

# パッケージ管理ツール

新井 友輔, 山下 俊樹

Yusuke ARAI, Toshiki YAMASHITA

## 1 はじめに

従来、ソフトウェアの導入には、ソースコードをコンパイルし、作成された実行ファイルを手動でシステムのディレクトリに追加することでインストールしていた。しかし近年では、ソフトウェアの大規模化が著しく、手動による方法では、ソフトウェアの導入時や導入後の管理が煩雑になった。そこで、いくつかのソースコードをコンパイルしパッケージとしてまとめ、パッケージを組み合わせることで、ソフトウェアを管理するようになった。ソースコードのパッケージ化により、ソフトウェア管理は簡略化され、作業の効率化に貢献した。そして、パッケージ管理ツールの登場により、パッケージの操作が単純化した。本稿ではパッケージ管理ツールについて述べる。

## 2 パッケージ

### 2.1 概要

パッケージとは、設定ファイル、データファイル、プログラム本体などソフトウェアを構成する要素を一式にまとめたものである。パッケージには複数のユーザが同じ開発環境を実現するために、パッケージの依存関係に関する情報を付与する。パッケージには rpm や deb のような形式があり、その形式はパッケージ管理ツール毎に異なり、コマンド操作も異なる。

### 2.2 パッケージ間の関係性

パッケージを正しく動作させるために、パッケージ間の関係性が重要となる。パッケージ A が別のパッケージ B に依存するとは、A が適切に機能するために B が必要ということの意味する。パッケージ間の関係性には、依存、推奨、提案、衝突の 4 種類がある。Fig. 1 にパッケージ間の関係性の例を示す。

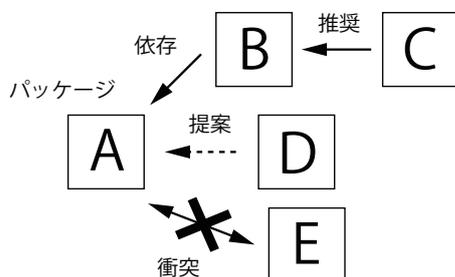


Fig.1 パッケージ間の関係性

例としてパッケージ A が正しく動作するために、パッケージ B, C をインストールする必要があるものとする。パッケージ B がパッケージ C を推奨するとは、B が要望する重要な機能を C が提供することを意味する。依存と

は異なり、パッケージのインストールが必須とは限らないが、ユーザがそれを必要としないことを把握している場合を除いてユーザは通常インストールする。パッケージ A が別のパッケージ D を提案するとは、D が提供する機能が A の機能を強化するという意味する。ただし、D の提供する機能強化は必要とは限らない。パッケージ A が別のパッケージ E と衝突するとは、A と E のパッケージが同時にインストールできないという意味である。パッケージ A がパッケージ B を推奨しており、かつパッケージ B がパッケージ C を推奨していれば、パッケージ A もまたパッケージ C を推奨していることになる。従って、1 つのアプリケーションをインストールしようとしているにも関わらず、多数のパッケージをインストールする必要がある場合がある。

## 3 パッケージ管理ツール

### 3.1 概要

かつて UNIX 系の OS では、ソースコードをコンパイルし、作成された実行ファイルを手動でシステムのディレクトリに加え、ソフトウェアをインストールしていた。例えば、この方法は、エラーが生じた場合、誤りが生じているソースコードを探索するために、ソースコードの依存関係を全て理解する必要があった。そのため、確認と訂正の作業に時間を要した。ソースコード単位でのソフトウェアの管理は煩雑であった。そこで、ソースコードをあらかじめビルドし、パッケージ化したものをディレクトリで管理するパッケージ管理ツールが考え出された。パッケージ管理ツールは主に、パッケージ間の依存関係の管理、パッケージのインストールの簡略化、およびパッケージのアップデートの簡略化を行う。

### 3.2 パッケージ管理ツールの利点と欠点

パッケージ管理ツールは、パッケージ単位でソフトウェアを導入する。インストールは、パッケージの概要や説明文、正しく動作するために必要となる他のパッケージに関する情報が記述されたメタデータをもとに行う。パッケージをインストールすると、システムのデータベースにパッケージをインストールしたという情報を登録する。

パッケージ内のファイルには、ユーザが使用しないファイルが含まれている場合がある。例として、CD/DVD のサポートを行うパッケージを考える。DVD を使用しないユーザにとって、パッケージ内に含まれている DVD のサポート機能は不要となる。不要なファイルを含むパッケージを使用すると、ソフトウェアの規模が拡大し、PC のメモリ容量を不要に圧迫する<sup>1)</sup>。パッケージ単位で行うインストールでは、ユーザにとって不要な機能の有無を変更

