

## RPM 関連のFAQ

- [RPM 関連のFAQ](#)
  - [RPM の基本コマンド](#)
    - [新規インストール](#)
    - [アップグレード](#)
    - [アンインストール](#)
    - [インストール済みリスト](#)
    - [参考情報](#)
  - [あるディストリビューションで glibc-x.x が必要と出て RPM パッケージが導入できない。](#)
  - [だれかつくってねーかなあ](#)
  - [別のディストリビューション用の RPM パッケージを利用できますか?](#)
  - [RPM パッケージの中身を取り出したい。](#)
  - [依存関係で入れたいパッケージをインストールできません](#)
  - [RPM ファイルの命名規則を知りたい。](#)
  - [インストールされているパッケージを知りたい。](#)
  - [パッケージをアップデートすると設定ファイルが置き換わったりそのままだったりするのはなぜ?](#)
    - [パッケージに含まれる設定ファイルが同じ場合](#)
    - [パッケージに含まれる設定ファイルに違いがあるが、従来のものも使える\(互換性がある\)場合](#)
    - [パッケージに含まれる設定ファイルに違いがあり、従来のものは使えない\(互換性がない\)場合](#)
    - [注意: 正しく作られてないパッケージの場合](#)
  - [必要なライブラリ\(ソフトウェア\)がインストールされているのにそれを使用するソフトをコンパイルできません。](#)
  - [xx コマンドのソースはどこにあるのですか?](#)
  - [RPMファイルの作り方](#)
  - [Source RPMのリビルドからインストールまでの作業は実際どのようにしますか。](#)
  - [Debian\(deb\)/Slackware\(tgz\) パッケージから RPM を作成できますか。\(Alien\)](#)
    - [インストール](#)
    - [パッケージを変換](#)
  - [ソースファイルから RPM を簡単に作成できますか。\(CheckInstall\)](#)
    - [CheckInstall の使い方 \(hoge-1.2.3.tar.gz をインストールするとき\)](#)
    - [関連日本語解説サイト](#)
  - [ダウングレード\(古いバージョンのパッケージをインストール\)したいとき](#)
  - [2ch Linux板 スレッド](#)

## RPM の基本コマンド

パッケージの追加や削除、アップグレードといったよく使う基本的な操作を説明します。

- オプションの `vh` は詳細表示のためにつけてるだけで、無くてOK。

## 新規インストール

# rpm -ivh パッケージファイル名

新規インストール時にはさらに rpm -ivh --test パッケージファイル名のように次のオプションが追加可能です。

- --test (テストのみ実行し、実際にインストールはしない)
- --force (インストール済であっても強制的にインストール実行する)
- --nodeps (依存性を無視してインストールを行う。非推奨)

## アップグレード

- # rpm -Fvh パッケージファイル名
  - 通常のオプションは -Fvh です
  - kernel のアップグレード時のみ -ivh オプションでインストールするのが無難です。
- # rpm -Uvh パッケージファイル名
  - アップグレードと新規インストールを兼ねます

## アンインストール

- # rpm -e パッケージ名
- # rpm -e --nodeps パッケージ名
  - 依存性を無視して強制的に削除を行う。非推奨

たとえばインストールしたパッケージのファイル名が hoge-1.2.3-4.i386.rpm なら、パッケージ名は hoge となります。

## インストール済みリスト

- \$ rpm -qa
  - 表示されますが恐らく量が半端ではありません。
- \$ rpm -qa | grep hoge
  - 普通はこうします。hogeに当てはまったもののみリストに表示されます。
- \$ rpm -qa | less
  - どうしても全部表示したい時はページャーと一緒に使うのが普通。

## 参考情報

- RPM HOWTO

<http://linuxjf.sourceforge.jp/JFdocs/RPM-HOWTO.html>

- Manpage of RPM - JF Project

<http://linuxjm.sourceforge.jp/html/rpm/man8/rpm.8.html>

ターミナルから `man rpm` と入力しても詳細な解説が見れます。

## あるディストリビューションで `glibc-x.x` が必要と出て RPM パッケージが導入できない。

ソースパッケージ (`src.rpm`) を探してリビルドしパッケージを作り直すか、ソースからコンパイルするか、必要とされているバージョンの `glibc` を採用しているディストリビューションのバージョンにアップグレードしましょう。

