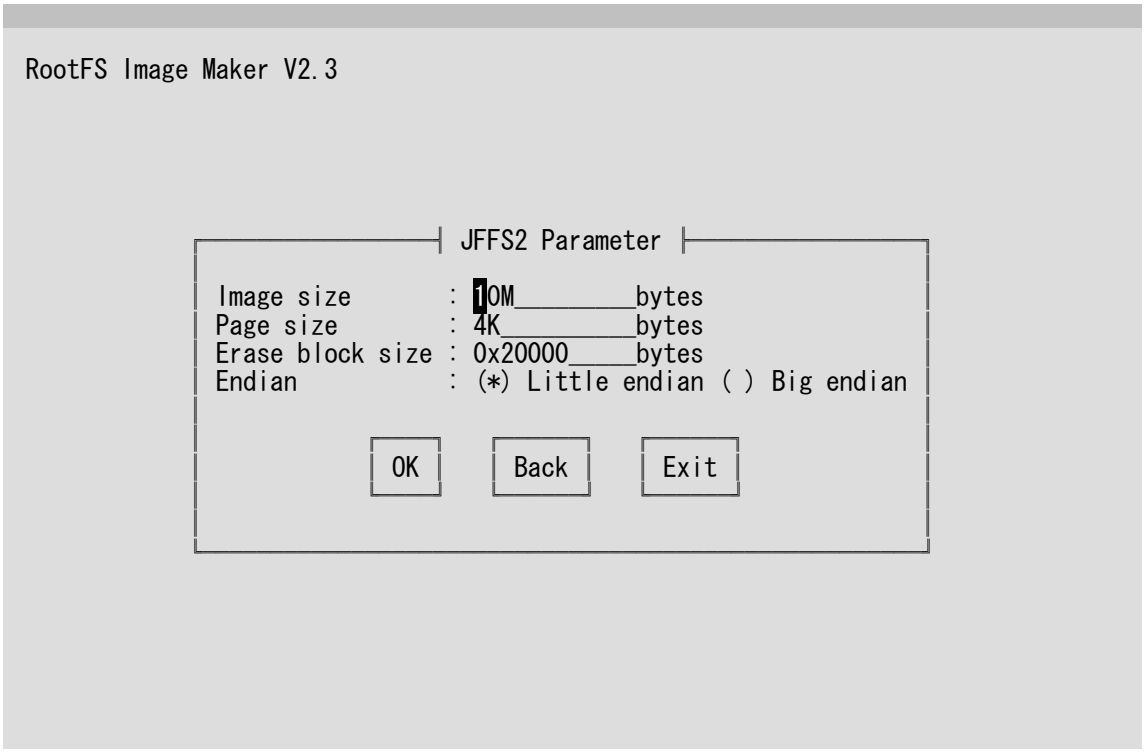


図 24 JFFS2 イメージパラメータの入力画面

以下のパラメータを入力してください。

パラメータの入力後は、"OK"ボタンを押してください。設定内容の確認画面が表示されます。

入力項目	入力値
Image size	希望するイメージファイルのサイズを入力してください。
Page size	ファイルシステムのページサイズを入力してください。 デフォルトは、4KB です。
Erase block size	flash メモリの Erase block サイズを入力してください。
Endian	ファイルシステムのエンディアンを選択してください。 デフォルトは、リトルエンディアンです。

※数値入力データは、最後の文字が"K"であればキロバイト、"M"であればメガバイトとして扱われます。先頭に "0x" を付けると 16 進数値として扱われます。

7.6.2 JFFS2 パラメータの確認

パラメータに設定された内容が表示されます。

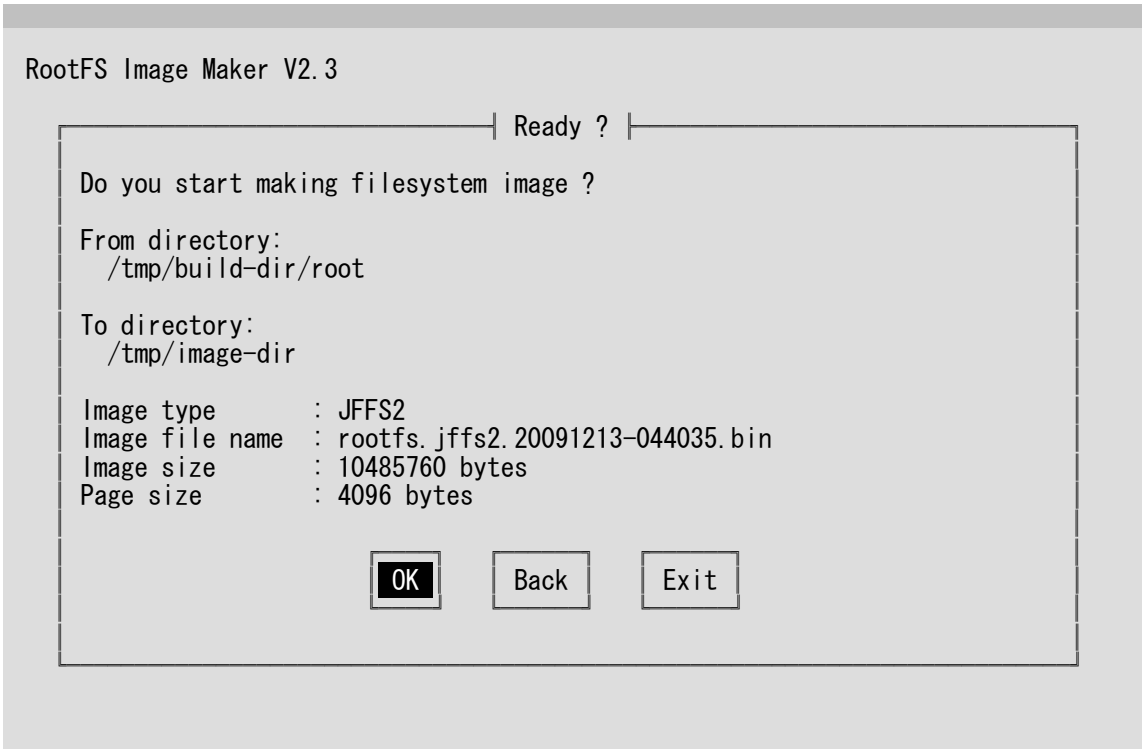


図 25 JFFS2 イメージパラメータの確認画面

表示内容に問題がなければ、"OK"ボタンを押してください。イメージの作成が開始されます。

本ツール起動時に、-o オプションでイメージファイル名が指定されていない場合、イメージファイル名は、以下になります。

［イメージファイル名］

"rootfs.jffs2.YYYYMMDD-hhmmss.bin"

YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒

7.6.3 JFFS2 イメージの作成

作成中は以下のメッセージが出力されます。

このメッセージは、ログファイル to-dir/イメージファイル名.log にも出力されます。

- 作成を開始する旨のメッセージ
- 作成を終了する旨のメッセージ
- エラーメッセージ（エラー発生時のみ）

以下は正常終了時のメッセージです。

```
Making the JFFS2 image start.
Making the JFFS2 image succeeded.
RootFS Image Maker finish.
```

以下の警告メッセージが出力された場合は、作成されたファイルシステムイメージが、指定されたサイズを超えています。

［警告メッセージ］

"WARNING: The image file size is larger than the specified size. !!"

