Nixpkgs Reference Manual を読む③（Nixpkgs lib）

Mutsuha Asada
@mutsuha_asada

Nix 日本語コミュニティ ゼミ 2026/01/20

前回までのあらすじ

- ソフトウェアを学習する際に最も効率が良いのはマニュアルの初手通読だが、ハードルが高く、行間が広めなのでどうしても手を動かすことで理解しようとしてしまう
- このシリーズ（Nixpkgs Reference Manual を読む）では、マニュアルを通読して、必要に応じて行間を埋めて発表するという形式を取る
- 前回はライブラリ関数のうち、マイナーで面白いものを 4 つ選んで紹介しました
  - 誰が知ってるねんということで非常に好評（？）でした

目次

1. lib.attrsets

概要

- `lib` は、nixpkgs が提供する Nix 式向けの標準ライブラリ
- 今回は `lib.attrsets` にフォーカスします
  - 属性セット（attrset）を操作する関数群
- `attrset` は nixpkgs を扱う上で本当に出番が多い
  - パッケージ集合（pkgs）
  - module system（options / config）
  - overlay

attrset（属性セット）とは

- Nix の基本データ構造のひとつ
  - キーは文字列
  - 値は任意
  - `{ a = 1; b = 2; }`
- ネストできる
  - `{ a = { b = 3; }; }`
- 順序は基本的に意味を持たない

今回紹介する関数

- `lib.attrsets.optionalAttrs`
- `lib.attrsets.genAttrs`
- `lib.attrsets.mapAttrs`
- `lib.attrsets.mapAttrs'`
- `lib.attrsets.recursiveUpdate`

`lib.attrsets.optionalAttrs`

- 条件が `true` のときだけ属性を追加する
- `if cond then { ... } else {}` を毎回書かなくてよい
- overlay や mkDerivation の引数組み立てでよく使う

lib.attrsets.optionalAttrs

```
let
  pkgs = import <nixpkgs> {};
  lib = pkgs.lib;
  enableDocs = true;
  enableDebug = false;
  attrs =
    {
      pname = "demo";
      version = "0.1.0";
    }
    // lib.attrsets.optionalAttrs enableDocs {
      doCheck = true;
      nativeBuildInputs = [ pkgs.pandoc ];
    }
    // lib.attrsets.optionalAttrs enableDebug {
      NIX_CFLAGS_COMPILE = "-O0 -g";
    };
in attrs
```

lib.attrsets.genAttrs

- 文字列リスト `names` から属性セットを生成する
- `genAttrs names (name: value)`
- 列挙したキーに対して同じ変換を当てたい時に便利

lib.attrsets.genAttrs

```
let
  pkgs = import <nixpkgs> {};
  lib = pkgs.lib;
  names = [ "clang" "gcc" "rustc" ];
  toolchainByName = lib.attrsets.genAttrs names (name: {
    package = pkgs.${name};
    version = pkgs.${name}.version or "unknown";
  });
in
toolchainByName
```

lib.attrsets.mapAttrs

- attrset の値だけを写像したいときに使う（map）
- builtins.mapAttrs を使いやすくしたもの
- mapAttrs :: (name: value: newValue) -> AttrSet -> AttrSet

lib.attrsets.mapAttrs

```
let
  pkgs = import <nixpkgs> {};
  lib = pkgs.lib;
  x = {
    a = 1;
    b = 20;
    c = 300;
  };
  y = lib.attrsets.mapAttrs (name: value: value + 1) x;
in
y
```

lib.attrsets.mapAttrs'

- キーも変えたい場合に使う
- `mapAttrs` はキーが固定なので、キー変換が必要なら `mapAttrs'`
- `nameValuePair` を返す
  - `{ name = "..."; value = ...; }`

lib.attrsets.mapAttrs'

```
let
  pkgs = import <nixpkgs> {};
  lib = pkgs.lib;

  x = { foo = 1; bar = 2; };

  y = lib.attrsets.mapAttrs' (name: value:
    lib.attrsets.nameValuePair ("prefix-" + name) (value * 10)
  ) x;
in
y
```

`lib.attrsets.recursiveUpdate`

- `//` は 1 段だけのマージ
- `recursiveUpdate` はネストした attrset を深くマージする
- module system や config の上書きに近い操作ができる

lib.attrsets.recursiveUpdate

```
let
  pkgs = import <nixpkgs> {};
  lib = pkgs.lib;
  base = {
    a = { b = 1; c = 2; };
    d = 10;
  };
  patch = {
    a = { c = 999; };
    e = 42;
  };
  shallow = base // patch;
  deep = lib.attrsets.recursiveUpdate base patch;
in
{
  inherit shallow deep;
}
```

2. まとめ

まとめ

- `lib.attrsets` は nixpkgs の巨大な属性セットを扱うためのライブラリ
- 今日紹介したのは以下の関数
  - `optionalAttrs`
  - `genAttrs`
  - `mapAttrs`
  - `mapAttrs'`
  - `recursiveUpdate`