`glibc` (GNU C Library) は `init`, `Bash` を含め、Linux 上の C 言語で書かれたほとんど全てのソフトウェアが使用する極めて重要なライブラリです。`glibc` のアップグレードに失敗するとシステムが動作しなくなる危険があります。`glibc` を自前でアップグレードするのは避けた方が無難です。

## だれかつくってねーかなあ

<http://rpmfind.net/>

ただし、つぎの項目を読んだうえで利用してください。

## 別のディストリビューション用の RPM パッケージを利用できますか？

ディストロによって

- ビルドに使ったツール
  - コンパイラのバージョン
  - ライブラリのバージョン
  - コンパイル時のオプション
- インストールするファイルの場所と構成
  - 起動スクリプト/メニュー
  - 初期化設定ファイル
- ソースコードを修正するためのパッチ

とかが微妙に異なっていたりするのであまりおすすめしない。

特にビルド済みのバイナリパッケージは、できるだけその環境でビルドされたものを使った方がいいかと。

大体のシステムが似通っていれば、ソースパッケージを自分でビルドすれば使えることもある。

- RHEL クローンなど、全体が大体同じ構成で作られている場合は、バイナリパッケージもそのまま使えたりする。

## RPM パッケージの中身を取り出したい。

いったん `rpm2cpio` コマンドで `cpio` アーカイブに変換すれば取り出せます。

```
rpm2cpio hoge-1.0.0-1.i386.rpm | cpio -idm
```

また `tar.gz` 形式に変換する `rpm2tgz` コマンドが用意されていることもあります。

```
rpm2tgz hoge-1.0.0-1.i386.rpm
```

```
tar xvfz hoge-1.0.0-1.i386.tgz
```

rpm-utils というのもあります。rpmパッケージを操作するスクリプト集です。rpm-get hoge-1.0.0-1.i386.rpm と入力するとパッケージに含まれるファイルが番号つきリストで出力され、rpm-get hoge-1.0.0-1.i386.rpm -n 3 と入力すると3番目を標準出力に出力します。

## 依存関係で入れたいパッケージをインストールできません

まとめて指定しましょう。

例えば rpm -Uhv hoge-1.0.0-1.i386.rpm と入力して

Failed dependencies:

fuga = 2.0.0-1 is needed by 1.0.0-1

と表示される場合は、rpm -Uhv hoge-1.0.0-1.i386.rpm fuga-2.0.0-1.i386.rpm のように**同時**に入れればいいです。

## RPM ファイルの命名規則を知りたい。

i686 は Pentium Pro 以降用に最適化がされたパッケージです。386, 486, Pentium マシンで動く保証はありません。

これに対して i386 は 386 以降の x86 系すべての CPU 用です。Intel の 386, 486, Pentium, Pentium Pro 以降のすべてで使用可能です。AMD の K6, Athlon, Duron でも使用可能です。

表示または名称	説明
i386	<b>386 以降用</b> 。x86 (IA-32) 系の全ての CPU で動く万能選手。その反面、あまり最適化されていない。
i486	<b>486 以降用</b> に最適化されたもの。386 マシンで動く保証なし。
i586	<b>Pentium 以降用</b> に最適化されたもの。386, 486 マシンで動く保証なし。
i686	<b>Pentium Pro 以降用</b> に最適化されたもの。386, 486, Pentium, K6 マシンで動く保証なし。
k6	<b>AMD の K6 以降用</b> に最適化されたもの。Intel 製の CPU で動く保証なし。K6 は基本的に i586 に含まれるので、K6 用の RPM がない場合、i586 で代用できる。
k7, athlon	<b>AMD の Athlon 以降用</b> に最適化されたもの。Intel 製の CPU, K6 シリーズで動く保証なし。Athlon, Duron マシンは基本的に i686 に含まれるので、Athlon 用の RPM がない場合、i686 で代用できる。
ia32e	Intel の 32/64bit CPU (EM64T) マシン用のもの。32bit CPU マシンで動く保障なし。
amd64, x86_64	AMD の 32/64bit CPU (AMD64) マシン用のもの。32bit CPU マシンで動く保障なし。
ia64	Intel の 64bit CPU (IA-64) マシン用のもの。32bit CPU マシンで動く保障なし。
ppc, ppc64, ppcseries, ppcpseries	IBM の <b>Power</b> アーキテクチャ系 (PowerPC, POWER5, etc) の CPU を搭載した Mac とか iSeries 等のマシン用。x86 系のマシンに入れないように。
sparc, sparc64	Sun や富士通の <b>SPARC</b> 系の CPU を搭載したワークステーション等のマシン用。x86 系のマシンに入れないように。
alpha	HP (旧 DEC, 旧 Compaq) の <b>Alpha</b> 系の CPU を搭載したマシン用。x86 系のマシンに入れないように。