7.7 INITRAMFS イメージ

7.7.1 INITRAMFS パラメータの確認

INITRAMFS イメージ作成時に入力するパラメータはありません。

以下の確認画面が表示されます。

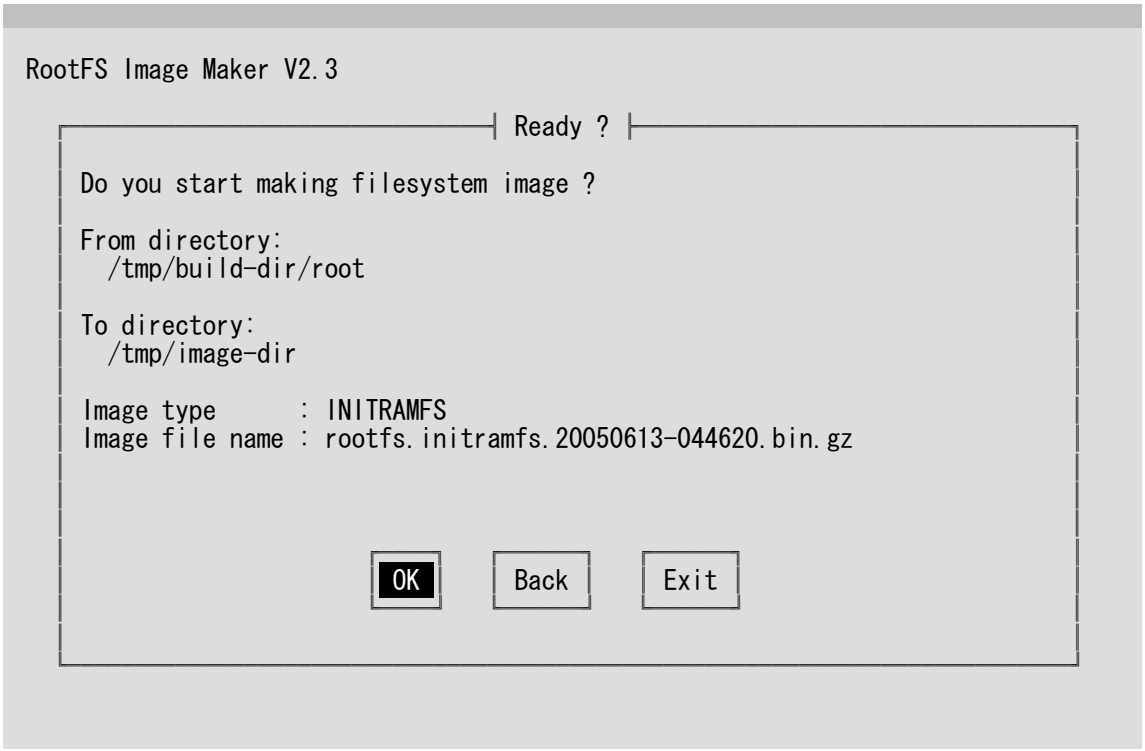


図 26 INITRAMFS イメージパラメータの確認画面

表示内容に問題がなければ、"OK"ボタンを押してください。イメージの作成が開始されます。

本ツール起動時に、-o オプションでイメージファイル名が指定されていない場合、イメージファイル名は、以下になります。

［イメージファイル名］

"rootfs.initramfs.YYYYMMDD-hhmmss.bin.gz"

YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒

7.7.2 INITRAMFS イメージの作成

作成中は以下のメッセージが出力されます。

このメッセージは、ログファイル to-dir/イメージファイル名.log にも出力されます。

- 作成を開始する旨のメッセージ
- 作成を終了する旨のメッセージ
- エラーメッセージ（エラー発生時のみ）

以下は正常終了時のメッセージです。

```
Making the INITRAMFS image start.
27235 blocks
Making the INITRAMFS image succeeded.
RootFS Image Maker finish.
```

エラーが発生した場合にはディスク容量等をご確認下さい。

7.8 INITRD イメージ

7.8.1 INITRD パラメータの設定

イメージの作成に必要な各種パラメータを問い合わせる画面が表示されます。

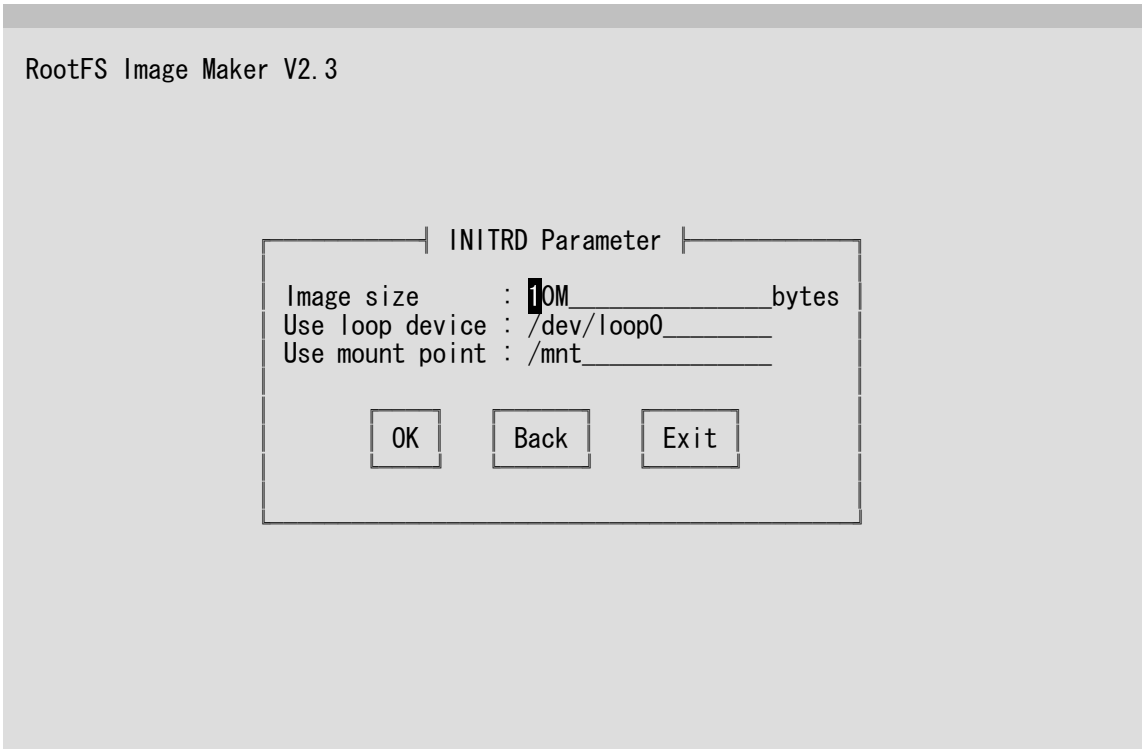


図 27 INITRD イメージパラメータの入力画面

以下のパラメータを入力してください。

パラメータの入力後は、"OK"ボタンを押してください。設定内容の確認画面が表示されます。

入力項目	入力値
Image Size	希望するイメージファイルのサイズを指定してください。（実際にはイメージ内に作成するファイルシステムの容量になります。ファイルシステムは、gzip で圧縮されますのでここに指定したサイズより小さいイメージが作成されます。）
Use loop device	イメージ作成時に本ツールが使用可能なループデバイス名を入力してください。デフォルトは、/dev/loop0 です。
Use mount point	イメージ作成時にループデバイスのマウントが可能なディレクトリを入力してください。デフォルトは、/mnt です。

7.8.2 INITRD イメージパラメータの確認

パラメータに設定された内容が表示されます。

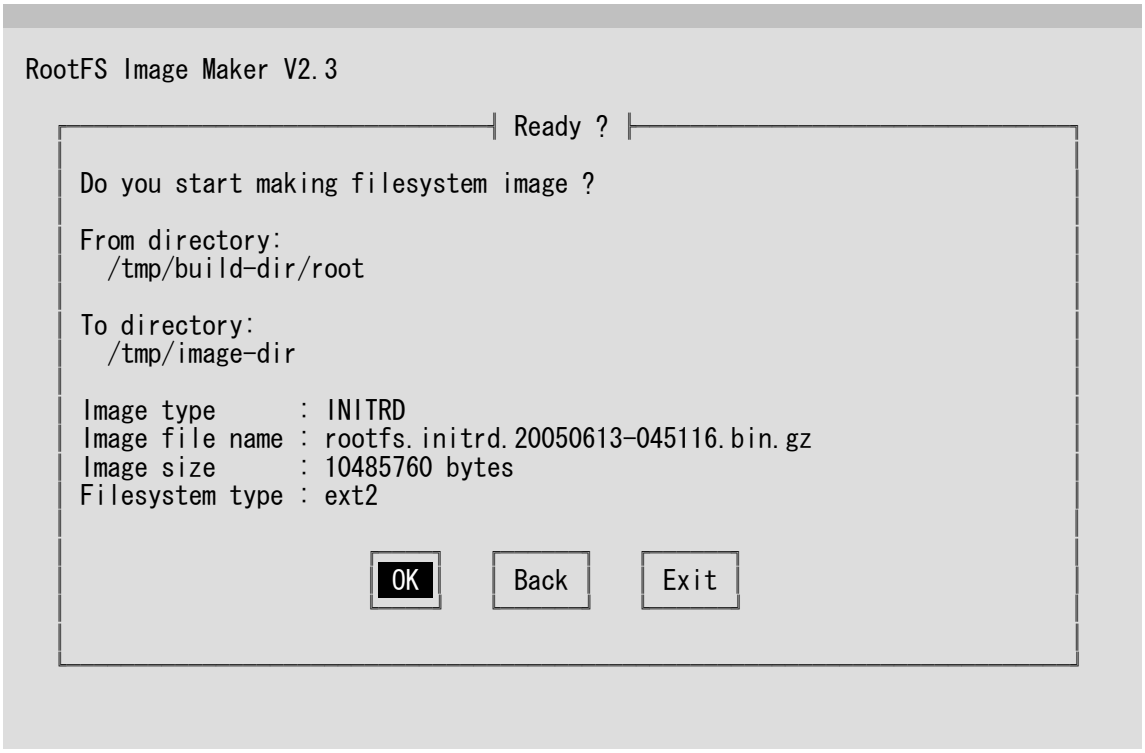


図 28 INITRD イメージパラメータの確認画面

表示内容に問題がなければ、"OK"ボタンを押してください。イメージの作成が開始されます。

本ツール起動時に、-o オプションでイメージファイル名が指定されていない場合、イメージファイル名は、以下になります。

［イメージファイル名］

"rootfs.initrd.YYYYMMDD-hhmmss.bin"

YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒

7.8.3 INITRD イメージの作成

作成中は以下のメッセージが出力されます。

このメッセージは、ログファイル to-dir/イメージファイル名.log にも出力されます。

- 作成を開始する旨のメッセージ
- 作成を終了する旨のメッセージ
- エラーメッセージ（エラー発生時のみ）

以下は正常終了時のメッセージです。

```
Making the INITRD image start.
20480+0 records in
20480+0 records out
mke2fs 1.35 (28-Feb-2004)
3498 blocks
Making the INITRD image succeeded.
RootFS Image Maker finish.
```

