lib.mkMerge

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

`lib.mkMerge` — это функция, которая позволяет **объединять несколько наборов атрибутов в один**, решая конфликты через **приоритеты**.
Это особенно полезно в модульной конфигурации NixOS, когда вы хотите разделить конфигурацию на части и объединить их без ручного разрешения конфликтов.
Когда вы объединяете наборы атрибутов в Nix, могут возникать конфликты (одинаковые ключи с разными значениями). `lib.mkMerge` решает эти конфликты, используя систему приоритетов: каждый элемент получает приоритет (по умолчанию 0), и выигрывает значение с наивысшим приоритетом.

???????

????????

Исходный код

```
{ lib, ... }:

let
  # Три конфигурации, которые мы хотим объединить
  config1 = { boot.loader.grub.enable = true; };
  config2 = { boot.loader.systemd-boot.enable = true; };
  config3 = { networking.hostName = "myhost"; };
in
lib.mkMerge [
  config1
  config2  # Конфликт с config1 по boot.loader.*
  config3
]
```

Описание

В этом примере `config1` и `config2` конфликтуют (оба определяют загрузчик). По умолчанию `lib.mkMerge` просто выберет последнее значение, но с приоритетами можно контролировать результат.

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

```
{ lib, ... }:

lib.mkMerge [
  # Приоритет 1000 (высокий)
  (lib.mkIf false {
    services.nginx.enable = true;
  })

  # Приоритет 100 (средний)
  {
    services.nginx.enable = lib.mkDefault false;
    services.nginx.virtualHosts."example.com".root = "/var/www";
  }

  # Приоритет 150 (выше среднего)
  (lib.mkForce {
    services.nginx.enable = true;  # Переопределит mkDefault
  })
]
```

Описание:

- `mkIf` имеет приоритет 1000
- `mkDefault` имеет приоритет 100
- `mkForce` имеет приоритет 150
- Результат: `services.nginx.enable = true` (побеждает `mkForce`)

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

```
{ lib, config, ... }:

let
  commonNetwork = {
    networking.networkmanager.enable = true;
    networking.firewall.enable = true;
  };

  homeConfig = {
    networking.hostName = "home-pc";
```

```
    networking.firewall.allowedTCPPorts = [ 80 443 ];
  };

  workConfig = {
    networking.hostName = "work-laptop";
    networking.firewall.allowedTCPPorts = [ 22 3389 ];
    networking.proxy.default = "http://proxy.company.com:8080";
  };

  # Динамически выбираем конфигурацию
  environmentConfig = if config.isWorkEnvironment then workConfig else homeConfig;
in
{
  imports = [ ./hardware-configuration.nix ];

  options.isWorkEnvironment = lib.mkOption {
    type = lib.types.bool;
    default = false;
  };

  config = lib.mkMerge [
    commonNetwork
    environmentConfig
    {
      # Этот блок имеет самый высокий приоритет
      networking.nameservers = lib.mkForce [ "1.1.1.1" "8.8.8.8" ];
    }
  ];
}
```

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

```
# hardware/base.nix
{ lib, ... }:

{
  options.hardware = {
    profile = lib.mkOption {
      type = lib.types.enum [ "desktop" "laptop" "server" ];
      default = "desktop";
```

```
    };

    hasBluetooth = lib.mkOption {
      type = lib.types.bool;
      default = false;
    };
  };

  config = lib.mkMerge [
    # Базовая конфигурация для всех систем
    {
      hardware.enableRedistributableFirmware = true;
      powerManagement.enable = true;
    }

    # Конфигурация для ноутбуков
    (lib.mkIf (config.hardware.profile == "laptop") {
      services.tlp.enable = true;
      services.auto-cpufreq.enable = true;
      hardware.hasBluetooth = lib.mkDefault true;
    })

    # Конфигурация для серверов
    (lib.mkIf (config.hardware.profile == "server") {
      powerManagement.enable = lib.mkForce false;  # Отключаем на серверах
      services.openssh.enable = true;
    })

    # Конфигурация Bluetooth
    (lib.mkIf config.hardware.hasBluetooth {
      hardware.bluetooth.enable = true;
      services.blueman.enable = true;
    })
  ];
}
```

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

```
{ lib, ... }:

lib.mkMerge [
  {
    services.postgresql = lib.mkMerge [
      {
        enable = true;
        package = pkgs.postgresql_15;
      }
      (lib.mkIf config.services.grafana.enable {
        authentication = ''
          host grafana all ::1/128 md5
        '';
      })
    ];
  }

  {
    environment.systemPackages = with pkgs; [
      vim
      htop
    ];
  }
]
```

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

Приоритеты по умолчанию (от высокого к низкому):

1. `lib.mkIf` (1000) — условное включение
2. `lib.mkOverride` — явное указание приоритета
3. `lib.mkForce` (50) — принудительное значение
4. `lib.mkDefault` (1000 для false, 100 для true) — значения по умолчанию
5. Обычные значения (0) — стандартный приоритет

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

1. **Порядок важен только при равных приоритетах** — при одинаковых приоритетах побеждает последнее значение

2. **Глубокое слияние** — `mkMerge` рекурсивно объединяет вложенные атрибуты
3. **Не путать с** `mkOverride` — `mkMerge` объединяет списки наборов, а `mkOverride` изменяет приоритет конкретного атрибута

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

```
# Объединение конфигураций
config = lib.mkMerge [
  commonConfig
  (lib.mkIf condition conditionalConfig)
  (lib.mkForce forcedConfig)
  userConfig
];

# Эквивалентно (но более читаемо и модульно):
# {
#   commonConfig
#   conditionalConfig (если condition == true)
#   forcedConfig (с приоритетом)
#   userConfig
# }
```

Revision #2
Created 2025-12-22 07:50:58 UTC by Антон Сергеевич Абраменко
Updated 2025-12-22 08:03:44 UTC by Антон Сергеевич Абраменко