lib.genAttrs

`lib.genAttrs` — функция, которая принимает список имён и функцию-генератор, создавая набор атрибутов, где каждый элемент списка становится ключом, а значение вычисляется функцией-генератором на основе этого ключа.
Инструмент для избежания повторения в конфигурациях NixOS, особенно когда нужно применить похожую конфигурацию к множеству сущностей(службы, пользователи, способы взаимодействия(interfaces))

?????????

```
genAttrs :: [String] -> (String -> Any) -> AttrSet
```

- первый аргумент — список строк (имена будущих атрибутов);
- второй аргумент — функция, принимающая имя и возвращающая значение для этого атрибута;

Возвращает набор атрибутов `{ name1 = value1; name2 = value2; ... }`

???????

????????

```
{ lib }:
let
  names = [ "foo" "bar" "baz" ];
  # Функция-генератор: добавляет "-suffix" к имени
  addSuffix = name: "${name}-suffix";

  result = lib.genAttrs names addSuffix;
in
result
```

Результат:

```
{
  foo = "foo-suffix";
  bar = "bar-suffix";
  baz = "baz-suffix";
}
```

??????? ????????? ?????

```
{ config, lib, pkgs, ... }:
let
  myServices = [ "nginx" "postgresql" "redis" ];

  # Создаём атрибутный набор включённых сервисов
  enabledServices = lib.genAttrs myServices (name: {
    enable = true;
  });
in
{
  # Включаем все сервисы одним выражением
  services = enabledServices;

  # Эквивалентно:
  # services.nginx.enable = true;
  # services.postgresql.enable = true;
  # services.redis.enable = true;
}
```

??????? ????????????

```
{ config, lib, ... }:
let
  users = [ "alice" "bob" "charlie" ];

  # Создаём базовую конфигурацию для каждого пользователя
  userConfigs = lib.genAttrs users (name: {
    isNormalUser = true;
    extraGroups = [ "wheel" "networkmanager" ];
    createHome = true;
    home = "/home/${name}";
  });
in
{
  users.users = userConfigs;
}
```

????????? ???????

```
{ pkgs, lib, ... }:
let
  languages = [ "python3" "go" "nodejs" "rustc" ];

  # Устанавливаем последние версии языков программирования
  devPackages = lib.genAttrs languages (name: pkgs.${name});
in
{
  environment.systemPackages = builtins.attrValues devPackages;

  # Или с дополнительной конфигурацией:
  languages = lib.genAttrs languages (name: {
    enable = true;
    package = pkgs.${name};
  });
}
```

????????? ???????????????? ??????

```
{ config, lib, ... }:
let
  domains = [ "example.com" "test.com" "admin.example.com" ];

  nginxVhosts = lib.genAttrs domains (domain: {
    serverName = domain;
    root = "/var/www/${domain}";
    locations."/".proxyPass = "http://localhost:8080";
    enableSSL = true;
    sslCertificate = "/var/ssl/${domain}.crt";
    sslCertificateKey = "/var/ssl/${domain}.key";
  });
in
{
  services.nginx.virtualHosts = nginxVhosts;
}
```

?????????? ?????? ? ???????????? ?? ?????

```
{ lib, ... }:
let
```

```
  interfaces = [ "eth0" "eth1" "wlan0" ];

  networkConfig = lib.genAttrs interfaces (name:
    if lib.hasPrefix "eth" name then {
      useDHCP = false;
      ipv4.addresses = [{
        address = "192.168.1.${toString (10 + lib.elemIndex name interfaces)}";
        prefixLength = 24;
      }];
    } else {
      useDHCP = true;
      wireless.enable = true;
    }
  );
in
{
  networking.interfaces = networkConfig;
}
```

??????? ? `listToAttrs`

Часто путают `genAttrs` с `listToAttrs`. Вот ключевое отличие:

```
# genAttrs - проще, когда нужно создать значения на основе имён
lib.genAttrs [ "a" "b" ] (name: "value-${name}")
# => { a = "value-a"; b = "value-b"; }

# listToAttrs - когда у вас уже есть пары {name, value}
lib.listToAttrs [
  { name = "a"; value = 1; }
  { name = "b"; value = 2; }
]
# => { a = 1; b = 2; }
```

?????????????

????????????? ? ??????? ?????????

```
lib.genAttrs (lib.filter (s: lib.hasPrefix "dev" s) allNames) (name: ...)
```

????????????? ? ??????? NixOS

```
options = lib.genAttrs [ "foo" "bar" ] (name: lib.mkOption {
  type = lib.types.str;
  default = name;
});
```

???????????? ???????? ????????? ?? ??????

```
let
  keys = builtins.attrNames someInputSet;
  processed = lib.genAttrs keys (key: transformFunction someInputSet.${key});
in
processed
```

Revision #4
Created 2025-12-24 03:04:36 UTC by Антон Сергеевич Абраменко
Updated 2025-12-24 03:25:59 UTC by Антон Сергеевич Абраменко