エラーが発生した場合にはパラメータ等を見直してください。

7.9 RAW イメージ

7.9.1 RAW パラメータの設定

イメージの作成に必要な各種パラメータを問い合わせる画面が表示されます。

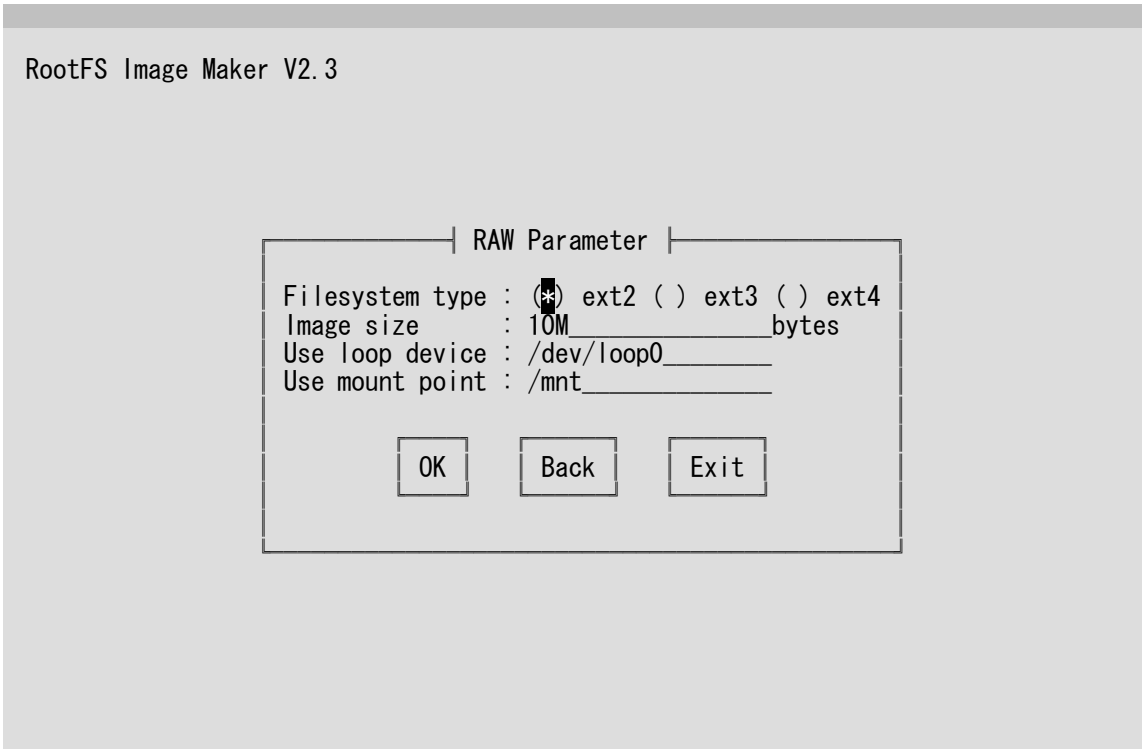


図 29 RAW イメージパラメータの入力画面

以下のパラメータを入力してください。

パラメータの入力後は、"OK"ボタンを押してください。設定内容の確認画面が表示されます。

入力項目	入力値
Filesystem type	イメージファイル内に作成するファイルシステムの種別を選択してください。
Image Size	希望するイメージファイルのサイズを指定してください。
Use loop device	イメージ作成時に本ツールが使用可能なループデバイス名を入力してください。デフォルトは、/dev/loop0 です。
Use mount point	イメージ作成時にループデバイスのマウントが可能なディレクトリを入力してください。デフォルトは、/mnt です。

7.9.2 RAW イメージパラメータの確認

パラメータに設定された内容が表示されます。

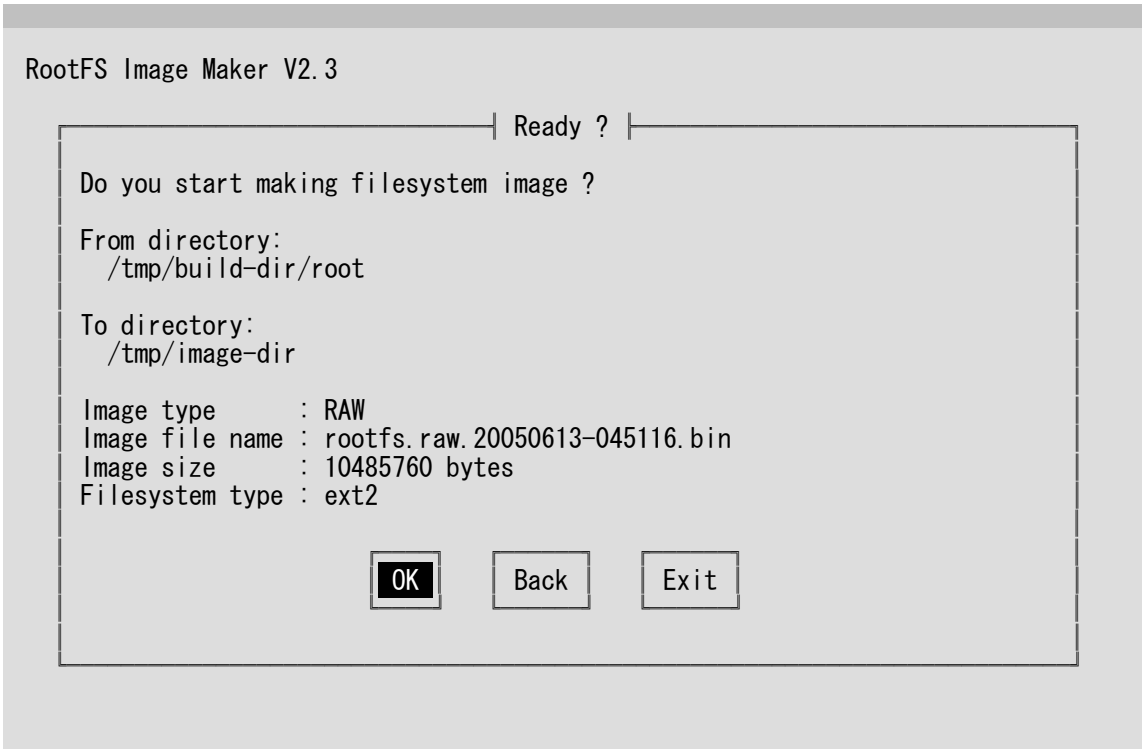


図 30 RAW イメージパラメータの確認画面

表示内容に問題がなければ、"OK"ボタンを押してください。イメージの作成が開始されます。

本ツール起動時に、-o オプションでイメージファイル名が指定されていない場合、イメージファイル名は、以下になります。

［イメージファイル名］

"rootfs.raw.YYYYMMDD-hhmmss.bin"

YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒

7.9.3 RAW イメージの作成

作成中は以下のメッセージが出力されます。

このメッセージは、ログファイル to-dir/イメージファイル名.log にも出力されます。

- 作成を開始する旨のメッセージ
- 作成を終了する旨のメッセージ
- エラーメッセージ（エラー発生時のみ）

以下は正常終了時のメッセージです。

```
Making the RAW image start.
Filesystem Type : ext3
20480+0 records in
20480+0 records out
mke2fs 1.35 (28-Feb-2004)
3498 blocks
Making the RAW image succeeded.
RootFS Image Maker finish.
```

