lib.mapAttrsToList

???????????, ???????? ? ????????????

`lib.mapAttrsToList` — это функция, которая **преобразует набор атрибутов в список, применяя функцию к каждой паре ключ-значение.**

```
lib.mapAttrsToList f attrs
```

Описание:

- `f` — функция, принимающая два аргумента: `name` (ключ) и `value` (значение);
- `attrs` — набор атрибутов.

Возвращает **список** результатов применения `f` к каждому элементу.

Использование:

- Генерация конфигурационных файлов.
- Создания списков для динамической конфигурации.
- Преобразования данных между форматами.

???????

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

Исходный код:

```
{ lib }:
let
  users = {
    alice = 25;
    bob = 30;
    charlie = 35;
  };

  result = lib.mapAttrsToList (name: age: "${name} is ${toString age} years old") users;
in
```

```
result
```

Результат:

```
[ "alice is 25 years old" "bob is 30 years old" "charlie is 35 years old" ]
```

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

```
{ lib, ... }:
let
  userConfigs = {
    alice = {
      uid = 1001;
      group = "users";
      shell = "/bin/bash";
    };
    bob = {
      uid = 1002;
      group = "developers";
      shell = "/bin/zsh";
    };
  };

  # Преобразуем в формат, понятный для users.users
  usersList = lib.mapAttrsToList (username: config: {
    name = username;
    inherit (config) uid group shell;
    isNormalUser = true;
  }) userConfigs;
in
{
  users.users = builtins.listToAttrs (map (u: { name = u.name; value = u; }) usersList);
}
```

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

```
{ lib, ... }:
let
  services = {
    nginx = {
      port = 80;
```

```
      enable = true;
    };
    postgres = {
      port = 5432;
      enable = true;
    };
    redis = {
      port = 6379;
      enable = false;
    };
  };

  # Фильтруем и создаем конфиги только для включенных сервисов
  enabledServices = lib.mapAttrsToList (name: config:
    lib.optional config.enable {
      serviceName = name;
      inherit (config) port;
      configFile = ./${name}-config.conf;
    }
  ) services;
in
{
  # enabledServices будет списком списков, flatten его
  services = lib.flatten enabledServices;
}
```

????????? firewall

```
{ lib, ... }:
let
  openPorts = {
    http = 80;
    https = 443;
    ssh = 22;
    smtp = 25;
  };

  firewallRules = lib.mapAttrsToList (name: port: {
    # Создаем правило для каждого порта
    rule = "-p tcp --dport ${toString port} -j ACCEPT";
```

```
    description = "Allow ${name} (port ${toString port})";
  }) openPorts;
in
{
  networking.firewall.extraCommands = lib.concatMapStringsSep "\n" (rule:
    "iptables -A INPUT ${rule.rule} # ${rule.description}"
  ) firewallRules;
}
```

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

```
{ lib }:
let
  data = { a = 1; b = 2; c = 3; };
in
{
  # mapAttrsToList: возвращает список
  mapAttrsToList = lib.mapAttrsToList (n: v: "${n}=${toString v}") data;
  # Результат: [ "a=1" "b=2" "c=3" ]

  # mapAttrs: возвращает набор
  mapAttrs = lib.mapAttrs (n: v: v * 2) data;
  # Результат: { a = 2; b = 4; c = 6; }

  # attrValues: только значения
  attrValues = lib.attrValues data;
  # Результат: [ 1 2 3 ]
}
```

????????? systemd ?????

```
{ lib, ... }:
let
  backupJobs = {
    "backup-home" = {
      source = "/home";
      destination = "/backup/home";
      schedule = "daily";
    };
    "backup-etc" = {
```

```
      source = "/etc";
      destination = "/backup/etc";
      schedule = "weekly";
    };
  };

  systemdServices = lib.mapAttrsToList (jobName: config:
    {
      name = "backup-${jobName}";
      value = {
        description = "Backup ${config.source}";
        script = ''
          rsync -av ${config.source} ${config.destination}
        '';
        startAt = config.schedule;
      };
    }
  ) backupJobs;
in
{
  systemd.services = builtins.listToAttrs systemdServices;
}
```

Revision #5
Created 2025-12-22 07:26:59 UTC by Антон Сергеевич Абраменко
Updated 2025-12-22 08:07:16 UTC by Антон Сергеевич Абраменко