A tour of Nix

29 / 35 Fold: Introduction

prev next

Exercise

Your job:

- `ex0`: use `fold` to write a function that counts all `strings` with value "a" in a given `list`
- `ex1`: use `fold` to multiply each element by 2 and add the [8] to it

fold (aka foldr)

`fold func init [x_1 x_2 ... x_n] == func x_1 (func x_2 ... (func x_n init))`

- `func`, a function like `(el: container: container + el)`
- `init` as the initial starting value

String examples:

- `lib.fold (el: c: el + c) "z" [ "a" "b" "c" ] => "abcz"`
- `lib.fold (el: c: c ++ [el]) [0] [1 2 3] => [ 0 3 2 1 ]`

foldl

`foldl func init [x_1 x_2 ... x_n] == func (... (func (func init x_1) x_2) ... x_n)`.

- `func`, a function like `(container: el: container + el)`
- `init` as the initial starting value

List examples:

- `lib.foldl (c: el: el + c) "z" [ "a" "b" "c" ] => "cbaz"`
- `lib.foldl (c: el: c ++ [el]) [0] [1 2 3] => [ 0 1 2 3 ]`

Note: See https://nixos.org/manual/nixpkgs/stable/#chap-functions, search **lib.lists.foldr**

Note: See https://nixos.org/manual/nix/stable/command-ref/new-cli/nix3-repl

Note: See video @youtube

```
1  with import <nixpkgs> { };
2  with lib;
3  let
4    list = ["a" "b" "a" "c" "d" "a"];
5    intList = [ 1 2 3 ];
6    countA = XXX
7    #mulB = XXX
8  in
9  rec {
10   example = fold (x: y: x + y) "z" ["a" "b" "c"]; #is "abcz"
```

```
  ex0 = countA list; #should be 3
  #ex1 = mulB intList; #should be [ 2 4 6 8 ]
}
```

reset solution run

```
with import <nixpkgs> { };
with lib;
let
  list = ["a" "b" "a" "c" "d" "a"];
  intList = [ 1 2 3 ];
  countA = l: fold (el: c: if el == "a" then c + 1 else c) 0 l;
  mulB = l: fold (el: c: [(el * 2)] ++ c) [8] l;
in
rec {
  example = fold (x: y: x + y) "z" ["a" "b" "c"]; #is "abcz"
  ex0 = countA list; #should be 3
  ex1 = mulB intList; #should be [ 2 4 6 8 ]
}
```