エラーが発生した場合にはパラメータ等を見直してください。

7.10 SquashFS イメージ

7.10.1 SquashFS パラメータの設定

イメージの作成に必要な各種パラメータを問い合わせる画面が表示されます。

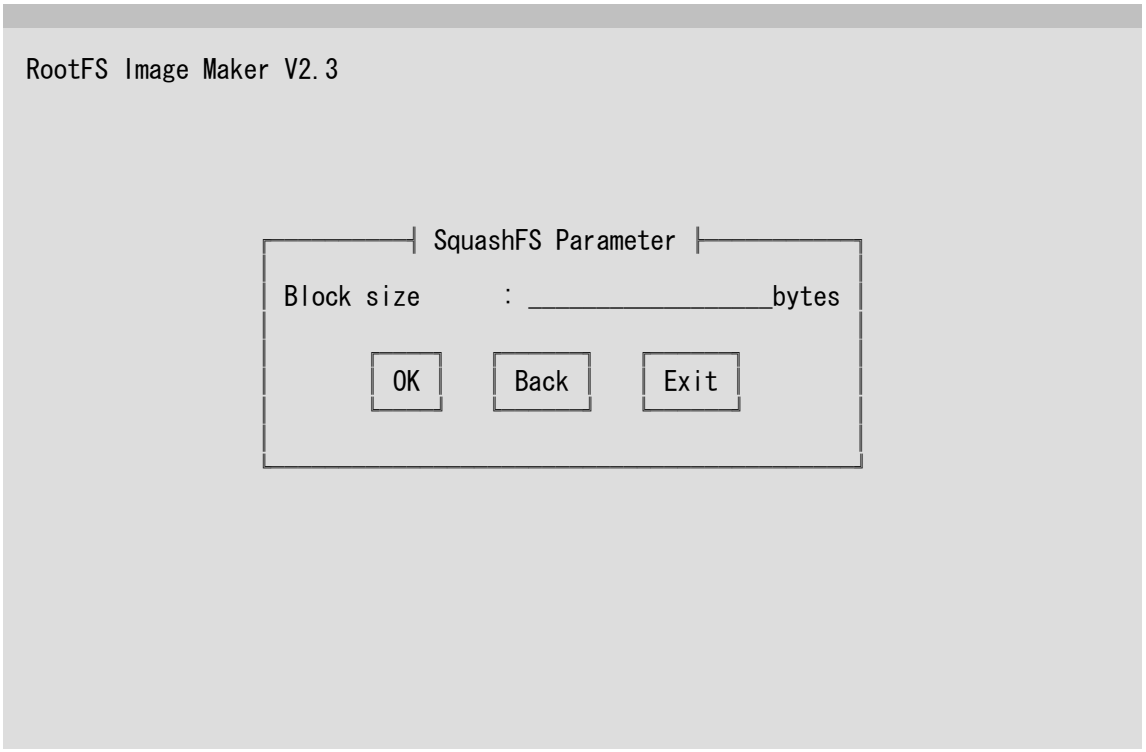


図 31 SquashFS イメージパラメータの入力画面

以下のパラメータを入力してください。

パラメータの入力後は、"OK"ボタンを押してください。設定内容の確認画面が表示されます。

入力項目	入力値
Block size	希望するブロックサイズを指定してください。

7.10.2 SquashFS イメージパラメータの確認

パラメータに設定された内容が表示されます。

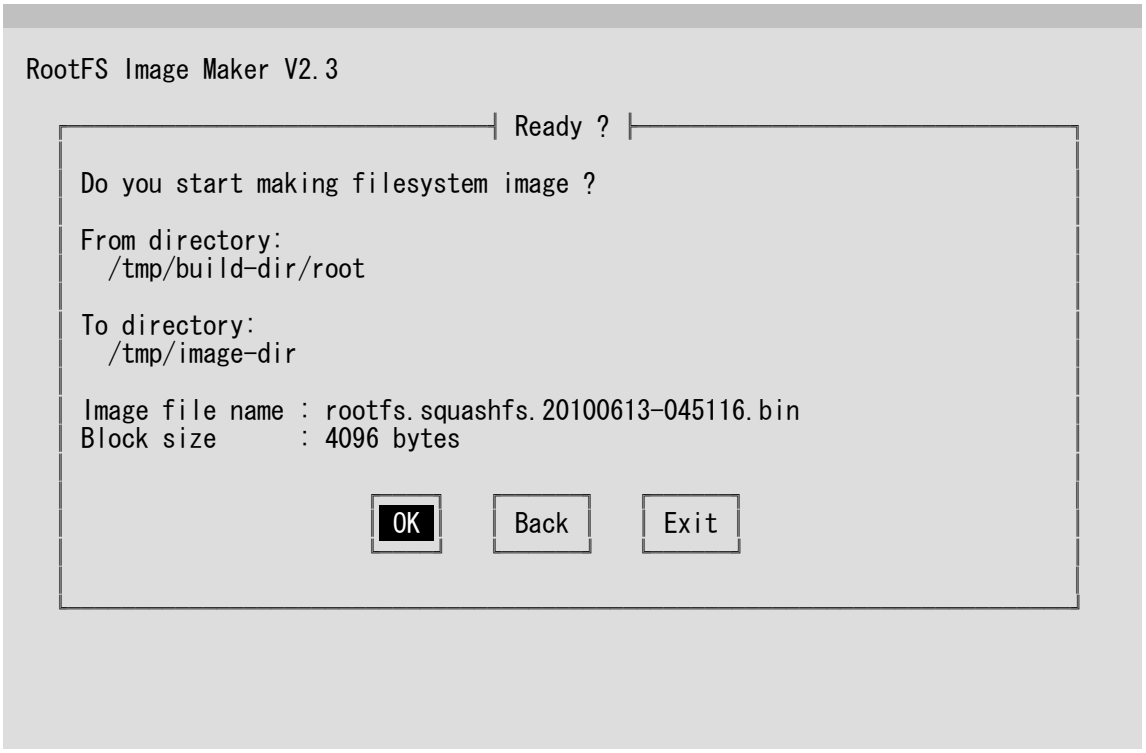


図 32 SquashFS イメージパラメータの確認画面

表示内容に問題がなければ、"OK"ボタンを押してください。イメージの作成が開始されます。
本ツール起動時に、-o オプションでイメージファイル名が指定されていない場合、イメージファイル名は、以下になります。

［イメージファイル名］
"rootfs.squashfs.YYYYMMDD-hhmmss.bin"
YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒

7.10.3 SquashFS イメージの作成

作成中は以下のメッセージが出力されます。

このメッセージは、ログファイル to-dir/イメージファイル名.log にも出力されます。

作成を開始する旨のメッセージ

作成を終了する旨のメッセージ

エラーメッセージ（エラー発生時のみ）

以下は正常終了時のメッセージです。

```
Making the SquashFS image start.
Making the SquashFS image succeeded.
RootFS Image Maker finish.
```

エラーが発生した場合にはパラメータ等を見直してください。

7.11 UBIFS イメージ

7.11.1 UBIFS パラメータの設定

イメージの作成に必要な各種パラメータを問い合わせる画面が表示されます。

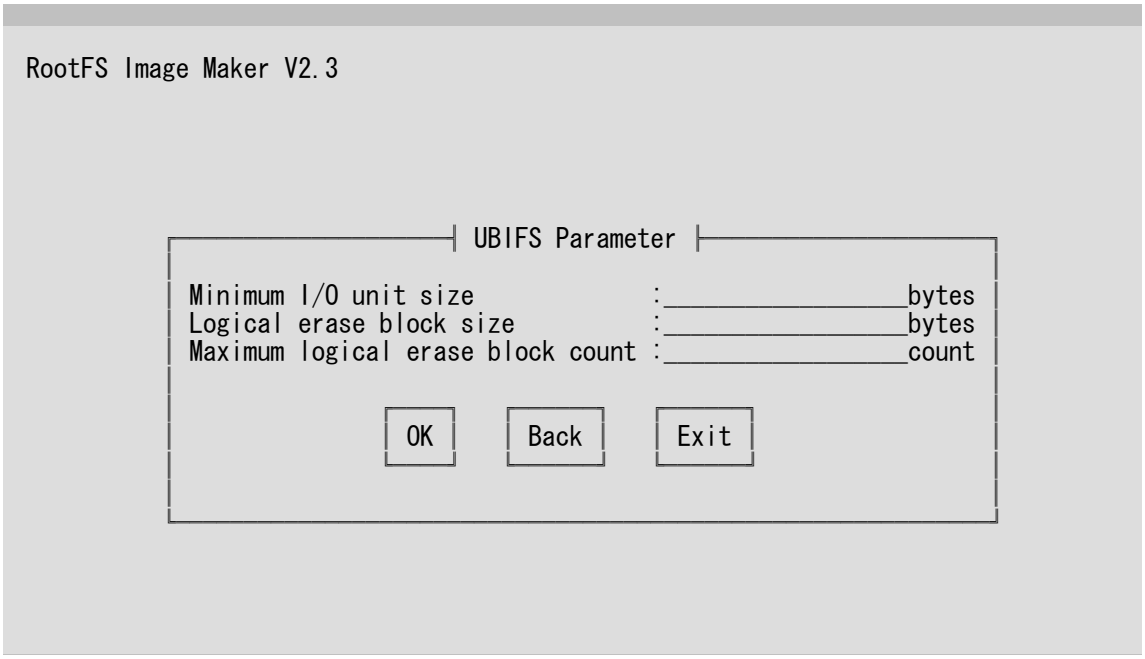


図 33 UBIFS イメージパラメータの入力画面

以下のパラメータを入力してください。

パラメータの入力後は、"OK"ボタンを押してください。設定内容の確認画面が表示されます。

入力項目	入力値
Minimum I/O unit size	希望する最小の I/O ユニットのサイズを入力してください。
Logical erase block size	ロジカルイレーズブロック（LEB）のサイズを入力してください。
Maximum logical erase block count	ロジカルイレーズブロックの最大カウント数を入力してください。