表示または名称	説明
noarch	特定の CPU アーキテクチャに依存しないパッケージ。フォント、PHP、Perl、Python スクリプト、設定ファイルなどが収録されています。
src	ソースパッケージ (SRPM と呼ばれる)、rebuild することで上記のパッケージをつくることができます。

## インストールされているパッケージを知りたい。

ターミナルで `rpm -qa` と入力するとインストールされているパッケージの一覧を表示できます。一画面では収まらないので `less`、`grep` コマンドを併用してください。

## パッケージをアップデートすると設定ファイルが置き換わったりそのままだったりするのはなぜ？

正しく作られた RPM パッケージなら、アップデートする際以下のような動作をするためです。

### パッケージに含まれる設定ファイルが同じ場合

現在の設定ファイルをそのまま保持。

### パッケージに含まれる設定ファイルに違いがあるが、従来のものも使える(互換性がある)場合

現在の設定ファイルはそのまま保持した上で、新規パッケージに含まれる設定ファイルは\*。  
`rpmnew`という名前でインストール。

### パッケージに含まれる設定ファイルに違いがあり、従来のものは使えない(互換性がない)場合

現在の設定ファイルと新規パッケージの設定ファイルを置き換え、従来の設定ファイルを\*。  
`rpmshave`という名前で保管。

ただし、普通は通常のアップデートで設定の書式の互換性がなくなったりすることはほとんどないので、設定ファイルが置き換わることはほとんどありません。

ディストリビューションのアップグレードの際には、設定ファイルの互換性がなくなる場合があるので、注意する必要があります。

### 注意: 正しく作られてないパッケージの場合

RPM パッケージの spec 内で、設定ファイルを設定ファイルと指定していない(普通のファイルになっている)場合は、問答無用で新しいパッケージの設定ファイルに上書きされます。

## 必要なライブラリ(ソフトウェア)がインストールされているのにそれを使用するソフトをコンパイルできません。

たとえば、コンパイルに `libhoge` が必要な `huga` というソフトをコンパイルしようとしているのにエラーが出てできない場合。

### devel パッケージを入れていますか??

`rpm -qa | grep libhoge` と入力してみて `libhoge-1.0.0-1` とかしか表示されないならば、`libhoge-devel-1.0.0-1.i386.rpm` ような devel パッケージを入れて下さい。

devel パッケージにはインクルードファイルなど開発に必要な物が入っています。

## ××コマンドのソースはどこにあるのですか?

rpm -qf ×× でそのコマンドが属する RPM パッケージを調べる

そのパッケージの \*.src.rpm を探す。

## RPMファイルの作り方

簡単な方法は [MakeRPMs](#) にあります。

詳しく知りたい人は RPM HOWTO などを参照のこと。(日本語訳の方は最新の環境ではうまくいかない場合があるかもしれません。)

- RPM HOWTO (日本語訳)

<http://linuxjf.sourceforge.jp/JFdocs/RPM-HOWTO.html>

- RPM HOWTO (原文)

