第5章 生産性と性能を両立させるモダン・メカニズム

ラズパイ4で動くJuliaアプリケーションの作成

寺崎 敏志

```
├──Project.toml
└──src
   └──CameraApp.jl
```

図1 プロジェクトで使う最低限のファイル

ここまでJuliaのパッケージを導入し，幾つかコードを書いてきました．

ここでは，パッケージをあらかじめコンパイルして高速化したり，Juliaの起動イメージに組み込んだりできる，事前（AOT：Ahead of Time）コンパイルを試してみます．

ラズベリー・パイでカメラの画像を表示する，単純なアプリケーションのプロジェクトを例に実験します．

プロジェクトを用意する

● プロジェクトの基本構成

Juliaのプロジェクト（Project）は，

- ソースコードを集めたsrc
- 使い方，マニュアル，API仕様書をまとめたdoc
- ソフトウェアとしての品質を維持・管理するテストコードをまとめたtest
- 依存関係をまとめたProject.toml，Manifest.tomなどのファイル

で構成されます．

Project.tomlに，依存するパッケージを記述しておくと，プロジェクトに必要なパッケージを導入できます．

プロジェクト，パッケージ，アプリケーションと用語が出てきました．プロジェクトはパッケージ，アプリケーションを包括する概念です．

パッケージ（Package）は再利用性が高い機能を集めたプロジェクトのことで，ソフトウェアとしてのライブラリです．

アプリケーション（Application）はエンドユーザ向けに使われるJuliaのプロジェクトです．これらの概念は厳密に区別されているわけではないので，インターネット上では，プロジェクトよりもパッケージの話題を調べると情報が多く出てきやすいでしょう．

● まず最小構成のプロジェクトを作る

ターミナルで次のコマンドを実行します[注1]．

```
$ julia -e 'using Pkg; Pkg.generate("CameraApp")'↵
```

CameraApp以下に**図1**のようなファイル構造が得られます．

これがJuliaの必要最小限のプロジェクト構成です．

GitHubでリポジトリを作り，管理したい場合はCameraApp以下のファイルをそっくりそのままリポジトリの直下に移動します．

● プロジェクトのプロンプトを起動できるようにする

末尾に“.jl”を付けておくとインターネットで，あなたのパッケージを公開するときに検索されやすくなります．以下，リポジトリの名前がCameraApp.jlと仮定して話を進めます．

Juliaのプロジェクトを開発するには，その環境のなかで作業することになります．そのためにJuliaの起動時にオプションとして，

```
--project=＜Project.tomlがあるディレクトリ＞
```

を指定します．

```
$ cd CreateApp.jl↵
$ julia --project=.↵
$ julia> ]↵
$ (CameraApp) pkg>
```

“]”を入力して，Pkg REPLに切り替えると元々，

```
(@v1.4) pkg>
```

と表示されていたプロンプトが，

注1：環境によってはコマンド・プロンプトでは動作しないようです．Juliaを起動し，“;”を入力するとShellプロンプトが表示されるので，そこにコマンドを入力してください．