7.11.2 UBIFS イメージパラメータの確認

パラメータに設定された内容が表示されます。

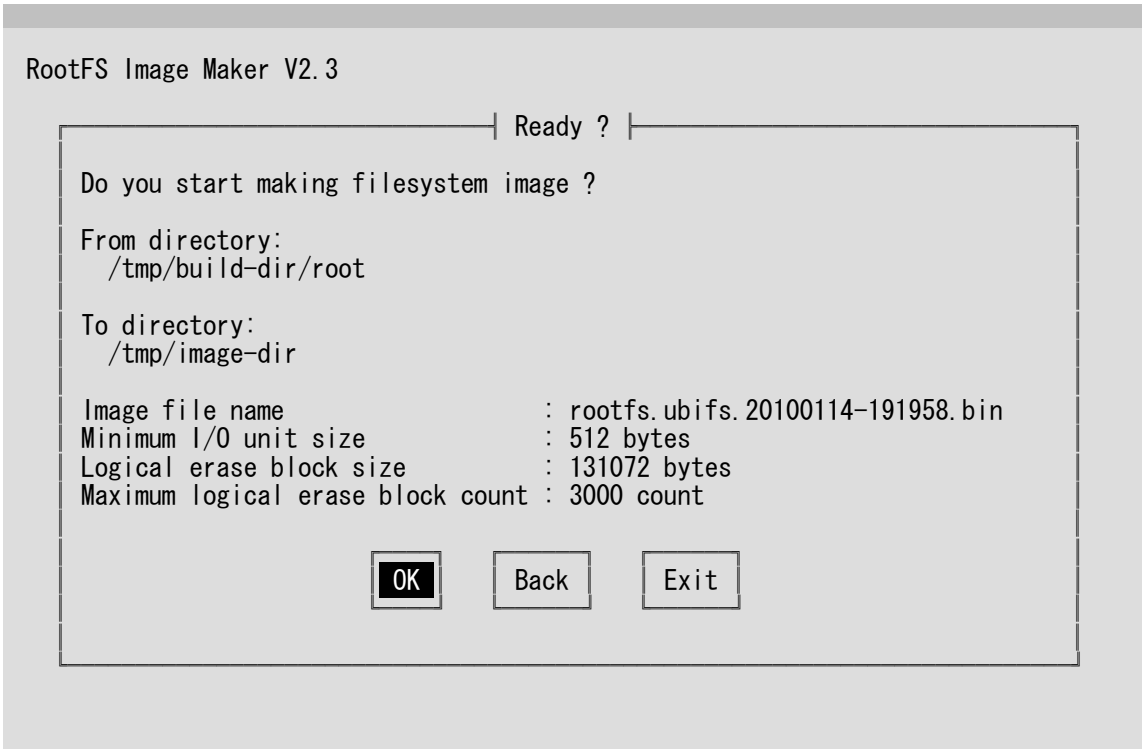


図 34 UBIFS イメージパラメータの確認画面

表示内容に問題がなければ、"OK"ボタンを押してください。イメージの作成が開始されます。
本ツール起動時に、-o オプションでイメージファイル名が指定されていない場合、イメージファイル名は、以下になります。

［イメージファイル名］
"rootfs.ubifs.YYYYMMDD-hhmmss.bin"
　YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒

7.11.3 UBIFS イメージの作成

作成中は以下のメッセージが出力されます。
このメッセージは、ログファイル to-dir/イメージファイル名.log にも出力されます。
作成を開始する旨のメッセージ
作成を終了する旨のメッセージ
エラーメッセージ（エラー発生時のみ）

以下は正常終了時のメッセージです。

```
Making the UBIFS image start.
Making the UBIFS image succeeded.
RootFS Image Maker finish.
```

エラーが発生した場合にはパラメータ等を見直してください。

7.12 Cramfs イメージ

7.12.1 Cramfs パラメータの設定

イメージの作成に必要な各種パラメータを問い合わせる画面が表示されます。

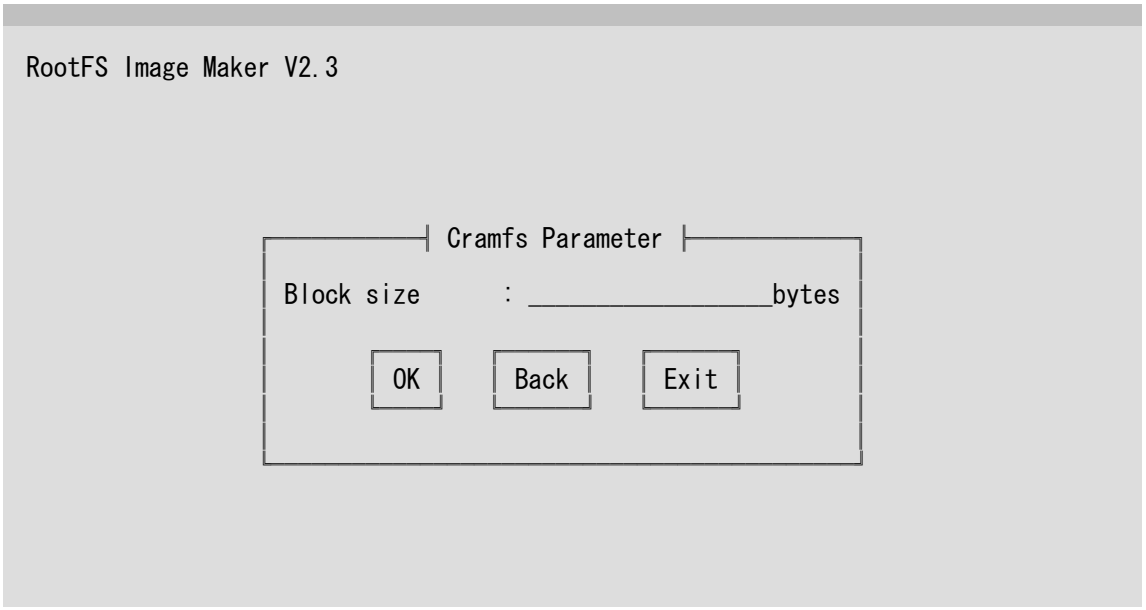


図 35 Cramfs イメージパラメータの入力画面

以下のパラメータを入力してください。

パラメータの入力後は、"OK"ボタンを押してください。設定内容の確認画面が表示されます。

入力項目	入力値
Block Size	希望するブロックサイズを指定してください。

7.12.2 Cramfs イメージパラメータの確認

パラメータに設定された内容が表示されます。

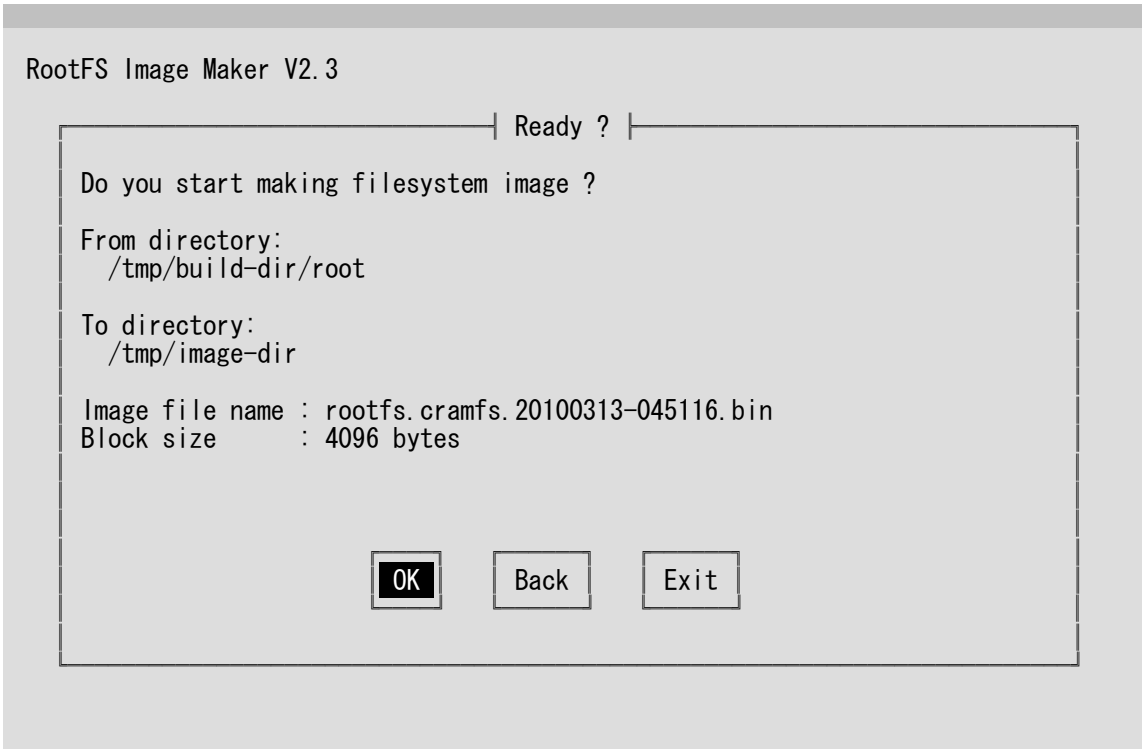


図 36 Cramfs イメージパラメータの確認画面

表示内容に問題がなければ、"OK"ボタンを押してください。イメージの作成が開始されます。
本ツール起動時に、-o オプションでイメージファイル名が指定されていない場合、イメージファイル名は、以下になります。

［イメージファイル名］
"rootfs.cramfs.YYYYMMDD-hhmmss.bin"
YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒

7.12.3 Cramfs イメージの作成

作成中は以下のメッセージが出力されます。
このメッセージは、ログファイル to-dir/イメージファイル名.log にも出力されます。
作成を開始する旨のメッセージ
作成を終了する旨のメッセージ
エラーメッセージ（エラー発生時のみ）