<http://www.tldp.org/HOWTO/RPM-HOWTO/index.html>

- Maximum RPM

<http://www.rpm.org/max-rpm/>

- RPMパッケージの作成

<http://vinelinux.org/manuals/making-rpm.html>

## Source RPMのリビルドからインストールまでの作業は実際どのようにしますか。

パッケージは、通常 Source RPM(SRPM) の形式でも配布されています。

メーカー Sylpheed の場合を例にして、簡単に説明します。

まず Sylpheed 公式サイトから ソースRPMとして提供されている sylpheed-1.0.0-1.src.rpm をダウンロードし、コンソール(ターミナル)より次のコマンドを入力します。

```
# rpmbuild --rebuild sylpheed-1.0.0-1.src.rpm
.....(途中経過省略)
Checking for unpackaged file(s): /usr/lib/rpm/check-files /var/tmp/sylpheed-root
書き込み中: /usr/src/redhat/RPMS/i386/sylpheed-1.0.0-1.i386.rpm
書き込み中: /usr/src/redhat/RPMS/i386/sylpheed-debuginfo-1.0.0-1.i386.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.21632
+ umask 022
+ cd /usr/src/redhat/BUILD
+ cd sylpheed-1.0.0
+ rm -rf /var/tmp/sylpheed-root
+ exit 0
Executing(--clean): /bin/sh -e /var/tmp/rpm-tmp.21632
+ umask 022
+ cd /usr/src/redhat/BUILD
+ rm -rf sylpheed-1.0.0
```

+ exit 0

このログから分かるとおり、sylpheed のRPMパッケージは /usr/src/redhat/RPMS/i386/ ディレクトリに作成されました。そこで、今度はこのディレクトリに移動し、通常のインストール手順を実行すれば、Sylpheed のインストール作業は完了します。

- 詳しくは Manpage of RPMBUILD ([JM Project](#) 内のページ) にあります。

<http://linuxjm.sourceforge.jp/html/rpm/man8/rpmbuild.8.html>

## Debian(deb)/Slackware(tgz) パッケージから RPM を作成できますか、(Alien)

Alien パッケージをインストールすれば、それぞれのあいだで相互変換することが可能となります。

- Alien package converter

<http://kitenet.net/~joey/code/alien/>

### インストール

root で rpmbuild -ta alien\_8.53.tar.gz と実行します。

```
Checking for unpackaged file(s): /usr/lib/rpm/check-files /tmp/alien-8.53.build
書き込み中: /usr/src/redhat/SRPMS/alien-8.53-1.src.rpm
書き込み中: /usr/src/redhat/RPMS/noarch/alien-8.53-1.noarch.rpm
```

/usr/src/redhat/RPMS/noarch に alien-8.53-1.noarch.rpm ができるので、これを通常の手順で展開すればOKです。

### パッケージを変換

パッケージを変換するには、alien -\* 変換するパッケージと入力すればOKです。-\* には -r (rpm), -d (deb), -t (tgz) が選べます。

- rpm > deb 変換: alien -d hoge.rpm
- rpm > tgz 変換: alien -t hoge.rpm
- deb > rpm 変換: alien -r hoge.deb
- tgz > rpm 変換: alien -r hoge.tgz

## ソースファイルから RPM を簡単に作成できますか、(CheckInstall)

ソースから RPM を作成したいのなら、CheckInstall を利用すると便利です。

- CheckInstall 公式ウェブサイト

<http://asic-linux.com.mx/%7Eizto/checkinstall/>

- ドキュメント (documentation)

<http://asic-linux.com.mx/%7Eizto/checkinstall/docs.php>

- ダウンロード (Download)

<http://asic-linux.com.mx/%7Eizto/checkinstall/download.php>

ソースコード: checkinstall-1.6.0.tgz

バイナリーパッケージ:

RPM バイナリー checkinstall-1.6.0-1.i386.rpm

Debian バイナリー checkinstall\_1.6.0-1\_i386.deb

**CheckInstall の使い方 (hoge-1.2.3.tar.gz をインストールするとき)**

```
$ tar xzvf hoge-1.2.3.tar.gz
$ cd hoge-1.2.3
$ ./configure
$ make
$ su
# checkinstall
```

以上の作業で hoge-1.2.3-1.rpm ができるので、これを通常の手順でインストールします。

**関連日本語解説サイト**

[@IT : ソースファイルからRPMファイルを作成するには](#)

**ダウングレード(古いバージョンのパッケージをインストール)したいとき**

アップグレードしたパッケージに不具合があった場合、もう一度、バージョンが古いパッケージをインストールしたいことがあります。このような場合には、'-U'オプションとともに '--oldpackage' オプションを用います。

```
# rpm -Uvh --oldpackage パッケージファイル名
```

**2ch Linux板 スレッド**

rpm作成スレッド

<http://pc11.2ch.net/test/read.cgi/linux/1034402194/>