できない。従って、必要な機能のみを選択し導入する場合は、パッケージごとインストールするのではなく、必要な機能に該当するソースコードをインストールする。

アンインストールもパッケージ単位で行う。例えば、アンインストールするパッケージが、他のパッケージに依存されている場合、依存しているパッケージもアンインストールする必要がある。パッケージ管理ツールを利用することで、ユーザはコマンド入力するだけで、アンインストールするパッケージとそのパッケージに依存するパッケージをまとめてアンインストールできる。CentOS で使用するパッケージ管理ツールである yum の場合、yum remove とコマンド入力すると、インストールしたパッケージとそのパッケージに依存するパッケージも同時にアンインストールする。

パッケージ管理ツールではソフトウェアをアップデートする場合、あるパッケージの新しいパッケージが公開されると、そのパッケージをアップデートする。Fig. 2 にパッケージの入れ替えによるソフトウェアのアップデートの例を示す。

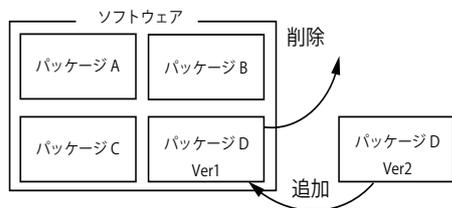


Fig.2 パッケージ単位でのアップデート

あるソフトウェアは 4 つのパッケージで構成されており、パッケージ D の新しいパッケージが公開されたとする。ソフトウェアをアップデートする場合、4 つすべてのパッケージを更新する必要はなく、新しいパッケージを追加し、以前のパッケージを削除することで完了する。追加されたパッケージに依存する他のパッケージがある場合、依存関係に従ってパッケージ管理ツールが自動で追加する。ソフトウェアのアップデートには、ソフトウェアのバグやセキュリティホールのような脆弱性を修正する役割がある。ソフトウェアのアップデートを行うために機能を停止する場合、停止期間中にユーザはそのソフトウェアを利用することはできない<sup>2)</sup>。このため、可能な限り早く脆弱性を修正する必要がある。パッケージを入れ替える方法は、ソフトウェアの修正時間を短縮させるために効果的である。

## 4 パッケージ管理ツールとリポジトリ

### 4.1 リポジトリ

リポジトリとは、パッケージを管理する場所であり、リモートリポジトリとローカルリポジトリの 2 種類がある。リモートリポジトリは、専用のサーバに配置して複数人で共有するためのリポジトリである。ローカルリポジトリは、ユーザー一人ひとりが利用するために、ユーザ個人の PC に配置するリポジトリである。

### 4.2 パッケージの入手

多くのパッケージは、リモートリポジトリで管理されている。パッケージを入手するためには、使用したいパッケージを管理するリポジトリにアクセスする必要がある。Fig. 3 にパッケージ管理ツールとリポジトリの関係を示す。

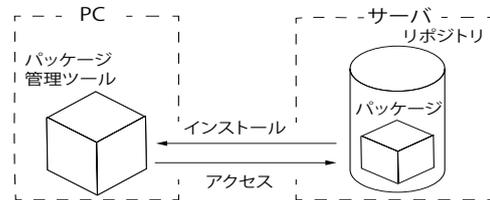


Fig.3 パッケージ管理ツールとリポジトリ

パッケージ管理ツールは、リポジトリにリポジトリの ID、名前、および場所をもとにアクセスし、リポジトリの情報を追加する。パッケージ管理ツールに複数のリポジトリの情報を追加することで、ユーザは特定のパッケージを所有しているリポジトリを探索することなくパッケージをインストールできる。パッケージ管理ツールはインストールするパッケージに依存する他のパッケージも自動でインストールする。情報を追加したりリポジトリ内に、特定のパッケージが存在しない場合、パッケージ管理ツールはパッケージ名を検索することで、そのパッケージを所持するリポジトリを特定できる。インストールしたいパッケージをキーワード検索する場合、該当するパッケージが多く存在するため、grep コマンドによって特定することが一般的である。

## 5 今後の展望

パッケージ管理ツールの登場により、ユーザはソフトウェアの管理が容易となった。しかし、パッケージ管理ツールの機能は OS や Linux のディストリビューション毎に異なり、依存関係解決や推奨パッケージの自動インストールなど操作の仕様が異なる。その結果、ユーザの意思に反して、パッケージの更新や削除を行う場合がある。OS や Linux のディストリビューションを問わない次世代パッケージ管理ツールが登場することで、ユーザの要望に合った操作で、異なる OS でも同じ開発環境が構築できると考える。

## 参考文献

- 1) りお：普通の東工大生が【gentoo】入れてみた。、入手先、(<http://titech-ssr.blog.jp/archives/1013311908.html>)
- 2) 宮本久仁男：第 1 回 脆弱性対策のためにパッケージを入れ替えるということ、入手先、(<http://gihyo.jp/admin/serial/01/infrasec/0001>)