以下は正常終了時のメッセージです。

```
Making the Cramfs image start.
Making the Cramfs image succeeded.
RootFS Image Maker finish.
```

エラーが発生した場合にはパラメータ等を見直してください。

7.13 Logfs イメージ

7.13.1 Logfs パラメータの設定

イメージの作成に必要な各種パラメータを問い合わせる画面が表示されます。

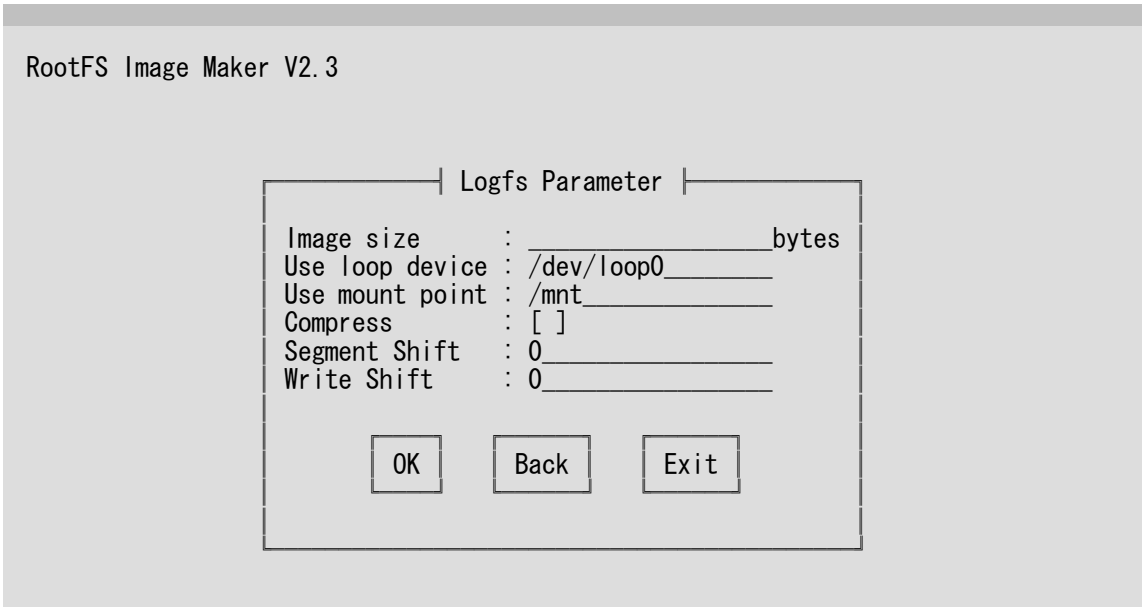


図 37 Logfs イメージパラメータの入力画面

以下のパラメータを入力してください。

パラメータの入力後は、"OK"ボタンを押してください。設定内容の確認画面が表示されます。

入力項目	入力値
Image Size	希望するイメージファイルのサイズを指定してください。
Use loop device	イメージ作成時に本ツールが使用可能なループデバイス名を入力してください。デフォルトは、/dev/loop0 です。
Use mount point	イメージ作成時にループデバイスのマウントが可能なディレクトリを入力してください。デフォルトは、/mnt です。
Compress	圧縮を有効にするかを指定してください。チェックすると有効になります。
Segment Shift	セグメントシフトを行うビット数を入力します。
Write Shift	ライトシフトを行うビット数を入力します。

7.13.2 Logfs イメージパラメータの確認

パラメータに設定された内容が表示されます。

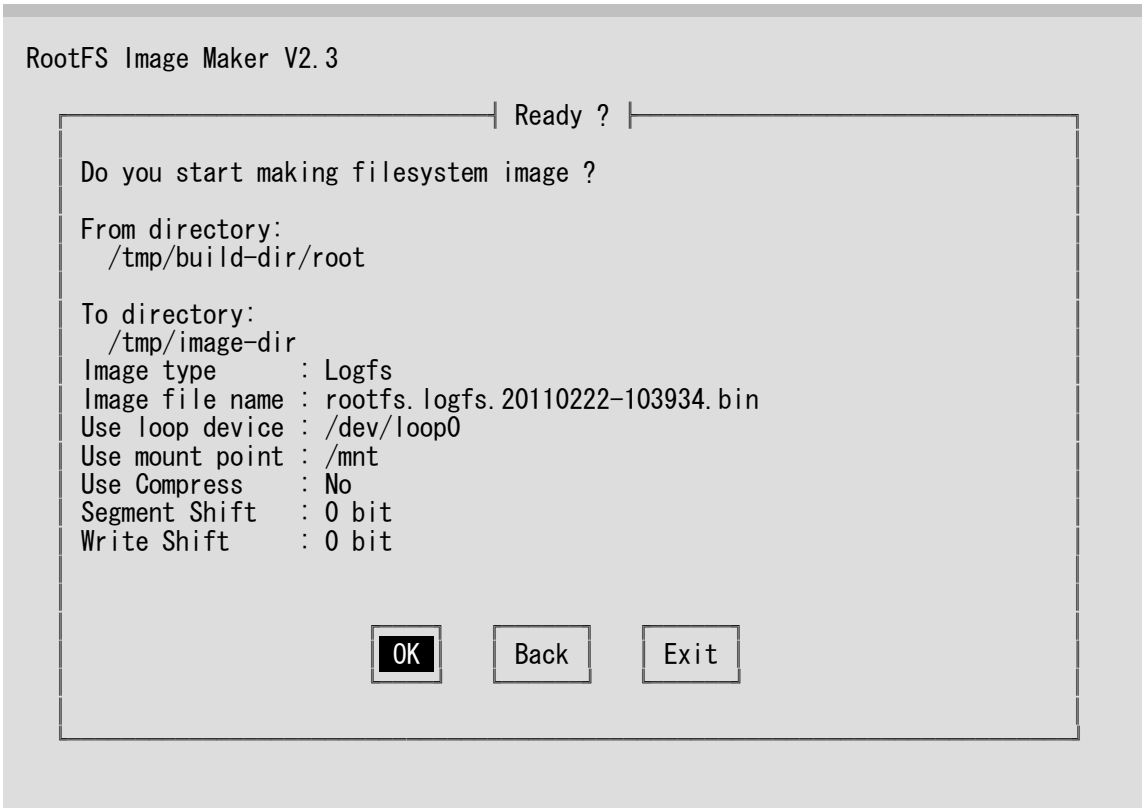


図 38 Logfs イメージパラメータの確認画面

表示内容に問題がなければ、"OK"ボタンを押してください。イメージの作成が開始されます。
本ツール起動時に、-o オプションでイメージファイル名が指定されていない場合、イメージファイル名は、以下になります。

［イメージファイル名］
"rootfs.logfs.YYYYMMDD-hhmmss.bin"
YYYY：年（西暦）、MM：月、DD：日、hh：時間、mm：分、ss：秒

7.13.3 Logfs イメージの作成

作成中は以下のメッセージが出力されます。

このメッセージは、ログファイル to-dir/イメージファイル名.log にも出力されます。

作成を開始する旨のメッセージ

作成を終了する旨のメッセージ

エラーメッセージ（エラー発生時のみ）

以下は正常終了時のメッセージです。

```
Making the logfs image start.
65536+0 records in
65536+0 records out
33554432 bytes (32 MB) copied, 0.267986 s, 125 MB/s
1 block
Making the Logfs image succeeded.
RootFS Image Maker finish.
```

エラーが発生した場合にはパラメータ等を見直してください。

付録１ 共有ライブラリ不足時の異常動作とその対策について

ルートファイルシステム作成ツールで作成したルートファイルシステムを使用して、システムを起動した場合に、共有ライブラリ（または、共有オブジェクト）のロードに失敗し、プログラムが起動しないことがあります。

この場合、当該のプログラムからロードされる共有ライブラリや共有オブジェクトをターゲット上に配置することで、プログラムが起動するようになります。

多くのプログラムでは､実行時にプログラムから動的にロードされる共有ライブラリ/共有オブジェクトを、プラグインと呼びます。一般に、プラグインは、拡張子に.so がついた通常のファイル※1 として作成されています。※2

プログラムが起動しないなどの場合は、当該のプログラムが含まれている rpm パッケージ内の拡張子.so のファイルがターゲット上に存在していることを確認してください。

代表的な既知のプラグインの不足で発生する異常現象と不足ライブラリや共有オブジェクトの例を**表 1** に示します。

表 1　代表的な既知のプラグイン形式の共有オブジェクト

No	パッケージ名	発生する現象	不足共有ライブラリ （共有オブジェクト）
1	glibc	マルチユーザモードで login する場合に、パスワード認証が成功しない	/lib/libnss_files.so.2 /lib/libnss_files-2.13.so
		テキストエンコーディングに関連する問題等	/usr/lib/gconv/ANSI_X3.110.so /usr/lib/gconv/ARMSCII-8.so /usr/lib/gconv/ASMO_449.so /usr/lib/gconv/BIG5.so /usr/lib/gconv/BIG5HKSCS.so /usr/lib/gconv/BRF.so /usr/lib/gconv/CP10007.so /usr/lib/gconv/CP1125.so ～～　省略 ～～ /usr/lib/gconv/UTF-16.so /usr/lib/gconv/UTF-32.so /usr/lib/gconv/UTF-7.so /usr/lib/gconv/VISCII.so /usr/lib/gconv/libCNS.so /usr/lib/gconv/libGB.so /usr/lib/gconv/libISOIR165.so /usr/lib/gconv/libJIS.so /usr/lib/gconv/libJISX0213.so /usr/lib/gconv/libKSC.so

※1 シンボリックリンクではないという意味です。

※2 プラグイン形式の共有オブジェクトは、実行環境に合わせて必要なライブラリをロード・実行することで、システムの総サイズを削減する意図で導入されます。

		その他 glibc に関連する共有オブジェクト	/lib/libBrokenLocale-2.13.so /lib/libBrokenLocale.so.1 /lib/libSegFault.so /lib/libanl-2.13.so /lib/libanl.so.1 /lib/libcidn-2.13.so /lib/libcidn.so.1 /lib/libnss_compat-2.13.so /lib/libnss_compat.so.2 /lib/libnss_dns-2.13.so /lib/libnss_dns.so.2 /lib/libnss_hesiod-2.13.so /lib/libnss_hesiod.so.2 /lib/libnss_nis-2.13.so /lib/libnss_nis.so.2 /lib/libnss_nisplus-2.13.so /lib/libnss_nisplus.so.2 /lib/libthread_db-1.0.so /lib/libthread_db.so.1
2	pam	PAM を用いた認証処理を行うシステムに、マルチユーザモードで login する場合、パスワード認証が成功しない。	/lib/security/pam_access.so /lib/security/pam_cap.so /lib/security/pam_chroot.so /lib/security/pam_console.so /lib/security/pam_cracklib.so /lib/security/pam_debug.so /lib/security/pam_deny.so /lib/security/pam_echo.so /lib/security/pam_env.so /lib/security/pam_exec.so /lib/security/pam_filter /lib/security/pam_filter.so /lib/security/pam_filter/upperLOWER /lib/security/pam_ftp.so /lib/security/pam_group.so /lib/security/pam_issue.so /lib/security/pam_keyinit.so /lib/security/pam_lastlog.so /lib/security/pam_limits.so /lib/security/pam_listfile.so /lib/security/pam_localuser.so /lib/security/pam_loginuid.so /lib/security/pam_mail.so /lib/security/pam_mkhomedir.so /lib/security/pam_motd.so /lib/security/pam_namespace.so /lib/security/pam_nologin.so /lib/security/pam_permit.so /lib/security/pam_postgresok.so /lib/security/pam_rhosts.so /lib/security/pam_rhosts_auth.so /lib/security/pam_rootok.so /lib/security/pam_rps.so

			/lib/security/pam_securetty.so /lib/security/pam_selinux.so /lib/security/pam_shells.so /lib/security/pam_stack.so /lib/security/pam_stress.so /lib/security/pam_succeed_if.so /lib/security/pam_tally.so /lib/security/pam_tally2.so /lib/security/pam_time.so /lib/security/pam_timestamp.so /lib/security/pam_umask.so /lib/security/pam_unix.so /lib/security/pam_unix_acct.so /lib/security/pam_unix_auth.so /lib/security/pam_unix_passwd.so /lib/security/pam_unix_session.so /lib/security/pam_userdb.so /lib/security/pam_warn.so /lib/security/pam_wheel.so /lib/security/pam_xauth.so /lib/libpamc.so.0 /lib/libpamc.so.0.82.1 /lib/security/pam_faildelay.so /lib/security/pam_pwhistory.so /lib/security/pam_selinux_permit.so /lib/security/pam_sepermit.so /lib/security/pam_tty_audit.so
3	libuser	以下の libuser 関連ツールを起動しようとしたときに、「 Error initializing libuser:/usr/lib/libuser/libuser_XXXX.so」というメッセージが表示され、異常終了する(XXXX の部分には、files/ldap/shadow が含まれます)。 ・/usr/sbin/lchage ・/usr/sbin/lgroupadd ・/usr/sbin/lgroupdel ・/usr/sbin/lgroupmod ・/usr/sbin/lid ・/usr/sbin/lnewusers ・/usr/sbin/lpasswd ・/usr/sbin/luseradd ・/usr/sbin/luserdel ・/usr/sbin/lusermod	/usr/lib/libuser/libuser_files.so /usr/lib/libuser/libuser_ldap.so /usr/lib/libuser/libuser_shadow.so

付録２ バッチインストールのサンプルスクリプト

パッケージインストーラ実行時に出力されるインストール済みパッケージの一覧を利用して、rpmパッケージをバッチインストールするためのサンプルスクリプトを以下に示します。

なお、本スクリプトを使用した場合、複数のパッケージで重複するファイルは、すべて、後にインストールされるパッケージのファイルによって上書きされます。

上書きを考慮したインストールを行う場合はパッケージインストーラを使用した自動インストールオプションを使用してください。

【スクリプト：/opt/ubinux_pkg_tools/share/tools/sample-script/install_sample.sh】

```
#!/bin/bash

INSTDIR=
TRPMDBDIR=
HRPMDNDIR=
PKGDIR=
PKGLIST=

PRELIST=/tmp/ubinux_install_prelist.$$
NORMALLIST=/tmp/ubinux_install_normallist.$$
POSTLIST=/tmp/ubinux_install_postlist.$$
HOSTLIST=/tmp/ubinux_install_hostlist.$$

function check()
{
    ### root check
    if [ "$UID" != "0" ] ; then
        echo "error: You must be root"
        return 1
    fi

    ### parameter check
    if [ "$#" != "5" ] ; then
        echo "error: parameter error"
        echo "install_sample: <install-dir> <target-rpmdb-dir> <host-rpmdb-dir>
<packages-dir> <packages-list>"
        return  1
    fi

   （つづく）
```

(つづき)

```
  INSTDIR=$1
  TRPMDBDIR=$2
  HRPMDBDIR=$3
  PKGDIR=$4
  PKGLIST=$5

 ### path check
 topchar=`echo "$INSTDIR" | cut -b1`
 if [ "$topchar" != "/" ]; then
     echo "error: install-dir must be specified absolute path: $INSTDIR"
     return 1
 fi

 topchar=`echo "$TRPMDBDIR" | cut -b1`
 if [ "$topchar" != "/" ]; then
     echo "error: target-rpmdb-dir must be specified absolute path: $TRPMDBDIR"
     return 1
 fi

 topchar=`echo "$HRPMDBDIR" | cut -b1`
 if [ "$topchar" != "/" ]; then
     echo "error: host-rpmdb-dir must be specified absolute path: $HRPMDBDIR"
     return 1
 fi

 if [ -d "$INSTDIR" ]; then
     echo "error: already exists install-dir: $INSTDIR"
     return 1
 fi

 if [ -d "$HRPMDBDIR" ]; then
     echo "error: already exists host-rpmdb-dir: $HRPMDBDIR"
     return 1
 fi

 if [ ! -d "$PKGDIR" ]; then
     echo "error: not found packages-dir: $PKGDIR"
     return 1
 fi

 if [ ! -f "$PKGLIST" ]; then
     echo "error: not found packages-list: $PKGLIST"
     return 1
 fi
```

(つづく)

（つづき）

```
    ### make direcroty
    mkdir -p $INSTDIR/$TRPMDBDIR
    if [ "$?" != "0" ]; then
        echo "error: Can't create directory: $INSTDIR/$TRPMDBDIR"
        return 1
    fi

    mkdir -p $HRPMDBDIR
    if [ "$?" != "0" ]; then
        echo "error: Can't create directory: $HRPMDBDIR"
        return 1
    fi

    return 0
}

function search_file()
{
    while read str
    do
        pkgtype=`echo "$str" | cut -d: -f1`
        pkgname=`echo "$str" | cut -d: -f2`

        pkgfile=`find $PKGDIR -name $pkgname`
        if [ "$?" != "0" ]; then
            echo "error: not found: $pkgname"
            return 1
        fi

        case "$pkgtype" in
            PRE)    echo "$pkgfile" >> $PRELIST;;
            NORMAL) echo "$pkgfile" >> $NORMALLIST;;
            POST)   echo "$pkgfile" >> $POSTLIST;;
            HOST)   echo "$pkgfile" >> $HOSTLIST;;
            *)      echo "error: invalid package-list file"; return 1;;
        esac
    done < $PKGLIST
}
```

（つづく）

```
   (つづき)

function pre_post_install()
{
    if [ ! -s $1 ]; then
        # no install packages
        return 0
    fi

    while read pkgfile
    do
        rpm -ivh --root "$INSTDIR" --dbpath "$TRPMDBDIR" ¥
        --noscripts --notriggers --ignorearch --force ¥
        --nodeps $pkgfile
        if [ $? != "0" ]; then
            return 1
        fi
    done < $1

    return 0
}

function normal_install()
{
    if [ ! -s $NORMALLIST ]; then
        # no normal-install packages
        return 0
    fi

    cat $NORMALLIST | ¥
    xargs rpm -ivh --root "$INSTDIR" --dbpath "$TRPMDBDIR" ¥
    --noscripts --notriggers --ignorearch --force ¥
    --nodeps

    return $?
}

function host_install()
{
    if [ ! -s $HOSTLIST ]; then
        # no post-install packages
        return 0
    fi

    cat $HOSTLIST | ¥
    xargs rpm -ivh --dbpath "$HRPMDBDIR" ¥
    --noscripts --notriggers --ignorearch --force ¥
    --nodeps

    return $?
}
   (つづく)
```

```
   （つづき）

function remove_tmpfile()
{
    rm -f $PRELIST > /dev/null
    rm -f $NORMALLIST > /dev/null
    rm -f $POSTLIST > /dev/null
    rm -f $HOSTLIST > /dev/null
}

####### main ###########

# parameter check
check $@
if [ "$?" != "0" ]; then
    exit 1
fi

# package file search
search_file
if [ "$?" != "0" ]; then
    exit 1
fi

# install
pre_post_install $PRELIST
if [ "$?" != "0" ]; then
    remove_tmpfile
    exit 1
fi

normal_install
if [ "$?" != "0" ]; then
    remove_tmpfile
    exit 1
fi

pre_post_install $POSTLIST
if [ "$?" != "0" ]; then
    remove_tmpfile
    exit 1
fi

host_install
if [ "$?" != "0" ]; then
    remove_tmpfile
    exit 1
fi

remove_tmpfile
exit 0
```

【サンプルスクリプトの起動例】

```
$ sh install_sample.sh /opt/ubq/devkit/x86/x86/target /opt/ubq/trpmdb
                                  ①                    ②
 /opt/ubq/devkit/x86/x86/hrpmdb  /opt/ubq/devkit/x86/x86/rpms
              ③                          ④
 /root/install_pakage.pkglist
            ⑤
```

① ターゲット向けパッケージをインストールするディレクトリを指定します。
② ターゲット向けパッケージの rpm データベースを作成するディレクトリを指定します。
③ ホスト向けパッケージ（クロス環境ツール）の rpm データベースを作成するディレクトリを指定します。ホスト向けパッケージは、パッケージ内に定義されたディレクトリにインストールされます。
④ パッケージファイルが格納されているディレクトリを指定します。
⑤ パッケージインストーラで出力されたインストール一覧ファイルを指定します。

ターゲット向けパッケージの rpm データベースは、実際には、① + ②のディレクトリに作成されます。上記例の場合、ターゲット向けパッケージの rpm データベースは、/opt/ubq/devkit/x86/x86/target/opt/ubq/trpmdb 配下に作成されます。

【サンプルスクリプトの実行結果例】

```
Preparing...                ########################################### [100%]
   1:setup                  ########################################### [100%]
Preparing...                ########################################### [100%]
   1:filesystem             ########################################### [100%]
Preparing...                ########################################### [100%]
   1:busybox                ########################################### [  0%]
   2:bash                   ########################################### [  1%]
   3:perl                   ########################################### [  1%]
    .
    .
    .
Preparing...                ########################################### [100%]
   1:x86-cross-binutils     ########################################### [  6%]
   2:x86-cross-gcc          ########################################### [ 11%]
   3:x86-cross-gcc-awk      ########################################### [ 17%]
    .
    .
    .
```

付録３ ubinux_make_rootfs.dev.conf の例

ルートファイルシステム作成ツールから使用される ubinux_make_rootfs.dev.conf の例を図 39 から 図 42 に示します。 本サンプルファイルは、ubinux インストーラのインストール後,
/opt/ubinux_pkg_tools/share/tools/sample-config/ubinux_make_rootfs.dev.conf
に格納されています。

```
#-------------------------------------------------------------
# Make device file config
#-------------------------------------------------------------

#-------------------------------------------------------------
# Charactor device
#-------------------------------------------------------------

# Memory devices (Major=1)
- mem c 1 1 root kmem 0640
- kmem c 1 2 root kmem 0640
- null c 1 3 root root 0666
CONFIG_ISA port c 1 4 root kmem 640
- zero c 1 5 root root 0666
- full c 1 7 root root 0644
- random c 1 8 root root 0644
- urandom c 1 9 root root 0644
- kmsg c 1 11 root root 0600
- shm d root root 01777

# Serial 8250/16550 (Major=4)
CONFIG_SERIAL_8250 ttyS0 c 4 64 root uucp 0660
CONFIG_SERIAL_8250 ttyS1 c 4 65 root uucp 0660
```

図 39 ubinux_make_rootfs.dev.conf の例(その１)

```
# Alternate TTY devices (Major=5)
- tty c 5 0 root root 0666
- console c 5 1 root root 0600
CONFIG_UNIX98_PTYS ptmx c 5 2 root root 0666
CONFIG_UNIX98_PTYS pts d root root 0755

#---------------------------------------------------------------
# Block device
#---------------------------------------------------------------

# RAM disk (Major=1)
CONFIG_BLK_DEV_RAM ram0 b 1 0 root disk 0640
CONFIG_BLK_DEV_RAM ram1 b 1 1 root disk 0640
CONFIG_BLK_DEV_RAM ram2 b 1 2 root disk 0640
CONFIG_BLK_DEV_RAM ram3 b 1 3 root disk 0640
CONFIG_BLK_DEV_RAM ram4 b 1 4 root disk 0640
CONFIG_BLK_DEV_RAM ram5 b 1 5 root disk 0640
CONFIG_BLK_DEV_RAM ram6 b 1 6 root disk 0640
CONFIG_BLK_DEV_RAM ram7 b 1 7 root disk 0640
CONFIG_BLK_DEV_RAM ram8 b 1 8 root disk 0640
CONFIG_BLK_DEV_RAM ram9 b 1 9 root disk 0640
CONFIG_BLK_DEV_RAM ram10 b 1 10 root disk 0640
CONFIG_BLK_DEV_RAM ram11 b 1 11 root disk 0640
CONFIG_BLK_DEV_RAM ram12 b 1 12 root disk 0640
CONFIG_BLK_DEV_RAM ram13 b 1 13 root disk 0640
CONFIG_BLK_DEV_RAM ram14 b 1 14 root disk 0640
CONFIG_BLK_DEV_RAM ram15 b 1 15 root disk 0640
CONFIG_BLK_DEV_RAM ramdisk s ram0
CONFIG_BLK_DEV_RAM ram s ram1
```

図 40 ubinux_make_rootfs.dev.conf の例(その２)

```
# Floppy disks (Major=2)
CONFIG_BLK_DEV_FD fd0 b 2 0 root disk 0660

# Loopback devices (Major=7)
CONFIG_BLK_DEV_LOOP loop0 b 7 0 root disk 0640
CONFIG_BLK_DEV_LOOP loop1 b 7 1 root disk 0640
CONFIG_BLK_DEV_LOOP loop2 b 7 2 root disk 0640
CONFIG_BLK_DEV_LOOP loop3 b 7 3 root disk 0640
CONFIG_BLK_DEV_LOOP loop4 b 7 4 root disk 0640
CONFIG_BLK_DEV_LOOP loop5 b 7 5 root disk 0640
CONFIG_BLK_DEV_LOOP loop6 b 7 6 root disk 0640
CONFIG_BLK_DEV_LOOP loop7 b 7 7 root disk 0640
```

図 41 ubinux_make_rootfs.dev.conf の例(その３)

```
# MTD devices (Major=31)
CONFIG_MTD_BLOCK mtdblock0 b 31 0 root disk 0640
CONFIG_MTD_BLOCK mtdblock1 b 31 1 root disk 0640
CONFIG_MTD_BLOCK mtdblock2 b 31 2 root disk 0640
CONFIG_MTD_BLOCK mtdblock3 b 31 3 root disk 0640
CONFIG_MTD_BLOCK mtdblock4 b 31 4 root disk 0640
CONFIG_MTD_BLOCK mtdblock5 b 31 5 root disk 0640
CONFIG_MTD_BLOCK mtdblock6 b 31 6 root disk 0640
CONFIG_MTD_BLOCK mtdblock7 b 31 7 root disk 0640

CONFIG_MTD_BLOCK_RO mtdblock0 b 31 0 root disk 0640
CONFIG_MTD_BLOCK_RO mtdblock1 b 31 1 root disk 0640
CONFIG_MTD_BLOCK_RO mtdblock2 b 31 2 root disk 0640
CONFIG_MTD_BLOCK_RO mtdblock3 b 31 3 root disk 0640
CONFIG_MTD_BLOCK_RO mtdblock4 b 31 4 root disk 0640
CONFIG_MTD_BLOCK_RO mtdblock5 b 31 5 root disk 0640
CONFIG_MTD_BLOCK_RO mtdblock6 b 31 6 root disk 0640
CONFIG_MTD_BLOCK_RO mtdblock7 b 31 7 root disk 0640

#-----------------------------------------------------------------
# Symbolic link
#-----------------------------------------------------------------
CONFIG_PROC_FS fd s ../proc/self/fd
CONFIG_PROC_FS stdin s ../proc/self/fd/0
CONFIG_PROC_FS stdout s ../proc/self/fd/1
CONFIG_PROC_FS stderr s ../proc/self/fd/2
CONFIG_PROC_FS core s ../proc/kcore
```

図 42 ubinux_make_rootfs.dev.conf の例(その４)